MATHEMATICS OF COMPUTATION
VOLUME 46, NUMBER 173
JANUARY 1986, PAGES 175–189

Numerical Solution of Large Sets of Algebraic Nonlinear Equations

By Ph. L. Toint

Abstract. This paper describes the application of the partitioned updating quasi-Newton methods for the solution of high-dimensional systems of algebraic nonlinear equations. This concept was introduced and successfully tested in nonlinear optimization of partially separable functions (see [6]). Here its application to the case of nonlinear equations is explored. Nonlinear systems of this nature arise in many large-scale applications, including finite elements and econometry. It is shown that the method presents some advantages in efficiency over competing algorithms, and that use of the partially separable structure of the system can lead to significant improvements also in the more classical discrete Newton method.

1. Introduction. In recent years, many researchers have investigated the solution to nonlinear problems involving an increasingly large number of variables. The finite element method has been very instrumental in this interest, since nonlinear partial differential equations give rise, by this method, to sets of nonlinear algebraic equations whose number of variables is proportional to the number of points considered in the discretization of the problem (see [13] for example). In this field, it is not uncommon that the Jacobian matrix of the system is unavailable or costly to compute, and one may be tempted to use quasi-Newton approximations for this important matrix. This type of procedure has indeed proven to be useful in small dense problems [2], and had been extended [10] to take into account the sparsity inherent in many of the large problems.

In the related field of unconstrained optimization, similar efforts were made to obtain methods that could handle efficiently a large number of nonlinear variables. Sparse quasi-Newton algorithms were proposed [12], [7], [11], and, more recently, a new class of methods, applicable to so-called "partially separable" functions has shown a lot of promise for the efficient solution of minimization problems involving several thousands of nonlinear variables (see [3], [4], [5] and [6]). These partially separable functions are functions that can be written as

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

where x is the vector of variables belonging to R^n, and where each "element function" $f_i(x)$ involves only a few components of this vector, or has a low-rank Hessian matrix for other reasons. Problems of this nature arise in discretized variational calculations, free-knots splines, nonlinear least squares, nonlinear net-

Received October 2, 1984; revised February 22, 1985.
1980 *Mathematics Subject Classification*. Primary 65H10, 65M99, 65N10.

0025-5718/86 $1.00 + $.25 per page

works, shape transformation and many other fields. The procedure proposed in [3], and subsequently analyzed and tested in the other papers referenced above, takes advantage of this low-rank property by using low-rank quasi-Newton approximations to the Hessian matrices of the various $f_i(x)$, giving up the idea of approximating the Hessian of $f(x)$ as a whole; hence the name "partitioned updating methods". Classical updating formulae, such as BFGS or rank one (RK1), were used for these low-rank approximations.

The numerical efficiency of this type of algorithm in unconstrained optimization leads naturally to the question as to whether the same kind of approach would be successful in the nonlinear equation setting. It is the purpose of this paper to throw some light on this issue.

In Section 2, we analyze more formally the partially separable structure that is present in many large-scale applications, as well as the various approximating schemes for the Jacobian matrix that result from this analysis. We also describe our algorithm and discuss a special memory management method and its implication on some parts of the calculation. In Section 3, we describe the test problems that have been used for the numerical experiments and present their results. We also state some conclusions and perspectives concerning the numerical solution of partially separable nonlinear problems in many variables.

2. Algorithmic Framework.

2.1. *Partially Separable Systems of Nonlinear Equations.* In the following, we will be concerned with the solution of the equation

$$f(x) = 0, \tag{2}$$

where x is a vector of unknowns belonging to the n-dimensional real vector space, and where $f(x)$ is a vector of the same space that can be computed, at a cost, for any given value of x. We will be interested in the case where $f(x)$ is a nonlinear function of x that is at least once continuously differentiable.

To solve our problem, we will, at a given point x, consider the local linear model

$$f(x + s) = f(x) + Js, \tag{3}$$

for sufficiently small displacement s, where J is a suitable approximation to $J(x)$, the Jacobian matrix of $f(x)$. We can then solve the corresponding linear system, and iterate on this process, which yields an iterative scheme of the type

$$x^{k+1} = x^k - [J^k]^{-1} f(x^k) \qquad (k = 0, 1, 2, \ldots), \tag{4}$$

where x^0 and J^0 are given. Clearly, and as is common in this field, we will, in practice, use a variant of (4) that allows for some damping along the direction d^k defined by the (approximate) solution to

$$J^k d^k = -f(x^k). \tag{5}$$

The interested reader is referred to [2] for more details.

We will say that the function $f(x)$ is partially separable if and only if two conditions are satisfied:

1. $f(x)$ is described as a sum of "element functions", i.e,

$$f(x) = \sum_{i=1}^{m} f_i(x), \tag{6}$$

2. each one of the m element functions $f_i(x)$ has a low-dimensional range and/or domain in R^n.

The term "low-dimensional" means that, in practice, this notion will be of interest when the maximal dimension of these ranges/domains will be much smaller than n, the total number of variables, although the formal definition does not prevent this quantity to be as great as $n - 1$.

It is important to realize that this notion covers most of the problems involving many variables. A typical case is when each $f_i(x)$ only involves a few of the n variables and has an image with only a few of its components being nonzero. This particular structure naturally results in a sparse Jacobian matrix for $f(x)$, but it is interesting to observe that this sparsity is only a consequence of the partially separable structure, and not the other way round. Partial separability is, in essence, a geometric description of the underlying structure of the considered problem, and is invariant with respect to the particular basis chosen, in contrast to sparsity.

One of the important cases where partial separability arises is in the finite element method, where $f(x)$ is decomposed into a sum of functions related to each element of the discretization (see [13]), hence the name "element functions". In this setting, the domain of each of these functions is contained in the subspace spanned by the canonical basis vectors corresponding to the variables occurring in a single element (the "elemental" variables), and its range is also contained in a low-dimensional subspace. We have used the expression "is contained in" on purpose, since, in a fair number of practical instances, these element functions are also invariant with respect to certain translations of their elemental variables, and this reduces even further the dimensionality of their domain. For example, in stress analysis, the value of the internal stress in a particular element is invariant for any translation of the complete element in the three-dimensional space.

Other large partially separable nonlinear problems arise in boundary value computations, and other discretized nonlinear functional equations.

2.2. *Partitioned Updating.* It is now interesting to see that some of the classical methods for computing a (quasi-)Newton approximation J to $J(x)$ can be deduced from particular choices of element functions in the decomposition (6).

1. If one chooses, for any choice of $f_i(x)$, to compute their Jacobians $J_i(x)$ analytically or by finite difference, it is easy to see that, because

$$J(x) = \sum_{i=1}^{m} J_i(x) \tag{7}$$

for all x, the overall matrix will be the analytical Jacobian of $f(x)$, or a finite difference estimation of it. A typical iterative method to solve (2) is then identified with (the discrete) Newton's algorithm.

2. One may also choose to describe $f(x)$ as the sum of canonical basis vectors multiplied by the corresponding component of $f(x)$, namely

$$f(x) = \sum_{i=1}^{n} e_i^T f(x) \cdot e_i, \tag{8}$$

where e_i denotes the ith vector of the canonical basis. For these element functions, we observe immediately that the range of each one of them is of dimension 1 (it is

span[e_i]), and hence the definition of partial separability applies. It is well-known (see [2], for example), that a least-change secant update for this decomposition, i.e., a row-by-row partitioning of the Jacobian, yields the classical Broyden formula

$$J^+ = J + (y - Js)s^T/s^Ts, \tag{9}$$

where we have used the notation $^+$ to denote a quantity related to the next iteration of the algorithmic process (4), and where

$$s = x^+ - x \tag{10}$$

and

$$y = f(x^+) - f(x). \tag{11}$$

3. Specializing the previous choice, when the number of variables n is large, we assume that each element function in (8) depends only on a few variables, those that actually appear in the ith nonlinear equation. This modification now restricts the domain of each element function, in addition to the previous case, which only restricts its range. Classically again, the least-change secant update corresponding to this choice is Schubert's update [10].

A unified local convergence theory based on this observation has been published in [4], and Q-superlinear convergence to an isolated solution x^* is shown for these quasi-Newton methods under standard assumptions.

Why write more about partially separable nonlinear systems, since well-known methods seem to exploit this structure already? Firstly, we must emphasize the geometrical viewpoint that is missing in the more traditional approaches to large problems based on sparsity. One can also see that, for some examples, the row-by-row partitioning of the Jacobian matrix may be rather unnatural. In finite elements again, the Jacobian is described as the sum of the Jacobians of the element functions, which are usually small square matrices of dimension larger than one, and this partitioning, although fitting in our partially separable context, is totally ignored by Broyden's or Schubert's methods. We may therefore prefer to preserve this structural information, and use a quasi-Newton approximation to the Jacobian of each element function, as given in (6). This amounts to setting up a partially separable local linear model of the form

$$f(x + s) = \sum_{i=1}^{m} \{ f_i(x) + J_i s \} \tag{12}$$

instead of (3), and modifying the equation (5) to read

$$\left[\sum_{i=1}^{m} J_i^k \right] \cdot d^k = - \sum_{i=1}^{m} f_i(x). \tag{13}$$

This approach has been called "partitioned updating" in [3], where quasi-Newton formulae were used in the optimization context; we will retain this terminology here, and use the Broyden update (9) for improving J_i^k, our current approximation to $J_i(x^k)$.

The resulting algorithm may then be broadly described as follows, if we assume that x^0 and $\{ J_i^0 \}_{i=1}^m$ are given.

1. Compute $f_i(x^0)$ for $i = 1, \ldots, m$, and hence $f(x^0)$. Set $k = 0$.

2. Compute the search direction by solving (approximately) the linear system (13).

3. Compute x^{k+1} and $f(x^{k+1})$ by using a line search from x^k along d^k, i.e., compute a scalar t^k such that

$$x^{k+1} = x^k + t^k \cdot d^k \tag{14}$$

and the norm of $f(x^{k+1})$ is sufficiently smaller than that of $f(x^k)$.

4. Test for termination.

5. Update the element Jacobian approximations by applying Broyden's formula (9) to each one of them, with s and y being defined, for the ith element, by

$$s = x^{k+1} - x^k \tag{15}$$

and

$$y_i = f_i(x^{k+1}) - f_i(x^k). \tag{16}$$

6. Set k to $k + 1$, and return to step 2.

We again refer the interested reader to [2] for further details concerning the line search, termination criteria and other points not explicitly discussed above.

2.3. *Storing Approximate Element Jacobians.* We now investigate the consequences of partial separability on the storage scheme that is used for the element Jacobian approximations J_i.

One very common reason for the range and domain of an element function to be of low dimension is that only a few of the components of their vectors are nonzero. In other words, a given element function does not contribute to all n equations and does not involve all n variables. In this frequent case, it is clear that the Jacobian J_i of this element will only contain a few nonzero rows and columns, and that the superposition of those small (possibly) dense submatrices induces, in J, a well-defined sparsity pattern. Instead of keeping track of the total sparsity pattern of J, as is usually done, we will, in our framework, keep track, for each element, of the respective variables (the elemental variables) and respective equations. This can be done conveniently, and this kind of reduction in dimension can therefore be handled by the data structure.

It may also happen that the domain and/or range of an element Jacobian is of low dimension, but is spanned by vectors other than those of the canonical basis. In this case, the information can no longer be represented by lists of components, but some explicit reduction technique is needed. We propose to consider the representation

$$J_i = U_i T_i W_i \qquad (i = 1, \ldots, m), \tag{17}$$

where the columns of U_i span the range of J_i and the rows of W_i its domain, for all values of the variables. (In this equation, we assume that J_i is a dense matrix, i.e., that the possible reduction explained in the previous paragraph has already been carried out. This assumption will also be made further below.) T_i is, in fact, the part of the Jacobian that we really wish to approximate. It is often the case that the matrices U_i and W_i are identical for a large number of elements, except for the assignment of the elemental variables; advantage can be taken of this structure to store only the different ones, or to ask the user to provide routines for their manipulations, as is done in [6]. Then only T_i is stored and estimated by the numerical algorithm.

This can be best illustrated by a small example. Consider the nonlinear system of equations where the ith element function is given by

$$
\begin{aligned}
[f_i(x)]_{2i-1} &= 3(x_{2i-1} - x_{2i+1})^3 + 2x_{2i} - 5 \\
&\quad + \sin(x_{2i-1} - x_{2i} - x_{2i+1})\sin(x_{2i-1} + x_{2i} - x_{2i+1}), \\
[f_i(x)]_{2i} &= -(x_{2i-1} - x_{2i+1})\exp[x_{2i-1} - x_{2i} - x_{2i+1}] + 4x_{2i} - 3, \\
[f_i(x)]_{2i+1} &= -2[f_i(x)]_{2i-1},
\end{aligned} \tag{18}
$$

for $i = 1$ up to m, and where the other components of $f_i(x)$ are zero. Clearly, only the components $2i - 1$, $2i$ and $2i + 1$ of x and $f(x)$ play a role in this element, so its domain and range are restricted to the subspace spanned by e_{2i-1}, e_{2i} and e_{2i+1}. Furthermore, it can easily be seen that, in fact,

$$
\text{range}[f_i(x)] = \text{span}[e_{2i}, e_{2i-1}, -2e_{2i+1}] \tag{19}
$$

while

$$
\text{domain}[f_i(x)] = \text{span}[e_{2i}, e_{2i-1} - e_{2i+1}] \tag{20}
$$

for all x. Hence, in this case the lists of elemental variables and equations will both contain $2i - 1$, $2i$, $2i + 1$, and the matrices U_i and W_i will be given by the expressions

$$
U_i^T = \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \end{pmatrix} \tag{21}
$$

and

$$
W_i = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{22}
$$

It is interesting to observe that, since structural information is stored twice for the same entries of the overall Jacobian at positions where two or more element Jacobians overlap, pointer storage needed for the proposed scheme is usually slightly larger than in more classical approaches to sparsity. On the other hand, if the matrices U_i and/or W_i are not reduced to the identity and do not vary too much with i, storage for real numbers can be less for the new technique than for the traditional storage procedure.

We now wish to rewrite the equations from which we estimate J_i in terms of the matrix T_i. Consider the quasi-Newton or secant equation first. We obtain

$$
y_i = J_i s = U_i T_i W_i s_i, \tag{23}
$$

where both y_i defined in (16) and s have been restricted to the subspaces spanned by the elemental rows and variables of the ith element (hence the subscript i in s_i). This yields, as a new secant equation,

$$
T_i(W_i s_i) = U_i^I y_i, \tag{24}
$$

where the superscript I denotes the generalized inverse (remember that y_i is in the column space of U_i!).

Similarly, assume that one wishes to estimate the jth column of T_i by differences. We want to use the relation

$$
T_i e_j = \text{some difference in } f_i(\cdot). \tag{25}
$$

The estimation process can now be rewritten as:

1. compute

$$h_j = aW_i^I e_j \tag{26}$$

for some steplength a,

2. evaluate the quantity

$$z_j = f_i(x + h_j) - f_i(x), \tag{27}$$

3. estimate the jth column of T_i by setting the difference in (25) equal to

$$\left(U_i^I z_j\right)/a. \tag{28}$$

Observe also here that z_j is always contained in the column space of U_i, so that the computation of equations like (28) is quite simple.

3. Some Numerical Tests.

3.1. *More Details About the Tested Algorithms.* We now present some numerical experiments that were performed with methods of the type just described. Before presenting the test problems, however, it is helpful to give some more details about the numerical procedures that were actually implemented.

Earlier on, we mentioned that the linear system (13) need not be solved to full accuracy. In practice, we have used an iterative linear least squares method (LSQR, by Paige and Saunders [9]) until the residual was reduced by a sufficient amount. This is only one possible choice that has been made for simplicity, and there are clearly a number of interesting alternatives, such as frontal methods (especially in the finite element context), SOR or more general sparse solvers. However, it was not our purpose to test efficiency of the linear algebra part of the nonlinear algorithm, but, instead, to compare different ways of handling the nonlinearity itself. Hence, these more efficient procedures for solving the linear system were not implemented.

At the starting point, and following [2] again, we choose to evaluate the initial Jacobian (i.e., the Jacobians of the element functions at the starting point) by finite differences. This strategy was also used when the step computed by solving (13) was not a descent direction for the l_2-norm of the residual, the merit function that was used throughout the calculation.

The line search was implemented using safeguarded cubic/quadratic interpolation in a very classical fashion, but care was taken not to reevaluate an element function whose elemental variables were not modified since its last evaluation. This "special" feature, together with the difference estimation of the Jacobians element-by-element, account for the fractional number of function evaluations that appear in the numerical results presented below.

Different approximation schemes for the Jacobian(s) are tested below. First, a method is examined that uses finite differences to estimate the element Jacobians. This amounts to a rather traditional discrete Newton algorithm. However, the element-by-element estimation and the particular storage scheme we use for the elements will allow some further refinements: since the dimension of T_i can be smaller than that of J_i, fewer differences are needed to estimate it. The gains resulting from this observation will be investigated by comparing an algorithm that estimates J_i by finite differences and one that only estimates T_i by the same method.

Amongst the partially separable quasi-Newton updates, we will restrict ourselves to the application of Broyden's formula (9) to different partitionings: the classical Broyden update for the row-by-row description of the nonlinear system, Schubert's update when advantage is taken of the fact that not all variables occur in all equations, and the general partitioned updating based on the natural decomposition of the problem into elements.

It is worth mentioning, at this point, that the updating formula (9) is scale invariant on the range of $f_i(x)$, but not on its domain, in contrast to quasi-Newton updates for minimization, like the BFGS formula. Another formula, also due to Broyden [1] and usually called the "bad" Broyden formula, provides scale invariance on the domain, but not on the range. Testing various partitionings of the Jacobians together with this update has been left for further work. Other combinations are still possible, like using Schubert's formula together with nontrivial W_i for each equation, but the results available for this algorithm are not sufficiently complete to be reported here.

We may also try to predict the behavior of the two algorithms which takes information in U_i and W_i into account (discrete Newton and partitioned Broyden), when U_i and/or W_i are wrongly defined. A typical case may be when one of these matrices is, in fact, different from the identity, but when this structure is ignored by the user and the identity is used instead. We anticipate that the discrete Newton method would be substantially better than partitioned Broyden, because it estimates the true Jacobian, and therefore the structure of its domain and range as well, while the quasi-Newton approximation using (9) relies on an external specification of the domain. If this specification, i.e., W_i, is wrong or incomplete, one may expect the quasi-Newton approximations to be of poor quality, and hence the resulting convergence to be significantly impaired. Using the identity as U_i when the range of $J_i(x)$ has a dimension lower than its number of elemental variables should be, according to the previous paragraph, is of less importance.

Finally, the case where s_i is zero for some element i has been dealt with by simply skipping updating J_i, as no information on this element can be gained from such a step.

3.2. *The Test Problems.* The nonlinear systems that were used for the numerical tests are now described. We first used three classical problems from the Argonne test set [8]: Broyden tridiagonal and banded systems, and the discretized boundary value problem. We also used the gradient of the linear minimum surface problem described in [3] and [6]. Finally, we included five new problems with variable dimension, two of them being of the finite element type.

3.2.1. *A trigonometric system.* The element functions are defined by

$$[f_i(x)]_{5i-j} = 5 - i\left[1 - \cos(x_{5i-j})\right] - \sin(x_{5i-j}) - \sum_{k=5i-4}^{5i} \cos(x_k) \tag{29}$$

for $j = 0$ to 4, the other components of $f_i(x)$ being zero. The starting point is defined by

$$x_i = 1/n \qquad (i = 1, \ldots, n). \tag{30}$$

3.2.2. *A trigonometric-exponential problem* [trigexp 1]. The element functions are defined by

$$
\begin{aligned}
\left[f_i(x)\right]_i &= 3x_i^3 + 2x_{i+1} - 5 + \sin(x_i - x_{i+1})\sin(x_i + x_{i+1}),\\
\left[f_i(x)\right]_{i+1} &= -x_i \exp[x_i - x_{i+1}] + 4x_{i+1} - 3,
\end{aligned} \tag{31}
$$

the other components of $f_i(x)$ being zero. The starting point is at the origin.

3.2.3. *A variant of the preceding system* [trigexp 2]. The element function components are described in (18) and the starting point is given by

$$x_i = 1 \qquad (i = 1, \ldots, n). \tag{32}$$

3.2.4. *Two finite element type problems*. The last two problems are the bilinear finite element discretization of the heat radiation equation on the unit square. The basic equation is

$$\left[k_x\left(T_x', T_y'\right)T_x'\right]_x' + \left[k_y\left(T_x', T_y'\right)T_y'\right]_y' + Q = 0, \tag{33}$$

where the gradient of the temperature has the components T_x' and T_y', and where $[\cdot]_x'$ denotes the partial derivative of the quantity inside the bracket with respect to the variable x. Dirichlet boundary conditions were chosen, inasmuch as the temperature T was constrained to be zero on the boundary. In the tests, Q was chosen to represent point sources/sinks at the points

$$(0.9, 0.1), \quad (0.1, 0.3), \quad (0.5, 0.5), \quad (0.1, 0.9) \quad \text{and} \quad (0.9, 0.9)$$

with respective values

$$-1.0, \quad -0.5, \quad 1.83, \quad -0.6 \quad \text{and} \quad 0.27.$$

In the first problem (nlheat1), the conductivities were chosen according to the relations

$$
\begin{aligned}
k_x\left(T_x', T_y'\right) &= \max\left(0, 1 - 69.444445\left[T_x'\right]^2\right),\\
k_y\left(T_x', T_y'\right) &= \max\left(0, 1 - 69.444445\left[T_y'\right]^2\right),
\end{aligned} \tag{34}
$$

while in the second of these tests they were both set to

$$k_x\left(T_x', T_y'\right) = k_y\left(T_x', T_y'\right) = 0.01 + 100 \exp\left[-0.1\|T'\|\right], \tag{35}$$

where $\|T'\|$ is the Euclidean norm of the gradient of the temperature. The starting point is at the origin for both problems.

It can easily be seen that the form of (34) implies that each element function has a range and a domain of dimension two, even though there are four variables (the temperature values at the four corners of a square element); this is not the case when (35) is used.

3.2.5. *Summary of the tests*. We now summarize the characteristics of the tests that were run. In the following, we indicate by the symbol # the problems where the natural decomposition into elements is the decomposition into the equations of the nonlinear system. Since we assume that all variables do not necessarily appear in all equations, the partitioning corresponding to the partially separable approach is identical to the partitioning corresponding to Schubert's update. We also mark the problems with the symbol \$ when the matrix U_i and/or W_i is not reduced to the identity for at least one (usually most) element(s) of the decomposition.

Nonlinear heat conduction 1	(, $) :	$n =$	100, 169, 225, 324, 400, 484.
Nonlinear heat conduction 2	(,) :	$n =$	100, 169, 225, 324, 400, 484, 576.
Minimum surface	(, $) :	$n =$	25, 49, 64, 81, 100, 121, 169, 196, 225, 289, 324, 400, 484.
Broyden tridiagonal	(#, $) :	$n =$	50, 100, 250, 500, 750, 1000.
Broyden banded	(#,) :	$n =$	50, 100, 250, 500, 750, 1000.
Trigonometric	(,) :	$n =$	51, 101, 251.
Trigexp 1	(,) :	$n =$	100, 250, 500, 750, 1000.
Trigexp 2	(, $) :	$n =$	51, 75, 101, 125.
Discretized boundary value	(#, $) :	$n =$	50, 75, 100, 120.

Using these 54 problems, 223 runs were made with the algorithms described above. All calculations were done in double precision on a DEC2060 of the Facultés Notre-Dame de la Paix in Namur (Belgium), using the Fortran 77 compiler without optimization. This machine has a wordlength of 36 bits and uses 63 bits to hold the mantissa of a double-precision number. All methods were stopped when the relative max norm of the residual (see [2] for details) was below 10^{-7}.

3.3. *Results and Discussion.*

3.3.1. *Using the structure of the range and/or domain.* We first examine the effect, in the numerical computations, of using the structure of the range and/or domain of the element Jacobians $J_i(\cdot)$, as described in (17). For this purpose, we selected, in the test problems mentioned above, those where advantage could be taken of this structure (flagged with $). We compare the two methods that can deal with this situation (discrete Newton and partitioned Broyden). A summary of their performance can be found in Table 1. In this table, we use the following abbreviations:

- nprob is the number of problems considered in this set,
- av.dim. is the average dimension of the considered problems,
- str is set to "yes" when the structure of the range and/or domain was correctly incorporated in the calculations, and to "no" when it was ignored,
- fls is the number of times where the considered algorithm failed to satisfy the stopping criterion,
- it/prob is the average number of iterations required by the considered method
- nfev/prob is the average number of function evaluations required by the considered method.

The names of the methods are self-explanatory. The problems sets were chosen to represent "small" problems (n at most 100), and "larger" ones (n above 100). Here

TABLE 1

Performance of the methods when using the range/domain structure

nprob	av.dim.	method	str	fls	it/prob	nfev/prob
13	70.77	D. Newton	yes	0	5.80	21.68
			no	0	5.80	28.88
		Part. Br.	yes	0	14.23	21.53
			no	0	30.54	42.77
18	320.77	D. Newton	yes	0	8.72	33.13
			no	0	9.06	48.48
		Part. Br.	yes	0	25.61	35.87
			no	1	49.65	64.93

and below, the number of function evaluations is calculated as the total number of element function evaluations divided by the total number of elements. Therefore, it represents the number of times the complete right-hand side has been computed.

Inspection of these results clearly shows the advantage of taking the structure into account, for both methods. One can also observe the behavior predicted above: ignoring the structure causes a much worse degradation in performance (and, to some extend, reliability) of the partitioned Broyden method than for the discrete Newton algorithm. This remark is most apparent when one considers the iteration numbers needed for solution. Hence, we can conclude that analyzing this type of structure, and using it when available, can be of importance when solving sets of nonlinear equations.

3.3.2. *Comparing the methods tested.* One of the major issues in running the numerical experiments was to compare, if possible, the relative efficiency and reliability of the different methods discussed in the beginning of this paper. The data for such a comparison appear in Table 2. Conventions in this table (and the following ones) are similar to those used in Table 1. Because of the conclusions of the previous paragraph, the discrete Newton and partitioned Broyden methods were used with correct information on the ranges and domains of the element Jacobians.

Although this constitutes only a rather limited set of experiments, and caution must be exercised before extrapolating any conclusion to a more general framework, one feels justified to make the following observations when analyzing this data.

TABLE 2

Comparison between the methods

nprob	av.dim.	method	str	fls	it/prob	nfev/prob
18	73.39	D. Newton	yes	0	5.55	24.93
		Part. Br.	yes	0	13.78	20.31
		Schubert	no	0	22.66	37.53
		Broyden	no	5	26.23	117.46
36	393.36	D. Newton	yes	0	6.92	31.35
		Part. Br.	yes	0	18.39	25.57
		Schubert	no	0	43.67	73.76

TABLE 3
Results for the trigexp 1 *test problem*

n	D. Newton	Part. Br.	Schubert
100	7/25.0	13/18.0	16/23.0
250	7/25.0	13/17.6	16/21.6
500	7/25.0	13/16.1	16/19.5
1000	7/25.0	13/13.7	16/22.0

1. Schubert's method is never better than discrete Newton.

2. Schubert's method is not better, on average, than partitioned Broyden. Here one must remember that these two methods may coincide when the natural partitioning of the nonlinear system is row by row. Therefore, a finer analysis is needed to justify this remark, which will be given below using problems for which the two methods are different.

3. Even for small problems, the full Broyden method is usually outperformed by other techniques that can use more of the problem structure, if any.

4. Partitioned Broyden updating is clearly competitive compared to the discrete Newton approach. The choice between these two algorithms may depend, in practical problems, upon the relative costs of the linear algebra involved and that of function evaluations. If this latter cost is high and the problem is partially separable, some consideration should be given to the partitioned Broyden algorithm.

5. The above remarks are more relevant when the problem dimension increases.

We nevertheless feel that these tentative conclusions should be tempered by a closer examination of some particular instances in our test set. Two of these instances are therefore more detailed than in the above tables. We first present the results obtained for the "trigexp1" problem (see Table 3).

In Table 3, the number appearing before the slash is the iteration number and that appearing after the slash is the number of function evaluations needed to obtain the solution to the desired accuracy. The classical Broyden method was also tested for $n = 100$, and needed 53 iterations and 202 function evaluations to obtain the solution. The constancy of the number of iterations for the three first methods, as the dimension of the problem increases, is quite remarkable here. Partitioned Broyden has the best performance in terms of function evaluations, as is the case on average for the problems tested.

However, this is not the case for the minimum surface test problem, whose results appear in Table 4. The pure Broyden update was also used for the problems of dimension 25, 49, 64, 81 and 100, and needed 35, 30, 33, and 34 iterations respectively, as well as 64, 90, 118, 162 and 178 function evaluations. In this example, it is therefore advantageous to use the partially separable structure of the problem, even for small dimensions.

We observe also the appreciable degradation due to ignoring the structure of the range and domain of the element Jacobians, especially when using the partitioned Broyden update.

3.3.3. *Partitioned Broyden vs. Schubert when they are different.* As mentioned above, we end this comparison by a closer look at the respective performance of the partitioned Broyden and Schubert approach, when the structure of the problem does

TABLE 4
Results for the minimum surface test problem

n	D. Newton		Part. Br.		Schubert
	str	no str	str	no str	
25	8/26.8	8/ 31.3	18/30.5	25/ 39.5	20/ 35.5
49	8/28.0	8/ 36.0	27/35.9	56/ 74.8	35/ 68.8
64	10/36.1	10/ 49.7	29/46.8	33/ 52.9	31/ 53.1
81	10/37.3	10/ 49.7	32/42.9	77/ 89.1	56/ 82.0
100	10/37.5	10/ 50.8	35/47.9	81/103.2	25/ 63.3
121	9/33.6	9/ 46.4	30/50.9	87/120.5	52/ 84.1
169	12/44.6	12/ 63.7	44/57.0	93/114.0	135/219.8
196	11/41.7	11/ 58.9	41/55.0	90/118.8	75/114.2
225	16/60.7	14/ 75.7	38/53.0	52/ 77.9	71/128.0
289	19/71.7	18/ 98.8	61/80.0	150/176.5	102/197.0
324	18/65.7	18/ 95.3	59/76.0	158/181.5	164/299.7
400	14/53.8	22/117.6	60/81.0	68/ 93.2	179/339.0
484	19/71.8	19/104.6	64/92.0	--fail--	397/619.5

not make them coincide. We therefore only consider the test problems where the natural partitioning is not row-by-row, or, if it is, where the true domain of some of the element Jacobians is smaller than their number of elemental variables. The results of this comparison are presented in Table 5.

Two observations arise from the examination of Table 5:

1. the performance of partitioned Broyden with the structure of ranges/domains correctly taken into account is slightly better than in Table 2, where all problems were considered, while that of Schubert's update is slightly worse;

2. on average, partitioned Broyden without taking the range/domain structure into account is a little bit better than Schubert's method. We nevertheless feel that this should be taken with some caution, since the detailed results show the matter to be very much problem dependent.

We end this comparison by comparing the storage requirements of partitioned Broyden and Schubert's methods. As has been said above, we expect Schubert's method to use less real storage than partitioned Broyden, because this last approach must store each element Jacobian completely, even for the components that overlap others. Using the structure of the range and domain also affects the storage for the latter method, because, when U_i and/or W_i are not reduced to the identity, the size of T_i is then smaller than that of J_i. These storage requirements are compared in Table 6. The numbers appearing in this table are the average number of reals to be stored for the complete algorithm, including workspace. However, the differences

TABLE 5
Schubert vs. Partitioned Broyden when different

nprob	av.dim.	method	str	fls	it/prob	nfev/prob
16	73.19	Part. Br.	yes	0	13.37	19.75
			no	0	26.37	37.03
		Schubert	no	0	23.37	39.13
30	359.23	Part. Br.	yes	0	19.60	27.27
			no	1	33.10	44.02
		Schubert	no	0	49.93	84.73

TABLE 6
Storage needs for Schubert and partitioned Broyden

nprob	av.dim.	Part. Br.		Schubert
		str	no str	
16	73.19	1060	1413	988
30	359.23	5985	7381	4936

reflect only the differences in the storage scheme of the approximate Jacobians, because the remaining space is independent of the method considered. These numbers do indeed confirm our expectations.

3.3.4. *General comments and perspectives.* Although the tests we reported upon can certainly not be considered as final or conclusive, we think that some general comments can be made concerning the solution of partially separable systems of nonlinear equations.

Aside from the interest of the theoretical concept of partial separability and partitioned updating techniques, it seems clear that the partitioned Broyden approach has to be considered seriously when dealing with such problems, especially if one has good knowledge of the geometry of the ranges and domains of the element Jacobians.

In this context, we also believe that our results show the importance of using the properties of these subspaces when using a more traditional discrete Newton approach.

In view of the above results, one also feels justified to question the efficiency of Schubert's algorithm for the class of problems that we have considered, except maybe when storage is extremely scarce.

Finally, we must reassert the need for further testing of the issues we raised in this paper. We believe that tests made by a single person and on a single set of test problems are not sufficient to draw firm conclusions as to the relative merits of numerical algorithms. Open questions also include a proof of global convergence for the structured quasi-Newton algorithms that we have described.

Department of Mathematics
Facultés Universitaires de Namur
Namur, Belgium

1. C. G. BROYDEN, "A class of methods for solving nonlinear simultaneous equations," *Math. Comp.*, v. 19, 1965, pp. 577–593.

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

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

4. A. GRIEWANK & PH. L. TOINT, "Local convergence analysis for partitioned quasi-Newton updates," *Numer. Math.*, v. 39, 1982, pp. 429–448.

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

6. A. GRIEWANK & PH. L. TOINT, "Numerical experiments with partially separable optimization problems," in *Numerical Analysis, Proceedings Dundee* 1983 (D. F. Griffiths, ed.), Lecture Notes in Math., vol. 1066, Springer-Verlag, Berlin, 1984, pp. 203–220.

7. E. MARWIL, *Exploiting Sparsity in Newton-Like Methods*, Ph.D. thesis, Cornell University, Ithaca, New York, 1978.

8. J. J. MORÉ, B. S. GARBOW & K. E. HILLSTROM, "Testing unconstrained optimization software," *ACM Trans. Math. Software*, v. 7(1), 1981, pp. 17–41.

9. C. C. PAIGE & M. A. SAUNDERS, "LSQR: An algorithm for sparse linear equations and sparse least squares," *ACM. Trans. Math. Software*, v. 8(1), 1982, pp. 43–71.

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

11. D. F. SHANNO, "On variable metric methods for sparse Hessians," *Math. Comp.*, v. 34, 1980, pp. 499–514.

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

13. O. C. ZIENKIEWICZ, *The Finite Element Method*, McGraw-Hill, London, 1977.