Large-scale nonlinear constrained optimization

A. R. Conn, Nick Gould and Ph. L. Toint

January 16, 2018

Abstract

During the past ten years, much progress has been made in the theory and practice of constrained nonlinear optimization. However, considerable obstacles appear when these ideas are applied to large-scale problems. This is important as many real applications require the solution of problems in thousands of unknowns. In some areas, in particular linear programming, considerable progress has been made. But even modest departures into nonlinearity, for example the solution of large, general quadratic programs, present considerable challenges. This is apparent when one views the paucity of software for solving such problems. Unsurprisingly, the position does not improve as more drastic forms of nonlinearity are encountered.

In this paper, we will try to explain why the difficulties arise, how attempts are being made to overcome them and what the problems are that still remain.

1 Introduction

Our purpose in this paper is to present an overview of the state-of-the art in large-scale nonlinear optimization. This article is a personal response to the questions, why we are interested in large-scale nonlinear optimization, what are the difficulties and what kind of progress has been made. Although we have made some effort to be complete in our references and thus hope to provide a useful bibliography, we have not attempted to be as complete in our overview. Rather, we have tried to include enough details of the general issues to indicate the nature and reasons for some of the current research in the field. Moreover, the length of treatment is frequently an indicator that the work is new and less well-known, and a brief mention does not mean that the work is relatively less important.

It seems appropriate to first state the most general form of the problem that we are addressing, namely

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

subject to the general (possibly nonlinear) inequality constraints

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

to the (possibly nonlinear) equality constraints

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

and the simple bounds

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

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

At the outset we should make it clear that we only expect to obtain local minimizers. This is in marked contrast to combinatorial optimizers who, typically, are only interested in global solutions. This presents no problems in convex programming, where all local minima are indeed global (for example, in linear programming), but even for small, general nonlinear programming problems it is usually extremely difficult to verify globality. For large problems, it is practically impossible. Fortunately, in many situations, an algorithm that determines local optima suffices.

Our primary interest here is in problems that involve a large number of variables and/or constraints. Consequently, it seems worthwhile to elaborate as to what we mean by large.

Firstly, this notion is clearly *computer dependent*. What is large on an Apple Macintosh is significantly different from what is large on an IBM 3090 or a Cray 2. The first machine has a substantially smaller memory and storage than the other two, 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.

One might suppose that intellectual curiosity alone is sufficient reason to be interested in large-scale nonlinear optimization. However, although we readily admit to the fact that this is an important element of our interest (and indeed if our research had been confined to the publication of theoretical articles, arguably the main one), much of our joint effort has been devoted to the time

consuming and often tedious task of writing software, preparing input and testing. The salient point is that there is a need for algorithms to solve large-scale nonlinear optimization problems. The accurate modelling of physical and scientific phenomena frequently leads to such problems. Nature loves to optimize: minimum energy, minimum potential difference, shortest paths. Moreover, the universe certainly is not linear. If the model is to be accurate (for example, if it is derived from a discretization of a continuous process), the number of variables is necessarily large. Another area where large nonlinear problem arise naturally is in economics, where one often wishes to maximize profit (or minimize losses) in complex situations involving many parameters. The proliferation of large linear models, rather than nonlinear ones, is sometimes a consequence of our lack of knowledge concerning the phenomena being modelled, in which case assuming linearity is about the simplest assumption one can make. As our knowledge improves, often the models are refined and nonlinearity should be introduced. In our opinion, the frequent use of linear models is not an indication that nonlinear problems do not abound. Rather, it is a statement of the desire to use an algorithm (the simplex method) that is readily understood and is well-known to be suitable for large problems. In particular, it is one of our tasks to convince you that you should consider solving nonlinear programs, when they are more appropriate. As a necessary corollary, it should be emphasized that solutions to large nonlinear problems on moderate workstations in a reasonable amount of time are currently quite possible. Furthermore, in practice one is often only seeking marked improvement rather than assured optimality (another reason why globality is not necessarily an issue). This fact makes even problems that at first sight seem impossible (for example, control problems that one wishes to solve in something like real-time), tractable.

Without a doubt, the ubiquity of powerful workstations and the availability of supermachines (both parallel and sequential) have encouraged research in algorithms for large-scale problems. However, we concur with a remark that Martin Beale once made that he would 'much rather work with today's algorithms on yesterday's computers than with yesterday's algorithms on today's computers' [128].

2 Examples of Applications

As we already stated, our interest in developing algorithms for large-scale optimization was created out of necessity. There is an increasing demand for such software as the size and nonlinearity of the problems that practitioners are interested in solving steadily grows. The same evolution that leads to larger and larger nonlinear optimization models for physical phenomena is observed in data fitting, econometrics and operations research models. In particular, it is perhaps worth listing some examples:

- Discretizations of variational calculations and optimal control problems involving both state and control variables.

  These arise, for example, in quantum physics, tidal flow analysis,

design (of aircraft, journal bearings and other mechanical devices), structural optimization, ceramics manufacturing, chemical process control and satellite piloting.

- Nonlinear equations arising both in their own right and in the solution of ordinary and partial differential equations.

  These occur, for instance, in elasticity, semiconductor simulation, chemical reaction modelling and radiative transfer.

- Nonlinear least squares or regression.

  Some examples include fluid dynamic calculations, tomography (both seismic and medical), combustion, isomerization and metal coating thickness assessment.

- Nonlinear approximation.

  These include antenna design, power transmission, maximum likelihood and robust regression.

- Nonlinear networks.

  Examples occur in traffic modelling, energy and water distribution/management systems, and neural networks. These problems can have hundreds of thousands of variables but they are tractable because of their very special structure.

- Other interesting problems occur in macro- and micro-economics, equilibrium calculations, production planning, energy scenario appraisal and portfolio analysis. These problems often give rise to quadratic programs, particularly in portfolio analysis.

The increasing interest in the solution of large-scale nonlinear optimization problems is also related to the realisation by users that today's advances in computer technology are making the solution of such problems possible.

Recent articles and books devoted primarily to large-scale optimization include [19], [20], [24], [33], [35], and [133]. Some examples of applications are given in [59], [87], [95], [96], [123] and [131]. Background material on nonlinear optimization is given in [54], [60], [61], [70] and [108]. An overview of what is involved in a mathematical programming system, especially with respect to linear problems, is given in [129].

3 What are the difficulties?

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 [108] for good surveys). As a consequence, it is just not adequate to take existing optimization software for small problems and apply it to large ones, hoping that the increased capacity in computing will take care of the growth in problem size. By contrast, we could expect that an efficient method for large-scale problems be at least moderately efficient for small-scale problems.

Perhaps the most important difficulty in the large-scale context is that of *exploiting structure*. The fact that we are able to solve large problems at all is because they are structured. Even in linear programming, a problem in one thousand variables devoid of structure (happily, usually an indication of a bad formulation) severely taxes codes. 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.

Another significant difficulty is that of *scaling*. Perhaps the main reason algorithms for small-scale problems do not necessarily translate into efficient algorithms for large-scale problems 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.

One consequence of the necessity to keep the algorithms for large-scale nonlinear optimization '*simple*' is that typically one has rather incomplete information available at every iteration. Thus it becomes difficult to successfully merging two distinct (and in many ways, conflicting) aspects of any nonlinear programming algorithm. The first aspect is that which guarantees global convergence. By global convergence (not to be confused with convergence to global optima), we mean convergence to a stationary point from any starting point. Essentially, this is a weak requirement (the method of steepest descent with a suitable line search condition will suffice) which combined with the desire to stress simplicity encourages us to use a steepest-descent-like method. On the other hand, ultimately we want faster convergence. This means using Newton's method or quasi-Newton methods, although in the case of large problems a fast linear rate might suffice.

From a more mundane point of view there are very real difficulties *inputting* large problems. In particular, the amount of information present in the structure of a large-scale optimization problem, although crucial for the acceptable performance of algorithms, is also very difficult to specify in a complete and understandable format. A standard input that is a simple formal language in which these structural concepts could be expressed unambiguously, has been rather well-established and successful in the more restricted domain of linear

programming [47]. We have extended this input format to the nonlinear case (see [36]). However, particularly because of the necessity of exploiting structure in a rather general sense, the resulting standard is not as simple as it was in the linear case. Other approaches are based upon using a high-level modelling language. The high-level aspect makes these rather more user friendly but they require an interpreter and are thus not normally in the public domain. Moreover, they do not, currently, exploit structure as generally as we would like. Well-known examples of this approach are GAMS ([14]) and AMPL ([65]).

Obviously, a primary requirement in evaluating the quality of an algorithm is to have a good set of *test problems*. In the first instance, it is by no means obvious what constitutes such a set for large-scale optimization, in the main because of the complexity of the problems and the lack of experience in solving them. Nevertheless, important starts have been made, and although we do not yet have collections as readily available as those for linear programming and sparse linear algebra (see [57] and [66]), we currently have over nine hundred problem instances in (our) 'standard input format' [40]. We are also asking for more test problems from the community ([125]). Moreover, as a part of the MINPACK-2 project 'a collection of significant optimization problems' is being made, see [4]. Earlier collections for unconstrained problems include [15] and [102].

Just as important is the ability to *evaluate results*. Given the number of variables (in both senses of the word) at hand, it is a complex task to interpret the results of testing. It is fair to say that, at present, we require more established test problems and a broader experience of the behaviour with various algorithms.

4 Current Approaches

Having considered the difficulties, we now examine how they are addressed. It is convenient to consider three broad classes, namely, approaches based upon classical large linear programming, approaches based upon small-scale nonlinear programming and approaches based upon a mixed linear programming/interior point method, even though the ideas in each approach are not mutually exclusive.

4.1 Approach based upon classical large linear programming

We will begin with a terse and somewhat eccentric summary of the simplex method. For those who need further details, an excellent recent survey article, that includes the interior point method that is relevant to the third approach below, is given in [73]. Consider the linear programming problem in the form

$$\underset{x \in \Re^n}{\text{minimize}} \quad c^T x \tag{4.5}$$

subject to the m $(\leq n)$ general linear constraints

$$Ax = b \tag{4.6}$$

and to the simple bound constraints

$$x \geq 0. \tag{4.7}$$

The solution to the above problem normally occurs at a vertex of the feasible region, that is a point defined by the equations (4.6) and $n-m$ of the variables lying on their bounds (4.7). Without loss of generality, we can assume that the last $n-m$ components of x are at their bounds. We call such variables *non-basic* — the remaining m variables are termed *basic*. We may thus consider the activities (i.e. those constraints satisfied as equalities) to be given by

$$Cx = \begin{pmatrix} B & N \\ & I \end{pmatrix} x = \begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{4.8}$$

Now

$$C^{-1} = \begin{pmatrix} B^{-1} & -B^{-1}N \\ & I \end{pmatrix}. \tag{4.9}$$

The fact that the k^{th} column of C^{-1} is orthogonal to the other $n-1$ rows of C, along with the fact that we start at a vertex of the feasible polytope and insist on following a path of objective-improving feasible vertices to optimality is really the heart of the simplex method. The first statement means that by moving along this k^{th} column the remaining equations corresponding to the other $n-1$ rows of C stay active. Being at a vertex ensures that we can refer to C^{-1}. Objective-improving is just a matter of sign and maintaining feasibility requires that we move to an adjacent vertex.

More importantly, from the point of view of this article, the method is efficient because it exploits heavily the structure of B, making use of techniques such as the Markowitz strategy, [93], and sparse Bartels-Golub, [6], updating of LU factors (see [117] for further details and the first implementation in Fortran). Another important feature of linear programming software is the ability to have crash starts, i.e. a relatively simple method for finding a good starting basis (see, for example [76]).

The highly successful package MINOS, [106] can be viewed as an extension of the simplex method as a reduced gradient technique. Its origins come from [118] and [119]. One should also note that for practitioners who are used to linear programming approaches, MINOS serves as an *extremely* useful bridge to nonlinear programming. In particular, MINOS replaces

$$\begin{array}{ll} \underset{x\in\mathbf{R}^n, y\in\mathbf{R}^m}{\text{minimize}} & F(x) + c^T x + d^T y \\ \text{subject to} & f(x) + A_1 y = b_1 \\ & A_2 x \ + A_3 y = b_2 \\ \text{and} & \\ & l_x \leq x \leq u_x \\ & l_y \leq y \leq u_y \end{array} \tag{4.10}$$

with

$$\begin{array}{ll} \text{minimize} & F(x) + c^T x + d^T y + \lambda_k^T (f(x) - \tilde{f}(x)) + \frac{1}{2}\rho(f(x) - \tilde{f}(x))^T (f(x) - \tilde{f}(x)) \\ x \in \mathbf{R}^n, y \in \mathbf{R}^m & \end{array}$$

subject to

$$\begin{aligned} \tilde{f}(x) + A_1 y &= b_1 \\ A_2 x + A_3 y &= b_2 \end{aligned}$$

and

$$\begin{aligned} l_x \leq x \leq u_x \\ l_y \leq y \leq u_y, \end{aligned}$$

where

$$\tilde{f}(x) = f(x_k) + J_k(x - x_k), \tag{4.11}$$

and J_k denotes the Jacobian of f evaluated at x_k. In other words, the nonlinear contribution to the constraints is linearized so that we can exploit linear programming technology. It then formulates a quadratic model for the artficial objective function. A reduced gradient technique is used, that is one determines a search direction that maintains the current activities to first-order (i.e. the linearized approximations that were active stay active). Writing the activities that are determined by the general linear constraints as

$$\hat{A}x = \begin{pmatrix} B & S & N \end{pmatrix} x = b, \tag{4.12}$$

this means that our search direction is given by

$$h = Zd, \tag{4.13}$$

where

$$Z^T = \begin{pmatrix} -[B^{-1}S]^T & I & 0 \end{pmatrix}. \tag{4.14}$$

This follows directly from the fact that

$$\hat{A}Z = 0 \;\text{ and }\; (0 \quad 0 \quad I)Z = 0. \tag{4.15}$$

Analogously to the simplex method, the columns of B correspond to basic variables and the columns of N correspond to non-basic variables. However, because of the nonlinearity of the objective function, we are no longer able to ensure that optima lie at vertices (the number of columns of B and N may not add up to the dimension of the space). The ensuing deficiences are made up by the columns of S, the so-called superbasic columns. Because of the similarities in the resulting linear algebra the exploitation of structure is much the same as that in the simplex method. It is worth pointing out that exploitation of the structure of Z and the simple bounds is especially attractive in the context of network problems (see, for example, [51], [82], [127] and [126]).

It should be clear that the fewer superbasic columns, the closer the problem is to a linear programming problem. MINOS works particularly well when there are relatively few superbasics.

A related approach that was one of the earliest successful pieces of software that could handle large nonlinear problems was an implementation of the generalised reduced gradient method of Abadie ([1]) by Lasdon ([90]). A quadratic

programming algorithm that uses similar ideas to MINOS and is for large-scale problems is given in [75].

Not surprisingly, the earliest approach to large-scale nonlinear optimization was a successive linear programming technique (see [81]). A more recent successive linear programming technique that uses an exact l_1 penalty function and incorporates trust region constraints is given in [64], although numerical results are given for small problems only.

4.2 Approaches based upon small-scale nonlinear programming

4.2.1 Sequential quadratic programming

One of the best known techniques for nonlinear programming is the so-called sequential quadratic programming approach (see, for example, [61], Chapter 12 and [70], Chapter 6). Recent work by Eldersveld [58] and colleagues uses the augmented Lagrangian. The vector s represents slack or surplus variables (see below for some motivation for this function and the introduction of slack or surplus variables)

$$f(x) - \lambda^T[c(x) - s] + [c(x) - s]^T[c(x) - s]/\mu, \tag{4.16}$$

with the quadratic programming search-direction subproblem

$$\begin{aligned} \underset{p,q}{\text{minimize}} \quad & \tfrac{1}{2}p^T H p + g^T p \\ & Ap - q \;=\; -[c(x_k) - s_k] \\ & \hat{l} \le \begin{bmatrix} p \\ q \end{bmatrix} \le \hat{u}, \end{aligned} \tag{4.17}$$

using a suitable symmetric matrix H, to solve (1.1) to (1.4). A protoype implementation has been developed that uses a modification of the MINOS code. They use for the active set

$$\hat{A} = \begin{pmatrix} B & S & N \\ 0 & 0 & I \end{pmatrix} \tag{4.18}$$

and solve

$$Z^T H Z y_z = -Z^T(g + Hp),$$

$$y = Z y_z,$$

where

$$Z^T = \begin{pmatrix} -[B^{-1}S]^T & I & 0 \end{pmatrix}$$

and $Z^T H Z$ is small. Noting that one needs H to evaluate the gradient of the quadratic objective function, they make use of the fact that if we define

$Q = [Z\ Y]$, choosing Z and Y so that $Z^T HY = 0$ and $Y^T HY = I$, then we can write $H = Q^{-T}(Q^T HQ)Q^{-1}$, where

$$Q^T HQ = \begin{bmatrix} Z^T HZ & 0 \\ 0 & I \end{bmatrix}.$$

For example, taking

$$Y = \begin{bmatrix} B^{-1} & 0 \\ 0 & 0 \\ 0 & I \end{bmatrix}$$

then

$$Q^{-1} = \begin{bmatrix} 0 & I & 0 \\ B & S & 0 \\ 0 & 0 & I \end{bmatrix}.$$

Details are given in [58]. An approach that also uses sequential quadratic programming, but 'solves' the quadratic program using an interior point method (see below) is given by [13]. Although these methods hold promise, computational experience to date has been insufficient to make definitive statements as to their effectiveness. Other sequential quadratic programming based methods include [92] and [105].

4.2.2 The LANCELOT project

The approach to which we wish to devote much of the rest of this article is based upon the adaptation of trust region methods to the problem with simple bounds. The method is extended to general constraints by using an augmented Lagrangian function and the bounds are handled directly via projections that are easy to compute. We use group partial separability (a generalisation of sparsity, introduced in [79]) to allow efficient storage and updating of matrices in matrix-vector product form. This approach has the further advantage that accurate approximations to the second derivatives of the element functions, normally being of low rank, are easier to obtain than for the assembled matrices. This structure is extremely general. Indeed, any sufficiently differentiable function with a sparse Hessian matrix may be written in this form. An introduction to group partial separability is given by [34]. The entire project has resulted in a substantial amount of software that is available at nominal cost for research purposes. There is also a book ([41]) to accompany the software. Returning to the underlying concepts of LANCELOT, we will now give some details.

Trust region methods in the context of unconstrained optimization have been able to combine a rather intuitive framework and robust numerical implementations with a powerful and elegant theoretical foundation. An excellent reference is [101]. The basic idea is to model the objective function (by a quadratic given by the first three terms of a Taylor's series expansion about the current point x^k, for example). One then 'trusts' this model in a neighbourhood (called the trust region) of x^k. The next step is to approximately minimize the model in the trust region, thereby obtaining a point $x^k + s^k$, say. One now determines how well the model actually predicted the change in the

true objective function. If good descent is obtained, the next iterate, x^{k+1}, is set to $x^k + s^k$ and the trust region is expanded. If moderate descent is obtained, the next iterate, x^{k+1} is set to $x^k + s^k$ and the trust region remains unchanged. Otherwise, x^{k+1} is set to x^k and the trust region is contracted. The beauty of such an approach is that, when the trust region is small enough and the problem smooth, the approximation is good, provided the model gradient is sufficiently accurate. Moreover, assuming one does at least as well as the minimum along the steepest descent direction of the model within the trust region (that determines the so-called Cauchy point), one can ensure convergence to a stationary point ([18]). In addition, eventually the trust region is expanded sufficiently that it does not interfere with the subsequent iterates, and thus, assuming that in this situation the underlying algorithm is sufficiently sophisticated, one can ensure fast asymptotic convergence. Details are given in [101].

The algorithm that is at the heart of LANCELOT is a method for which all the constraints are just simple bounds. The extension of the above ideas are relatively straightforward in this case. Essentially, one generalizes the Cauchy point to the minimum along the *projected* gradient path within the trust region, where the projection is with respect to the simple bounds. It is important to note that it is trivial computationally to compute such a projection: components of x that hit a bound just remain fixed. This approach was first carried out by McCormick in [94], and independently by Bertsekas [10] and Levitin and Polyak [91]. More recently it has been exploited extensively in the context of large-scale optimization by many authors, see for example [32], [52], [103], and [104]. As in the unconstrained case, global convergence can be guaranteed, provided one does at least as well as the generalized Cauchy point. One obtains better convergence, and ultimately a satisfactory asymptotic convergence rate, by further reducing the model function. In the context of LANCELOT, this is achieved by fixing the activities determined by the generalized Cauchy point and further reducing the model within the feasible region and trust region using just the remaining free variables. Updating of the trust region size is handled in exactly the same way as it is in the unconstrained case. The basic algorithm can be summarised as follows:

- Find the generalized Cauchy point based upon a local (quadratic) model.
- Fix activities to those at the generalized Cauchy point.
- Using the free variables further reduce the model within the feasible region and the trust region. Of course this may, and typically does, introduce new activities in addition to those determined by the generalized Cauchy point.
- Determine whether the current point is acceptable and update the trust region radius accordingly.

Provided the quadratic model is reasonable, we are able to prove that we converge to a Kuhn-Tucker point. Moreover, we identify the correct active constraints (activities) after a finite number of iterations assuming that strict complementarity is satisfied and the activities determined by the generalised

Cauchy point are kept active when the model is further reduced. Details are given in [31].

The extension to general constraints is carried out by means of an augmented Lagrangian function. In order to understand this extension we need to motivate this function. We have known for nearly two hundred years ([89], part 1, section 4, article 2), that a solution to

$$\begin{array}{ll} \underset{x\in\mathbf{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) = 0 \end{array} \tag{4.19}$$

is a feasible stationary point of the Lagrangian

$$f(x) - \lambda^T c(x). \tag{4.20}$$

It is only since the Second World War, that we have recognised ([48]) that we can solve (4.19) using the quadratic penalty function

$$\underset{x\in\mathbf{R}^n}{\text{minimize}} \quad f(x) + c^2(x)/\mu \tag{4.21}$$

as μ tends to zero from above. The idea here is that as μ becomes small the 'penalty term' $c^2(x)/\mu$ forces one to become feasible. This intuitive idea was not accorded a sound theoretical basis until the work of Fiacco and McCormick (see for example [60]). Augmented Lagrangians combine both ideas, thereby convexifying the Lagrangian and circumventing the necessity of requiring small μ by, instead, approximating the Lagrange multipliers, λ. Thus we use the problem

$$\underset{x\in\mathbf{R}^n}{\text{minimize}} \quad f(x) - \lambda^T c(x) + c^2(x)/\mu. \tag{4.22}$$

This approach was first suggested by K. J. Arrow and R. M. Solow in [3] but is better known through the work of Hestenes and Powell in [85] and [115].

In LANCELOT one thus solves the general problem by first introducing slack or surplus variables, if necessary, to change inequalities to equalities. Subsequently one minimizes the augmented Lagrangian

$$\Phi(x, \lambda, S, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x) + \frac{1}{2\mu} \sum_{i=1}^{m} s_{ii} c_i(x)^2 \tag{4.23}$$

(where the diagonal matrix S is introduced to incorporate scalings) subject to the explicit bounds, using the earlier algorithm[1]. This approach can be summarised as follows:

1. Test for convergence using the two following conditions.
   Sufficient stationarity — the projected gradient of the augmented Lagrangian with respect to the simple bounds is sufficiently small;
   Sufficient feasibility — the norm of the constraint violations is sufficiently small .

[1]It is worth noting that MINOS uses for its objective function an augmented Lagrangian function with corresponding constraints $f - \tilde{f} = 0$, whose relaxation can be considered as *trusting* the linear approximation to f.

2. Use the simple bounds algorithm to find a sufficiently stationary approximate minimizer of Φ (considered as a function of x only) subject to simple bounds.

3. If sufficiently feasible (both the 'local convergence' values here and in 2) are greater than the test for convergence, in general) update the multipliers and decrease the tolerances for stationarity and feasibility.

4. Otherwise, decrease the penalty parameter and reset tolerances for stationarity and feasibility.

We are able to show, under suitable conditions, that we converge to a first-order stationary point for the nonlinear programming problem. Furthermore, if we have a single limit point, we eventually stop reducing the penalty parameter, μ. Under somewhat stronger conditions we are able to show that one requires only a single iteration of the simple bounds algorithm to satisfy the conditions of the third item above. Details of these important properties are given in [38] and [42].

As we have already seen, a significant (and often dominant) cost in optimization is solving a linear system. Typically these arise from the necessity to determine an approximate stationary point for a quadratic function — equivalently, the necessity to solve a linear system whose coefficient matrix is a symmetric matrix. If the system is large there are two possible approaches. The first is to use direct methods based upon multifrontal techniques. These use partial assembly and dense matrix technology on sparse matrices. General details are given in [56] and an application in the context of LANCELOT is given in [43]. Our experience to date, however, has been that an iterative approach is more robust. The most popular such approach is preconditioned conjugate gradients. For ease of motivation we first consider conjugate gradients without a preconditioner. Directions, d_i, are called conjugate with respect to a positive definite matrix A if $d_1^T A d_2 = \langle d_1, d_2 \rangle_A = 0$. In other words, they are orthogonal in the A-metric. The best known conjugate set of vectors are the set of orthonormal eigenvectors. If one considers minimizing a strictly convex quadratic form $\frac{1}{2}x^T Ax - b^T x + c$, it is easy to see from the geometry that if one minimizes along the eigenvectors of A, then at each stage one determines the minimum of the quadratic on the space spanned by the eigenvectors used, and thus, after at most n steps, if A is n by n, the quadratic function's (unique) minimum is determined.

The appeal of conjugate-gradient methods is that this finite termination result for quadratics is true for general conjugate directions. Moreover, the attraction for large-scale optimization is that such an orthogonalisation can be determined via a three term recurrence and thus the method is particularly simple and only requires that we store three vectors.

However, if n is large, performing n steps may be prohibitively expensive. Moreover, a quadratic is only being used to model a nonlinear problem and so what we have is really a 'moving quadratic'. What makes this technique remain attractive is the use of preconditioners. The essential result is that whenever one has multiple eigenvalues, the conjugate gradient method *minimizes the*

quadratic in the space spanned by the corresponding eigenvectors. If we can cluster eigenvectors (i.e. approximately have multiple eigenvectors) we can reduce the number of iterations for good approximations to minimizers, from n to the number of clusters. The perfect way to do this in the quadratic case is to precondition with A^{-1} — but then this is equivalent to doing Newton's method. Surprisingly one can often do very well by using very crude approximations to A^{-1} (diagonal matrices, for instance). A good reference is [67].

It is fortunate that most optimization problems in thousands of variables are structured; fortunate but demanding of respect. If one considers the arrowhead matrix ($a_{i,1}, a_{1,i}, a_{i,i}$ non-zero, $i = 1, 2, \ldots n$, all other entries zero), it is clear that one neither wants to input or nor wants to store this matrix as a dense matrix, for large n. A little less obvious is the fact that if one does Gaussian elimination without pivoting, after the first column is updated the remaining $n-1$ by $n-1$ block will be full — in other words, fill-in is disastrous. On the other hand, if we first reverse the ordering of the rows and columns (which amounts to changing the orderings of the equations and the labellings of the variables) there is no fill-in at all. Since optimal numerical stability typically dictates row and column orderings and limitations on storage motivate one to minimize fill-in, we are immediately aware of a major conflict in numerical linear algebra when one wants to account for structure. One form of compromise is known as threshold pivoting (see, for example, [56]).

In LANCELOT we take the point of view that invariant subspaces are more important than sparsity. For example, consider $f(x) = x_{50}^4$, and $F(x) = \left(\sum_{i=1}^{5,000,000} x_i\right)^4, x \in \mathbf{R}^{5,000,000}$. In the first case the Hessian is sparse, while in the second case the Hessian is dense. But they both have an invariant subspace of dimension $n-1$. In the first case, it is the orthogonal complement of e_{50} (the vector with a single non-zero entry, one, in the fiftieth component), and in the second case it is the orthogonal complement of e (the vector of all ones). We exploit invariant subspaces by writing our functions as $f(x) = \sum_{i=1}^{m} g_i \left(l_i(x) + \sum_{j \in I_i} w_j f_j(x)\right)$, where g_i is a scalar function, the l_i are linear functions and the w_j are weights for the nonlinear functions $f_j(x)$. The essential point is that the rank of $\nabla_{xx} f_j$) is much smaller than n and the null space of $\nabla_{xx} f_j$ is fixed. This, and the use of linear transformations to consider expressions like $e^{(x+y)}$ as $e^u, u = x + y$, enables us to use very compact representations of the problems we are optimizing. In particular we can store these as *dense* matrices. We note, however, that it is no longer reasonable to expect these matrices to be positive definite. This has led to a revival of interest in rank one secant methods. For an introduction to these considerations, with many more details, the reader is urged to read [34]. A related means of exploiting structure is given in [96]. For recent work on rank one updating see [37], [55], [88], [111], and [132].

Although we are unaware of the details, an augmented Lagrangian approach that is designed for large-scale optimization has been developed by Contesse, [46]. There are some similarities between the approach of [38], [45] and [64]. Fletcher and Sainz de la Maza use a piecewise linear model and an l_1 penalty function for the merit function but improvements over the Cauchy point involve

Problem	n	m
1-d nonlinear boundary value problem	5002	0
Economic model from Thailand	2230	1112
1-d variational problem from ODEs	1001	0
2-d variational problem from PDEs	5184	0
3-d variational problem from PDEs	4913	0
Nonlinear network gas flow problem	2734	2727
Chemical reaction problem	5000	5000
Oscillation problem in structural mechanics	5041	0
Nonlinear optimal control problem	9006	7000
Maximum pivot growth in Gaussian Elimination	3946	3690
Nonlinear network problem on a square grid	13284	6724
Nonlinear optimal control problem	10001	5000
Hydro-electric reservoir management	2017	1008
Minimum surface problem with nonlinear boundary conditions	15625	0
Economic equilibrium	1825	730
Nonlinear optimal control problem	7011	5005
Orthogonal regression problem	8197	4096
Analysis of semiconductors	1002	1000
Elastic-plastic torsion problem	14884	0

Figure 1: Some typical examples.

projected Hessian approximations.

Current work in LANCELOT has been to consider the special case of convex constraints. The motivation is that, in particular, linear constraints are very common, are too simple to handle effectively in the same manner as we handle general constraints, but are nevertheless too complicated to handle the projections easily in the context of large problems. We use a combination of a simple piecewise linear line search (with only two pieces) and the trust region approach. The projections are handled approximately, but the approximation has to be 'good enough'. Details are given in [45]. In particular, a special case gives a convergence proof for sequential linear programming in the case of convex constraints. In the case of nonlinear networks, Sartenaer, in [122], has obtained some very encouraging numerical results.

In addition we have a test-bed of around nine hundred problems written in our standard data format. We have solved almost all these examples, many of which are of a substantial size. We expect to have them available electronically via netlib or something similar some time this year (1992). Some typical examples of applications we have solved using LANCELOT are tabulated above, in Figure 1. Detailed analysis of these results and our interpretation will be reported separately.

What about other current work on LANCELOT and the future? As an alternative to the augmented Lagrangian we are implementing a Lagrangian barrier method [44], that is closely related to the modified and shifted barrier

function method (see, for example, [68], and [114]). We use the Lagrangian barrier function

$$\Psi(x, \lambda, s) = f(x) - \sum_{i=1}^{m} \lambda_i s_i \log(s_i - c_i(x)),$$

which we can then optimize with respect to the simple bounds. A possible choice is $s_i = \mu \lambda_i^{\alpha_\lambda}$, where μ is a penalty parameter and $0 < \alpha_\lambda \leq 1$.

We also would like to exploit group partial separability at a more fundamental level in our trust region algorithm. We have been able to use the structure of the problem explicitly in the definition of the trust region and have suitable convergence properties [39] and we are currently investigating the computational implications. In order to exploit linear constraints more successfully we are investigating interior point methods. Finally, we always need to perform more testing.

4.3 Approach based upon a mixed linear programming/interior point method

Although not strictly speaking interior point methods, we have already mentioned modified barrier methods above. More generally, interior point methods, although originally developed for nonlinear programming, have had a spectacular success in the context of linear programming and have thereby generated new interest in their use in nonlinear optimization. See [134], for example, for background material. Encouraging results have already been obtained in the context of linear-like problems (see [23], [25], [26], [100] and [120]) and quadratic programming ([2], [9], [12], [22] [72], [83], [84], [99], [112], and [137]). Nash and Sofer have computational experience with a barrier method applied to a thousand variable nonlinear problem with bound constraints, [107]. Work in convex programming includes [5], [53], [86], [97], [98], [135] and [136]. It seems reasonable to expect further developments within nonlinear programming, especially since these techniques appear to be especially appropriate for large-scale problems.

4.4 Other issues

There is obviously a number of issues that are relevant to our subject, but that fall slightly out of the context of the current paper. We now briefly mention some of the most important ones.

Both primal and dual degeneracy are intrinsic difficulties in that they are often a manifestation of the problem that can be troublesome to the method of solution. By primal degeneracy, we mean that the dual variables are not uniquely defined. By dual degeneracy, we mean that some of the dual variables are zero. Moreover degeneracy, especially primal degeneracy, is not a rare occurrence. Revelent work includes [17], [50], [62], [63], [69] and [121]. This is an important area in which there is a need for further research.

The primary level of formulation is clearly important. Augmented Lagrangians are rather different from barrier functions. If one exploits the structure using group partial separability this has a profound effect on the design

of the software and the algorithmic techniques used. Trust region methods typically make different demands from line search approaches. Small scale sequential quadratic programming techniques do not have much in common with the approach taken in LANCELOT.

Parallelism is starting to have its effect on optimization. Firstly, numerical linear algebra plays a fundamental role in optimization. Recently there has been much work implementing such methods on advanced architectures. An overview of many of the major issues is given in [74]. A lucid description, illustrated by considering the Cholesky factorization, is given in [130]. Secondly, parallelism can be exploited at the basic level of the optimization algorithm. Simple examples are the computation of 'extra' quantities in parallel (for example, speculative steps), or independent runs with different choices for some of the algorithm parameters. Often the most effective algorithms for very large problems are those which are very simple but not efficient if implemented in a sequential environment. However, with the possibility of intelligently running what amount to several instances in parallel, a much more effective algorithm results.

We think that although they are not scale invariant, truncated Newton methods are especially important in the context of large-scale nonlinear optimization. When incorporated with an iterative technique such as preconditioned conjugate directions, they enable us to handle the difficulty of deciding whether to solve inner iterations accurately or inaccurately, with the limited information available, whilst still being able to ensure a satisfactory asymptotic convergence rate and global convergence. For example, it is not easy to see how to achieve the same ends in the context of sequential quadratic programming using active set strategies.

Anyone who has tried inputting significant problems appreciates the potential of automatic differentiation. Recently, this has become a very active area of research (see, for example, [77] and [78]), and efforts are underway to provide automatic differentiation tools that promise to make this technology readily usable to the optimization community ([11] and [80]).

After the great success of quasi-Newton methods there was much hope that such techniques could be adapted to the manipulation of large Hessian approximations. Unfortunately, this turned out not to be the case and the results have been disappointing (see [124]). On the other hand, sparse finite difference schemes have been successful. Since the pioneering work of Curtis, Powell and Reid [49] there has been significant progress, some of which exploits parallelism (see [21], [27], [28], [29], [30], [71] [113], and [116]).

An alternative approach is to use limited memory quasi-Newton updating. This uses the information of only a few, most recent, steps to define a variable metric approximation to the Hessian (see, for example [16] and [110]). These methods have proved to be very useful for solving certain large unstructured problems and Nocedal, [109], claims it is competitive with the partitioned quasi-Newton method on partially separable problems in which the number of element variables exceeds five or six, at least when the cost of evaluating the objective function is relatively low. Recently Bartholomew-Biggs and Hernandez ([7]) have been able to solve large problems using limited memory approxima-

tions to the inverse of the Lagrangian in the sequential quadratic programming framework of [8].

We should also mention that the current familiarity of users with sophisticated computer environments has created a high expectation for a user friendly interface to software. Unfortunately, developers have been too busy, as yet, coping with the algorithmic complexities to have devoted much time to the important practicalities of a first rate interface.

4.5 In conclusion

We would like to emphasize that it is possible to solve large nonlinear constrained problems in thousands of variables in acceptable time on reasonable workstations. At least two software packages, LANCELOT and MINOS, are available. Input is important, and significant progress in both modelling languages and a standard input format, have been made.

This is a vibrant, challenging and useful research area. Our hope is that, in the not too distant future, practitioners will be solving nonlinear models rather than linear ones, when the former is the most appropriate one to consider. Prefering to solve linear models, only because we understand how to solve large linear programs, should no longer be the normal practice.

References

[1] J. Abadie and J. Carpentier. Généralisation de la méthode du gradient réduit de Wolfe au cas des contraintes non-linéaires. In D.B. Hertz and J. Melese, editors, *Proceedings IFORS Conference*, pages 1041–1053, Amsterdam, 1966. John Wiley.

[2] K. M. Anstreicher, D. den Hertog, C. Roos, and T. Terlaky. A long step barrier method for convex quadratic programming. Technical Report 90-53, Faculty of Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands, 1990.

[3] K. J. Arrow and R. M. Solow. Gradient methods for constrained maxima with weakened assumptions. In L. Hurwicz Arrow and Hirofumi Uzawa, editors, *Studies in Linear and Nonlinear Programming*, Stanford, CA, 1958. Stanford University Press.

[4] B. M. Averick, R. G. Carter, and J. J. Moré. The MINPACK-2 test problem collection. Technical Report ANL/MCS-TM-150, Applied Mathematics Division, Argonne National Labs, Argonne, IL, 1991.

[5] O. Bahn, J. L. Goffin, J. P. Vial, and O. Du Merle. Implementation and behaviour of an interior point cutting plane algorithm for convex programming: an application to geometric programming. Technical report, GERARD, Faculty of Management, McGill University, McGill University, Montreal, 1991.

[6] R. H. Bartels and G. H. Golub. The simplex method of linear programming using the LU decomposition. *Communications of the ACM*, 12:266–268, 1969.

[7] M. C. Bartholomew-Biggs and M. de F. G. Hernandez. Some improvements to the subroutine OPALQP for dealing with large problems. Technical report, Numerical Optimization Center, Hatfield Polytechnic, Hatfield, UK, 1992.

[8] M.C. Bartholomew-Biggs. Recursive quadratic programming methods based on the augmented Lagrangian function. *Mathematical Programming Study*, 31:21–42, 1987.

[9] M. Ben Daya and C. M. Shetty. Polynomial barrier function algorithm for convex quadratic programming. Report J85-5, School of ISE, Georgia Institute of Technology, Atlanta, Georgia, 1988.

[10] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM J. Control Optim.*, 20:221–246, 1982.

[11] C. Bischof, A. Carle, G. Corliss, P. Hovland, and A. O. Griewank. ADIFOR: Generating derivative codes from Fortran programs. Technical Report MCS-P263-0991, Argonne National Labs, Argonne, IL, 1991.

[12] P. T. Boggs, P. D. Domich, J. E. Rogers, and C. Witzgall. An interior point method for linear and quadratic programming problems. Mathematical Programming Society COAL Newsletter, August 1991.

[13] P. T. Boggs and J. W. Tolle. A truncated SQP algorithm for large scale nonlinear programming problems, January 1992. Presented at the sixth IIMAS-UNAM workshop on numerical analysis and optimization.

[14] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: a User's Guide*. The Scientific Press, Redwood City, USA, 1988.

[15] A. G. Buckley. Test functions for unconstrained minimization. Technical Report CS-3, Computing Science Division, Dalhousie University, Dalhousie, Canada, 1989.

[16] A. G. Buckley and A. LeNir. QN-like variable storage conjugate gradients. *Math. Prog.*, 27:155–175, 1983.

[17] S. Busovača. Handling degeneracy in a nonlinear l_1 algorithm. Technical Report Tech. Rept. CS-85-34, Univ. of Waterloo, Dept. of Computer Science, Univ. of Waterloo, Waterloo, Ontario N2L 3G1, 1985.

[18] A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultannées. *Comptes Rendus de l'Académie des Sciences*, pages 536–538, 1847.

[19] T. F. Coleman. Large Sparse Numerical Optimization. In *Lecture Notes in Computer Science #165*. Springer-Verlag, Berlin, 1984.

[20] T. F. Coleman. Large Scale Numerical Optimization: Introduction and overview. In *Encyclopedia of Computer Science and Technology*. Marcel Dekker, Inc., New York, 1992 (to appear).

[21] T. F. Coleman and J-Y. Cai. The cyclic coloring problem and estimation of sparse Hessian matrices. *SIAM J. Appl. Math.*, 20:187–209, 1983.

[22] T. F. Coleman and L. Hulbert. A globally and superlinearly convergent algorithm for convex quadratic programs with simple bounds. Technical Report TR 90-1092, Department of Computer Science, Cornell University, Ithaca, NY, 1990. To appear in SIAM Journal on Optimization.

[23] T. F. Coleman and Y. Li. A global and quadratic affine scaling method for (augmented) linear l_1 problems. In G.A. Watson, editor, *Proceedings of the 13th Biennial Numerical Analysis Conference Dundee 1989*. Longmans, 1989.

[24] T. F. Coleman and Y. Li, editors. *Large-Scale Numerical Optimization*. SIAM, Philadelphia, 1990.

[25] T. F. Coleman and Y. Li. A global and quadratic affine scaling method for linear l_1 problems. *Mathematical Programming*, 1992 (to appear).

[26] T. F. Coleman and Y. Li. A global and quadratically convergent method for linear l_∞ problems. *SIAM Journal on Optimization*, 1992 (to appear).

[27] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20:187–209, 1983.

[28] T. F. Coleman and J. J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Mathematical Programming*, 28:243–270, 1984.

[29] T. F. Coleman and J. J. Moré. Software for estimating sparse Jacobian matrices. *TOMS*, 10:329–345, 1984.

[30] T. F. Coleman and P. E. Plassman. Solution of nonlinear least-squares problems on a multiprocessor. *SIAM Journal on Scientific and Statistical Computing*, 13, 1992.

[31] 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*, 25:433–460, 1988. See also *SIAM Journal on Numerical Analysis,* 26:764-767, 1989.

[32] 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*, 50:399–430, 1988.

[33] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, editors. *Large-Scale Optimization*, volume 45, no. 3 of *Mathematical Programming*. North-Holland, 1989.

[34] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. An introduction to the structure of large scale nonlinear optimization problems and the lancelot project. In R. Glowinski and A. Lichnewsky, editors, *Computing Methods in Applied Sciences and Engineering*, pages 42–51, Philadelphia, 1990. SIAM.

[35] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, editors. *Large-Scale Optimization — Applications*, volume 48, no. 1 of *Mathematical Programming*. North-Holland, 1990.

[36] 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. Report CS-89-61, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, 1990.

[37] 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*, 50:177–195, 1991.

[38] 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. *SIAM Journal on Numerical Analysis*, 28:545–572, 1991.

[39] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Convergent properties of minimization algorithms for convex constraints using a structured trust region. Technical report, Department of Mathematics, FUNDP, Namur, Belgium, 1992.

[40] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: a collection of constrained and unconstrained test examples for nonlinear programming, 1992.

[41] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization*. Springer-Verlag, Berlin, Heidelberg and New York, 1992.

[42] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. On the number of inner iterations per outer iteration of a globally convergent algorithm for optimization with general nonlinear constraints and simple bounds. In D. F. Griffiths and G.A. Watson, editors, *Proceedings of the 14th Biennial Numerical Analysis Conference Dundee 1991*. Longmans, 1992 (to appear).

[43] A. R. Conn, N. I. M. Gould, M. Lescrenier, and Ph. L. Toint. Performance of a multifrontal scheme for partially separable optimization. In S. Gómez, J.P. Hennart, and R.A. Tapia, editors, *Proceedings of the Sixth Mexico-United States Numerical Analysis Workshop*, Philadelphia, 1992 (to appear). SIAM.

[44] A. R. Conn, N. I. M. Gould, R. Polyak, and Ph. L. Toint. A globally convergent Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. Technical report, IBM Thomas J. Watson Research Center, P.O.Box 218, Yorktown Heights, NY 10598, U.S.A., 1992.

[45] 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. *SIAM Journal on Optimization*, 1992 (to appear).

[46] L. Contesse and J. Villavicencio. Resolución de un modelo económico de despacho de carga eléctrica mediante el método de penalización Lagrangeana con cotas. *Revista del Instituto Chileno de Investigación Operativa*, pages 80–112, 1982.

[47] International Business Machines Corporation. Mathematical programming system/360 version 2, linear and separable programming-user's manual. Technical Report H20-0476-2, IBM Corporation, 1969.

[48] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc.*, 49:1–23, 1943.

[49] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *IMA J. Appl. Math.*, 13:117–120, 1974.

[50] A. Dax. A note on optimality conditions for the Euclidean multifacility location problem. *Math. Prog.*, 36:72–80, 1986.

[51] R. S. Dembo and J. G. Klincewicz. A scaled reduced gradient algorithm for network flow problems with convex separable costs. *Mathematical Programming*, 15:125–147, 1981.

[52] R. S. Dembo and U. Tulowitski. On the minimization of quadratic functions subject to box constraints. Technical Report B71, Yale School of Management, Yale University, New Haven, USA, 1983.

[53] D. den Hertog, C. Roos, and T. Terlaky. A potential reduction method for a class of smooth convex programming problems. Technical Report 90-01, Faculty of Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands, 1990.

[54] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1983. Russian edition, Mir Publishing Office, Moscow, 1988, O. Burdakov, translator.

[55] J. E. Dennis Jr. and H. Wolkowicz. Sizing and least-change secant methods. *SIAM J. Numer. Anal.*, 1992 (to appear).

[56] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices.* Clarendon Press, Oxford, UK, 1986.

[57] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15:1–14, 1989.

[58] S. K. Eldersveld. *Large-scale Sequential Quadratic Programming Algorithms*. PhD thesis, Department of Operations Research, Stanford University, Stanford, CA, 1991. Also available as a technical report.

[59] J. E. Falk and G. P. McCormick. Computational aspects of the international coal trade model. In P.T. Harker, editor, *Spacial price equilibrium: Advances in theory, computation and application, Lecture Notes in Economics and Mathematical Systems #249*, pages 73–117. Springer-Verlag, Berlin, Heidelberg and New York, 1986.

[60] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, NY, 1968. Reprinted as *Classics in Applied Mathematics 4*, SIAM, 1990.

[61] R. Fletcher. *Practical Methods of Optimization*. J. Wiley and Sons, Chichester, second edition, 1987.

[62] R. Fletcher. Degeneracy in the presence of roundoff errors. *Linear Algebra and its Applications*, 106:149–183, 1988.

[63] R. Fletcher. Resolving degeneracy in quadratic programming. Technical Report NA/135, Department of Mathematical Sciences, University of Dundee, 1991.

[64] R. Fletcher and E. Sainz de la Maza. Nonlinear programming and nonsmooth optimization by successive linear programming. *Math. Prog.*, 43:235–256, 1989.

[65] R. Fourer, D. M. Gay, and B. W. Kernighan. AMPL: A mathematical programming language. Computer science technical report, AT&T Bell Laboratories, Murray Hill, USA, 1987.

[66] D. M. Gay. Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Newsletter, December 1985.

[67] P. E. Gill and W. Murray. Conjugate-gradient methods for large-scale nonlinear optimization. Technical Report SOL79-15, Operations Research Department, Stanford University, Stanford, USA, 1979.

[68] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Shifted barrier methods for linear programming. Technical Report SOL88-9, Operations Research Department, Stanford University, Stanford, USA, 1988.

[69] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. A practical anti-cycling procedure for linearly constrained optimization. *Mathematical Programming*, 45(3):437–474, 1989.

[70] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, New York, London, Toronto, Sydney and San Francisco, 1981.

[71] D. Goldfarb and Ph. L. Toint. Optimal estimation of Jacobian and Hessian matrices that arise in finite difference calculations. *Math. Comp.*, 43:69–88, 1984.

[72] D. Goldfarb and S. Liu. An $O(n^3L)$ primal interior point algorithm for convex quadratic programming. *Math. Prog.*, 49:325–343, 1991.

[73] D. Goldfarb and M. J. Todd. Linear programming. In G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, chapter 2. North-Holland, Amsterdam, 1989.

[74] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, second edition, 1989.

[75] N. I. M. Gould. An algorithm for large-scale quadratic programming. *IMA Journal of Numerical Analysis*, 11:299–324, 1991.

[76] N. I. M. Gould and J. K. Reid. New crash procedures for large systems of linear constraints. *Mathematical Programming*, 45:475–502, 1989.

[77] A. O. Griewank. Direct calculation of Newton steps without accumulating Jacobians. In T. F. Coleman and Yuying Li, editors, *Large-Scale Numerical Optimization*, pages 115–137. SIAM, Philadelphia, 1990.

[78] A. O. Griewank and G. F. Corliss. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*. SIAM, Philadelphia, 1991.

[79] A. O. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M.J.D. Powell, editor, *Nonlinear Optimization*. Academic Press, London, 1982.

[80] A. O. Griewank, D. Juedes, J. Srinivasan, and C. Tyner. ADOL-C, a package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Software*, to appear. Also appeared as Preprint MCS–P180–1190, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439, 1990.

[81] R. E. Griffith and R. A. Stewart. A nonlinear programming technique for the optimization of continuous processing systems. *Manage. Sci.*, 7:379–392, 1961.

[82] M. D. Grigoriadis. An efficient implementation of the network simplex method. *Mathematical Programming*, 26:83–111, 1986.

[83] C-G. Han, P. M. Pardalos, and Y. Ye. Computational aspects of an interior point algorithm for quadratic programming problems with box constraints. In T.F. Coleman and Y. Li, editors, *Large-Scale Numerical Optimization*, pages 92–102, Philadelphia, 1990. SIAM.

[84] C-G. Han, P. M. Pardalos, and Y. Ye. Solving some engineering problems using an interior-point algorithm. Technical Report CS-91-04, Computer Science Department, The Pennsylvania State University, University Park, PA, 1991.

[85] M. R. Hestenes. Multiplier and gradient methods. *J. Optim. Theory Appl.*, 4:303–320, 1969.

[86] F. Jarre and M. A. Saunders. Practical aspects of an interior-point method for convex programming. Technical Report SOL91-9, Department of Operations Research, Stanford University, Stanford, USA, 1991.

[87] A. P. Jones. The chemical equilibrium problem: An application of SUMT. Technical Report RAC-TP-272, Reserach Analysis Corporation, McLean, Virginia, 1967.

[88] H. Khalfan, R. H. Byrd, and R. B. Schnabel. A theoretical and experimental study of the symmetric rank one update. *SIAM J. on Optimization*, 1992 (to appear).

[89] J. L. Lagrange. *Théorie des Fonctions Analytiques.* Impr. de la République, Paris, 1797.

[90] L. S. Lasdon, A. D. Waren, A. Jain, and M. Ratner. Design and testing of a generalized reduced gradient code for nonlinear programming. *TOMS*, 4:34–50, 1978.

[91] E.S. Levitin and B.T. Polyak. Constrained minimization problems. *USSR Comput. Math. and Math. Phys.*, 6:1–50, 1966.

[92] D. Mahidhara and L. Lasdon. An SQP algorithm for large sparse nonlinear programs. Working paper, MSIS Department, School of Business, University of Texas, Austin, TX 78712, 1990.

[93] H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Sci.*, 3:255–269, 1957.

[94] G. P. McCormick. Anti-zig-zagging by bending. *Manage. Sci.*, 15:315–320, 1969.

[95] G. P. McCormick. Computational aspects of nonlinear programming solutions to large-scale inventory problems. Technical Report Technical Memorandum Serial T-63488, George Washington University, Institute of Management Science and Engineering, Washington,DC, 1972.

[96] G. P. McCormick and A. Sofer. Optimization with unary functions. *Mathematical Programming*, 52:167–179, 1991.

[97] S. Mehrotra and J. Sun. An interior point algorithm for solving smooth convex programs based on Newton's method. In J.C. Lagarias and M.J. Todd, editors, *Contemporary Mathematics*, pages 265–284. AMS, Providence, Rhode Island, 1990.

[98] R. C. Monteiro. The global convergence of a class of primal potential reduction algorithms for convex programming. Report, Systems and Industrial Engineering Department, University of Arizona, Tucson, Arizona, 1991.

[99] R. C. Monteiro and I. Adler. Interior path following primal-dual algorithms, part ii: convex quadratic programming. Report, Department of IEOR, University of California, Berkeley, California, 1987.

[100] R. C. Monteiro, I. Adler, and M. G. Resende. A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. Report ESRC 88-8, Department of IEOR, University of California, Berkeley, California, 1987.

[101] J. J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: The State of the Art*, pages 258–287, Berlin, 1983. Springer Verlag.

[102] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[103] J. J. Moré and G. Toraldo. Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, 14:14–21, 1979.

[104] J. J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1:93–113, 1991.

[105] W. Murray and F. J. Prieto. A sequential quadratic programming algorithm using an incomplete solution of the subproblem. Technical Report SOL90-12, Operations Research Department, Stanford University, Stanford, USA, 1990.

[106] B. A. Murtagh and M. A. Saunders. MINOS 5.1 user's guide. Technical Report SOL83-20R, Department of Operations Research, Stanford University, Stanford, USA, 1987.

[107] S. G. Nash and A. Sofer. A barrier method for large-scale constrained optimization. Technical Report 91-10, Department of Operations Research and Applied Statistics, George Mason University, Fairfax, Virginia 22030, 1991.

[108] G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors. *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*. North-Holland, Amsterdam, 1989.

[109] J. Nocedal. Theory of algorithms for uncontrained optimization. *Acta Numerica*, 1, 1992 (to appear).

[110] J. Nocedal and D. C. Liu. On the limited memory BFGS method for large scale optimization. *Mathematical Programming, Series B*, 45:503–528, 1989.

[111] M. R. Osborne and L. P. Sun. A new approach to the symmetric rank-one updating algorithm. *Math. Prog.*, 1992 (to appear).

[112] P. M. Pardalos, C-G. Han, and Y. Ye. Interior point algorithms for solving nonlinear optimization problems. Mathematical Programming Society COAL Newsletter, August 1991.

[113] P.E.Plassman. Sparse Jacobian estimation and factorization on a multiprocessor. In T.F. Coleman and Y. Li, editors, *Large-Scale Numerical Optimization*, pages 152–179, Philadelphia, 1990. SIAM.

[114] R. Polyak. Modified barrier functions (theory and methods). Research report RC 15886 (#70630), IBM T. J. Watson Research Center, P.O.Box 218, Yorktown Heights, NY 10598, U.S.A., 1990.

[115] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, NY, 1969.

[116] M. J. D. Powell and Ph. L. Toint. On the estimation of sparse Hessian matrices. *SIAM J. Numer. Anal.*, 16:1060–1074, 1979.

[117] J. K. Reid. Fortran subroutines for handling sparse linear programming base. Technical Report AER-R-8269, AERE Harwell Laboratory, Harwell, UK, 1976.

[118] S. M. Robinson. A quadratically convergent algorithm for general nonlinear programming problems. *Math. Prog.*, 3:145–156, 1972.

[119] J. B. Rosen and J. Kreuser. A gradient projection algorithm for nonlinear constraints. In F. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*, pages 297–300. Academic Press, New York, NY, 1972.

[120] A. S. Ruzinsky and E. T. Olson. l_1 and l_∞ minimization via a variant of Karmarkar's algorithm. *IEEE Trans. Acoustics, Speech and Signal Processing*, 37:245–253, 1989.

[121] D. M. Ryan and M. R. Osborne. On the solution of highly degenerate linear programs. *Mathematical Programming*, 41:385–392, 1988.

[122] A. Sartenaer. *On some strategies for handling constraints in nonlinear optimization.* PhD thesis, Department of Mathematics, FUNDP, Namur, Belgium, 1991.

[123] D. A. Schrady and U. C. Choe. Models for multi-item continuous review inventory policies subject to constraints. *Naval Research Logistics Quarterly*, 18:451–463, 1971.

[124] D. C. Sorensen. An example concerning quasi-Newton estimation of a sparse Hessian. *SIGNUM Newsletter*, 16:8–10, 1981.

[125] Ph. L. Toint. Call for test problems in large scale nonlinear optimization. *COAL Newsletter*, 16:5–10, 1987.

[126] Ph. L. Toint and D. Tuyttens. On large scale nonlinear network optimization. *Mathematical Programming, Series B*, 48(1):125–159, 1990.

[127] Ph. L. Toint and D. Tuyttens. LSNNO: a Fortran subroutine for solving large scale nonlinear network optimization problems. *ACM Transactions on Mathematical Software*, 1990 (to appear).

[128] J. A. Tomlin. The influences of algorithmic and hardware developments on computational mathematical programming. In M. Iri and K. Tanabe, editors, *Mathematical Programming— Recent Developments and applications*, pages 159–175, Tokyo, 1988. KTK Scientific Publishers.

[129] J. A. Tomlin and J. S. Welch. Mathematical programming systems. Technical Report RJ 7400 (69202), IBM Thomas J. Watson Research Center, P.O.Box 218, Yorktown Heights, NY 10598, U.S.A., 1990.

[130] C. F. Van Loan. A survey of matrix computations. Technical Report CTC90TR26, Cornell Theory Center, Cornell University, Ithaca, NY, 1990.

[131] P. Werbos. Backpropagation: past and future. In *Proceedings of the 2nd International Conference on Neural Networks*, New York, 1988. IEEE.

[132] H. Wolkowicz. Measures for symmetric rank-one updates. *Math. Prog.*, 1992 (submitted).

[133] M. H. Wright. Optimization and large scale computation. In J.P. Mesirov, editor, *Very Large Scale Computation in the* 21^{st} *Century*, Philadelphia, 1991. SIAM.

[134] M. H. Wright. Interior methods for constrained optimization. *Acta Numerica*, 1, 1992 (to appear).

[135] S. J. Wright. An interior-point algorithm for linearly constrained optimization. Technical Report MCS-P162-0790, Argonne National Labs, Argonne, IL, 1990.

[136] Y. Ye. *Interior point algorithms for linear, quadratic and linearly constrained convex programming.* PhD thesis, Engineering and Economic Systems Department, Stanford University, Stanford, CA, 1987.

[137] Y. Ye. Interior point algorithms for quadratic programming. In S. Kumar (to appear), editor, *Recent Developments in Mathematical Programming.* Gordan and Breach Scientific Publishers, London, 1992.