An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project

A.R. Conn* N.I.M. Gould† Ph.L. Toint‡

December 7, 1989

Abstract

This paper presents the authors' personnal views on two fundamental aspects amongst the recent developments in the growing field of large scale nonlinear mathematical programming. Important concepts for the description of problem structure are discussed in detail. A systematic approach to software development for this class of problems is also presented. The approach incorporates both suitable numerical algorithms and user oriented standard format for problem specification in a modular and coherent system.

Keywords : Large scale problems, nonlinear programming, problem structure, software development.

1 Introduction

The field of numerical optimization, one of the most challenging areas of numerical analysis, has recorded an impressive growth in the past few years. Apart from widely known developments in linear programming (see [23], for instance), this growth has also been fueled by a steadily increasing interest in nonlinear problems involving a large number of variables. However, the importance of the new results and the power of the new algorithms developed are probably not fully appreciated by the larger community of numerical analysts and, even more importantly, by the community of potential users of large scale nonlinear optimization methods. Recent publications devoted specifically to large scale problems include [18], [9], [10], and [2].

It is a widely held view that only rather small problems can be handled by the techniques available. Specifying nonlinear optimization problems in more than 20 variables, say, is still considered by many as a risky modelling approach, mostly because the problem's solution is likely to be impossible with the existing algorithms. It is true that this view was justified ten years ago. The present situation is however quite different and it is the purpose of this paper to stress this change and to present some of the concepts that resulted in this progress.

*Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.
The research of this author was supported in part by NSERC grant A8639.
†Computer Science and Systems Division, Harwell Laboratory, Oxfordshire, England.
‡Department of Mathematics, Facultés Universitaires ND de la Paix, Namur, Belgium

These concepts will be presented here from the authors' very personnal (and maybe biased) point of view. In particular, no attempt is made to discuss every concept and significant recent development in the area, rather the exposition will focus on two topics that are considered to be fundamental by the authors. We will also outline our present and forthcoming research in this area, both from the algorithmic and software perspective.

The first part of this paper is devoted to an introduction to the classes of partially separable and group partially separable functions. Understanding and exploiting these concepts is, in our opinion, central to the development of efficient and reliable methods for large scale problems, much in the same way that sparsity is a key to large scale numerical linear algebra. This introduction is contained in Section 2.

The second part of the paper discusses the LANCELOT software project, whose purpose is to produce a self contained system for large scale nonlinear optimization. The discussion emphasizes the main objectives of the system, with a brief discussion of some data input and implementation issues.

2 The structure of nonlinear problems

2.1 A tutorial on partial separability

Very few numerical analysts would contest today the crucial role played by the various techniques for exploiting the structure of a problem in the development of practical computational methods for large scale problems. Sparsity in large systems of linear equations, domain decomposition in the numerical solution of partial differential equations and structure in the interpolation equations for function approximation are probably the three examples that come first to mind. The situation is entirely similar in large scale nonlinear optimization, where exploiting the structure of large problems is the only reasonable way to tackle their solution.

Introduced in 1981, by Andreas Griewank and the third author in [15], the notion of a partially separable function can be viewed as a way to describe the structure of a nonlinear function in terms of an underlying geometry (i.e. subspaces and their relations).

It is the authors' belief that this notion can be very helpful, not only to algorithm designers, but also to potential users of these algorithms. Indeed, for a method to exploit structure, it is very beneficial that the user describes this structure in a way coherent with the implementation. This in turn supposes that the user should be at least moderately familiar with the principles of the description used.

The concept of partial separability will be introduced and analysed by a simple yet meaningful example: the discretised minimum surface problem over the unit square. We will not be interested in this problem as such, but rather in showing that one of its discretised formulations exhibits the type of structure that we want to exploit.

The minimum surface variational problem is well known, and consists in finding the surface of minimum area that interpolates a given continuous function on the boundary of the unit square. The unit square itself is discretised uniformly into m^2 smaller squares, as shown in Figure 1.

The variables of the problem will be chosen as the "height" of the unknown surface above the

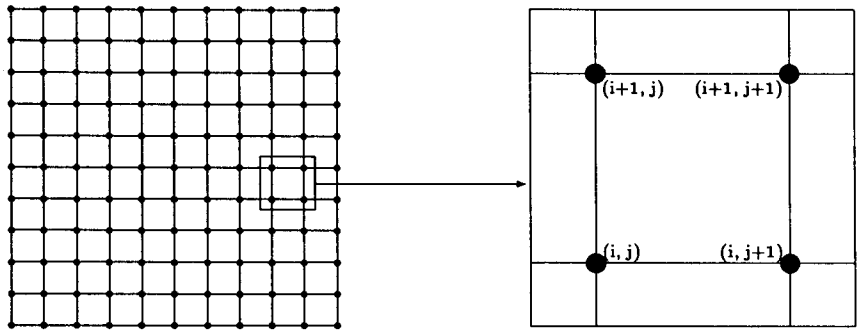

Figure 1: Discretisation of the unit square

$(m+1)^2$ vertices of the m^2 smaller squares. The area of the surface above the (i,j)-th discretised square is then approximated by the formula

$$s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}) = \frac{1}{m^2}\sqrt{1 + \frac{m^2}{2}\left[(x_{i,j} - x_{i+1,j+1})^2 + (x_{i,j+1} - x_{i+1,j})^2\right]}, \qquad (1)$$

where the squares and variables are indexed as shown in Figure 1. The precise justification of this formula does not matter here, but we concentrate instead on its form.

The variables in the sets

$$\{x_{1,j}\}_{j=1}^{m+1}, \quad \{x_{i,m+1}\}_{i=1}^{m+1}, \quad \{x_{m+1,j}\}_{j=1}^{m+1}, \text{ and } \{x_{i,1}\}_{i=1}^{m+1}, \qquad (2)$$

correspond to the boundary conditions and are assigned given values, while the others are left free. The complete problem is then to minimise the objective function

$$f(x) = \sum_{i,j=1}^{m} s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}) \qquad (3)$$

over all the free variables[1].

A first analysis quickly shows that, using a row-wise ordering of the variables, the Hessian $\nabla^2 f(x)$ has a block-tridiagonal sparsity pattern with tridiagonal blocks. We note that storing this matrix, which is typically required in one form or the other in most efficient algorithms, therefore requires storing $O[5(m+1)^2]$ real numbers. More importantly, this sparsity structure may be discovered by considering the Hessian of $s_{i,j}$ as a function of $x_{i,j}$, $x_{i+1,j}$, $x_{i,j+1}$ and $x_{i+1,j+1}$, and then by assigning each of the rows and columns of this dense 4×4 matrix to the relevant row and column of the larger $\nabla^2 f(x)$. If we consider each $s_{i,j}$ as a function of the complete set of $(m+1)^2$ variables, denoted by $s_{i,j}(x)$, we see that it satisfies the important property that

$$s_{i,j}(x) = s_{i,j}(x + w) \qquad (4)$$

[1]Various obstacle problems can also be obtained by specifying suitable bounds on the variables.

for all vectors w in the *invariant subspace*

$$N^s_{i,j} = \{x \in \mathbf{R}^{(m+1)^2} | w_{i,j} = w_{i,j+1} = w_{i+1,j} = w_{i+1,j+1} = 0\}. \tag{5}$$

Each $s_{i,j}$ is therefore invariant with respect to all translations corresponding to vectors of $N^s_{i,j}$. From this invariance, it is again easy to deduce that $\nabla^2 s_{i,j}(x)$ is a (very) sparse matrix, but we stress the point that (4)–(5) is in fact the most important observation when analysing the structure of the second derivatives of our model problem. *We may then interpret the sparsity pattern of this last matrix as a* consequence *of the problem structure: sparsity is easily derived from the structure, but the reverse is not true.*

If we now examine the function $s_{i,j}$ in more detail, we easily see that, instead of being a function of the four variables $x_{i,j}$, $x_{i+1,j}$, $x_{i,j+1}$ and $x_{i+1,j+1}$, it is, in fact, a function of the two *internal variables*

$$u_{i,j} \stackrel{\text{def}}{=} x_{i,j} - x_{i+1,j+1} \text{ and } v_{i,j} = x_{i,j+1} - x_{i+1,j}. \tag{6}$$

If we write the simple linear transformation

$$\begin{pmatrix} u_{i,j} \\ v_{i,j} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_{i,j} \\ x_{i,j+1} \\ x_{i+1,j} \\ x_{i+1,j+1} \end{pmatrix}, \tag{7}$$

we can then reformulate $s_{i,j}$ in terms of these internal variables as

$$s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}) \stackrel{\text{def}}{=} \hat{s}_{i,j}(u_{i,j}, v_{i,j}), \tag{8}$$

where the new function $\hat{s}_{i,j}(u_{i,j}, v_{i,j})$ is easily derived from (1) and is given by

$$\hat{s}_{i,j}(u_{i,j}, v_{i,j}) = \frac{1}{m^2}\sqrt{1 + \frac{m^2}{2}(u^2_{i,j} + v^2_{i,j})}. \tag{9}$$

Computing now the gradient and Hessian of $\hat{s}_{i,j}$ with respect to its two arguments, we obtain that

$$W^T \nabla \hat{s}_{i,j}(u_{i,j}, v_{i,j}) = \nabla s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}), \tag{10}$$

where the matrix

$$W = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}, \tag{11}$$

and also that

$$W^T \nabla^2 \hat{s}_{i,j}(u_{i,j}, v_{i,j}) W = \nabla^2 s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}), \tag{12}$$

where $s_{i,j}$ is again considered as a function of four variables. But the Hessian of $\hat{s}_{i,j}$ is now a 2×2 symmetric matrix, while that of $s_{i,j}$ is 4×4. Furthermore, we did not index the matrix W, because the same linear transformation holds for all values of i and j. Storing the second derivatives of our problem therefore requires storing W once, plus storing the m^2 Hessians of the functions $\hat{s}_{i,j}$, which amounts to $O[3(m+1)^2]$ real numbers. This is a substantial reduction compared to the $O[5(m+1)^2]$ required by the more classical "sparsity oriented" approach.

In order to abstract from these observations, we simply note that (4) holds not only for the vectors w in $N^s_{i,j}$, but for all w in

$$N_{i,j} = N^s_{i,j} + \{w \in \mathbf{R}^{(m+1)^2} | w_{i,j} = w_{i+1,j+1} \text{ and } w_{i,j+1} = w_{i+1,j}\}. \tag{13}$$

We can then use all the above remarks to define *partially separable functions* as follows.

We say that $f(x)$ is partially separable if and only if

1. it can be written as a sum of *element functions*, that is

$$f(x) = \sum_{i=1}^{p} f_i(x), \tag{14}$$

2. each of these element functions has a nontrivial *invariant subspace*, that is if, for each $i \in \{1, \ldots p\}$, there exists a subspace $N_i \neq \{0\}$ such that, for every $x \in \mathbf{R}^n$ and $w \in N_i$, we have that

$$f_i(x) = f_i(x + w). \tag{15}$$

This definition is not the most general one (see, for example [15]), but has the advantage of being rather intuitive. As in the minimum surface example above, we will be mostly interested in the case where *the dimension of the invariant subspaces N_i is large compared to the total dimension of the problem.* We may then set up a linear transformation for each element, that transforms the *problem variables* $\{x_i\}_{i=1}^n$ into *internal variables* for the element, $\{u_i\}_{i=1}^{n_i}$ say, as expressed by the relation

$$u = W_i x. \tag{16}$$

In our example, this transformation matrix is obtained by composing the four variables at the corners of the considered discretised square with (11).

Once the transformation from problem variables to internal ones is defined, it is only necessary to store, compute and/or update the derivatives

$$\nabla \hat{f}_i(u) \text{ and } \nabla^2 \hat{f}_i(u), \tag{17}$$

whose dimensions are small.

Moreover, the situation arising in our model example is very common. The transformation can be separated into the product of two distinct operators: the first selects a (usually small) number of problem variables that explicitly appear in a given element, and the second defines a further linear transformation of these selected variables yielding the internal ones. We note that the first of these operators need not to be expressed in terms of a matrix, but merely in terms of a list of problem variables associated with the considered element. This saves a substantial amount of matrix manipulation. As in our example, it also frequently happens that the second of these transformations, that is from the selected problem variables to the internal ones, is independent of the element under consideration. Otherwise, frequently there are only a few different such transformations[2], which again saves storage and computation.

[2] In a finite element application, for instance, there are as many transformations as distinct element types in the problem.

Since the invariant subspaces are not necessarily spanned by vectors of the canonical basis (as in our example), we see that the notion of partial separability extends that of sparsity.

More formally, we may state the following result.

Theorem 1 *Every twice continuously differentiable function from* $\mathbf{R}^n$ *into* $\mathbf{R}$ *having a sparse Hessian matrix is partially separable.*

The reader is referred to [15, section 2] for a more detailed discussion of this basic property.

Finally, it may be worthwhile to note that separable functions, that is functions of the form

$$f(x) = \sum_{i=1}^{p} f_i(x_i), \tag{18}$$

are clearly a very restricted case of partially separable functions. Because the variables x_i must all be different in (18), we prefer to call these functions *totally separable.*

2.2 Group partial separability

A significant proportion of practical large optimization problems exhibit another very important structure: the assignment of sets of element functions to *groups.* The most typical and pervasive example is probably that of the least-squares problem, where element functions are gathered into groups which are then squared.

Grouping nonlinear elements into sets is also desirable if we consider solving constrained problems: it is indeed necessary to distinguish the element functions associated with the objective from those associated with the constraints.

In order to achieve this grouping, we have to extend the notion of partial separability and define a slightly more general class. We will say that the real function $f(x)$ is a *group partially separable function* if and only if it is of the form

$$f(x) = \sum_{j=1}^{\ell} g_j(h_j(x)), \tag{19}$$

where the *group functions* g_j are twice continuously differentiable functions from $\mathbf{R}$ into itself, and where their arguments $h_j(x)$ are partially separable functions from $\mathbf{R}^n$ into $\mathbf{R}$. Expanding the h_j, we obtain the expression

$$f(x) = \sum_{j=1}^{\ell} g_j \left(l_j(x) + \sum_{i \in J_j} f_i(x) \right), \tag{20}$$

where $l_j(x)$ is the linear part of $h_j(x)$, if any: the element functions $f_i(x)$ $(i \in J_j)$ now contain the purely nonlinear part of the j-th group.

Our model problem of the previous subsection also exhibits this structure. Indeed, we can choose the group functions as

$$g_{i,j}(y) = \frac{1}{m^2}\sqrt{y}, \tag{21}$$

(where y is called the *group variable*), the linear part of the groups as

$$l_{i,j}(x) = 1 \tag{22}$$

and the two nonlinear element functions of the (i,j)-th group as

$$f_1 = \frac{m^2}{2}(x_{i,j} - x_{i+1,j+1})^2 \text{ and } f_2 = \frac{m^2}{2}(x_{i,j+1} - x_{i+1,j})^2, \tag{23}$$

where we used again our double indexing convention for the group indices. In the same spirit as above, each of these element functions clearly has two elemental variables and only one internal.

An algorithm capable of minimising group partially separable functions is a very powerful tool, as it can be applied, without any modification, to nonlinear least-squares problems, constrained problems (in particular, to their (augmented) Lagrangian formulations) and many other cases. Such an algorithm, called SBMIN, has been developed by the authors in the context of the LANCELOT project that is presented below.

Clearly, we can interpret the use of group partial separability as an additional step (compared to partial separability alone) in the exploitation of the computational tree associated with a given real function of several variables, and then wonder if one more step could not bring further advantages. The resulting procedure would then become closer and closer to the exploitation of the complete tree, as advocated by McCormick and co-workers in the concept of *factorable functions* (see [19], for instance). A complete discussion of the relative merits of partial vs complete exploitation of the computational tree associated with a given problem is outside the scope of the present paper, but should, at least, take the following arguments into account.

- There is a difference between using the complete computational graph [14] for efficient calculation of various quantities (for example, derivatives) that are requested by a given algorithm, and using the complete tree in the algorithm itself.

  The first approach is used by the new promising automatic differentiation algorithms, as discussed in [14], while the second raises the question of the possible use of derivatives of an order higher than two. This very interesting approach has been taken by R. Schnabel and co-authors (see [22] for an introduction to the subject), but the applications have been restricted to small problems and specific subsets of the higher derivatives. Whether or not this type of techniques can be extended to large problems and complete higher derivatives (and whether this is desirable) remains an open question: if a Hessian matrix is large, a third order derivative tensor is huge... but again structure might play an important role here.

- Storage being a factor of importance for large problems, one has to reach a compromise between storage needs and efficiency for a given algorithm.

  The storage required for an algorithm using group partial separability is of the same order as that required for a specialised nonlinear least squares solver, and using specialised software for this last application is widely recommended. This indicates that the balance between storage and efficiency is reasonably achieved in our context.

- A number of problems involve "black boxes", that is user supplied routines for computing some problem dependent numerical functions, for which the structure is unknown. Although it may be possible in the future to process these routines using a specially designed

"precompiler" that would extract information about their underlying structure, it is still necessary for today's algorithms to use these black boxes as they stand. Furthermore, if these black boxes link some specific subsets of variables together, this property should be expressed in the structural description of the complete problem. This can be handled very naturally in the (group) partially separable framework by identifying such black boxes with element functions.

2.3 Structure and new computer architectures

An important issue in the design of algorithms for large scale problems is their potential use of advanced computer architecture. The question is already important for small problems (see [22]), but is even more so for large ones, because the amount of calculation that is purely internal to the algorithm (therefore excluding problem functions evaluation) rises significantly and may well become the dominant computational cost.

The exploitation of partial separability and group partial separability on parallel computers is quite straightforward and efficient (see [17] and [16]). The overall idea is very simple. We note that the computational tasks in an algorithm using partial separability are of three types:

functions and derivatives evaluations : Because of the partial separable structure, all the element functions are independent, and their evaluation can be spread over the available processors in a purely asynchronous manner, an ideal situation in parallel computing.

internal linear algebra : This is one of the areas where the use of parallel computers have been most studied and for which efficient algorithms are now available. At the k-th iteration of a typical Newton-type method, partially separable problems give rise to linear systems of the form

$$\nabla^2 f(x_k)s_k = -\nabla f(x_k),. \tag{24}$$

From the linear algebra point of view, the structure of such systems is extremely similar to that of finite element systems, as the coefficient matrix is the sum of element Hessians which must ultimately be assembled. Efficient algorithms for this class of problems are well studied and developed (see [1] and [12] for instance). Both iterative methods (eg. conjugate gradients variants) or direct algorithms (eg. multifrontal techniques) have been applied to large scale partially separable optimization problems in this context (see [17] and [6]).

internal element handling and updating : These are the algorithm's inner calculations that handle the element functions (including the Hessian approximation update for quasi-Newton methods). Again, these can be shared by the available processors because of the independence of these functions.

Of course, if one restricts one's attention to sparsity of the Hessian matrix, important speedups can still be achieved in internal linear algebra, as discussed above, but parallelisation of the two other types of computational tasks is then more difficult.

As a conclusion, we may say that the use of partial and group partial separability facilitates the algorithms use of the potentialities of parallel computers, providing a natural problem de-

composition, which then results in an efficient partitioning of the computational work amongst the processors.

2.4 The sources of (group) partially separable problems

One of the major sources of (group) partially separable is the discretisation of continuous problems. Both finite differences and finite elements approximations result in problems of this type, mostly because of the "locality" of the involved operators, that only relate variables corresponding to "neighbouring" discretisation points. An interesting collection of such problems has been recently gathered by J. Moré in [20]. This collection features, amongst others, chemical engineering applications, variational inequalities, biomedical modelling, boundary value problems and elasticity analysis.

Nonlinear network problems form another important source of group partially separable problems, ranging from urban traffic equilibria (see [13] for a excellent survey of this type of applications) to water and gas resource management [21]. The structure again results from the same "locality" property that we mentioned for discretised problems: nonlinear variables explicitly interact when they are close to each other in the considered network (they are typically associated to arcs incident to a given node, or to nodes at the extremities of a given arcs).

Other classes of problems that often exhibiting partially and/or group partially separable structure include

- multiperiod planning models,
- input-output macro-economic models,
- multiobjective optimization,
- nonlinear matrix equations.

It seems therefore fair to say that (group) partially separable problems actually occur in most fields where large scale nonlinear optimization is itself relevant. This is not surprising if one recalls Theorem 1, but it is worthwhile to note that the decomposition (14) or (19) often arises very naturally in the problem formulation, its description therefore requiring a minimal amount of additional effort.

3 The LANCELOT software project

The LANCELOT software project was started by the authors more than two years ago. The meaning of the LANCELOT acronym is explained by the banner displayed in Figure 2.

According to this banner, LANCELOT 's purpose is to attack large problems involving nonlinear objectives and/or constraints by using techniques based on the Lagrangian function[3].

The main characteristics of the LANCELOT software can be described as follows.

[3]The "s" at the end of "techniques" is intentional: the present pilot version of the software already uses two different extensions or augmentations of the Lagrangian.

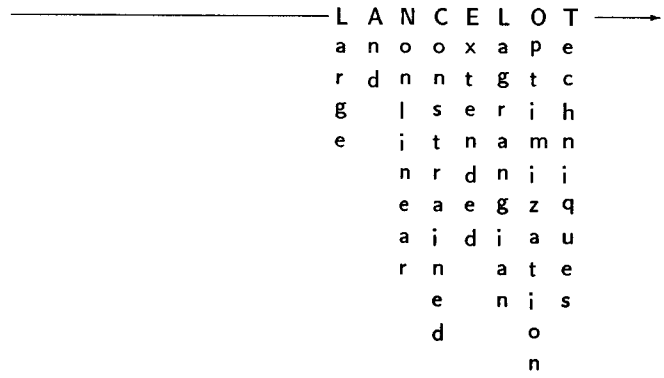

Figure 2: The LANCELOT banner

Use of problem structure: As discussed in the first part of the paper, the use of problem structure is the only reasonable way to tackle large problems. For the reasons explained above, (group) partial separability seems the right concept to invoke for this purpose: LANCELOT will therefore explicitly handle these types of structure.

On the other hand, this capacity to exploit structure will inevitably result is some inefficiency when handling unstructured problem. Care will be taken to ensure that this inefficiency is not too severe.

Efficiency and reliability: LANCELOT will be efficient and reliable. At the beginning of a software project, such a statement is of course a little preposterous. What is meant is that the algorithm design and implementation will systematically use efficient and reliable structures and methods.

An important point is that strong emphasis is put on the theoretical justification of the algorithms and structures used by the software. Suitable convergence theory should be available (and, in part, already is: see [3], [5], [7], [11]). Furthermore, the final implementations and the studied algorithms should differ as little as possible. This requirement of a well established supporting theory is not considered as a sufficient condition guaranteeing software reliability, but as truly necessary.

This theoretical support will be completed by intensive testing on model problems, both practical and more academic. The first ones are crucial because they reflect best the situation in which the software will be applied. The second ones are important too because they introduce sometimes extreme numerical difficulties: the performance of the system when faced with these difficulties is easier to isolate and to improve on such idealised problems (see [4] and [6] for preliminary tests).

Of course, the final efficiency and reliability achieved is best judged by the end-users!

Scope: The domain of application of the LANCELOT system will include a large part of smooth

nonlinear optimization problems. Although primarily focussed on large scale problems, LANCELOT will also cover small and medium size ones. It will exploit specific types of constraints, including

- none (unconstrained problems),
- simple bounds on the problem's variables,
- linear network-type constraints,
- general linear constraints,
- convex constraints, where the feasible domain is such that a (possibly approximate) projection can be efficiently computed,
- general nonlinear nonconvex constraints.

Extension to nonsmooth problems is of interest, but is not planned at this stage. The emphasis will be on nonlinear problems: LANCELOT is not presently intended to compete with large linear programming packages.

Ease of use: Inputting problems to LANCELOT will be reasonably easy. A standard input format for nonlinear problems (SDIF) has been proposed by the authors to achieve this objective. It features a number of facilities to describe problem structure, along the lines analysed in Section 3 of this paper. For instance, it automatically handles multi-indexed variables as they naturally arise from discretizations of multi-dimensional problems (as the minimum surface example presented above). This format has been formalised in [8] and has already been used for the input of a fairly sizeable set of large problems. Although not as complete as a true modelling language, it nevertheless provides an important practical help in specifying structured problems, as well as invaluable internal data consistency checks.

It is the authors' experience that large structured nonlinear problems arising from applications[4] have been fully specified using the SDIF, validated and solved by the pilot version of LANCELOT , the whole process taking less than one hour (which we consider quite reasonable).

Adaptability: The LANCELOT system is also designed in a very modular and hierarchical way which, in turn, provides a good adaptability of the system to extensions, both algorithmic and implementation oriented.

This organisation is also made necessary by the need to provide more than one methodology for some of the algorithmic parts of the system: preconditioning the large linear systems arising from Newton's equation requires, for example, that several strategies (simple diagonal scaling, incomplete factorisation, modified band techniques, ...) be available to the user.

The adaptability of the LANCELOT software is also enhanced by the choice of a reverse communication interface for the system.

[4]The examples we have in mind here were proposed by practitioners in the fields of energy modelling and finite element applications. They involve more that 1000 nonlinear variables and some of them have nonlinear constraints.

Portability: Because the LANCELOT system is designed to be easily portable, the programming language most commonly used for scientific applications, Fortran 77, has been chosen for its development. Strict conformity with the standard of the language is enforced at all levels of the system. Transfer between different machines (CRAY, IBM, DEC and SUN mainframes and workstations, ...) and operating systems (VM/CMS, VMS, UNIX, ...) also takes place during the development phases, in order to ensure maximal portability, not only of the end-product, but also of the successive pilot codes.

Following the arguments of Section 2.3, the code is also designed in a way that has the potential to make it efficient on parallel and/or vector computers.

4 Conclusions

We have shown how the structure of large complex nonlinear problems can be analysed using the concept of (group) partial separability. We have also discussed some aspects of the LANCELOT project, whose purpose is the implementation of this approach in a practical software tool.

Development of the LANCELOT system is ongoing, both from the theoretical and software viewpoints. As alluded to above, some layers and functionalities of the system are already operational and being tested. Detailed numerical experiments with these modules will shortly be reported elsewhere.

The coherence of theoretical concepts with their applications to "real world" problems and the coherence of the theoretical concepts between themselves are considered by the authors to be of central importance. An approach to large scale nonlinear programming that has this type of coherency has been outlined in this paper, ranging from abstract convergence theory and structure analysis to practical software implementations. In a domain just reaching maturity, this unified perspective is desirable.

References

[1] P. Amestoy and I.S. Duff, "Vectorization of a multiprocessor multifrontal code", International Journal of Supercomputers Applications, vol. 3, pp. 41-59, 1989.

[2] T.F. Coleman and Y. Li (eds.), "Large Scale Numerical Optimization", SIAM Publications (to appear), 1990.

[3] A.R. Conn, N.I.M. Gould and Ph.L. Toint, "Global convergence of a class of trust region algorithms for optimization with simple bounds", SIAM Journal on Numerical Analysis, vol. 25, pp. 433–460, 1988. Correction, same journal, vol. 26, pp. 764–767, 1989

[4] A.R. Conn, N.I.M. Gould and Ph.L. Toint, "Testing a class of methods for solving minimization problems with simple bounds on the variables", Mathematics of Computation, vol. 50(182), pp. 399–430, 1988.

[5] A.R. Conn, N.I.M. Gould and Ph.L. Toint, "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds", Technical Report

CS-88-38, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, 1988.

[6] A.R. Conn, N.I.M. Gould, M. Lescrenier and Ph.L. Toint, "Performance of a multifrontal scheme for partially separable optimization", Technical Report CS-88-04, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, 1988.

[7] A.R. Conn, N.I.M. Gould and Ph.L. Toint, "Convergence of quasi-Newton matrices generated by the Symmetric Rank One update", Mathematical Programming (to appear), 1989.

[8] A.R. Conn, N.I.M. Gould and Ph.L. Toint, "A proposal for a Standard Data Input Format for large-scale nonlinear programming problems", Technical Report CS-89-61, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, 1989.

[9] A.R. Conn, N.I.M. Gould and Ph.L. Toint (eds.), "Large Scale Optimization", Mathematical Programming, vol. 45(3), 1989.

[10] A.R. Conn, N.I.M. Gould and Ph.L. Toint (eds.), "Large Scale Optimization — Applications", Mathematical Programming, vol. 48(1), 1990.

[11] A.R. Conn, N.I.M. Gould, A. Sartenaer and Ph.L. Toint, "Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints", Technical Report CS-89-60, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, 1989.

[12] I.S. Duff, N.I.M. Gould, M. Lescrenier and J.K. Reid, "The multifrontal method in a parallel environment", in "Reliable Scientific Computation", M.G. Cox and S.J. Hammarling (eds.), Oxford University Press, 1988.

[13] M. Florian, "Mathematical programming applications in national, regional and urban planning", in "Mathematical Programming: recent developments and applications", M. Iri and K. Tanabe (eds.), Kluwer Academic Publishers, Dordrecht, pp. 57–82, 1989.

[14] A. Griewank, "On automatic differentiation", in "Mathematical Programming: recent developments and applications", M. Iri and K. Tanabe (eds.), Kluwer Academic Publishers, Dordrecht, pp. 83–108, 1989.

[15] A. Griewank and Ph.L. Toint, "On the unconstrained optimization of partially separable functions", in "Nonlinear Optimization 1981" (M.J.D. Powell, ed.), Academic Press, London, 1982.

[16] M. Lescrenier, "Partially separable optimization and parallel computing", in "Parallel Optimization on Novel Computer Architectures", R.R. Meyer and S. Zenios (eds.), A.C. Baltzer Scientific Publishing Co, Switzerland, 1989.

[17] M. Lescrenier and Ph.L. Toint, "Large scale nonlinear optimization on the FPS164 and CRAY X-MP vector processors", International Journal of Supercomputer Applications, vol. 2(1), pp. 66–81, 1988.

[18] O.L. Mangasarian and R.R. Meyer (eds.), "Parallel Methods in Mathematical Programming", Mathematical Programming Vol. 42(2), 1988.

[19] G.P. McCormick, "Nonlinear programming: theory, algorithms and applications", Academic Press, New York, 1983.

[20] J.J. Moré, "A collection of nonlinear model problems", Report ANL/MCS-P60-0289, Argonne National Laboratory, Argonne (USA), 1989.

[21] A.J. Osiadacz and D.J. Bell, "Optimization techniques for large networks: gas and water", in "Simulation and optimization of large systems", A.J. Osiadacz (ed.), Clarendon Press, Oxford, pp. 175–192, 1988.

[22] R.B. Schnabel, "Sequential and parallel methods for unconstrained optimization", in "Mathematical Programming: recent developments and applications", M. Iri and K. Tanabe (eds.), Kluwer Academic Publishers, Dordrecht, pp. 227–262, 1989.

[23] M.J. Todd, "Recent developments and new directions in linear programming", in "Mathematical Programming: recent developments and applications", M. Iri and K. Tanabe (eds.), Kluwer Academic Publishers, Dordrecht, pp. 109–158, 1989.