Mathematical Programming Study 16 (1982) 84–117
North-Holland Publishing Company

A PROJECTED LAGRANGIAN ALGORITHM AND ITS IMPLEMENTATION FOR SPARSE NONLINEAR CONSTRAINTS†

Bruce A. MURTAGH
The University of New South Wales, Sydney, N.S.W., Australia

Michael A. SAUNDERS*
Stanford University, Stanford, CA, U.S.A.

Received 29 January 1980
Revised manuscript received 2 February 1981

An algorithm is described for solving large-scale nonlinear programs whose objective and constraint functions are smooth and continuously differentiable. The algorithm is of the projected Lagrangian type, involving a sequence of sparse, linearly constrained subproblems whose objective functions include a modified Lagrangian term and a modified quadratic penalty function.

The algorithm has been implemented in a general purpose FORTRAN program called MINOS/AUGMENTED. Some aspects of the implementation are described, and computational results are given for some nontrivial test problems.

The system is intended for use on problems whose Jacobian matrix is sparse. (Such problems usually include a large set of purely linear constraints.) The bulk of the data for a problem may be assembled using a standard linear-programming matrix generator. Function and gradient values for nonlinear terms are supplied by two user-written subroutines.

Future applications could include some of the problems that are currently being solved in industry by the method of successive linear programming (SLP). We would expect the rate of convergence and the attainable accuracy to be better than that achieved by SLP, but comparisons are not yet available on problems of significant size.

One of the largest nonlinear programs solved by MINOS/AUGMENTED involved about 850 constraints and 4000 variables, with a nonlinear objective function and 32 nonlinear constraints. From a cold start, about 6000 iterations and 1 hour of computer time were required on a DEC VAX 11/780.

Key words: Large-scale Optimization, Optimization Software, Nonlinear Programming, Projected Lagrangian.

1. Introduction

The work reported here was prompted by consideration of various ways to extend the linearly constrained optimization code MINOS [35, 36, 52] to prob-

† Presented at the Tenth International Symposium on Mathematical Programming, Montréal, August 1979.

* Supported by the Department of Energy Contract DE-AS03-76SF00326, PA DE-AT-03-76ER72018; the National Science Foundation Grants MCS-7926009 and ENG-7706761 A01; the U.S. Army Research Office Contract DAAG29-79-C-0110; and the Applied Mathematics Division, DSIR, New Zealand.

lems containing both linear and nonlinear constraints. In particular, we are concerned with large, sparse problems, in the sense that each variable is involved in relatively few constraints.

Ignoring sparsity for the moment, consider the model problem

$$\begin{aligned} &\text{minimize} \quad f^0(x), \\ &\text{subject to} \quad f(x) = \mathbf{0}, \quad l \le x \le u \end{aligned} \tag{1.1}$$

where the functions of x are assumed to be twice differentiable with bounded Hessians. For this problem, the algorithm presented here would solve a sequence of linearly constrained subproblems of the following form: given x_k, $\boldsymbol{\lambda}_k$ and ρ,

$$\text{minimize} \quad L(x, x_k, \boldsymbol{\lambda}_k, \rho), \tag{1.2a}$$

$$\text{subject to} \quad \tilde{f} = \mathbf{0}, \quad l \le x \le u \tag{1.2b}$$

where the objective function is a *modified augmented Lagrangian,* and $\tilde{f}$ is the linear approximation to $f(x)$ at the point x_k. These quantities are defined as follows:

$$L(x, x_k, \boldsymbol{\lambda}_k, \rho) = f^0(x) - \boldsymbol{\lambda}_k^{\mathrm{T}}(f - \tilde{f}) + \tfrac{1}{2}\rho(f - \tilde{f})^{\mathrm{T}}(f - \tilde{f}),$$

$$\tilde{f} = f_k + J_k(x - x_k)$$

where f_k and J_k are the constraint vector and Jacobian matrix evaluated at x_k.

Our approach is based on the algorithm given by Robinson [45]. When applied to problem (1.1), Robinson's algorithm would solve a sequence of subproblems of the form (1.2), except that the penalty term involving ρ would not be present. If x_k and $\boldsymbol{\lambda}_k$ are taken to be the solution and corresponding Lagrange multipliers for the previous subproblem, Robinson has shown for the case $\rho = 0$ that the sequence $\{(x_k, \boldsymbol{\lambda}_k)\}$ will converge under certain circumstances to a solution of the original problem, and that the rate of convergence will be quadratic. A case for which convergence can be expected from an arbitrary starting point is when the modified Lagrangian $L(x, x_k, \boldsymbol{\lambda}_k, 0)$ is convex. In general, however, convergence may not occur unless the initial point $(x_0, \boldsymbol{\lambda}_0)$ is sufficiently close to the desired solution.

In order to allow convergence from a wider range of starting points, Rosen [48] proposed a two-phase algorithm in which a penalty function is first used to locate a point in a neighborhood of the solution. For the model problem, Phase 1 of Rosen's method would involve solving the problem

$$\text{minimize} \quad f^0(x) + \tfrac{1}{2}\rho f^{\mathrm{T}} f, \tag{1.3a}$$

$$\text{subject to} \quad l \le x \le u \tag{1.3b}$$

for some choice of ρ. (Phase 2 is then Robinson's method.) The general constraints are not linearized in Phase 1; they appear only in the objective function (1.3a). However, any that are known to be linear would be included in (1.3b). A linearly constrained optimizer is therefore needed in both phases.

This method depends heavily on a good choice for the penalty parameter ρ. If ρ is too large, the Phase 1 subproblem may be difficult and expensive to solve. Conversely, if ρ is too small, the point obtained from Phase 1 may be too far from the desired solution for Robinson's method to converge.

To overcome these difficulties, Best et al. [9] have recently proposed an algorithm similar to Rosen's, in which provision is made to return to Phase 1 with an increased ρ if it is determined that Robinson's method is not converging. Under certain weak assumptions they are then able to guarantee convergence to a Kuhn-Tucker point for the original problem.

As an alternative to the two-phase approach, the method we propose uses subproblem (1.2) throughout. The penalty term is chosen to ensure that the Hessian of $L(x, x_k, \lambda_k, \rho)$ is positive definite within an appropriate subspace. It also inhibits large discrepancies between f and $\tilde{f}$, thereby discouraging large changes in x in each subproblem if the nonlinearities are such that the linearized constraints have little meaning far from the point of linearization.

As in Rosen's method, we depend rather heavily on making a good choice for ρ. Heuristically, we increase ρ whenever it seems necessary to prevent non-convergence. More rigorously, we develop a mechanism for deciding when to reduce ρ to zero, in order to benefit from Robinson's method's quadratic convergence.

The reason for choosing the modified quadratic penalty function in (1.2a), rather than the conventional $\frac{1}{2}\rho f^{\mathrm{T}} f$, will become clear when sparsity is reintroduced (Section 3.1).

1.1. Other approaches

One of the few general-purpose methods for large-scale nonlinear programs is the method of approximation programming [23]. This is now often called successive linear programming (SLP), and has been implemented in various forms, typically in conjunction with an existing mathematical programming (MP) system (for example, [4–7]). As the name implies, the method involves a sequence of linear subproblems. These can be formulated and solved using all of the facilities provided by the MP system. Clearly this carries many advantages.

One difficulty with SLP is that the solution to an LP subproblem is unlikely to provide a good approximation to a nonlinear solution, since the latter is normally not at a vertex of the (linearized) constraint set. The bounds on the variables must therefore be manipulated between subproblems, and the methods for doing this tend to be heuristic.

An algorithm based on successive *quadratic* programs has been implemented for large problems by Escudero [16]. Again this takes advantage of many of the facilities in an existing MP system. GRG [1, 2, 30] is the only other general-purpose algorithm we know of that has been applied to large-scale problems with some success.

1.2. *Use of MINOS*

For a certain class of objective functions, the development of MINOS has opened the way to solving large, linearly constrained problems quite efficiently. Hence, for large versions of problem (1.1) involving a sparse Jacobian matrix and many purely linear constraints, it is natural to apply MINOS to the corresponding subproblems (1.2). The resulting system is called MINOS/AUGMENTED [37]. Our aim is to describe the algorithm used and some details of its practical implementation, and to discuss its performance on some nontrivial problems.

Note that the Lagrangian and penalty terms in (1.2a) require continual evaluation of the nonlinear constraint functions during the solution of the subproblems. In some cases this may be expensive. MINOS/AUGMENTED therefore allows the option of setting $\lambda_k = 0$ and $\rho = 0$, so that only the true objective $f^0(x)$ remains in (1.2a). Some results obtained using this option are also reported.

2. Brief description of MINOS

MINOS is a particular implementation of Wolfe's reduced-gradient algorithm [54]. It is designed to solve large problems with nonlinear objective functions, expressed in the following standard form:

$$\text{minimize} \quad f^0(x) + c^{\mathrm{T}}x + d^{\mathrm{T}}y, \tag{2.1a}$$

$$\text{subject to} \quad A\begin{bmatrix} x \\ y \end{bmatrix} = b, \tag{2.1b}$$

$$l \le \begin{bmatrix} x \\ y \end{bmatrix} \le u \tag{2.1c}$$

where A is m by n, $m \le n$, and the variables are partitioned into 'nonlinear' and 'linear' variables x and y respectively. (This standard form is a slight generalization of the one normally used for linear programming problems; it emphasizes the fact that nonlinearities often involve just a few of the variables. The concept of linear and nonlinear variables was introduced by Griffith and Stewart [23] in 1961.)

For numerous practical reasons the last m columns of A form the identity matrix I, and the last m components of y are the usual logical ('slack' or 'surplus') variables.

MINOS uses an 'active constraint' strategy, with the general constraints and some portion of the bound constraints being active at any given time. Thus if A is partitioned as $[B\,S\,N]$ where N is a set of 'nonbasic' columns, the active constraints are always of the form

$$\begin{bmatrix} B & S & N \\ & & I \end{bmatrix} \begin{bmatrix} x_B \\ x_S \\ x_N \end{bmatrix} = \begin{bmatrix} b \\ b_N \end{bmatrix}. \tag{2.2}$$

The first part of this equation is equivalent to

$$A\begin{bmatrix} x \\ y \end{bmatrix} = b,$$

while the second part reading $x_N = b_N$ indicates that the nonbasic variables x_N are being held equal to one or other of their bounds. (The components of b_N come from l or u as appropriate and the partition $[x_B, x_S, x_N]$ is some permutation of $[x, y]$.)

The remaining columns of A are partitioned into 'basic' and 'superbasic' sets B and S, such that the basis matrix B is square and nonsingular. The corresponding basic and superbasic variables x_B and x_S are free to vary between their bounds during the next iteration.

It can readily be shown that an optimal solution of the above form exists for which the number of superbasic variables is less than or equal to the number of nonlinear variables. This point is discussed later in a more general context (Sections 3.1 and 3.2).

2.1. *Some aspects of the algorithm used in MINOS*

The operators

$$\hat{A} = \begin{bmatrix} B & S & N \\ & & I \end{bmatrix}, \qquad Z = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \tag{2.3}$$

will be useful for descriptive purposes. The active constraints (2.2) are of the form $\hat{A}\hat{x} = \hat{b}$, and Z happens to satisfy $\hat{A}Z = 0$.

Under suitable conditions a *feasible descent direction* p may be obtained from the equations

$$Z^{\mathrm{T}}GZp_S = -Z^{\mathrm{T}}g, \qquad p = Zp_S$$

(see Gill and Murray [18]), where g is the gradient of the objective function (2.1a). Thus, if the *reduced gradient* $Z^{\mathrm{T}}g$ is nonzero and if the *reduced Hessian* $Z^{\mathrm{T}}GZ$ is positive definite (or if any positive definite matrix is used in place of $Z^{\mathrm{T}}GZ$), then the point $x + \alpha p$ lies on the active constraints and some scalar $\alpha > 0$ exists for which the objective function has a lower value than at the point x.

Other matrices Z exist satisfying $\hat{A}Z = 0$, but the form chosen above, together with a sparse LU factorization of B, allows efficient computation of the products $Z^{\mathrm{T}}g$ and Zp_S. (Neither B^{-1} nor Z is computed.) A positive-definite approximation to $Z^{\mathrm{T}}GZ$ is maintained in the form $R^{\mathrm{T}}R$ where R is upper triangular. Quasi-Newton updates to R lead to superlinear convergence.

Let the gradient of the objective function be partitioned as $[g_B, g_S, g_N]$. If π satisfies

$$B^T\pi = g_B$$

it is easily seen that the reduced gradient is

$$Z^Tg = g_S - S^T\pi.$$

Hence in linear programming terminology the reduced gradient is obtained by 'pricing' the superbasic columns S. This is a cheap operation once π has been computed.

Likewise for p we have $R^TRp_S = -Z^Tg$ and then

$$p = \begin{bmatrix} p_B \\ p_S \\ p_N \end{bmatrix} = Zp_S = \begin{bmatrix} -B^{-1}Sp_S \\ p_S \\ 0 \end{bmatrix},$$

so most of the work lies in solving $Bp_B = -Sp_S$. (The value $p_N = 0$ indicates that no change will be made to the current nonbasic variables. As long as the reduced gradient Z^Tg is nonzero, only the variables in $[B\ S]$ are optimized. If any such variables encounter an upper or lower bound they are moved into N and the partition $[B\ S]$ is suitably redefined.)

Note that if the reduced gradient does prove to be zero ($Z^Tg = 0$), the reduced objective has reached its optimal value. If we compute $\sigma = g_N - N^T\pi$ (i.e., the usual pricing of nonbasic columns) we then have

$$\begin{bmatrix} B^T & \\ S^T & \\ N^T & I \end{bmatrix} \begin{bmatrix} \pi \\ \sigma \end{bmatrix} = \begin{bmatrix} g_B \\ g_S \\ g_N \end{bmatrix},$$

so that π and σ are exact Lagrange multipliers for the current active constraints. The components of σ indicate whether any nonbasic variables should be released from their bounds. If so, one or more are moved from N into S and optimization continues for the new set $[B\ S]$. If not, an optimum has been obtained for the original problem.

In practice, optimization for each $[B\ S]$ will be curtailed when Z^Tg is sufficiently small, rather than zero. In this case π will be just an approximation to the Lagrange multipliers for the general constraints. The accuracy of π will depend on the size of $\|Z^Tg\|$ and on the condition number of the current basis B.

2.2. *Key points*

The algorithm implemented in MINOS provides a natural extension of linear programming technology to problems whose objective function is nonlinear. If the number of nonlinear variables is moderate (or more precisely, if the number of superbasic variables and hence the dimension of R is moderate), then the work per iteration is not substantially greater than for one iteration of the revised simplex method on the same data

$$A\begin{bmatrix}x\\y\end{bmatrix}=b,\qquad l\le\begin{bmatrix}x\\y\end{bmatrix}\le u.$$

Here we assume that the cost of evaluating the objective function and its gradient is moderate compared to manipulation of a sparse factorization of the basis matrix B. At the same time it is important that the step-size α be determined efficiently. The linesearch procedure used in MINOS/AUGMENTED is that of [19, 21], in the form of subroutine GETPTC. This employs cubic interpolation with safeguards, and allows the user to control the accuracy of the search by means of a parameter ETA, where $0.0 \le \text{ETA} < 1.0$. Even with a relatively accurate search (e.g., ETA $= 0.01$), the number of function and gradient evaluations required is typically very few (usually 1, 2 or 3 per search). This is increasingly beneficial for the algorithm discussed next, where the objective function is modified to include an arbitrary number of nonlinear functions.

3. Extension to nonlinear constraints

3.1. Statement of the problem

It is assumed that the nonlinearly constrained problem can be expressed in the following standard form:

$$\text{minimize}\quad f^0(x)+c^{\mathrm{T}}x+d^{\mathrm{T}}y, \tag{3.1a}$$

$$\text{subject to}\quad f(x)+A_1y=b_1\quad (m_1\text{ rows}), \tag{3.1b}$$

$$A_2x+A_3y=b_2\quad (m_2\text{ rows}), \tag{3.1c}$$

$$l\le\begin{bmatrix}x\\y\end{bmatrix}\le u,\qquad m=m_1+m_2 \tag{3.1d}$$

where $f(x)=[f^1(x),\dots,f^{m_1}(x)]^{\mathrm{T}}$. The first n_1 variables x are again called 'nonlinear variables'. They occur nonlinearly in either the objective function or the first m_1 constraints. There may be purely linear constraints, (3.1c). As before, a full set of slack variables is included as the last m components of the 'linear variables' y, so that general equality and inequality constraints can be accommodated in (3.1b, c) by means of suitable bounds in (3.1d).

We shall assume that the functions $f^i(x)$ are twice continuously differentiable with gradients $g^i(x)$ and bounded Hessians $G^i(x)$, $i=0,1,\dots,m_1$. We shall also assume that the 1st and 2nd order Kuhn-Tucker conditions hold for a local minimum x^* with corresponding Lagrange multipliers λ^*.

The standard form above is essentially the same as the one adopted by Griffith and Stewart [23]. Clearly, if the nonlinear variables x are given fixed values, the problem reduces to a linear program in y. Beale [6, 7] points out that it can be useful to partition the variables further, such that the problem reduces to a linear program when just some of the nonlinear variables are given fixed values. Without loss of generality, x and $f(x)$ can be expressed in the form

$$x = \begin{bmatrix} x^N \\ x^L \end{bmatrix}, \qquad f(x) = \hat{f}(x^N) + A(x^N)x^L \tag{3.2}$$

where the vector $\hat{f}(x^N)$ and the matrix $A(x^N)$ depend only on x^N. The objective function can be partitioned similarly. The extent to which an optimal solution deviates from a normal 'basic solution' is then bounded by the dimension of x^N rather than the dimension of x. Although we do not exploit such structure explicitly, it is important to be aware that real-life models often arise in this form. Hence, a model expressed in the simpler standard form (3.1) may not be as nonlinear as it looks. This is fortunate, and we bear it in mind in formulating the subproblem below.

The partitioning algorithms of Benders [8] and Rosen [46] were designed for problems in the form (3.1) and (3.2) respectively (see Lasdon [29, Ch. 7]). With certain restrictions, (3.2) also resembles the 'variable coefficients' form of Wolfe's generalized linear program [14, Ch. 22].

3.2. *The linearized subproblem*

The solution process consists of a sequence of *major iterations*, each one involving a linearization of the nonlinear constraints at some point x_k, corresponding to a first-order Taylor's series approximation:

$$f^i(x) = f^i(x_k) + g^i(x_k)^{\mathrm{T}}(x - x_k) + \mathrm{O}\|x - x_k\|^2.$$

We thus define

$$\tilde{f}(x, x_k) = f(x_k) + J(x_k)(x - x_k),$$

or

$$\tilde{f} = f_k + J_k(x - x_k) \tag{3.3}$$

where $J(x)$ is the $m_1 \times n_1$ Jacobian matrix whose ijth element is $\partial f^i/\partial x_j$. We then see that

$$f - \tilde{f} = (f - f_k) - J_k(x - x_k)$$

consists of the higher order ('nonlinear') terms in the Taylor's expansion of $f(x)$ about the point x_k.

At the kth major iteration of our algorithm, the following linearly constrained subproblem is formed:

$$\underset{x,\,y}{\text{minimize}} \quad L(x, y, x_k, \lambda_k, \rho) = f^0(x) + c^{\mathrm{T}}x + d^{\mathrm{T}}y - \lambda_k^{\mathrm{T}}(f - \tilde{f}) + \tfrac{1}{2}\rho(f - \tilde{f})^{\mathrm{T}}(f - \tilde{f}), \tag{3.4a}$$

$$\text{subject to} \quad \tilde{f} + A_1 y = b_1, \tag{3.4b}$$

$$A_2 x + A_3 y = b_2, \tag{3.4c}$$

$$l \le \begin{bmatrix} x \\ y \end{bmatrix} \le u. \tag{3.4d}$$

The objective function L is a modified augmented Lagrangian, in which $f - \tilde{f}$ is used in place of the conventional constraint violation, $f + A_1 y - b_1$. The partial derivatives are

$$\frac{\partial L}{\partial x} = g^0(x) + c - (J - J_k)^{\mathrm{T}}[\boldsymbol{\lambda}_k - \rho(f - \tilde{f})] \tag{3.5}$$

and $\partial L/\partial y = d$. We see that the nonlinearities in L involve x but not y, which means that the subproblem has the same nonlinear variables as the original problem. Further, if the functions are written in the form (3.2) above, $f(x)$ is by definition a linear function of x^L, and so $f - \tilde{f}$ is identically zero when x^N is held fixed. Thus, in Beale's generalized sense, L is effectively nonlinear in x^N alone. The dimension of the reduced Hessian for the subproblem is therefore bounded in a way that is consistent with the nonlinearity of the original problem. This property seems highly desirable.

The modified Lagrangian was used by Robinson [45] with $\rho = 0$. The use of a penalty term to ensure the augmented Lagrangian maintains a positive-definite Hessian in the appropriate subspace was suggested by Arrow and Solow [3] and adopted later by, among others, Hestenes [25] and Powell [39] in their sequential unconstrained procedures, and by Sargent and Murtagh [50] in conjunction with their 'variable-metric projection' algorithm involving a sequence of linearized constraints. To our knowledge, the modified penalty function has not been used elsewhere. Note that it is identical to the conventional penalty function in the subspace defined by the linearized constraints.

A potential disadvantage of the quadratic penalty function has been pointed out by Greenberg [22], namely, that it destroys separability if any $f^i(x)$ is the sum of two or more separable functions. This means that the subproblems can be solved by a separable programming code only in the case where each $f^i(x) = f^i(x_j)$ for some j, or in the convex programming case where the penalty term is not required anyway.

3.3. *Choice of* $\boldsymbol{\lambda}_k$

The choice $\boldsymbol{\lambda}_k = \mathbf{0}$, $\rho = 0$ corresponds to simple sequential linearization of the nonlinear constraints, with no additional terms to $f^0(x)$ in the objective function. We shall call this the *Newton strategy*, although it should not be confused with applying Newton's method to the Kuhn-Tucker equations for a solution of (3.1). This simplified procedure does converge in certain cases, particularly if the solution is at a vertex (e.g., reverse-convex programs [47, 33, 26]).

Ideally, λ_k should be as close as possible to λ^*, but of course the optimal multipliers are normally unknown. The simplest choice is $\lambda_k = \hat{\lambda}$, the multipliers corresponding to linearized constraints at the solution of the previous subproblem. As we shall see, this choice is the best of several alternatives. For convenience suppose there are no linear constraints, so that $\hat{\lambda} = \pi$ is the solution of $B^{\mathrm{T}}\pi = g_B$ at the end of the previous major iteration. We know that π also satisfies $S^{\mathrm{T}}\pi = g_S$ (at least to within the convergence tolerance used for the subproblem). We thus have

$$\begin{bmatrix} B^{\mathrm{T}} \\ S^{\mathrm{T}} \end{bmatrix} \hat{\lambda} = \begin{bmatrix} g_B \\ g_S \end{bmatrix}$$

and it is immaterial which variables are in B and which are in S. Now g is zero for all slack variables and it follows immediately that $\hat{\lambda}_i = 0$ if the ith linearized constraint is inactive. The choice $\lambda_k = \hat{\lambda}$ therefore ensures that an apparently inactive nonlinear constraint will be excluded from the Lagrangian term $\lambda_k^{\mathrm{T}}(f - \tilde{f})$ in the next subproblem. This is a desirable property.

It may seem that a better approximation to λ^* could be obtained by evaluating the new Jacobian $J(\hat{x})$ which is required anyway for the next subproblem. Let the resulting 'new' $[B\ S]$ be denoted by $[\bar{B}\ \bar{S}]$. One possibility is to define λ_k as the solution of the least-squares problem

$$\begin{bmatrix} \bar{B}^{\mathrm{T}} \\ \bar{S}^{\mathrm{T}} \end{bmatrix} \lambda \approx \begin{bmatrix} g_B \\ g_S \end{bmatrix}$$

where the rhs is still the 'old' gradient vector for the previous augmented Lagrangian. However, this least-squares problem would be very expensive to solve for large problems. Furthermore it is not guaranteed that $\lambda_i = 0$ would result where desired.

A cheaper alternative would be to solve $\bar{B}^{\mathrm{T}}\bar{\pi} = g_B$ and take $\lambda_k = \bar{\pi}$, but then $\lambda_i = 0$ for inactive constraints would be assured only if the corresponding slack variable happened to be basic and not superbasic.

If the new $\bar{B}$ is to be used, the method of Sargent and Murtagh [50] shows that

$$\bar{B}^{\mathrm{T}}\lambda = g_B + [I\ 0\ 0]G_L \Delta x$$

would produce the correct multipliers for the solution of the *new* subproblem if the original objective and constraints were quadratic and G_L was an adequate approximate to the Hessian of the new Lagrangian. (See equation (12) in [35].) However, this again is not a practical alternative for large problems.

3.4. Choice of ρ

It is well known that x^* need not be a local minimum of the Lagrangian function (with $\rho = 0$). If we assume that $J(x^*)$ is of full rank, then λ^* exists and is such that

$$L(x, \lambda) = f^0(x) + c^T x + d^T y - \lambda^T[f + A_1 y - b_1]$$

is stationary at (x^*, λ^*), but $L(x^*, \lambda^*)$ may well display negative curvature in x at x^*.

The most that can be said [55] is that, if we consider the constraints satisfied at x^* as equalities and ignore the inactive ones, then a necessary (sufficient) condition that x^* is a local minimum is

$$Z(x^*)^T \frac{\partial L}{\partial x}(x^*, \lambda^*) = 0$$

and

$$Z(x^*)^T \frac{\partial^2 L}{\partial x^2}(x^*, \lambda^*) Z(x^*)$$

is positive semidefinite (positive definite), where $Z(x^*)$ is as defined in (2.3) using $J(x^*)$ in the appropriate part of $\hat{A}$.

Thus if we restrict our search to the linearly constrained subspace defined by $Z(x^*)$ we do indeed seek a minimum of the Lagrangian, and we may expect that when x_k is sufficiently close to x^* for $J(x_k)$ to be close to $J(x^*)$ we may minimize (3.4a) with $\rho = 0$. This is confirmed by Robinson's theorem on quadratic convergence [45].

Difficulty arises when x_k is far removed from x^*, since the linearized constraints may define a subspace where perhaps a saddle-point would be closer to x^* than a minimum would be. Successive minima of (3.4) with $\rho = 0$ may therefore fail to converge to x^*. The addition of a penalty term $\rho[f - \tilde{f}]^T[f - \tilde{f}]$ imposes the correct curvature properties on (3.4a) for a sufficiently large $\rho > 0$.

For general nonconvex problems it is not practical to determine a priori what the appropriate order of magnitude ρ should be (indeed, $\rho = 0$ is often adequate even in the nonconvex case). The more important consideration is when to reduce ρ to zero, for we know that there is a radius of convergence around (x^*, λ^*) within which Robinson's theorem holds for $\rho = 0$, and we can then expect a quadratic rate of convergence.

Two parameters we can monitor at the solution $(\hat{x}, \hat{\lambda})$ to each linearized subproblem are the constraint violation or 'row error',

$$\|f(\hat{x}) + A_1 y - b_1\| = \|f(\hat{x}) - \tilde{f}(\hat{x}) - \tilde{f}(\hat{x}, x_k)\|,$$

and the change in multiplier estimates, $\|\hat{\lambda} - \lambda_k\|$. The question that arises is whether these can be used to provide adequate measures of convergence toward x^*.

For simplicity, consider the equality-constrained problem

$$\text{P}_0: \quad \text{minimize } f^0(x), \quad \text{subject to } f(x) = 0$$

where the functions of x are twice continually differentiable with bounded Hessians. We shall assume that at some point x^* the Jacobian $J(x^*)$ is of full rank, there exists a λ^* such that $\partial f^0/\partial x = J(x^*)^T\lambda^*$, and the reduced Hessian $Z(x^*)^T\partial^2 L(x^*, \lambda^*)/\partial x^2 Z(x^*)$ is positive definite (i.e., the sufficiency conditions are satisfied for x^* to be a local optimum).

Theorem 1. *Let* (x_k, λ_k) *be an approximate solution to* P_0 *and let* $(\hat{x}, \hat{\lambda})$ *be a solution to the linearized subproblem*

$$\mathrm{S}_1: \quad \text{minimize} \quad f^0(x) - \lambda_k^T(f - \tilde{f}) + \tfrac{1}{2}\rho(f - \tilde{f})^T(f - \tilde{f}),$$
$$\text{subject to} \quad \tilde{f}(x, x_k) = 0.$$

If $\hat{\lambda} - \lambda_k = \epsilon_1$ *and* $f(\hat{x}) = \epsilon_2$, *then* $(\hat{x}, \hat{\lambda})$ *is also a solution to the perturbed problem*

$$\mathrm{P}_1: \quad \text{minimize} \quad f^0(x) + (\epsilon_1 + \rho\epsilon_2)^T(f - \tilde{f}),$$
$$\text{subject to} \quad f(x) = \epsilon_2$$

for sufficiently small ϵ_1 *and* ϵ_2.

Proof. *If* $(\hat{x}, \hat{\lambda})$ is a solution of S_1 we must have $\tilde{f} = 0$ and

$$g^0(\hat{x}) - (\hat{J} - J_k)^T\lambda_k + \rho(\hat{J} - J_k)^T(f - \tilde{f}) = J_k^T\hat{\lambda}$$

where J_k is the Jacobian at x_k but $\hat{J}$, f and $\tilde{f}$ are evaluated at $\hat{x}$. Adding $(\hat{J} - J_k)^T\hat{\lambda}$ to both sides and inserting the expressions for ϵ_1 and ϵ_2 gives

$$g^0(\hat{x}) + (\hat{J} - J_k)^T\epsilon_1 + \rho(\hat{J} - J_k)^T\epsilon_2 = \hat{J}^T\hat{\lambda}$$

which shows that $(\hat{x}, \hat{\lambda})$ also satisfies the conditions for a stationary point of P_1. Now it can be shown that the Hessians for the Lagrangian functions of S_1 and P_1 differ only by the amount $\rho(\hat{J} - J_k)^T(\hat{J} - J_k)$ at the solution of P_1, which is of order $\rho\|\Delta x_k\|^2$ where $\Delta x_k = \hat{x} - x_k$. Hence for sufficiently small ϵ_1, ϵ_2 and Δx_k, if the reduced Hessian of S_1 is positive definite at $\hat{x}$, then by continuity the reduced Hessian of P_1 will also be positive definite, thus satisfying the sufficiency conditions for a local minimum of P_1 at $\hat{x}$.

It is of interest to examine the corresponding result for the conventional penalty term.

Theorem 2. *Let* (x_k, λ_k) *be an approximate solution to* P_0 *and let* $(\hat{x}, \hat{\lambda})$ *be a solution to the linearized subproblem*

$$\mathrm{S}_2: \quad \text{minimize} \quad f^0(x) - \lambda_k^T(f - \tilde{f}) + \tfrac{1}{2}\rho f^T f,$$
$$\text{subject to} \quad \tilde{f}(x, x_k) = 0.$$

If $\hat{\lambda} - \lambda_k = \epsilon_1$ *and* $f(\hat{x}) = \epsilon_2$, *then* $(\hat{x}, \hat{\lambda})$ *is also a solution to the perturbed problem*

$$\begin{aligned} \mathrm{P}_2: \quad & \text{minimize} \quad f^0(x) + \boldsymbol{\epsilon}_1^{\mathrm{T}}(f - \tilde{f}) + \rho \boldsymbol{\epsilon}_2^{\mathrm{T}} f, \\ & \text{subject to} \quad f(x) = \boldsymbol{\epsilon}_2. \end{aligned}$$

Proof. Analogous to the proof of Theorem 1.

Again it follows that if $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are sufficiently small, $(\hat{x}, \hat{\boldsymbol{\lambda}})$ will be within the radius of convergence of Robinson's theorem and ρ can safely be reduced to zero. A point of interest is that problem P_1 appears to be less sensitive than P_2 to deviations from its optimum. Thus, let Δx be an arbitrary small change to the solution $\hat{x}$ of P_1. The objective function for P_1 then differs from the true objective $f^0(x)$ by an amount

$$\begin{aligned} &\delta_1 = (\boldsymbol{\epsilon}_1 + \rho \boldsymbol{\epsilon}_2)^{\mathrm{T}}(f - \tilde{f}), \\ &|\delta_1| \le (\|\boldsymbol{\epsilon}_1\| + \rho\|\boldsymbol{\epsilon}_2\|)\mathrm{O}\|\Delta x\|^2. \end{aligned}$$

For P_2 the analogous deviation is

$$\begin{aligned} \delta_2 &= \boldsymbol{\epsilon}_1^{\mathrm{T}}(f - \tilde{f}) + \rho \boldsymbol{\epsilon}_2^{\mathrm{T}} f \\ &= \boldsymbol{\epsilon}_1^{\mathrm{T}}(f - \tilde{f}) + \rho \boldsymbol{\epsilon}_2^{\mathrm{T}}(\hat{f} + \hat{J}\Delta x + \mathrm{O}\|\Delta x\|^2), \\ |\delta_2| &\le (\|\boldsymbol{\epsilon}_1\| + \rho\|\boldsymbol{\epsilon}_2\|)\mathrm{O}\|\Delta x\|^2 + \rho\|\boldsymbol{\epsilon}_2\|^2 + \rho\|\hat{J}^{\mathrm{T}}\boldsymbol{\epsilon}_2\|\,\|\Delta x\|. \end{aligned}$$

Since δ_1 is of order $\|\Delta x\|^2$ while δ_2 is of order $\|\Delta x\|$, it appears that the modified penalty term in S_1 has a theoretical advantage over the conventional penalty term of S_2.

3.5. Summary of procedure

The cycle of major iterations can be described as follows:

Step 0. Set $k = 0$. Choose some initial estimates x_0, y_0 and $\boldsymbol{\lambda}_0$. Specify a penalty parameter $\rho \ge 0$ and a convergence tolerance $\epsilon_c > 0$.

Step 1. (a) Given x_k, y_k, $\boldsymbol{\lambda}_k$ and ρ, solve the linearly constrained subproblem (3.4) to obtain new quantities x_{k+1}, y_{k+1} and $\boldsymbol{\pi}$ (where $\boldsymbol{\pi}$ is the vector of Lagrange multipliers for the subproblem).

(b) Set $\boldsymbol{\lambda}_{k+1}$ = the first m_1 components of $\boldsymbol{\pi}$.

Step 2. (a) Test for convergence (see Section 4.8). If optimal, exit.

(b) If

$$\|f(x_{k+1}) + A_1 y_{k+1} - b_1\|/(1 + \|[x_{k+1}, y_{k+1}]\|) \le \epsilon_c$$

and

$$\|\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k\|/(1 + \|\boldsymbol{\lambda}_{k+1}\|) \le \epsilon_c,$$

then set $\rho = 0$.

(c) Relinearize the constraints at x_{k+1}.

(d) Set $k = k + 1$ and repeat from Step 1.

This procedure would not be complete without an algorithm for increasing the penalty parameter in certain circumstances. In Step 2(b) of the present implementation, we raise ρ by some factor if the relative change in $\boldsymbol{\lambda}_k$ proves to be very large.

4. Computer implementation

4.1. Sparse matrices

Using equation (3.3), the linearized constraints (3.4b) can be expressed in the form:

$$J_k x + A_1 y = b_1 + J_k x_k - f_k \tag{4.1}$$

where $f_k = f(x_k)$. The terms on the right-hand side of (4.1) are constant and become part of 'b', the current right-hand side. The set of linear constraints '$Ax = b$' for each major iteration is thus of the form:

$$\begin{bmatrix} J_k & A_1 \\ A_2 & A_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b_1 + J_k x_k - f_k \\ b_2 \end{bmatrix}. \tag{4.2}$$

The major implication of A being large and sparse is that efficient methods are available for forming and updating an LU factorization of the basis matrix B (cf. (2.2)). (In particular, a 'bump and spike' algorithm [24] is used to preserve sparsity at each refactorization of B. This occurs at the start of every relinearization and occasionally thereafter as necessary. At each intervening change of basis the LU factors are updated using the scheme described by Saunders [51] to preserve both sparseness and numerical stability.)

4.2. Infeasible subproblems

One of the difficulties with sequential linearization is that some of the linearized subproblems may prove to be infeasible. In particular, the point $[x_k, y_k]$ used to define subproblem k is usually not a feasible point for the subproblem. However, it will typically be feasible for the previous subproblem (and possibly optimal). This can be used to advantage in the manner suggested by Powell [41]. Thus we write the linearized constraints (4.1) in the form

$$J_k x + A_1 y = b_1 + J_k x_k - f_k + \gamma q \tag{4.3}$$

where γq is a perturbation to the right-hand side. If $[x_k, y_k]$ is the final feasible point from subproblem $k-1$, we can show that it will also be feasible with respect to the new linearized constraints (4.3) if $\gamma = 1$ and $q = f(x_k) - \tilde{f}(x_k, x_{k-1})$. (Thus q is the value of $f - \tilde{f}$ at the end of the previous major iteration.)

In MINOS/AUGMENTED the right-hand side of (4.3) is initialized with $\gamma = 0$. If the subproblem proves to be infeasible we add $\frac{1}{2}q$ to the right-hand side and continue the solution process. If there is still no feasible solution we add $\frac{1}{4}q$, $\frac{1}{8}q$ and so on. This simulates the sequence of values $\gamma = \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$ tending to 1 as desired.

If the above procedure fails after 10 modifications, or if it is not applicable (e.g., when $k = 0$ or the previous subproblem was infeasible), a new linearization is requested as long as at least one minor iteration has been performed. Otherwise the algorithm is terminated with the assumption that the original problem itself is infeasible.

In [48], Rosen guards against infeasible subproblems by linearizing perhaps only some of the nonlinear constraints, namely those that have been active or reasonably close to active at any earlier stage. This alternative could be implemented in MINOS/AUGMENTED by adjusting the bounds on the slack variables associated with the linearized constraints.

4.3. *User options*

Various implementation options are discussed in the following sections. Capitalized keywords at the head of each section illustrate the input data needed to select any particular option. Fuller details are given in the user's manual [37].

4.4. *Subroutines CALCFG and CALCON*

VERIFY OBJECTIVE GRADIENT
VERIFY CONSTRAINT GRADIENTS

As in the linearly constrained version of MINOS, a user-written subroutine CALCFG is required to calculate the objective function $f^0(x)$ and its gradient. The Lagrangian terms in (3.4a) are calculated internally.

The user also supplies a subroutine CALCON to define the constraint vector $f(x)$ and the current Jacobian $J(x)$. The nonzeros in J are returned column-wise in an output vector and must be in the same order as the corresponding entries in the MPS file (see below).

Subroutine CALCON is called every time the constraints are linearized. It is also called one or more times each line search (except with the Newton strategy), to allow computation of (3.4a) and (3.5). The expense of evaluating the constraints and their gradients should therefore be taken into account when specifying the linesearch accuracy.

Note that every function and Jacobian element is computed in every call to CALCON. Although some of these values may effectively be wasted (e.g. if some of the constraints are a long way from being active), the resulting simplicity of the subroutine from the user's point of view cannot be overemphasized.

Since the programming of gradients is notoriously prone to error, the VERIFY option is an essential aid to the user. This requests a check on the output from

CALCFG and/or CALCON, using finite differences or $f^0(x)$ of $f(x)$ along the coordinate directions. The check is performed at the first feasible point obtained (where feasibility is with respect to the first linearized subproblem). This point will not satisfy the nonlinear constraints in general, but at least it will satisfy the linear constraints and the upper and lower bounds on x. Hence it is usually possible to avoid singularities in the nonlinear functions, both in the gradient check and in subsequent iterations.

4.5. Jacobian option

JACOBIAN = SPARSE or DENSE

The submatrices A_1, A_2, A_3 and vectors b_1, b_2 in (4.2) are constant data and so may be entered using a standard MPS input file, as in linear programming, whereby only the nonzero coefficients and their row locations are entered column-by-column. Since we envisage that the Jacobian submatrix J will also be large and sparse we use the same scheme for recording the row and column locations of the nonzeros. Thus (with JACOBIAN = SPARSE) the sparsity pattern of J is entered as part of the MPS file. The corresponding numerical values in the MPS file may be genuine coefficients (if they are constant) or else dummy values, such as zero. Each call to subroutine CALCON subsequently replaces all dummy entries by their actual value at the current point x.

Of course the intention here is to allow use of standard matrix generators to specify as much of the constraint matrix as possible. Pinpointing the nonzeros of J by name rather than number has the usual advantages, and in subroutine CALCON some code of the form

```
LJAC = LJAC + 1
G(LJAC) = ···
```

is usually adequate to define the next nonzero in a column of the Jacobian, without explicit reference to any row or column numbers. Nevertheless, the user is effectively required to give the sparsity pattern *twice* (in the MPS file and in CALCON), and it is essential that mismatches be avoided. At present the VERIFY option is the only aid to detecting incompatibility.

In the interest of simplicity, the option JACOBIAN = DENSE allows J to be treated as a dense matrix. In this case the MPS file need not specify any elements of J, and subroutine CALCON can use assignment statements of the form $G(I, J) = \cdots$ to specify J_{ij} by row and column number. The danger of mismatches is thereby eliminated, but the storage requirements may be excessive for large problems. It may also give rise to an unnecessarily large 'bump' in the basis factorizations.

4.6. Partial completion

COMPLETION = PARTIAL or FULL

Partial completion is a compromise between the two extremes of relinearizing after each linesearch and solving each subproblem to high accuracy (*full completion*).

The idea of attaining only partial completion at each major iteration can be accommodated effectively via the convergence tolerances. Following Gill and Murray [20], MINOS uses relatively loose tolerances for minimizing the reduced objective, until it appears that the optimal partition $[B\ S\ N]$ has been identified. The partial completion option is effected by terminating a major iteration at this stage.

Otherwise for full completion the normal optimization procedure is continued using tighter tolerances to measure the change in x, the size of the reduced gradient ($\|Z^{\mathrm{T}}g\|/\|\pi\|$), etc. This usually gives only small changes in x and π without changing the partition $[B\ S\ N]$.

An alternative means for achieving partial completion for early major iterations is via the MINOR ITERATIONS limit (see Section 4.8).

4.7. Lagrangian option, penalty parameter

Newton strategy:	LAGRANGIAN	NO
	PENALTY PARAMETER	0.0

With this option the constraint functions and gradients are evaluated only once per major iteration.

Augmented Lagrangian:	LAGRANGIAN	YES
	PENALTY PARAMETER	ρ $(\rho \geq 0)$

Here the constraints and Jacobian are evaluated as often as the objective. Evaluation of the augmented Lagrangian and its gradient with $\rho > 0$ is negligibly more expensive than with $\rho = 0$.

4.8. Convergence conditions

MAJOR ITERATIONS	60
MINOR ITERATIONS	40
RADIUS OF CONVERGENCE	ϵ_c $(\approx 10^{-2})$
ROW TOLERANCE	ϵ_r $(\approx 10^{-6})$

Apart from terminating within each major iteration, we also need a terminating condition for the cycle of major iterations (Step 2(a), Section 3.5).

The point $[x_k, y_k]$ is assumed to be a solution to the nonlinearly constrained problem (3.1) if both the following conditions are satisfied:

(1) $[x_k, y_k]$ satisfies the nonlinear constraints (3.1b) to within a predefined error tolerance, i.e.,

$$\|f(x_k) + A_1 y_k - b_1\|/(1 + \|[x_k, y_k]\|) \le \epsilon_r.$$

(2) $[x_k, y_k]$ satisfies the first-order Kuhn-Tucker conditions for a solution to the linearized problem.

The tolerance parameter ϵ_r is specified by the user, and was set equal to 10^{-6} for most of the test problems described in the subsequent sections.

If the partial completion option is used, then once these two criteria are satisfied, a switch to full completion is invoked to obtain an accurate solution for the current subproblem, as well as for any possible further linearizations.

The error tolerance ϵ_c is used to define a radius of convergence about (x^*, λ^*) within which Robinson's theorem is assumed to hold. If the row error defined above and the relative change in Lagrange multipliers between successive subproblems are both less than ϵ_c in magnitude, then the penalty term is dropped (i.e. ρ is set to 0.0).

The MINOR ITERATIONS limit is used to terminate each linearized subproblem when the number of linesearches becomes excessive. The limit of 40 was used in most of the numerical experiments. A much smaller number would result in more frequent use of expensive housekeeping operations such as the refactorization of B. Similarly a much larger number may be wasteful; if significant changes to x have occurred, then a new linearization is appropriate, while if there has been little progress, then updating the Lagrangian information gives some hope of more rapid progress.

It must be noted that for some choices of x_0, λ_0 and ρ the sequence of major iterations may not converge. The MAJOR ITERATIONS limit provides a safeguard for such circumstances.

For a discussion of linearly constrained Lagrangian methods and their drawbacks see Wright [55, pp. 137–147].

In the present implementation of MINOS/AUGMENTED, if convergence does not occur, the only recourse is to restart with a different (x_0, λ_0) or a larger value for the penalty parameter ρ (or both).

5. Test problems

Most of the test examples reported here appear in the published literature. The last two examples are new and are described in detail. They can be made arbitrarily large by increasing one parameter.

5.1. Colville No. 2

This well known problem is one of the more testing of the Colville series of problems [12] and has been used frequently to compare different algorithms [2, 17, 45, 50]. It has a cubic objective function and 15 quadratic constraints. Even in this small problem the variables can be partitioned into linear and nonlinear sets, of dimension 10 and 5 respectively.

(a) Feasible starting point.

(b) Infeasible starting point.

5.2. *Colville No. 3*

This problem has a quadratic objective function of five variables and three quadratic constraints. It also has upper and lower bounds on all the variables, and upper and lower limits on the constraints. These can be accommodated effectively by using the BOUNDS and RANGES options in the MPS file; the BOUNDS option allows variables to be nonbasic at their upper or lower bound, and the RANGES option assigns both upper and lower bounds to the slack variables associated with the constraints (thus allowing the right-hand side to range between bounds).

(a) Feasible starting point.

(b) Infeasible starting point.

5.3. *Colville No. 8*

This is a typical process design problem, where some of the variables are determined by solving nonlinear equations. It has 3 independent variables and 7 constraints.

5.4. *Powell's problem* [40]

This has 5 variables and 3 constraints. Although small, this is a good test problem as the nonlinearities in the constraints are quite significant.

$$\begin{array}{ll} \text{minimize} & \exp(x_1x_2x_3x_4x_5), \\ \text{subject to} & x_1^2+x_2^2+x_3^2+x_4^2+x_5^2 = 10, \\ & x_2x_3-5x_4x_5 = 0, \\ & x_1^3+x_2^3 = -1. \end{array}$$

Starting point: $(-2, 2, 2, -1, -1)$.

5.5. *Power scheduling*

This is a comparatively large test problem reported recently by Biggs and Laughton [10], with 79 variables and 91 constraints (all nonlinear). It also has upper and lower bounds on some of the variables. Although all the variables and constraints are nonlinear, the linearized constraint matrix J_k (4.3) is quite sparse, with on average a little under 6 nonzero row entries per column. Treating it as a dense matrix could result in a 'bump' of 79 columns, which is clearly undesirable. A number of minor discrepancies between Biggs and Laughton's paper and the original statement of the problem [53] were resolved by using the original data.

5.6. Launch vehicle design

This problem appears in Bracken and McCormick's book on nonlinear programming applications [11] and also appears in [49]. There are 12 linear constraints and 10 nonlinear constraints, and the Jacobian of the nonlinear constraints is quite sparse (density 23%), yielding an overall matrix density of 15%. All 25 variables are nonlinear.

5.7. Quartic with quartic constraints

This problem appears in Pierre and Lowe [38]. Only a few terms are quartic, the remainder being predominantly quadratic. It has 20 variables (all nonlinear) and 17 constraints, 13 of which are nonlinear.

5.8. Dembo No. 7

This is one of Dembo's set of 8 Geometric Programming test problems [15]; in particular, it required the most computation time in Dembo's results. Other authors have reported difficulty with the problem [13, 42]. There are 16 variables (all nonlinear) and 19 general constraints (11 of them nonlinear). The solution has a few primal and dual degeneracies, but it is essentially at a vertex of the constraint space (i.e., a vertex of the final linearization).

5.9. Wright No. 4 [55]

This is a highly nonlinear non-convex problem for which four local minima have been determined.

$$\begin{aligned}\text{minimize}\quad & (x_1-1)^2+(x_1-x_2)^2+(x_2-x_3)^3+(x_3-x_4)^4+(x_4-x_5)^4,\\ \text{subject to}\quad & x_1+x_2^2+x_3^3=2+3\sqrt{2},\\ & x_2-x_3^2+x_4=-2+2\sqrt{2},\\ & x_1x_5=2.\end{aligned}$$

Starting points:
(a) (1, 1, 1, 1, 1),
(b) (2, 2, 2, 2, 2),
(c) $(-1, 3, -\frac{1}{2}, -2, -3)$,
(d) $(-1, 2, 1, -2, -2)$,
(e) $(-2, -2, -2, -2, -2)$.

Local optima:

$$x^*(1) = (1.11663,\ 1.22044,\ 1.53779,\ 1.97277,\ 1.79110),$$
$$x^*(2) = (-2.79087,\ -3.00414,\ 0.205371,\ 3.87474,\ -0.716623),$$
$$x^*(3) = (-1.27305,\ 2.41035,\ 1.19486,\ -0.154239,\ -1.57103),$$
$$x^*(4) = (-0.703393,\ 2.63570,\ -0.0963618,\ -1.79799,\ -2.84336).$$

5.10. Wright No. 9 [55]

This is another highly nonlinear example.

$$\begin{aligned} \text{minimize}\quad & 10x_1x_4 - 6x_3x_2^2 + x_2x_1^3 + 9\sin(x_5 - x_3) + x_5^4x_4^2x_2^3, \\ \text{subject to}\quad & x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \le 20, \\ & x_1^2x_3 + x_4x_5 \ge -2, \\ & x_2^2x_4 + 10x_1x_5 \ge 5. \end{aligned}$$

Starting points:
(a) (1, 1, 1, 1, 1),
(b) (1.091, −3.174, 1.214, −1.614, 2.134).

Local optima:

$$x^*(1) = (-0.0814522,\ 3.69238,\ 2.48741,\ 0.377134,\ 0.173983),$$
$$x^*(2) = (1.47963,\ -2.63661,\ 1.05468,\ -1.61151,\ 2.67388).$$

With the barrier trajectory algorithm, Wright [55] obtained convergence to $x^*(1)$ from (a) and convergence to $x^*(2)$ from (b). Note that starting point (b) was originally chosen to cause difficulty for the barrier algorithm and other methods that retain feasibility throughout.

5.11. Optimal control

This problem investigates the optimal control of a spring, mass and damper system. It is adapted from [44]. While it is acknowledged that there may be simpler ways of solving the problem by taking specific advantage of the nature of the constraints, it serves the present purpose of providing a large, sparse test problem.

$$\text{minimize}\quad f(x, y, u) = \tfrac{1}{2}\sum_{t=0}^{T} x_t^2,$$

$$\text{subject to}\quad \left.\begin{aligned} & x_{t+1} = x_t + 0.2y_t \\ & y_{t+1} = y_t - 0.01y_t^2 - 0.004x_t + 0.2u_t \\ & -0.2 \le u_t \le 0.2 \\ & y_t \ge -1.0 \end{aligned}\right\}\quad t = 0, \ldots, T-1$$

$$x_0 = 10,\ y_0 = 0,\ y_T = 0.$$

Starting point: $x_t = 0$, $y_t = -1$, $t = 1, \ldots, T$.

For the results below we set $T = 100$. There are thus 202 nonlinear variables ($x_t, y_t, t = 0, \ldots, 100$) and 100 linear variables ($u_t, t = 0, \ldots, 99$). There are also 100 quadratic constraints and 100 linear constraints. Note that the nonlinear constraints are very sparse, with only 4 nonzeros per row.

The solution is dominated to a large extent by the lower bound on y_t; the optimal y_t decreases with increasing t and remains at -1.0 from $t = 20$ to $t = 40$, and then increases again, goes slightly positive and settles to 0.0 at $t = 100$. The corresponding values of x_t can be calculated directly from the linear constraint $x_{t+1} = x_t + 0.2y_t$. The optimal value of the objective is $\frac{1}{2}\|x\|^2 = 1186.382$.

5.12. *Economic growth model* (Manne [31])

This is a model for optimizing aggregate consumption over time. The variables are C_t, I_t, and K_t which represent consumption, investment and capital in time period t for $t = 1, \ldots, T$.

Utility function:

$$\text{maximize} \sum_{t=1}^{T} \beta_t \log_e C_t.$$

Nonlinear constraints:

$$\alpha_t K_t^b \geq C_t + I_t, \quad 1 \leq t \leq T.$$

Linear constraints:

$$K_{t+1} \leq K_t + I_t, \quad 1 \leq t < T,$$
$$gK_T \leq I_T.$$

Bounds:

$$K_1 = I_0 + K_0,$$
$$\left.\begin{array}{l} K_t \geq I_0 + K_0 \\ C_t \geq C_0 \\ I_t \geq I_0 \\ I_t \leq (1.04)^t I_0 \end{array}\right\} \quad 1 \leq t \leq T.$$

Data:

$$\beta = 0.95, \qquad b = 0.25, \qquad g = 0.03,$$
$$C_0 = 0.95, \qquad I_0 = 0.05, \qquad K_0 = 3.0,$$
$$\beta_t = \beta^t \quad \text{except } \beta_T = \beta^T/(1-\beta),$$
$$\alpha_t = \alpha(1+g)^{(1-b)t} \quad \text{where } \alpha = (C_0 + I_0)/K_0^b.$$

With b in the range $[0, 1]$, this is a convex program with separable nonlinear functions. It should therefore be a useful test problem for more specialized convex programming algorithms.

For test purposes we have used $T = 100$, which gives a problem with 200 general constraints and 300 variables. The optimal value of the utility function is 9.287547. All general constraints are active at the solution, and the first 74 upper bounds on I_t are also active.

It should be mentioned that specialized methods are known to economists for solving this problem, with and without the upper bounds ('absorptive capacities') on the variables I_t. However, in more practical circumstances the model would be imbedded in a much larger model for which an analytical solution is not known. If the larger model contains no additional nonlinearities, the performance of MINOS/AUGMENTED should degrade only in proportion to the problem size.

6. Results and discussion

MINOS/AUGMENTED is implemented in standard FORTRAN. Various options can be selected at run-time by means of a SPECS file, and initial vectors x_0, y_0, and λ_0 can be specified in the MPS file containing the constraint data.

For the present results, some components of x_0 were specified to match the given starting point, if any. (The corresponding variables were initially superbasic at the specified values.) Any remaining variables in $[x_0, y_0]$ were subjected to a normal CRASH start, in which a triangular basis is selected from the associated columns of the Jacobian matrix. (The resulting initial values for these variables are not predictable.) The default value $\lambda_0 = 0$ was used. The following parameter values were also specified:

LINESEARCH PARAMETER ETA	$= 0.1$	(moderately accurate search),
RADIUS OF CONVERGENCE	$= 0.01$	(ϵ_c in Section 3.5),
ROW TOLERANCE	$= 10^{-6}$	(ϵ_r in Section 4.8),
MINOR ITERATIONS LIMIT	$= 40$	(not active on small examples).

Run-times are reported below in order to allow comparison among various algorithmic options. For the Lagrangian algorithm, the item labeled 'Functions' means the number of times the objective and the constraints and all of their gradients were evaluated. For the Newton procedure, 'Functions' means the number of times the *objective* and its gradient were evaluated; the number of constraint and Jacobian evaluations is equal to the number of major iterations.

6.1. *Problems 5.1–5.8*

These examples were solved on a CDC Cyber 70. In all cases, convergence was obtained with zero penalty parameter ρ. The results are summarized in Table 1. In general the partial completion option converged more rapidly than

Table 1

Summary of results for Test Problems 5.1–5.8

Problem	5.1(a) Colville No. 2				5.1(b) Colville No. 2				5.2(a) Colville No. 3		5.2(b) Colville No. 3		5.3 Colville No. 8	
Method	N	L	N	L	N	L	N	L	N	L	N	L	N	L
Completion	P.C.		F.C.		P.C.		F.C.		P.C.		F.C.		P.C.	
Major itns	10	4	10	4	4	4	9	4	3	3	3	3	3	3
Total itns	31	41	58	43	32	27	59	38	5	5	4	4	11	10
Functions	52	65	93	77	54	42	86	55	7	7	6	6	31	54
Time (s)	3.18	3.58	4.43	3.52	4.38	3.07	4.15	3.07	0.91	0.97	0.91	0.91	1.36	2.05

Problem	5.4 Powell			5.5 Power Scheduling		5.6 Launch Vehicle		5.7 Quadratic	
Method	N	L	L	N	L	N	L	N	L
Completion	F.C.	F.C.	P.C.	P.C.		P.C.		P.C.	
Major itns		4	6		5	3	3	4	27
Total itns	*	18	10	*	100	26	26	41	91
Functions		26	17		69	60	60	73	147
Time (s)		1.53	1.53		38.9	6.65	8.13	5.61	20.1

5.8, Method L, F.C. Dembo No. 7		
Minor itns limit	3	40
Major itns	6	7
Total itns	30	55
Functions	19	66
Time (s)	2.27	3.56

N Newton strategy ($\lambda_k = 0, \rho = 0$).
L Lagrangian algorithm, $\rho = 0$.
P.C. Partial Completion.
F.C. Full Completion.
* Non-convergence.

full completion. However, Example 5.4 illustrates that fewer major iterations are likely if subproblems are solved accurately once the correct subspace has been identified. This was observed in several other cases and is probably explained by the discussion of $\boldsymbol{\lambda}_k$ in Section 3.3. In terms of total run-time the Newton strategy was often superior, but it failed to converge on Problems 5.4 and 5.5. This deficiency becomes more prominent in the examples below.

Problem 5.8 was run with two different MINOR ITERATIONS limits (3 and 40, which could have been 3 and 15 without altering the comparison). The results illustrate that terminating major iterations early can sometimes help rather than hinder.

6.2. *Problems 5.9–5.12*

The results for these examples were obtained on an IBM 370/168 using the FORTRAN H Extended (Enhanced) compiler with full optimization (OPT(3)). Computation was performed in double precision, but the constraint data were stored in single precision, including J_k in the linearization of $f(x)$. This limits the achievable constraint error to about 10^{-6}, but that is usually adequate in practice. Full completion was used throughout.

Problem 5.9 (Wright No. 4)

This highly nonlinear problem illustrates the difficulties discussed in Section 3.4 when no penalty term is used. The Newton strategy and the Lagrangian algorithm with $\rho = 0$ both gave rise to subproblems which changed radically each major iteration.

The results using the Lagrangian algorithm with $\rho = 10$ and $\rho = 100$ are shown in Table 2.

Infeasible subproblems were encountered with starting point (e) using the penalty parameter $\rho = 10$, but the procedure discussed in Section 4.6 successfully overcame this difficulty. Case (e) was also the only one for which the larger ρ was important in stabilizing progress from the starting point to the solution.

6.3. *Problem 5.10 (Wright No. 9)*

Again the Newton strategy and the Lagrangian algorithm with $\rho = 0$ failed to converge. Results for the Lagrangian algorithm with $\rho = 10$ and $\rho = 100$ are shown in Table 3.

A value of $\rho = 10$ is almost too low for starting point (b), the subproblem solutions changing radically as they did for $\rho = 0$, but finally converging to a new local optimum, $x^*(3) = (-0.07427, -3.69170, 2.48792, 0.37693, 0.18446)^T$.

In general the results for these last two problems are inferior to those obtained by Murray and Wright [34, 55] with their trajectory algorithms, in terms of total function evaluations. However, the difference is less than a factor of 4 in all cases, and averaged 2.2 over the seven starting points.

Table 2

Results for Test Problem 5.9 (Wright No. 4)

Starting point	(a)		(b)		(c)		(d)		(e)	
ρ	100	10	100	10	100	10	100	10	100	10
Major itns	9	7	9	6	6	6	5	5	7	12
Total itns	47	33	41	25	24	26	22	21	30	67
Functions	84	53	70	40	46	42	37	30	55	120
Solution	$x^*(1)$	$x^*(1)$	$x^*(1)$	$x^*(1)$	$x^*(4)$	$x^*(4)$	$x^*(3)$	$x^*(3)$	$x^*(2)$	$x^*(3)$

Table 3
Results for Test Problem 5.10 (Wright No. 9)

Starting point	(a)		(b)	
ρ	100	10	100	10
Major itns	12	9	5	19
Total itns	92	71	32	201
Functions	183	146	61	386
Solution	$x^*(1)$	$x^*(1)$	$x^*(2)$	$x^*(3)$

Table 4
Results for Test Problem 5.11 (optimal control)

Method	N	L ($\rho = 0$)
Major itns	6	6
Total itns	254	247
Functions	213	203
Time (s)	10.55	11.56

6.4. Problem 5.11 (Optimal control)

Despite the large size of this problem both the Newton strategy and the Lagrangian algorithm with $\rho = 0$ converged rapidly. Recall that procedure N evaluates the constraint functions only once per major iteration (in this case 6 times compared to 203 times for procedure L). If $f(x)$ were more costly to compute, the time advantage would be that much greater. The Lagrangian algorithm displayed insensitivity to nonzero values of ρ in the range 0.01–10.0, taking the same iterations and calculations as shown in Table 4.

6.5. Problem 5.12 (Economic growth)

On this example the Newton strategy led to oscillation and no convergence. The Lagrangian algorithm did converge rapidly with $\rho = 0$. Without the upper bounds $I_t \leq (1.04)^t I_0$ it required 11 major iterations, 355 minor iterations, 859 function calculations and 34.3 seconds, and there were 99 superbasic variables at the optimal solution. However when these bounds were imposed, the optimal number of superbasics was only 25 and convergence was obtained in 7 major iterations, 183 minor iterations, 497 function calculations and 11.9 seconds. This illustrates the gains that are made when the presence of constraints reduces the dimensionality of the optimal subspace.

As an experiment on the effect of ρ on the rate of convergence the problem was solved several times with different values of ρ in the range $10^{-4} \leq \rho \leq 1.0$.

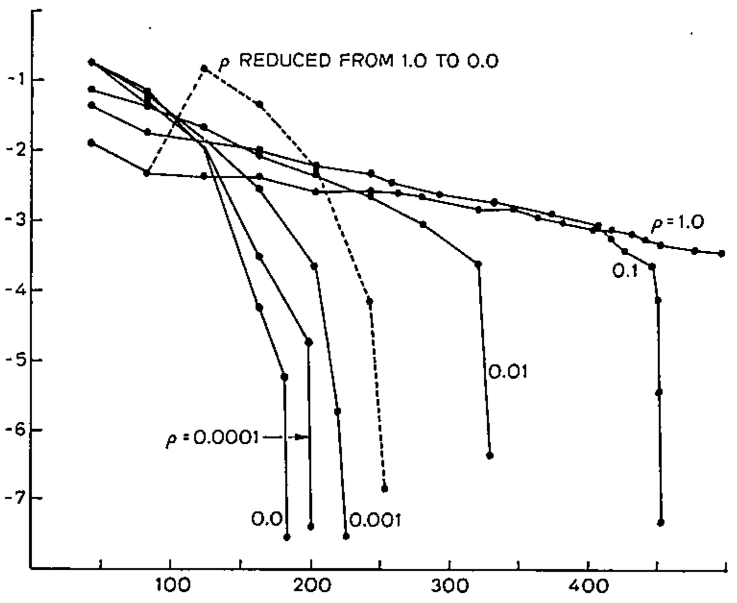

Fig. 1. Solution of Problem 5.12 (Economic growth) using various values for the penalty parameter ρ. The constraint violation, $\log_{10}\|f(x)+A_1y-b_1\|_\infty$, is plotted against minor iteration number. Dots signify the end of each major iteration.

(The tolerance ϵ_c for dropping ρ was set to zero, thus forcing ρ to stay constant for each run.) The results are shown in Fig. 1. This is a plot of minor iterations versus log of the constraint violation or 'row error', $\log_{10}\|f(x)+A_1y-b_1\|_\infty$, immediately following relinearization. The dots represent the end of each major iteration. Initially these occur every 40 iterations (the MINOR ITERATIONS limit) but later subproblems were solved accurately before the limit was reached. In fact the number of minor iterations reduces rapidly to only one or two per subproblem as convergence is approached. This behavior was also observed in all of the preceding examples.

It can be seen that higher values of ρ give lower row errors at the end of the first major iteration (as we would expect), but they lead to consistently slower convergence. It is interesting to note that rapid convergence does occur ultimately in all cases (except for $\rho = 1.0$ which was terminated after 500 iterations). However, this is not until the correct active constraints have been identified, by which time x_k is very close to the optimum and the penalty term is having a negligible effect on the Lagrangian and its reduced gradient.

These results confirm that the benefit of Robinson's proof of quadratic convergence for the case $\rho = 0$ cannot be realized unless ρ is actually reduced to zero as soon as precautions allow.

The dotted line in Fig. 1 shows the result for $\rho = 1.0$ (the worst case) with ϵ_c set to 0.01, allowing ρ to be reset to zero at the start of major iteration 3. (Note that on the highly nonlinear Examples 5.9 and 5.10, the same value of ϵ_c ensured that ρ was not set to zero until a near optimum point was reached. This was the desired effect since the multiplier estimates λ_k were changing radically in the early major iterations.)

It will be seen in Fig. 1 that the row error increases sharply once ρ is reset to zero. This is consistent with the algorithm suddenly being free to take a large step. We could therefore regard the first two major iterations as having served to verify that the problem is only mildly nonlinear.

6.6. Other results

Successive linear programming (SLP) algorithms have been used in industry for many years [4, 7, 23] on problems of the type discussed here. Some informal comparisons [32] on the Colville problems indicate that SLP is likely to require more iterations and function evaluations than the Lagrangian algorithm given here. However, it is clearly unwise to draw any firm conclusion from a few small test cases. We hope that comparison with SLP on a large model will be possible in the near future.

Generalized reduced-gradient (GRG) algorithms [1] have also been in use for many years. In particular, the large-scale implementation MINOS/GRG [27, 30] has been applied to a variant of Manne's economic growth model (Problem 5.12). The results reported in [30] show that the GRG algorithm required very many function evaluations for the case $T = 40$ (considerably more than the numbers given above for the Lagrangian algorithm in the larger case $T = 100$). This is consistent with GRG's emphasis on retaining feasibility at the expense of many short steps that follow the curvature of the constraints.

Table 5
Iterations for air pollution model

Major iteration k	Iterations to find a feasible point for subproblem S_k	Total minor iterations for S_k	Constraint violation after termination of S_k	$\frac{\|\lambda_{k+1} - \lambda_k\|}{1 + \|\lambda_{k+1}\|}$
1	1554	0	0	1554
2	0	1000	0.50	0
3	289	1000	0.27	0.02
4	11	600	0.58	0.28
5	126	626	0.51	0.45
6	12	204	0.29	0.40
7	7	172	0.15	0.49
8	1	167	0.067	0.38
9	4	196	0.022	0.27
10	0	82	0.0035	0.21
11	0	22	0.00041	0.05
12	0	2	0.0000058	0.002
13	0	1	—	—

6.7. *A practical application*

To date, the largest nonlinear programs solved by MINOS/AUGMENTED have come from an energy production model concerned with air pollution control [28]. One example of the model involved about 850 constraints and 4150 variables. The objective function was nonlinear in 225 of the variables (a sum of terms of the form e^{x_i} and e^{x_i/x_j}), and 32 of the constraints were quadratic in 78 of those variables (ellipsoids of the form $x^TE_ix \leq \beta_i$ with E_i positive semidefinite).

Some statistics follow for the solution of this problem [28]. A cold start was used with default values for all parameters, except that the MINOR ITERATIONS limit was set to the unusually high value of 1000.

Major iterations (number of subproblems)	13
Minor iterations	5626
Objective function and gradient evaluations	5955
Constraint function and gradient evaluations	5968
Active nonlinear constraints at optimum	12
Superbasic variables at optimum	18
CPU time on a DEC VAX 11/780	63 minutes

Details for each major iteration are given in Table 5, where S_k denotes the kth subproblem. The high MINOR ITERATIONS limit probably just reduced the number of major iterations, without having a substantial effect on the total work required. The first subproblem was terminated at the first feasible point, since the limit on minor iterations had then been exceeded. By chance, this point was also feasible for the original problem. As a result, the penalty parameter ρ was then reduced to zero, the ideal value for a convex program such as this. (In general, however, it seems clear that the criterion for reducing ρ should not take effect if the most recent subproblem was terminated short of optimality.)

The next two subproblems were principally concerned with optimizing the objective function, without much regard for the nonlinear constraints. Thereafter, the constraint violation decreased steadily and, ultimately, quadratically. (The quantity shown is the largest violation, normalized by the size of the corresponding constraint gradient, $\|e_i^T[J_{k+1} \ \ A_1]\|$, and by the solution size, $1+\|[x_k, y_k]\|$.) The Lagrange multiplier estimates converged more slowly as might be expected, but it is known that this does not inhibit the final rapid convergence of x_k.

7. Conclusions

Many real-life optimization problems originate as linear programming models that are not quite linear; i.e., they contain simple, differentiable functions in the constraints and objective, but otherwise the constraint set is large, sparse and linear. For such problems the Jacobian matrix is also likely to be sparse, and the strategy of solving a sequence of linearly constrained subproblems has many advantages. This is clear from the results obtained for the larger test problems above.

For convex problems the Lagrangian term in the objective of the subproblems is usually necessary to ensure convergence. The Newton strategy performed adequately without it on some occasions, but in general the saving in run time will seldom be substantial.

For non-convex problems, both the Lagrangian and the penalty term are clearly useful but the actual choice of the penalty parameter ρ remains a critical decision for the user. In practice, optimization problems are often solved repeatedly on a production basis. In this situation it is worthwhile experimenting with different values of the parameters and tolerances (and perhaps with the Newton strategy). However, the case of an isolated problem with unknown characteristics is no less important. A conservative (high) value of ρ is then virtually mandatory. One of our aims has been to minimize the risk of subsequent slow convergence by determining an opportune time to reduce ρ to zero. The analysis in Section 3.4 has suggested a practical procedure for achieving this aim. In particular, the 'radius of convergence' tolerance ϵ_c (applied to both the constraint violation and the relative change in the estimates of $\boldsymbol{\lambda}$) allows early removal of ρ in moderately nonlinear cases but otherwise retains it until convergence to a local solution is imminent.

The results reported here should provide a benchmark for measuring the performance of other algorithms and their implementations. Clearly no single algorithm can be expected to perform uniformly better than all others in such a diverse field as nonlinearly constrained optimization. As it happens, MINOS/AUGMENTED has proved to be reasonably efficient on small, highly nonlinear problems, but more importantly, it represents an advance in the development of general-purpose software for large-scale optimization. Future research could include an investigation of the following:

(a) Comparison with other large-scale algorithms such as SLP.

(b) An algorithm for adjusting the penalty parameter between subproblems.

(c) Alternative means for obtaining multiplier estimates $\boldsymbol{\lambda}_k$.

(d) Use of a merit function to evaluate the connection between consecutive subproblems; e.g., to allow interpolation between the points $(x_k, \boldsymbol{\lambda}_k)$ and $(x_{k+1}, \boldsymbol{\lambda}_{k+1})$ of the present algorithm.

(e) Methods for the case where some or all of the gradient functions are not available.

Acknowledgement

The authors wish to thank Noel Diaz, Philip Gill, Harvey Greenberg, Charles Kolstad, Alan Manne, Walter Murray, Ben Rosen, Ken Schofield, John Tomlin, Wesley Winkler, and Margaret Wright for various guiding comments and assistance with the test problems.

Availability

MINOS/AUGMENTED is distributed by the Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA 94305. Most of the source code is portable and can be installed on any reasonably large system that includes a FORTRAN IV compiler. The distribution tape contains source code for most of the current scientific computers. It also contains some routines [43] to facilitate the solution of sequences of related problems.

Added in proof

Since this paper was submitted, some particularly successful results have been obtained on electric power scheduling (optimal power flow) problems considerably larger than problem 5.5 above. This is an application where a good starting basis presents itself naturally. Some approximate statistics for a 600-bus Q-dispatch case follow [56]: 1200 nonlinear constraints, 1300 variables, 10 000 Jacobian nonzeros; 15 major iterations, 370 minor iterations, 700 function evaluations, 1 hour of CPU time on a DEC VAX 11/780.

References

[1] J. Abadie and J. Carpentier, "Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints", in: R. Fletcher, ed., *Optimization* (Academic Press, London, 1969) pp. 37–49.

[2] J. Abadie and J. Guigou, "Numerical experiments with the GRG method", in: J. Abadie, ed., *Integer and nonlinear programming* (North-Holland, Amsterdam, 1970) pp. 529–536.

[3] K.J. Arrow and R.M. Solow, "Gradient methods for constrained maxima, with weakened assumptions", in: K.J. Arrow, L. Hurwicz and H. Uzawa, eds., *Studies in linear and nonlinear programming* (Stanford University Press, Stanford, CA, 1958) pp. 166–176.

[4] T.E. Baker and R. Ventker, "Successive linear programming in refinery logistic models", presented at ORSA/TIMS joint national meeting, Colorado Springs, CO (November 1980).

[5] A.S.J. Batchelor and E.M.L. Beale, "A revised method of conjugate gradient approximation programming", presented to the Ninth International Symposium on Mathematical Programming (Budapest, 1976).

[6] E.M.L. Beale, "A conjugate gradient-method of approximation programming", in: R.W. Cottle and J. Krarup, eds., *Optimization methods for resource allocation* (English Universities Press, 1974) pp. 261–277.

[7] E.M.L. Beale, "Nonlinear programming using a general Mathematical Programming System", in: H.J. Greenberg, ed., *Design and implementation of optimization software* (Sijthoff and Noordhoff, The Netherlands, 1978) pp. 259–279.

[8] J.F. Benders, "Partitioning procedures for solving mixed variables programming problems", *Numerische Mathematik* 4 (1962) 238–252.

[9] M.J. Best, J. Bräuninger, K. Ritter and S.M. Robinson, "A globally and quadratically convergent algorithm for general nonlinear programming problems", *Computing* 26 (1981) 141–153.

[10] M.C. Biggs and M.A. Laughton, "Optimal electric power scheduling: a large nonlinear test problem solved by recursive quadratic programming", *Mathematical Programming* 13 (1977) 167–182.

[11] J. Bracken and G.P. McCormick, *Selected applications for unconstrained and linearly constrained optimization* (Wiley, New York, 1968) pp. 58–82.

[12] A.R. Colville, "A comparative study on nonlinear programming codes", Report 320–2949 (1968), IBM New York Scientific Center, New York.

[13] I.D. Coope and R. Fletcher, "Some numerical experience with a globally convergent algorithm for nonlinearly constrained optimization", *Journal of Optimization Theory and Applications* 32 (1) (1980) 1–16.

[14] G.B. Dantzig, *Linear programming and extensions* (Princeton University Press, New Jersey, 1963).

[15] R.S. Dembo, "A set of geometric programming test problems and their solutions", *Mathematical Programming* 10 (1976) 192–213.

[16] L.F. Escudero, "A projected Lagrangian method for nonlinear programming", Report G320–3401 (1980), IBM Palo Alto Scientific Center, CA.

[17] U.M. Garcia-Palomares and O.L. Mangasarian, "Superlinearly convergent quasi-Newton algorithms for nonlinearly constrained optimization problems", *Mathematical Programming* 11 (1976) 1–13.

[18] P.E. Gill and W. Murray, "Newton-type methods for unconstrained and linearly constrained optimization", *Mathematical Programming* 7 (1974) 311–350.

[19] P.E. Gill and W. Murray, "Safeguarded steplength algorithms for optimization using descent methods", Report NAC 37 (1974), National Physical Laboratory, Teddington, England.

[20] P.E. Gill and W. Murray, Private communication (1975), National Physical Laboratory, Teddington, England.

[21] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, "Two steplength algorithms for numerical optimization", Report SOL 79–25 (1979), Department of Operations Research, Stanford University, CA. (Revised 1981.)

[22] H.J. Greenberg, Private communication (1979), Department of Energy, Washington, DC.

[23] R.E. Griffith and R.A. Stewart, "A nonlinear programming technique for the optimization of continuous processing systems", *Management Science* 7 (1961) 379–392.

[24] E. Hellerman and D.C. Rarick, "The partitioned preassigned pivot procedure", in: D.J. Rose and R.A. Willoughby, eds., *Sparse matrices and their applications* (Plenum Press, New York, 1972) pp. 67–76.

[25] M.R. Hestenes, "Multiplier and gradient methods", *Journal of Optimization Theory and Applications* 4 (1969) 303–320.

[26] R.J. Hillestad and S.E. Jacobsen, "Reverse convex programming", Research paper (1979), RAND Corporation, Santa Monica, CA, and Department of System Science, University of California, Los Angeles, CA.

[27] A. Jain, L.S. Lasdon and M.A. Saunders, "An in-core nonlinear mathematical programming system for large sparse nonlinear programs", presented at ORSA/TIMS joint national meeting, Miami, FL (November 1976).

[28] C.D. Kolstad, Private communication (1980), Los Alamos Scientific Laboratory, New Mexico.

[29] L.S. Lasdon, *Optimization theory for large systems* (Macmillan, New York, 1970).

[30] L.S. Lasdon and A.D. Waren, "Generalized reduced gradient software for linearly and nonlinearly constrained problems", in: H.J. Greenberg, ed., *Design and implementation of optimization software* (Sijthoff and Noordhoff, The Netherlands, 1978) pp. 363–396.

[31] A.S. Manne, Private communication (1979), Stanford University, CA.

[32] P.H. Merz, Private communication (1980), Chevron Research Company, Richmond, CA.
[33] R.R. Meyer, "The validity of a family of optimization methods", *SIAM Journal on Control* 8 (1970) 41–54.
[34] W. Murray and M.H. Wright, "Projected Lagrangian methods based on trajectories of penalty and barrier functions", Report SOL 78–23 (1978), Department of Operations Research, Stanford University, CA.
[35] B.A. Murtagh and M.A. Saunders, "Large-scale linearly constrained optimization", *Mathematical Programming* 14 (1978) 41–72.
[36] B.A. Murtagh and M.A. Saunders, "MINOS user's guide", Report SOL 77–9 (1977), Department of Operations Research, Stanford University, CA.
[37] B.A. Murtagh and M.A. Saunders, "MINOS/AUGMENTED user's manual", Report SOL 80–14 (1980), Department of Operations Research, Stanford University, CA.
[38] D.A. Pierre and M.J. Lowe, *Mathematical programming via augmented Lagrangians* (Addison-Wesley, Reading, MA, 1975) pp. 239–241.
[39] M.J.D. Powell, "A method for nonlinear constraints in optimization problems", in: R. Fletcher, ed., *Optimization* (Academic Press, New York, 1969) pp. 283–297.
[40] M.J.D. Powell, "Algorithms for nonlinear constraints that use Lagrangian functions", *Mathematical Programming* 14 (1978) 224–248.
[41] M.J.D. Powell, "A fast algorithm for nonlinearly constrained optimization calculations", presented at the 1977 Dundee Conference on Numerical Analysis, Dundee, Scotland.
[42] M.J.D. Powell, "Constrained optimization by a variable metric method", Report 77/NA6 (1977), Department of Applied Mathematics and Theoretical Physics, Cambridge University, England.
[43] P.V. Preckel, "Modules for use with MINOS/AUGMENTED in solving sequences of mathematical programs", Report SOL 80-15 (1980), Department of Operations Research, Stanford University, CA.
[44] P.S. Ritch, "Discrete optimal control with multiple constraints I: constraint separation and transformation technique", *Automatica* 9 (1973) 415–429.
[45] S.M. Robinson, "A quadratically convergent algorithm for general nonlinear programming problems", *Mathematical Programming* 3 (1972) 145–156.
[46] J.B. Rosen, "Convex partition programming", in: R.L. Graves and P. Wolfe, eds., *Recent advances in mathematical programming* (McGraw-Hill, New York, 1963) pp. 159–176.
[47] J.B. Rosen, "Iterative solution of nonlinear optimal control problems", *SIAM Journal on Control* 4 (1966) 223–244.
[48] J.B. Rosen, "Two-phase algorithm for nonlinear constraint problems", in: O.L. Mangasarian, R.R. Meyer and S.M. Robinson, eds., *Nonlinear programming* 3 (Academic Press, London, 1978) pp. 97–124.
[49] B.C. Rush, J. Bracken and G.P. McCormick, "A nonlinear programming model for launch vehicle design and costing", *Operations Research* 15 (1967) 185–210.
[50] R.W.H. Sargent and B.A. Murtagh, "Projection methods for nonlinear programming", *Mathematical Programming* 4 (1973) 245–268.
[51] M.A. Saunders, "A fast, stable implementation of the simplex method using Bartels-Golub updating", in: J.R. Bunch and D.J. Rose, eds., *Sparse matrix computations* (Academic Press, New York, 1976) pp. 213–226.
[52] M.A. Saunders, "MINOS system manual", Report SOL 77–31 (1977), Department of Operations Research, Stanford University, CA.
[53] C.M. Shen and M.A. Laughton, "Determination of optimum power system operating conditions under constraints", *Proceedings of the Institute of Electrical Engineers* 116 (1969) 225–239.
[54] P. Wolfe, "Methods of nonlinear programming", in: J. Abadie, ed., *Nonlinear programming* (North-Holland, Amsterdam, 1967) pp. 97–131.
[55] M.H. Wright, "Numerical methods for nonlinearly constrained optimization", SLAC Report No. 193 (1976), Stanford University, CA (Ph.D. Dissertation).
[56] R.C. Burchett, Private communication (1981), General Electric Company, Schenectady, NY.