Large-scale nonlinear optimization and the LANCELOT package

Sartenaer, Annick

Published in:
Belgian Journal of Operations Research, Statistics and Computer Science

Publication date:
1995

Document Version
Peer reviewed version

Large-scale nonlinear optimization and the LANCELOT package

A. Sartenaer

Department of Mathematics
Facultés Universitaires ND de la Paix
61 rue de Bruxelles
5000 Namur, Belgium

e-mail : **as@math.fundp.ac.be**

(This work was supported by the Belgian
National Fund for Scientific Research.)

Abstract

This paper presents a general introduction to the optimization methods which form the basis of the large-scale nonlinear optimization package LANCELOT written by Conn, Gould and Toint. That is, we propose here a comprehensive overview of the principal tools selected by the authors of LANCELOT, among the classical large-scale nonlinear optimization tools, to develop their code. This description has the aim to be complementary to the description given by the authors in their book, while furnishing to the reader a further insight of the underlying ingredients of the LANCELOT package.

Keywords : nonlinear programming, large-scale optimization.

Large-scale nonlinear optimization and the LANCELOT package

by A. Sartenaer*

Report 95/9 October 10, 1995

Abstract. This paper presents a general introduction to the optimization methods which form the basis of the large-scale nonlinear optimization package LANCELOT written by Conn, Gould and Toint. That is, we propose here a comprehensive overview of the principal tools selected by the authors of LANCELOT, among the classical large-scale nonlinear optimization tools, to develop their code. This description has the aim to be complementary to the description given by the authors in their book, while furnishing to the reader a further insight of the underlying ingredients of the LANCELOT package.

* **Department of Mathematics, Facultés Universitaires ND de la Paix, 61, rue de Bruxelles, B-5000 Namur, Belgium (e-mail : as@math.fundp.ac.be).**
This work was supported by the Belgian National Fund for Scientific Research.

Keywords : nonlinear programming, large-scale optimization.

1 Introduction

Optimization problems arise within many areas (mathematics, applied science, engineering, economics, medicine, statistics, ...) and require, in particular, the development of mathematical models, in order to analyze and understand the phenomena. It is of great importance that the models chosen correspond most closely to reality, while being manageable under existing means. However, accurate modelling often leads to *large-scale nonlinear* optimization problems. Fortunately, advances in computer technology are nowadays making the solution of such problems more and more possible. Hence an increasing interest, from both the researchers and practitioners, for developing and using, respectively, software designed to solve larger and larger nonlinear problems.

Our primary interest here is thus in nonlinear problems that involve a large number of variables and/or constraints. Let us then elaborate as to what we generally mean by large. As discussed by Conn, Gould and Toint (1992b), this notion is first clearly *computer dependent*. What is large on a personal computer is significantly different from what is large on a super computer. The first machine has a substantially smaller memory and storage than the second one, and therefore has more difficulty handling problems involving a large amount of data. Secondly, a highly nonlinear problem in one hundred variables could be considered large, whereas in linear programming it is possible to solve problems in five million variables. The notion of size is thus *problem dependent*. It also depends upon the *structure of the problem*. Many large-scale nonlinear problems arise from the modelling of very complicated systems that may be subdivided into loosely connected subsystems. This structure may often be reflected in the mathematical formulation of the problem and exploiting it is often crucial if one wants to obtain an answer efficiently. The complexity of the structure is often a key factor in assessing the size of a problem. Lastly, the notion of a large problem depends upon the *frequency* with which one expects to solve a particular instance or closely related problem. When one anticipates solving the same class of problems many times, one can afford to expend a significant amount of energy analyzing and exploiting the underlying structure. Thus, although it is not possible to say categorically that a problem in say seven hundred variables is large, suffice it to say that, today, a problem in fifty variables is small and a generally nonlinear problem in five thousand variables and one thousand nonlinear constraints is large.

Efficient algorithms for small-scale problems do not necessarily translate into efficient algorithms for large-scale problems. This is unfortunate, since in the past twenty years rather sophisticated and reliable techniques for small-scale problems have been developed (see Kan and Timmer, 1989, for good surveys). Perhaps the main reason is that, in order to be able to handle large problems, the algorithms have to necessarily be *as simple as possible*. Consequently, relative to many of the more successful algorithms for small, dense problems, the amount of information available at any given iteration may be severely restricted. This makes designing algorithms that are scale invariant (in the sense that, assuming infinite precision arithmetic, quasi-Newton methods for unconstrained optimization are invariant under linear transformations) more difficult for large-scale problems. *Scaling* is hence a significant difficulty in the large-scale context.

Another important difficulty is that of *exploiting structure*. The fact that we are able to solve large problems at all is because they are structured. Thus, it is absolutely essential for efficient algorithms to exploit structure. Moreover, this means exploiting more than just sparsity. Unfortunately, this exploitation often complicates the question of stability, that is, the ability of an algorithm to guarantee that small perturbations in the data will only result in small perturbations to the solution for 'satisfactorily conditioned' problems. By contrast, algorithms for small problems have the possibility to ignore structure.

Our main purpose in this paper is to give a *general introduction* to the optimization methods which form the basis of the large-scale nonlinear optimization package LANCELOT (see Conn, Gould and Toint, 1992a). That is, we propose here a comprehensive overview of the principal tools selected by the authors of LANCELOT, among the large-scale nonlinear optimization tools, to develop their code. Doing so, we hope to be complementary to the description given in Conn et al. (1992a) (see Chapter 3), while furnishing to the reader a further insight of the underlying ingredients of the LANCELOT package.

The most general form of the problem that is addressed in LANCELOT is

$$\underset{x\in\Re^n}{\text{minimize}} \quad \mathbf{f}(x) \tag{1.1}$$

subject to the general (possibly nonlinear) inequality constraints

$$\mathbf{c_j}(x) \leq 0, \quad 1 \leq j \leq l, \tag{1.2}$$

to the (possibly nonlinear) equality constraints

$$\mathbf{c_j}(x) = 0, \quad l+1 \leq j \leq m, \tag{1.3}$$

and the simple bounds

$$l_i \leq x_i \leq u_i, \quad 1 \leq i \leq n. \tag{1.4}$$

Here, $\mathbf{f}$ and the $\mathbf{c_j}$ are all assumed to be twice-continuously differentiable and any of the bounds in (1.4) may be infinite.

Calling a *feasible point* a vector x satisfying all the constraints (1.2), (1.3) and (1.4), LANCELOT finds a *local minimizer* of problem (1.1)–(1.4), that is a feasible point x_* such that $f(x_*) \leq f(x)$ for all feasible x in a neighbourhood of x_*. For general nonlinear (non convex) problems indeed, the task of finding a *global minimizer* (a feasible point x_* such that $f(x_*) \leq f(x)$ for each feasible point x) may be computationally intractable, and it is only generally practicable to locate a local minimizer.

When solving the above problem, LANCELOT first transforms automatically the inequality constraints (1.2) to equality constraints by the addition of extra slack or surplus variables. Thereafter, the objective function and general (equality) constraints are combined into a composite function, the *augmented Lagrangian function*,

$$\Phi(x,\lambda,S,\mu) = \mathbf{f}(x) + \sum_{i=1}^{m} \lambda_i \mathbf{c_i}(x) + \frac{1}{2\mu}\sum_{i=1}^{m} s_{ii}\mathbf{c_i}(x)^2, \tag{1.5}$$

where the components λ_i of the vector λ are known as *Lagrange multiplier estimates*, the entries s_{ii} of the diagonal matrix S are positive scaling factors, and μ is known as the *penalty parameter*.

A solution of the constrained minimization problem (1.1)–(1.4) is then sought by solving a sequence of problems in which the current augmented Lagrangian function is approximately minimized within the region defined by the simple bounds (1.4). More precisely, at a major iteration of the process, an approximate minimizer of $\Phi(x, \lambda, S, \mu)$ in the feasible box (1.4) is found for given λ, S and μ. These three quantities are then carefully updated in such a way that convergence is ultimately guaranteed. We refer the reader to Conn, Gould and Toint (1991) for a complete discussion of this approach.

In this paper, we will rather focus on the methods used to solve the subproblem mentioned above at each major iteration, that is, the minimization of a twice continuously function subject to bound constraints. Moreover, in order to simplify the presentation, we will present the methods in an unconstrained context, giving further explanations for the handling of the bound constraints when necessary.

The paper is organized as follows. Section 2 introduces the classical Newton's method, with its advantages and its inconvenients. Section 3 presents methods designed to globalize the Newton one. The iterative solution of the Newton equations through a preconditioned truncated conjugate gradient method is then given in Section 4. The concept of partially separable function as a tool to exploit the structure of a problem is introduced in Section 5. Finally some conclusions are given in Section 6.

2 Newton's method

Newton's method is an iterative method for finding a solution of a system of n *nonlinear* equations of n variables. In many existing methods for solving nonlinear programming problems, these nonlinear equations form the gradient of some twice continuously differentiable function f[1]. That is, if we have to solve, as a subproblem of a nonlinear programming method, the unconstrained minimization problem

$$\min_{x \in \Re^n} f(x),$$

a first-order necessary condition for x_* to be the minimizer of f is that $\nabla f(x_*) = 0$. In this context, Newton's method may be described as an iterative method for finding a point x_* such that $\nabla f(x_*) = 0$.

Using the Taylor's expansion of ∇f at a particular iterate x_k,

$$\nabla f(x_k + p) = \nabla f(x_k) + \nabla^2 f(x_k) p + r(x_k, p), \tag{2.1}$$

where $r(x_k, p)$ is the remainder term, the basic idea of Newton's method is to approximate ∇f by an affine function, that is, neglecting the remainder term. Given this linearization, p_k is then determined so that the right-hand side of (2.1) is zero, that is, p_k satisfies the *linear* system of

[1] For instance in LANCELOT, this function is the function $\Phi(x, \lambda, S, \mu)$ defined in (1.5).

equations

$$\nabla^2 f(x_k)p_k = -\nabla f(x_k). \tag{2.2}$$

If $\nabla^2 f(x_k)$ is nonsingular, p_k is the unique solution of (2.2). Consequently, if $\nabla^2 f(x)$ is nonsingular for all x, Newton's method is well defined and generates a sequence of iterates $\{x_k\}$ as given by the following algorithm.

NT Algorithm.

Step 0. The starting point $x_0 \in \Re^n$ is given. Set $k = 0$.

Step 1. Set

$$p_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$$

and

$$x_{k+1} = x_k + p_k.$$

Increment k by one and go to Step 1.

If $\nabla^2 f(x_*)$ is nonsingular and $\nabla^2 f$ is Lipschitz continuous in a neighbourhood of the solution x_*, then there exists a region surrounding x_* in which Newton's method converges, and the asymptotic rate of convergence is quadratic. For a complete discussion of Newton's method, see Dennis and Schnabel (1983) and Ortega and Rheinboldt (1970).

When Newton's method converges, it is generally agreed to be the most efficient method for solving a system of equations of the type $\nabla f(x) = 0$. However, the method may not converge from every starting point, and if at some iterate x_k the Hessian $\nabla^2 f(x_k)$ is singular, (2.2) does not necessarily have a solution. Also, the need to solve a system of linear equations and to evaluate the second derivatives at each iteration may require a prohibitive amount of calculation if the dimension n is large. Moreover, from an optimization point of vue, what is desired is convergence to a point where not only the gradient vanishes, but where in addition the Hessian has some particular properties in order to ensure the minimizer character of the limit point. Consequently, Newton's method without suitable modifications is generally not an appropriate method. The next sections introduce the most important among these modifications which are used in the LANCELOT package.

3 Globalization methods

Globalization methods are methods that ensure convergence of the iterative process to a stationary point (that is a point that satisfies the first-order necessary conditions for optimality) *from every starting point.* There are two major approaches to obtain *global convergence.* One is called the *linesearch method* and the other is called the *trust region method.* Both methods are very important in nonlinear optimization, and even though the trust region approach has been chosen to be implemented in LANCELOT, we shall give here an overview of the two methods.

3.1 The linesearch approach

The idea of the linesearch method is simple: given a descent direction p_k (such that $\nabla f(x_k)^T p_k < 0$), we select a steplength α_k in that direction that yields an "acceptable" next iterate $x_{k+1} = x_k + \alpha_k p_k$, that is, an iterate that *sufficiently* decreases the value of the function f in order to assure convergence.

The simple requirement "$f(x_k+\alpha_k p_k) < f(x_k)$" is not acceptable to guarantee the convergence of $\{x_k\}$ to a minimizer of f, and we list here some of the main rules used in practice for choosing a stepsize that enforces the desired sufficient decrease in f (see Bertsekas, 1982, for instance).

1. *Minimization rule:* Choose α_k so that

$$f(x_k + \alpha_k p_k) = \min_{\alpha \geq 0} f(x_k + \alpha p_k).$$

2. *Limited minimization rule:* Select a fixed number $s > 0$ and choose α_k so that

$$f(x_k + \alpha_k p_k) = \min_{\alpha \in [0,s]} f(x_k + \alpha p_k).$$

3. *Goldstein rule:* Select a fixed scalar $\sigma \in (0, \frac{1}{2})$ and choose α_k so that

$$\sigma \leq \frac{f(x_k + \alpha_k p_k) - f(x_k)}{\alpha_k \nabla f(x_k)^T p_k} \leq 1 - \sigma.$$

4. *Armijo rule:* Select fixed scalars $s > 0$, $\beta \in (0,1)$ and $\sigma \in (0, \frac{1}{2})$ and set $\alpha_k = \beta^{n_k} s$ where n_k is the first nonegative integer n for which

$$f(x_k) - f(x_k + \beta^n s p_k) \geq -\sigma \beta^n s \nabla f(x_k)^T p_k.$$

The minimization and limited minimization rules are usually implemented with the aid of one-dimensional linesearch algorithms. In practice, the linesearch is stopped once a stepsize α_k satisfying some termination criterion is obtained. An example of such criterion is that α_k satisfies simultaneously

$$f(x_k) - f(x_k + \alpha_k p_k) \geq -\sigma \alpha_k \nabla f(x_k)^T p_k \tag{3.1}$$

and

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq \gamma \nabla f(x_k)^T p_k \tag{3.2}$$

for some scalars $\sigma \in (0, \frac{1}{2})$ and $\gamma \in (\sigma, 1)$. The condition $\gamma > \sigma$ ensures that (3.1) and (3.2) can be satisfied simultaneously. Condition (3.1) in fact guarantees a sufficient function decrease, while condition (3.2) is a test on the accuracy of the minimization, also interpreted as a test of sufficiently large step.

When a *backtracking* strategy is used, condition (3.2) needs not to be implemented, as this type of strategy avoids excessively small steps. The framework of such a strategy is the following.

BT Algorithm.

Step 0. Choose $\sigma \in (0, \frac{1}{2})$ and l, u satisfying $0 < l < u < 1$. Set $\alpha_0 = 1$ and $i = 0$.

Step 1. If

$$f(x_k) - f(x_k + \alpha_i p_k) < -\sigma \alpha_i \nabla f(x_k)^T p_k,$$

then set

$$\alpha_{i+1} = \rho_i \alpha_i \text{ for some } \rho_i \in [l, u],$$

increment i by one and go to Step 1. Else set

$$x_{k+1} = x_k + \alpha_i p_k$$

and STOP.

The Armijo rule is a particular case of this last algorithm where $s = 1$ and $\rho_i = \beta$ for all i (see Sartenaer, 1993). Another strategy for reducing α_i (choosing ρ) is to minimize a quadratic (or even cubic) model of $\hat{f}(\alpha) = f(x_k + \alpha p_k)$, the one-dimensional restriction of f to the line through x_k in the direction p_k, using the current information about $\hat{f}$ to model it (see Dennis and Schnabel, 1983).

Convergence results for the above list of rules or strategies may be found in Bertsekas (1982) and Dennis and Schnabel (1983), where superlinear asymptotic rate of convergence is obtained. It is important to note that these convergence results are derived under the condition that a *full* step (Newton step for instance) is tried first, as suggested in the backtracking algorithm given above by setting $\alpha_0 = 1$. Therefore, the performance of such algorithms will depend on the properties of fast local convergence of the full step used.

When constraints are present in the problem, we can adapt the linesearch methods so that the iterates be feasible, thus using feasible search directions and selecting feasible steplengths. For bound constraints, for instance, a method that maintains feasibility is appropriate and quite simple.

3.2 The trust region approach

When the full Newton step is unsatisfactory, in order to achieve global convergence, the linesearch approach will retain the same *step direction* and then search for an approximate local minimizer along the line defined by that direction, selecting a *shorter steplength*. The alternate approach to obtain global convergence is based on the observation that Newton's method models the function by a quadratic approximation around the current iterate and that if the full Newton step is not acceptable, that is because the quadratic model does not adequately model f in a region containing this full step. The quadratic being accurate only in a neighbourhood of the current iterate, the new approach consists in choosing the next iterate to be an (approximate) minimizer of the quadratic constrained to be in a *region* where we *trust* the approximation. Consequently, in trust region methods, when we need to take a shorter step, we first will choose a *shorter steplength* and then use the full n-dimensional quadratic model to choose the *step direction*.

Trust region methods can also be introduced as a way to circumvent the difficulty caused by non positive definite Hessian matrices in Newton's method. Indeed, in this case, the underlying quadratic model does not have a unique minimizer and the method is not defined. Another way

of regarding this fact is that the region around x_k in which the model is adequate does not include a minimizing point of this model, and one more time, a more realistic approach is to assume that some neighbourhood of x_k is defined in which the model agrees with the function in some sense.

If

$$m_k(x_k + p) \stackrel{\text{def}}{=} f(x_k) + \nabla f(x_k)^T p + 1/2 p^T \nabla^2 f(x_k) p \tag{3.3}$$

denotes the quadratic approximation of f around the current iterate x_k, the *trust region problem* is to find p_k so that

$$m_k(x_k + p_k) = \min\{m_k(x_k + p) \mid \|p\| \leq \Delta_k\} \tag{3.4}$$

for some given norm $\|\cdot\|$ and some positive scalar Δ_k called the *trust region radius*. Note that in practice, p_k will be chosen as an *approximate* minimizer of problem (3.4) (see below). Once a step p_k is selected, that guarantees a sufficient decrease on the model inside the trust region, the objective function is then evaluated at the new point $x_k + p_k$. If its value has decreased enough, the new point is accepted as next iterate and the trust region radius is possibly increased. Otherwise, the new point is rejected and the trust region radius is reduced. The updating of the trust region radius is directly depending on a certain measure of agreement existing between the model and the objective function, namely, the quantity ρ_k below, defined as the *actual reduction* in f divided by the *predicted reduction*, that is the reduction in m_k. The trust region approach may be summarized by the following general framework.

TR **Algorithm.**

Step 0. Choose η_1, η_2 satisfying $0 < \eta_1 < \eta_2 < 1$ and $\gamma_1, \gamma_2, \gamma_3$ satisfying $0 < \gamma_1 \leq \gamma_2 < 1 < \gamma_3$.

Step 1. Calculate $f(x_k)$ and the model $m_k(x_k + p)$.

Step 2. Determine a solution p_k to problem (3.4).

Step 3. Compute the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(x_k) - m_k(x_k + p_k)}.$$

Step 4. If $\rho_k > \eta_1$, then set

$$x_{k+1} = x_k + s_k$$

and

$$\Delta_{k+1} \in [\Delta_k, \gamma_3 \Delta_k], \text{ if } \rho_k \geq \eta_2,$$

or

$$\Delta_{k+1} \in [\gamma_2 \Delta_k, \Delta_k], \text{ if } \rho_k < \eta_2.$$

Otherwise, set

$$x_{k+1} = x_k$$

and

$$\Delta_{k+1} \in [\gamma_1 \Delta_k, \gamma_2 \Delta_k].$$

In practice, a more sophisticated update of the trust region radius is used in Step 4, in which if $\rho_k \leq \eta_1$, then Δ_{k+1} is chosen in an interval $(0.1, 0.5)\Delta_k$ on the basis of a polynomial interpolation (see Dennis and Schnabel, 1983). It can be shown (see Fletcher, 1987) that this framework is globally convergent and also retains the rapid rate of convergence of Newton's method. In fact, the heuristic of restricting the step by $\|p\| \leq \Delta_k$ ensures that significant reductions in f are made until x_k is close to a local solution. At this stage, the restriction becomes inactive and the iteration then reduces to the basic Newton's method with its rapid rate of convergence.

When (3.4) is defined in term of the Euclidean norm, a necessary and sufficient condition for the step p_k to be a global solution of (3.4) is that it solves the system

$$(\nabla^2 f(x_k) + \lambda I)p = -\nabla f(x_k) \tag{3.5}$$

for some nonnegative scalar λ such that the matrix $\nabla^2 f(x_k) + \lambda I$ is at least positive semi-definite and

$$\lambda(\Delta_k^2 - \|p_k\|^2) = 0.$$

(For a proof, see Fletcher, 1987, for instance.) Thus, if Δ_k is large enough and $\nabla^2 f(x_k)$ is positive definite, the solution of (3.4) is simply the Newton direction. Otherwise, the restriction on the norm will apply and $\|p_k\| = \Delta_k$. Methods for solving (3.5) where first suggested by Levenberg (1944) and Marquardt (1963) in the context of nonlinear least squares problems, and then by Goldfeldt, Quandt and Trotter (1966) for general nonlinear problems. All Levenberg-Marquardt algorithms find a value $\lambda \geq 0$ such that $\nabla^2 f(x_k) + \lambda I$ is positive definite and solve (3.5) to determine p_k. There are many variations on this theme, and in some earlier methods, the precise restriction $\|p_k\| \leq \Delta_k$ on the length of p_k is not imposed. Instead, λ is used as the controlling parameter in the iteration and the length of p_k is determine by whatever value λ happens to take. However, this has disadvantages, and more recent Levenberg-Marquardt type methods (see Hebden, 1973 and Moré, 1978) follow the model of Algorithm TR much more closely, controlling the iteration by using the radius Δ_k. In this case, equation (3.5) is regarded as defining a trajectory $p(\lambda)$, and the precise value of λ which makes $\|p(\lambda)\| = \Delta_k$ (or an approximation of it) is sought (see Fletcher, 1987). This may be carried out with any standard method, but Hebden (1973) suggested a very efficient method using safeguarded rational approximation (see Dennis and Schnabel, 1983, and Moré, 1978).

Another approach related indirectly to the use of (3.5) and introduced by Powell (1970) is based on the observation that the trajectory $p(\lambda)$ runs smoothly from the Newton step $[p(0) = -\nabla^2 f(x_k)^{-1}\nabla f(x_k)]$ to an incremental *steepest descent* step $[p(\lambda) \to -(1/\lambda)\nabla f(x_k)$ when $\lambda \to \infty]$. In the general case, $p(\lambda)$ can be interpreted as interpolating these extreme cases, and it is therefore possible to approximate $p(\lambda)$ by a piecewise linear trajectory made up of two (three) line segments and called the *dogleg (double dogleg) trajectory*. This path connects first $p = 0$ to the *Cauchy step*, that is the step p^C minimizing the quadratic model m_k in the steepest descent direction, and then the Cauchy step to the Newton step, p^N say. (p^C and p^N are defined by $p^C = -\nabla f(x_k)\|\nabla f(x_k)\|^2 / \nabla f(x_k)^T \nabla^2 f(x_k) \nabla f(x_k)$ and $p^N = -\nabla^2 f(x_k)^{-1}\nabla f(x_k)$, respectively.) In the double dogleg trajectory is introduced the extra point γp^N, where $0 < \gamma \leq 1$,

giving an earlier bias to the Newton step. The resulting trajectory is used in an algorithm similar to Algorithm TR, but the computation of $p(\lambda)$ from (3.5) is replaced by using the point p of length Δ_k on the (double) dogleg trajectory (see Dennis and Schnabel, 1983 for more details on that method).

In the LANCELOT package, an algorithm similar to Algorithm TR is used, where the function f is replaced by the augmented Lagrangian function $\Phi(x, \lambda, S, \mu)$ defined in (1.5), and the bound constraints (1.4) are added in the trust region problem (3.4). Moreover, the norm used to define the trust region in problem (3.4) may be either the Euclidean one or the infinity one. The advantage of this last possibility is that the trust region constraint and the bound constraints may then be merged together and be handled more easily.

Problem (3.4) in Step 2 of Algorithm TR is solved in two stages by LANCELOT. In the first stage, the *generalized Cauchy step* p^{GC} is computed. This step is defined as the (possibly approximate) minimizer of the quadratic model m_k in the steepest descent direction *subject to* the bound constraints (1.4). This step is important for two reasons. Firstly, convergence of the algorithm can be guaranteed provided the value of the quadratic model at the step p_k is no larger than that at the generalized Cauchy step (see Conn, Gould and Toint, 1988). Secondly, the set of variables which lie on their bounds at the generalized Cauchy point, $x_k + p^{GC}$, often provide an excellent prediction of those which will ultimately be fixed at the solution to the problem.

The second stage consists in further reducing the quadratic model (in order to guarantee a reasonable rate of convergence), by solving problem (3.4) *starting from* the generalized Cauchy point and *keeping fixed* the variables which lie on their bounds at the generalized Cauchy point. LANCELOT offers two possibilities to perform this stage. Either a direct method (using matrix factorization) is used to approximately solve the Newton equations of problem (3.4) (see Conn et al., 1992a), or an iterative method called the *conjugate gradient method* is performed. The next section gives a general introduction to this last method, which is reputed to be well suited in the context of large-scale nonlinear optimization, as explained below.

4 The conjugate gradient method

4.1 The preconditioned truncated conjugate gradient method

The *conjugate gradient* method is a particular case of the class of *conjugate direction* methods, originally developed for minimizing a quadratic function in a known finite number of iterations. Indeed, if we define a set of non-zero vectors $d_1, \ldots, d_n$ as *mutually conjugate with respect to some positive definite matrix* A when they satisfy the condition

$$d_i^T A d_j = 0 \ \text{ for } \ i \neq j, \ \ i = 1, \ldots, n \ \text{ and } \ j = 1, \ldots, n,$$

a conjugate direction method is one which generates such directions, when applied to a quadratic function with positive definite Hessian H, and terminates in at most n exact linesearches, each x_{i+1} being the minimizer in the subspace generated by x_1 and the directions $d_1, \ldots, d_i$ (that is the set of points $\{x \in \Re^n | x = x_1 + \sum_{j=1}^{i} \alpha_j d_j \ \forall \alpha_j\}$). In particular, x_{n+1} is the minimizer of the quadratic function over $\Re^n$ (see Fletcher, 1987 for a proof).

The aim of the conjugate gradient method is to generate mutually conjugate directions by associating conjugacy properties with the steepest descent method in an attempt to achieve both efficiency and reliability. This method, credited to Fletcher and Reeves (1964), deflects the direction of steepest descent by adding to it a positive multiple of the direction used in the last step. When applied for solving a set of positive definite symmetric linear equations $Ax = -b$, the conjugate gradient method may be described by the following algorithm, using the notation r_i for the residual vector $b + Ax_i$ (see Gill, Murray and Wright, 1981):

CG Algorithm.

Step 0. The starting point $x_0 \in \Re^n$ is given. Set $\beta_{-1} = 0$, $d_{-1} = 0$, $r_0 = b + Ax_0$ and $i = 0$.

Step 1. If $r_i = 0$, then STOP. Else set

$$d_i = -r_i + \beta_{i-1} d_{i-1},$$

$$\alpha_i = \frac{\|r_i\|_2^2}{d_i^T A d_i},$$

$$x_{i+1} = x_i + \alpha_i d_i,$$

$$r_{i+1} = r_i + \alpha_i A d_i$$

and

$$\beta_i = \frac{\|r_{i+1}\|_2^2}{\|r_i\|_2^2},$$

increment i by one and go to Step 1.

The remarkable feature of the conjugate gradient method is that it computes the solution of a linear system using only products of the matrix with a vector, and does *not* require the elements of the matrix explicitly. It has also the property that, if exact arithmetic is used, convergence will occur in m ($m \leq n$) iterations, where m is the number of distinct eigenvalues of A. Very fast convergence may thus be obtained if the eigenvalues of A are clustered into groups of approximately equal value. Therefore, the rate of convergence should be significantly improved if the original system can be replaced by an equivalent one, in which the matrix has many unit eigenvalues. This is the idea of *preconditioning*, that constructs a transformation, using a symmetric positive definite *preconditioning matrix*, to have this effect on A (see Gill et al., 1981). (Note that the LANCELOT package proposes a list of preconditioning which are presented in Conn et al., 1992a).

Preconditioned conjugate gradient methods are also well suited when applied to solve *approximately* the linear system that arises in Newton's method. Indeed, when far from the minimizer of the nonlinear objective function, it may not be useful to solve the Newton equations with full accuracy, and we may then perform only a limited number of iterations of the conjugate gradient method, computing a vector that interpolates between the steepest descent direction and Newton direction. This has been termed a *truncated Newton* method and is especially successful when combined with a preconditioning, since then a good direction can be produced in a small number of conjugate gradient iterations.

4.2 The conjugate gradient–trust region method

Of particular interest is the use of the preconditioned truncated conjugate gradient method to find an approximate solution of the trust region problem (3.4) (see Steihaug, 1983, and Toint, 1981). In that case, three different termination rules are included in the method. We terminate when we have a sufficiently good approximation to the Newton step (cf. truncated Newton method). Secondly, we terminate when the norm of the approximation is too large (with respect to the trust region constraint), in which case we take a linear combination of the previous iterate and the current one. Finally, when we encounter a direction of negative curvature, then we move to the boundary. This may be regarded as a generalized dogleg scheme. Consider the trust region problem (3.4) where m_k is defined in (3.3) and the preconditioning is used under the form of a weighted norm

$$\|d\|_C^2 = d^T C d, \tag{4.1}$$

where C is a symmetric positive definite matrix. If we use the notations H for $\nabla^2 f(x_k)$, g for $\nabla f(x_k)$ and r_i for $-(g + Hp_i)$, the algorithm is:

TRCG **Algorithm.**

Step 0. Set $p_0 = 0$ and $r_0 = -g$. Solve $C\tilde{r}_0 = r_0$. Set $d_0 = \tilde{r}_0$ and $i = 0$.

Step 1. Compute

$$\gamma_i = d_i^T H d_i.$$

If $\gamma_i > 0$, then go to Step 2. Otherwise, compute $\tau > 0$ so that

$$\|p_i + \tau d_i\|_C = \Delta_k,$$

set

$$p_k = p_i + \tau d_i$$

and STOP.

Step 2. Compute

$$\alpha_i = \frac{r_i^T \tilde{r}_i}{\gamma_i}$$

and

$$p_{i+1} = p_i + \alpha_i d_i.$$

If $\|p_{i+1}\|_C < \Delta_k$, then go to Step 3. Otherwise, compute $\tau > 0$ so that

$$\|p_i + \tau d_i\|_C = \Delta_k,$$

set

$$p_k = p_i + \tau d_i$$

and STOP.

Step 3. Compute

$$r_{i+1} = r_i - \alpha_i H d_i.$$

If

$$\frac{\|r_{i+1}\|_C}{\|g\|_C} < \xi,$$

then set

$$p_k = p_{i+1}$$

and STOP. Otherwise, go to Step 4.

Step 4. Solve

$$C\tilde{r}_{i+1} = r_{i+1}.$$

Compute

$$\beta_i = \frac{r_{i+1}^T \tilde{r}_{i+1}}{r_i^T \tilde{r}_i}$$

and

$$d_{i+1} = \tilde{r}_{i+1} + \beta_i d_i.$$

Increment i by one and go to Step 1.

In Steihaug (1983), the author has shown that, for the piecewise linear trajectory $p(\lambda)$ that connects $p_j, j = 0, 1, \ldots, i$, and p_k, we have that $m_k(x_k + p(\lambda))$ is strictly monotonic decreasing and $\|p(\lambda)\|_C$ is strictly monotonic increasing. The method is thus well defined and reasonable, as there exists a unique step of length Δ_k (for $\Delta_k \leq \|H^{-1}g\|_C$) on the trajectory $p(\lambda)$ that minimizes the model m_k on that trajectory. When the previous algorithm is used in Step 2 of Algorithm TR, we obtain a globally convergent method that retains the rapid rate of convergence of Newton's method (see Steihaug, 1983, for a complete summary of convergence results for that method).

The LANCELOT package uses a slightly modified version of Algorithm TRCG. That is, in order to take the bound constraints into account, LANCELOT temporarily stops Algorithm TRCG when a bound constraint is encountered, then fixes the correspondent variable to its bound and restarts the algorithm from the current point. This is intended to restore the conjugacy of the directions which may be lost when a bound is encountered.

5 Partial separability

An important feature of the LANCELOT package is the way it exploits the structure of the problem to be solved. Techniques for exploiting the structure of a problem – as sparsity in large systems of linear equations for instance – are indeed playing a crucial role in the development of practical computational methods for large-scale problems. In nonlinear optimization (and in particular in LANCELOT), the technique chosen makes use of the notion of *partial separability*, as introduced by Griewank and Toint (1982), and 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).

In order to indroduce and illustrate the concept of partial separability, we will consider a simple example, inspired by a more complex problem called "the discretised minimum surface problem over the unit square" (see Conn, Gould and Toint, 1990).

Consider the minimization of the objective function

$$f(x) = \sum_{i,j=1}^{3} s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}), \tag{5.1}$$

where

$$s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}) = \sqrt{1 + (x_{i,j} - x_{i+1,j+1})^2 + (x_{i,j+1} - x_{i+1,j})^2}, \tag{5.2}$$

and where the 16 variables correspond to the vertices of a unit square discretised in 9 smaller squares, as shown in Fig. 1.

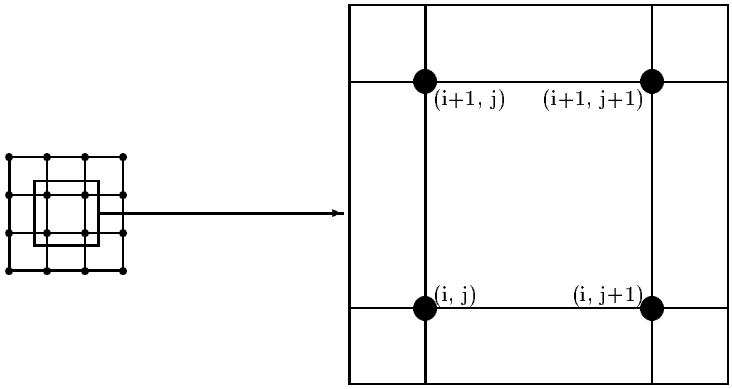

Figure 1: Discretisation of the unit square

It is easy to see that the Hessian, $\nabla^2 f(x)$, has a block-tridiagonal sparsity pattern with tridiagonal blocks (see Fig. 2). Actually, this sparsity structure may be discovered by considering the Hessian of each of the 9 functions $s_{i,j}$ (when taken as functions of the complete set of 16 variables), and by putting them on top of one another (see Fig. 3). So, considering each function $s_{i,j}$ as a function of only $x_{i,j}$, $x_{i+1,j}$, $x_{i,j+1}$ and $x_{i+1,j+1}$ (hence with a dense 4×4 Hessian matrix), the full Hessian $\nabla^2 f(x)$ may then be reconstituted by assigning each of the rows and columns of each 4×4 matrix to the relevant row and column of the larger $\nabla^2 f(x)$.

Let denote by $s_{i,j}(x)$ the function $s_{i,j}$ when considered as a function of the complete set of variables. We see that it satisfies the important property that

$$s_{i,j}(x) = s_{i,j}(x + w) \tag{5.3}$$

for all vectors w in the *invariant subspace*

$$N^s_{i,j} = \{w \in \Re^{16} | w_{i,j} = w_{i,j+1} = w_{i+1,j} = w_{i+1,j+1} = 0\}. \tag{5.4}$$

Each $s_{i,j}(x)$ 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

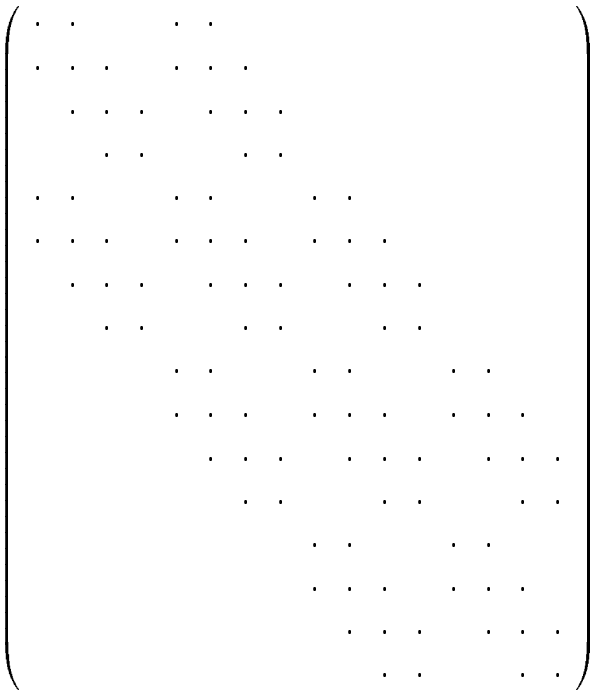

Figure 2: The Hessian pattern of function $f(x)$ in (5.1)

(5.3)–(5.4) is in fact the most important observation when analysing the structure of $\nabla^2 f(x)$. So, the sparsity pattern of this last matrix may be interpreted as a *consequence* of the problem structure: *sparsity is easily derived from the structure, but the reverse is not true.*

Examining the function $s_{i,j}$ in more detail, we also 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*

$$u_{i,j} \stackrel{\text{def}}{=} x_{i,j} - x_{i+1,j+1} \quad \text{and} \quad v_{i,j} \stackrel{\text{def}}{=} x_{i,j+1} - x_{i+1,j}. \tag{5.5}$$

If we write the simple linear transformation

$$\begin{pmatrix} u_{i,j} \\ v_{i,j} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_{i,j} \\ x_{i,j+1} \\ x_{i+1,j} \\ x_{i+1,j+1} \end{pmatrix}, \tag{5.6}$$

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

$$s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}) \stackrel{\text{def}}{=} \hat{s}_{i,j}(u_{i,j}, v_{i,j}), \tag{5.7}$$

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

$$\hat{s}_{i,j}(u_{i,j}, v_{i,j}) = \sqrt{1 + (u_{i,j}^2 + v_{i,j}^2)}. \tag{5.8}$$

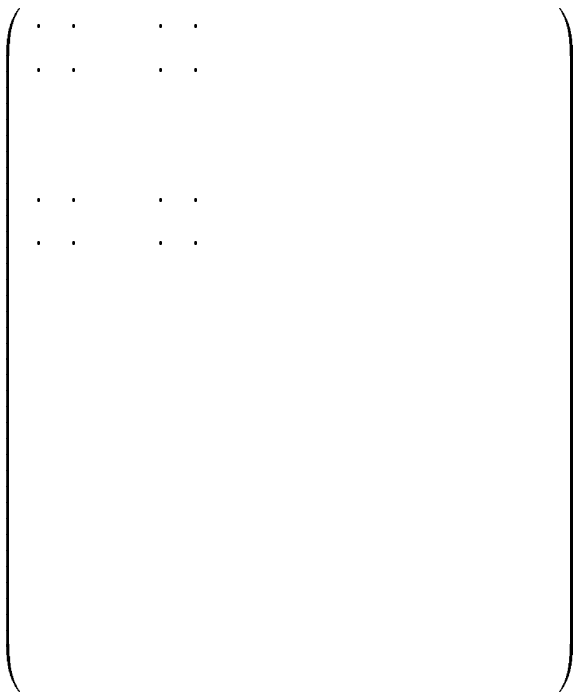

Figure 3: The Hessian pattern of function $s_{1,1}(x)$

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

$$W^T \nabla \hat{s}_{i,j}(u_{i,j}, v_{i,j}) = \nabla s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}), \tag{5.9}$$

where the matrix

$$W = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}, \tag{5.10}$$

and also that

$$W^T \nabla^2 \hat{s}_{i,j}(u_{i,j}, v_{i,j}) W = \nabla^2 s_{i,j}(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}), \tag{5.11}$$

where $s_{i,j}$ is again considered as a function of four variables. Observe that the Hessian of $\hat{s}_{i,j}$ is now a 2×2 symmetric matrix, while that of $s_{i,j}$ is 4×4. Furthermore, the matrix W needs not to be indexed, 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 9 Hessians of the functions $\hat{s}_{i,j}$.

In order to abstract from these observations, it is important to note that (5.3) holds not only for the vectors w in $N^s_{i,j}$, but for all w in

$$N_{i,j} = N^s_{i,j} \oplus \{w \in \Re^{16} | w_{i,j} = w_{i+1,j+1} \text{ and } w_{i,j+1} = w_{i+1,j}\}, \tag{5.12}$$

where the symbol $\oplus$ denotes the direct sum of two subspaces. 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

$$f(x) = \sum_{i=1}^{p} f_i(x), \tag{5.13}$$

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

$$f_i(x) = f_i(x + w). \tag{5.14}$$

As in the example above, 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

$$u = W_i x. \tag{5.15}$$

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

$$\nabla \hat{f}_i(u) \text{ and } \nabla^2 \hat{f}_i(u), \tag{5.16}$$

whose dimensions are smaller.

As we have seen, 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 $\Re^n$ into $\Re$ having a sparse Hessian matrix is partially separable.*

The reader is referred to Griewank and Toint (1982) for a more detailed discussion of this basic property.

The definition (5.13)–(5.14) is not the most general one (see, for example, Griewank and Toint, 1982), but has the advantage of being rather intuitive. The LANCELOT package uses the notion of *group partial separability* which is somewhat more complicated but allows to go one step further in the exploitation of the problem structure (see Conn et al., 1992a).

6 Conclusions

In this paper, we propose a general introduction to some classical optimization methods which have been used in the nonlinear optimization package LANCELOT. This package, written by Conn et al. (1992a) and particularly designed to solve large-scale nonlinear optimization problems, has been built using a performing combination of classical optimization methods together with new approaches which have been developed by the authors. While those approaches are extensively detailed in the literature, this paper presents the classical optimization background used in the framework of LANCELOT, showing how this background has been actually embedded in the somehow more complex structure and composition of the package.

Acknowledgement

The author wish to thank Philippe Toint who contributed to improve the present manuscript.

References

Bertsekas, D. P. (1982). *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, London.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1988). Global convergence of a class of trust region algorithms for optimization with simple bounds, *SIAM Journal on Numerical Analysis* **25**: 433–460. See also same journal 26:764–767, 1989.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1990). An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project, *in* R. Glowinski and A. Lichnewsky (eds), *Computing Methods in Applied Sciences and Engineering*, SIAM, Philadelphia, USA, pp. 42–51.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1991). A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, *SIAM Journal on Numerical Analysis* **28**(2): 545–572.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1992a). LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*, number 17 in *Springer Series in Computational Mathematics*, Springer Verlag, Heidelberg, Berlin, New York.

Conn, A. R., Gould, N. I. M. and Toint, P. L. (1992b). Large-scale nonlinear constrained optimization, *in* J. R. E. O'Malley (ed.), *Proceedings of the Second International Conference on Industrial and Applied Mathematics*, SIAM, Phildelphia, USA, pp. 51–70. (also in "Linear Algebra for Large-Scale and Real-Time Applications", (M.S. Moonen, G.H. Golub and B.L.R DeMoor, eds.), Kluwer Academic Publishers, NATO ASI Series E: Applied Sciences, vol. 232, 1993).

Dennis, J. E. and Schnabel, R. B. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall, Englewood Cliffs, USA.

Fletcher, R. (1987). *Practical Methods of Optimization*, second edn, J. Wiley and Sons, Chichester.

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients, *Computer Journal* **7**: 149–154.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press, London and New York.

Goldfeldt, S. M., Quandt, R. E. and Trotter, H. F. (1966). Maximization by quadratic hill-climbing, *Econometrica* **34**: 541–551.

Griewank, A. and Toint, P. L. (1982). On the unconstrained optimization of partially separable functions, *in* M. J. D. Powell (ed.), *Nonlinear Optimization 1981*, Academic Press, London and New York, pp. 301–312.

Hebden, M. D. (1973). An algorithm for minimization using exact second derivatives, *Technical Report T.P. 515*, AERE Harwell Laboratory, Harwell, UK.

Kan, A. H. G. R. and Timmer, G. T. (1989). Global optimization, *in* G. L. Nemhauser, A. H. G. R. Kan and M. J. Todd (eds), *Optimization*, Vol. 1 of *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, pp. 631–662.

Levenberg, K. (1944). A method for the solution of certain problems in least squares, *Quarterly Journal on Applied Mathematics* **2**: 164–168.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal on Applied Mathematics* **11**: 431–441.

Moré, J. J. (1978). The Levenberg-Marquardt algorithm: implementation and theory, *in* G. A. Watson (ed.), *Proceedings Dundee 1977*, Springer Verlag, Berlin. Lecture Notes in Mathematics.

Ortega, J. M. and Rheinboldt, W. C. (1970). *Iterative solution of nonlinear equations in several variables*, Academic Press, New York.

Powell, M. J. D. (1970). A hybrid method for nonlinear equations, *in* P. Rabinowitz (ed.), *Numerical Methods for Nonlinear Algebraic Equations*, Gordan and Breach, London, pp. 87–114.

Sartenaer, A. (1993). Armijo-type condition for the determination of a generalized Cauchy point in trust region algorithms using exact or inexact projections on convex constraints, *Belgian Journal of Operations Research, Statistics and Computer Science* **33**(4): 61–75.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization, *SIAM Journal on Numerical Analysis* **20**(3): 626–637.

Toint, P. L. (1981). Towards an efficient sparsity exploiting Newton method for minimization, *in* I. S. Duff (ed.), *Sparse Matrices and Their Uses*, Academic Press, London, pp. 57–88.