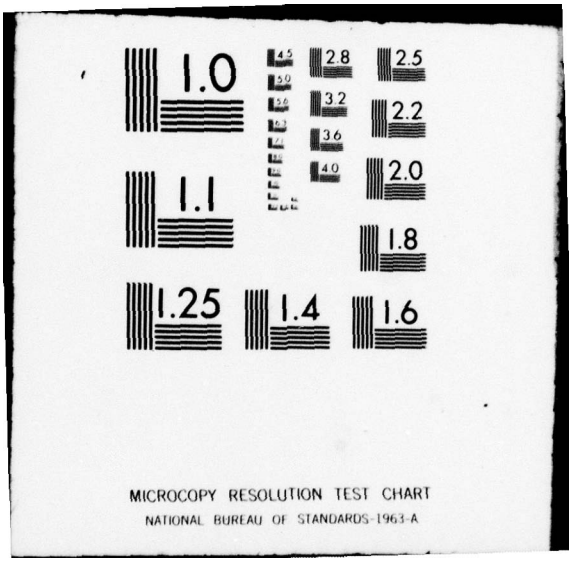
AD-A063 335 STANFORD UNIV CALIF SYSTEMS OPTIMIZATION LAB F/G 12/1
PROJECTED LAGRANGIAN METHODS BASED ON THE TRAJECTORIES OF PENAL--ETC(U)
NOV 78 W MURRAY, M H WRIGHT N00014-75-C-0267
UNCLASSIFIED SOL-78-23 NL

1 OF
AD
A063335

END
DATE
FILMED
3-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Systems
Optimization
Laboratory

LEVEL II

AD A063335

DDC FILE COPY

DDC
RECEIVED
JAN 17 1979
A

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Department of Operations Research
Stanford University
Stanford, CA 94305

79 01 16 120

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
Stanford University
Stanford, California
94305

(6) PROJECTED LAGRANGIAN METHODS BASED ON THE TRAJECTORIES OF PENALTY AND BARRIER FUNCTIONS

by

(10) Walter Murray* Margaret H. Wright

(9) TECHNICAL REPORT SOL 78-23

Nov 78

(11) (12) 71p.

(14) SOL-78-23

(15) NP00014-75-C-0267
NSF-MCS76-20019

*Division of Numerical Analysis and Computer Science, National Physical Laboratory, Teddington, England.

Research and reproduction of this report were partially supported by the National Science Foundation Grants ~~MCS76-20019 A01~~ and ENG77-06761; the Office of Naval Research Contract ~~N00014-75-C-0267~~; and the Department of Energy Contract EY-76-S-03-0326 PA #18.

Abstract

This report contains a complete derivation and description of two algorithms for nonlinearly constrained optimization which are based on properties of the solution trajectory of the quadratic penalty function and the logarithmic barrier function. The methods utilize the penalty and barrier functions only as merit functions, and do not generate iterates by solving a sequence of ill-conditioned problems. The search direction is the solution of a simple, well-posed quadratic program (QP), where the quadratic objective function is an approximation to the Lagrangian function; the steplength is based on a sufficient decrease in a penalty or barrier function, to ensure progress toward the solution.

The penalty trajectory algorithm was first proposed by Murray in 1969; the barrier trajectory algorithm, which retains feasibility throughout, was given by Wright in 1976. Here we give a unified presentation of both algorithms, and indicate their relationship to other QP-based methods. Full details of implementation are included, as well as numerical results that display the success of the methods on non-trivial problems.

1. Introduction

1.1. Statement of Problem and Notation

The problem of concern in this paper is the following:

P1:

$$\text{minimize} \quad F(x), \qquad x \in E^n ,$$

$$\text{subject to} \quad c_i(x) \geq 0 , \qquad i = 1, 2, \ldots, m ,$$

where $F(x)$ and $\{c_i(x)\}$ are prescribed nonlinear functions. The function $F(x)$ is usually termed the objective function, and the set $\{c_i(x)\}$ is the set of constraint functions. It will be assumed for simplicity that F and $\{c_i\}$ are twice continuously differentiable, although the methods to be discussed will cope with occasional discontinuities in the derivatives.

Equality constraints are not included in the statement of problem P1, in order to avoid the introduction of additional notation; the algorithms to be discussed can deal with equality constraints in a straightforward way.

A local minimum of P1 will be denoted by $\overset{*}{x}$. Suppose that t constraints are exactly satisfied at $\overset{*}{x}$, and let $\hat{c}(x)$ denote the vector of these active constraint functions, so that:

$$\hat{c}(\overset{*}{x}) = 0 .$$

The first- and second-order Kuhn-Tucker conditions are assumed to be satisfied at $\overset{*}{x}$, i.e.:

1) There is a Lagrange multiplier $\overset{*}{\lambda}_i$ corresponding to each active constraint, and the vector of Lagrange multipliers, $\overset{*}{\lambda}$, satisfies:

$$g(\overset{*}{x}) - \hat{A}(\overset{*}{x})\overset{*}{\lambda} = 0 \qquad (1a)$$

$$\overset{*}{\lambda} \geq 0 , \qquad (1b)$$

where g is the gradient of F, and the columns of $\hat{A}$ are the gradients of the constraints active at $\overset{*}{x}$. The condition (1a) may alternatively be stated in terms of the Lagrangian function,

$$L(x,\lambda) \equiv F(x) - \lambda^T \hat{c}(x) ,$$

since (1a) implies that $\overset{*}{x}$ is a stationary point of the Lagrangian function with respect to x when $\lambda = \overset{*}{\lambda}$.

2) Let $Z(x)$ denote a matrix whose columns span the space of vectors orthogonal to the columns of $\hat{A}(x)$; and let $W(x,\lambda)$ denote the Hessian matrix of the Lagrangian function with respect to x, that is,

$$W(x,\lambda) \equiv G(x) - \sum_{j=1}^{t} \lambda_j \hat{G}_j(x) ,$$

where $G(x)$ is the Hessian matrix of $F(x)$, and $\hat{G}_j(x)$ is the Hessian matrix of the j-th active constraint function.

The second assumed condition is that the projected Hessian matrix of the Lagrangian function with optimal multipliers--the matrix $Z(\overset{*}{x})^T W(\overset{*}{x},\overset{*}{\lambda}) Z(\overset{*}{x})$--is positive definite.

1.2. Overview of Trajectory Algorithms

The problem P1 can not, in general, be solved explicitly, and iterative methods are therefore required. A popular approach during the past decade has been to transform the problem of solving P1 into that of solving a sequence of related unconstrained minimization sub-problems. The most common such transformation has been effected by the use of penalty and barrier function methods (for a detailed description, see, for example, Fiacco and McCormick, 1968, and Ryan, 1974). These methods will not be reviewed here, but their properties are relevant in the derivation of the algorithms to be discussed.

The quadratic penalty function (with respect to the problem P1) is defined by:

$$P(x,\rho) \equiv F(x) + \frac{\rho}{2} \sum_{i\in I(x)} (c_i(x))^2 \equiv F(x) + \frac{1}{2r} \sum_{i\in I(x)} (c_i(x))^2 \qquad (2)$$

$$\equiv P(x,r),$$

where I is a subset of the indices $\{1, 2, \ldots, m\}$ (usually, the set of constraints whose values are less than a small positive number); and ρ is a positive scalar, termed the penalty parameter. It is sometimes convenient to deal with the second formulation in (2), in terms of the parameter $r = 1/\rho$. Let $\overset{*}{x}_P(\rho)$ denote an unconstrained minimum of $P(x,\rho)$ with respect to x, with the same meaning for $\overset{*}{x}_P(r)$.

The logarithmic barrier function (with respect to the problem P1) is defined as:

$$B(x,r) \equiv F(x) - r \sum_{j=1}^{m} \ell n(c_j(x)) ,$$

where r is a positive scalar termed the barrier parameter. This barrier function is defined only at points at which all the constraints are strictly satisfied. Let $\overset{*}{x}_B(r)$ denote an unconstrained minimum of $B(x,r)$.

Under certain mild conditions, there exists $\hat{r} > 0$ such that for $r < \hat{r}$, $\overset{*}{x}_P(r)$ and $\overset{*}{x}_B(r)$ are continuous functions of r, and:

$$\lim_{r \to 0} \overset{*}{x}_P(r) = \overset{*}{x} ;$$

$$\lim_{r \to 0} \overset{*}{x}_B(r) = \overset{*}{x} .$$

The continuous paths of minima in E^n defined by $\overset{*}{x}_P(r)$ and $\overset{*}{x}_B(r)$ as r approaches zero are termed the penalty trajectory and barrier trajectory of approach to $\overset{*}{x}$, respectively.

Penalty and barrier function methods display the following good features: (1) the iterates follow a non-tangential approach to x^* along the trajectory, and hence the use of linear approximations to the constraint functions is justified, even close to the solution; (2) the sub-problem of minimizing a penalty or barrier function provides straightforward criteria to measure progress at each iteration; (3) a single parameter is varied to control convergence; (4) at points on the trajectory, a special relationship exists between the values of the constraint functions and the Lagrange multiplier estimates; (5) in the barrier function case, all estimates of the solution are feasible. However, these methods also suffer from certain theoretical and numerical defects. In particular, convergence to x^* is achieved in theory only by solving an infinite sequence of sub-problems. Furthermore, as r approaches zero the Hessian matrices of the penalty and barrier functions become increasingly ill-conditioned, and are singular in the limit (Murray, 1969a, 1971). This unavoidable ill-conditioning means that in practice the unconstrained sub-problems corresponding to successively smaller values of r are more difficult to solve; consequently, the value of r to be used in solving the first unconstrained sub-problem must be "large enough", and there is a practical limit to the rate at which r may be decreased.

By exploiting the properties of the trajectories of penalty and barrier function methods, the algorithms to be described in this paper overcome the disadvantages of such methods without incurring

additional difficulties. The "trajectory" algorithms are so named because they are based on using these properties to generate a sequence of points that lie in a neighborhood of the appropriate trajectory, and thereby mimic the approach to x^* of the minima from a penalty or barrier function method.

Analysis of the behavior of the points generated by a penalty or barrier function method reveals that a close approximation to a step along the trajectory toward x^* solves an equality-constrained quadratic program. The linear constraints of the quadratic program involve the gradients and values of the active constraints, and the current estimates of the penalty or barrier parameter and the Lagrange multipliers; the quadratic objective function is related to the Lagrangian function. Murray (1969a,b) obtained this result for the quadratic penalty function, and proposed the original penalty trajectory algorithm, wherein at each iteration the search direction is given by the solution of a quadratic program, and the step taken is based on a satisfactory decrease in the penalty function; the analogous result for the logarithmic barrier function, and the method based on its trajectory, were given by Wright (1976).

Trajectory methods consequently belong to the class of "projected Lagrangian" methods, whose defining characteristic is that they contain a linearly constrained sub-problem based on the Lagrangian function. Two significant virtues of this "projection" approach are: (1) because of the linear constraints, the minimization in the sub-problem takes

place only within a subspace of reduced dimensionality; (2) the constraints can be chosen so that the Lagrangian function has a local minimum at x^* in this subspace. These properties contrast to those of "augmented Lagrangian" methods, where minimization in the full n-dimensional space is required, and a penalty term must, in general, be included to make x^* a local minimum rather than a stationary point.

The first projected Lagrangian method was proposed by Wilson (1963); in Wilson's algorithm, the linearly constrained sub-problem was specialized to a quadratic program, whose objective function was a quadratic approximation to the Lagrangian function. Other authors have subsequently proposed variations on the primary idea of constructing the sub-problem so as to minimize the Lagrangian function only in an appropriately chosen subspace--Robinson (1972), Rosen and Kreuser (1972), Biggs (1972,1974), Garcia-Palomares and Mangasarian (1974), Han (1976,1977), Rosen (1977), Powell (1977a,b).

Despite the common feature of a linearly constrained sub-problem, trajectory methods may be distinguished from the other methods because the quadratic programming sub-problem in the former is derived from analysis of the penalty and barrier trajectories. The sub-problems posed in other projected Lagrangian methods are usually based on the application of Newton's method to the nonlinear equations satisfied at x^*. This derivation has the unfortunate consequence that the sub-problem may be meaningless outside a close neighborhood of x^*. Therefore, it is necessary to extend the algorithms to include safeguards that allow

progress toward $\overset{*}{x}$ from an arbitrary starting point; furthermore, there is no obvious way to measure progress. However, because it is possible to characterize a step toward the penalty or barrier trajectory without assuming that the current iterate is in a close neighborhood of $\overset{*}{x}$, the derivation of the trajectory methods does not display a stringent dependence on properties that hold only in such a neighborhood; in addition, the underlying penalty or barrier function provides a natural criterion for measuring (and ensuring) progress.

At each iteration of a trajectory method, the search direction is computed as a step toward some point on the desired trajectory, where the particular point to be aimed for depends on the current value of the penalty or barrier parameter. This parameter may be adjusted at every iteration; however, the choice of the parameter value is not critical, since a step to a neighborhood of a point on the trajectory corresponding to $\bar{r}$ is also in the neighborhood of a point corresponding to $(1+\varepsilon)\bar{r}$, where ε is small. The "target point" may be adjusted both within and between iterations, so that a poor choice can be quickly corrected. Ultimately the target point becomes arbitrarily close to $\overset{*}{x}$ as the iterates converge.

The direction of search in the trajectory algorithms is computed by a well-posed numerical procedure. It is not necessary for any of the iterates to lie exactly on the trajectory (as in a penalty or barrier function method). Moreover, the approach to the limit of the penalty or barrier parameter does not induce ill-conditioning, and the influence of this parameter becomes negligible in the vicinity of the solution.

It is of interest to compare how other algorithms for solving P1 find a point that satisfies both the following conditions, which hold at $\overset{*}{x}$:

i) a subset of the constraints hold as equalities;

ii) the gradient of the objective function is a non-negative linear combination of the gradients of these active constraints.

Algorithms such as the gradient reduction and gradient projection methods attempt to produce iterates for which (i) always holds, and then progressively to reduce the discrepancies in (ii). Penalty function, barrier function, and augmented Lagrangian methods obtain points that satisfy (ii) at the end of their inner iterations; each successive outer cycle is designed to decrease the violation of (i). By contrast, the iterates generated by trajectory methods typically display approximately the same degree of satisfaction of (i) and (ii), with neither relationship holding exactly.

2. The Penalty Trajectory Algorithm

2.1. Derivation

Only an abbreviated derivation of the penalty trajectory algorithm will be presented here, in order to give the underlying motivation for the design of the algorithm (for a detailed derivation see Murray, 1969a, b; Wright, 1976). It will be assumed that: the first- and second-order Kuhn-Tucker conditions hold at $\overset{*}{x}$; $\hat{A}(\overset{*}{x})$ has full rank; $\overset{*}{\lambda}_j \neq 0$, $j = 1, 2, \ldots, t$; $\|G\|$ and $\|\hat{G}_j\|$, $j = 1, 2, \ldots, t$, are bounded at $\overset{*}{x}$; and $\lim_{\rho \to \infty} \overset{*}{x}(\rho) = \overset{*}{x}$, where $\overset{*}{x}(\rho)$ is taken to mean $\overset{*}{x}_p(\rho)$.

In general, there exists a value $\hat{\rho}$ such that for $\rho > \hat{\rho}$, the set of constraints violated at $\overset{*}{x}(\rho)$ is identical to the set of constraints active at $\overset{*}{x}$. As before, $\hat{c}$ will denote the vector of violated constraints, and $\hat{A}$ will denote the matrix whose columns are the gradients of these constraints. At $\overset{*}{x}(\rho)$, the following condition holds by definition:

$$\nabla P(\overset{*}{x}(\rho), \rho) = g + \rho \hat{A}\hat{c} = 0 ,$$

where g, $\hat{A}$ and $\hat{c}$ are evaluated at $\overset{*}{x}(\rho)$.

Consider the step Δx from $\overset{*}{x}(\rho)$ to $\overset{*}{x}(\bar{\rho})$, where $\bar{\rho} > \rho$, and $(1/\rho - 1/\bar{\rho})$ is small. By definition of $\overset{*}{x}(\bar{\rho})$:

$$g(\overset{*}{x}(\bar{\rho})) = -\bar{\rho}\, \hat{A}(\overset{*}{x}(\bar{\rho}))\hat{c}(\overset{*}{x}(\bar{\rho}))$$

$$g(\overset{*}{x}(\rho) + \Delta x) = -\bar{\rho}\, \hat{A}(\overset{*}{x}(\rho) + \Delta x)\hat{c}(\overset{*}{x}(\rho) + \Delta x) .$$

Expanding g, $\hat{A}$ and $\hat{c}$ in Taylor series about $\overset{*}{x}(\rho)$ yields:

$$g + G\Delta x = -\bar{\rho}\hat{A}\hat{c} - \bar{\rho}\hat{A}\hat{A}^T\Delta x - \sum_{j=1}^{t} \bar{\rho}\hat{c}_j(\overset{*}{x}(\bar{\rho}))\hat{G}_j\Delta x + \mathcal{O}(\|\Delta x\|^2) ,$$

where g, G, $\hat{c}$, $\hat{A}$ and $\hat{G}_j$ are evaluated at $\overset{*}{x}(\rho)$.

We now estimate the size of each term. Because of the properties of the trajectory of minima of $P(x, \rho)$, $\|\Delta x\| = \mathcal{O}(1/\rho - 1/\bar{\rho})$. If $\|G\|$ and $\|\hat{G}_j\|$ are bounded at $\overset{*}{x}$, then by continuity so are $\|G(\overset{*}{x}(\rho))\|$ and $\|\hat{G}_j(\overset{*}{x}(\rho))\|$ for ρ sufficiently large; however, this condition may also hold for modest values of ρ. By assumption, $\{|\overset{*}{\lambda}_j|\}$, $j = 1, 2, \ldots, t$, are bounded; since $\lim_{\rho\to\infty} \rho\hat{c}_j(\overset{*}{x}(\rho)) = -\overset{*}{\lambda}_j$, it follows that $|\hat{c}_j(\overset{*}{x}(\bar{\rho}))|$ is of order $1/\bar{\rho}$ for all j. Hence, $\|G\Delta x\|$ and $\|\sum_{j=1}^{t} \bar{\rho}\hat{c}_j(\overset{*}{x}(\bar{\rho}))\hat{G}_j\Delta x\|$ are of the order of $\|\Delta x\|$. Combining the terms thus far shown to be of order $\|\Delta x\|$ or higher, we have after re-arranging:

$$\bar{\rho}\hat{A}\hat{A}^T\Delta x = -g - \bar{\rho}\hat{A}\hat{c} + \mathcal{O}(\|\Delta x\|) ;$$

substituting the expression $(-\rho\hat{A}\hat{c})$ for g gives:

$$\bar{\rho}\hat{A}\hat{A}^T\Delta x = (\rho - \bar{\rho})\hat{A}\hat{c} + \mathcal{O}(\|\Delta x\|) .$$

Under the assumptions stated at the beginning of Section 2, it can be shown that the approach of the penalty trajectory to $\overset{*}{x}$ does not lie in a tangent place for any constraint with a non-zero Lagrange multiplier. Let

$$y = \lim_{r \to 0} \frac{d\overset{*}{x}_p(r)}{dr} ;$$

then:

$$\hat{A}(\overset{*}{x})^T y = -\overset{*}{\lambda} ,$$

(see Murray, 1969a, b, for further details).

Because of this non-tangential approach of $\overset{*}{x}(\rho)$ to $\overset{*}{x}$, the term $\bar{\rho}\hat{A}\hat{A}^T\Delta x$ will be of order $\bar{\rho}\|\Delta x\|$; the term $(\rho - \bar{\rho})\hat{A}\hat{c}$ is also of this order. Thus, these terms dominate the order $\|\Delta x\|$ portion of the relationship.

Since $\hat{A}(\overset{*}{x})$ has full rank, $\hat{A}(\overset{*}{x}(\rho))$ will be of full rank for ρ large enough, where again "large enough" need not imply a very large value. If the full-rank matrix $\hat{A}$ is cancelled, the resulting condition on Δx is:

$$\hat{A}^T\Delta x = -(1 - \frac{\rho}{\bar{\rho}})\hat{c} + \mathcal{O}(\frac{1}{\bar{\rho}}(\frac{1}{\rho} - \frac{1}{\bar{\rho}})) . \qquad (3)$$

Exactly the same result can be derived from an alternative viewpoint. Since

$$\lim_{\rho \to \infty} \rho\hat{c}(\overset{*}{x}(\rho)) = -\overset{*}{\lambda} ,$$

and $\overset{*}{\lambda}_j \neq 0$ for any j, the estimate of the Lagrange multipliers at $\overset{*}{x}(\rho)$, given by:

$$\lambda(\overset{*}{x}(\rho)) = -\rho\hat{c}(\overset{*}{x}(\rho)) ,$$

is bounded away from zero for ρ large enough, and is in error by at most order $(1/\rho)$ because of the properties of the penalty trajectory. If the requirement is imposed that these Lagrange multiplier estimates at $\overset{*}{x}(\rho)$ and $\overset{*}{x}(\bar{\rho})$, $\bar{\rho} > \rho$, agree to order $\|\Delta x\|$, the result is:

$$\rho \hat{c}(\overset{*}{x}(\rho)) = \bar{\rho} \hat{c}(\overset{*}{x}(\rho) + \Delta x) + \mathcal{O}\left(\frac{1}{\rho} - \frac{1}{\bar{\rho}}\right) .$$

Using the Taylor expansion of $\hat{c}$ about $\overset{*}{x}(\rho)$, we obtain:

$$\rho \hat{c} = \bar{\rho} \hat{c} + \bar{\rho} \hat{A}^T \Delta x + \mathcal{O}\left(\frac{1}{\rho} - \frac{1}{\bar{\rho}}\right) ,$$

which upon re-arrangement is identical to the previous result (3).

The characterization (3) of the portion of the move Δx along the trajectory in the range of $\hat{A}(\overset{*}{x}(\rho))$ exists because a change in the penalty parameter induces a specified first-order variation in the constraint values along the penalty trajectory.

Throughout this derivation, it has been assumed that ρ is "sufficiently large" for the various assumptions to hold. However, it should be emphasized that these restrictions do not imply that $\overset{*}{x}(\rho)$ is in a close neighborhood of $\overset{*}{x}$. The result (3) may hold even for modest values of ρ, provided that $\bar{\rho}$ is near enough to ρ.

This derivation suggests an algorithm for solving P1, since an approximation, ρ, to the step Δx from $\overset{*}{x}(\rho)$ to $\overset{*}{x}(\bar{\rho})$, $\bar{\rho} > \rho$, will, under certain conditions, satisfy the linear constraints of (3) with high accuracy. However, such an algorithm would impose an unnecessarily

restrictive property on p, since any direction that satisfies the constraints of (3) will always be a local direction of descent for $\hat{c}_j^2$ for every active constraint. Although this does not imply that $\hat{c}_j^2(x + p)$, or even $\|\hat{c}(x + p)\|$, is monotonically decreasing, nonetheless it may be desirable at times to move locally to attempt to increase the values of some, or even all, of the active constraints. A set of linear equality constraints that characterize a step toward the penalty trajectory in an alternative way can be derived if the current point, x, is "close" to the trajectory, and an estimate of the Lagrange multiplier vector is available.

In the penalty trajectory algorithm, the desired condition on the search direction, p, is that $x + p$ be a good approximation to $x^*(\bar{\rho})$ for some $\bar{\rho}$. At $x^*(\bar{\rho})$, the vector $-\bar{\rho}c(x^*(\bar{\rho}))$ is an estimate of the optimal multipliers, with accuracy related to $1/\bar{\rho}$. Hence, using the current multiplier approximation, λ, we seek a step p such that:

$$-\bar{\rho}\hat{c}_j(x + p) \approx \lambda_j, \qquad j = 1, 2, \ldots, t ,$$

where the terms of order $(1/\bar{\rho})$ are omitted.

Expanding $\hat{c}_j(x + p)$ in its Taylor series about x and ignoring all but first-order terms gives:

$$-\bar{\rho}\hat{c}_j - \bar{\rho}\hat{a}_j^T p = \lambda_j,$$

so that,

$$\hat{A}^T p = -\hat{c} - \frac{1}{\bar{\rho}}\lambda , \qquad (4)$$

where $\hat{A}$ and $\hat{c}$ are evaluated at the current point. This specification of the portion of p in the range of the current $\hat{A}$ is a first-order prediction that each constraint value at the updated point will satisfy the appropriate relationship with the corresponding multiplier estimate and the penalty parameter. If the current point is $\overset{*}{x}(\rho)$, the relationships (3) and (4) are identical, since the first-order multiplier estimate at $\overset{*}{x}(\rho)$ is $-\rho\hat{c}(\overset{*}{x}(\rho))$.

2.2. Properties of the Search Direction

The linear constraints (4) to be satisfied by the search direction specify the portion of the search direction in the range of the matrix of active constraint gradients only; we wish to choose the remainder of the search direction to minimize a quadratic approximation to the Lagrangian function. These two properties imply that at each iteration the search direction, p, should be constructed as the solution of the following quadratic program:

QP1:

$$\text{minimize} \quad \frac{1}{2} p^T S p + p^T g$$

$$\text{subject to} \quad \hat{A}^T p = -\hat{c} - \frac{1}{\bar{\rho}}\lambda \ ,$$

where $\hat{c}$ denotes the vector of constraints currently considered "active"; $\hat{A}$ is a matrix whose columns are the gradients of the active constraints; λ is an estimate of the Lagrange multipliers; $\bar{\rho}$ is the current value of the penalty parameter; g is the gradient of F; and S is a matrix that approximates the Hessian of the Lagrangian function.

Let Y be a matrix whose columns form an orthogonal basis for the range of the columns of $\hat{A}$, and let Z be a matrix whose columns form an orthogonal basis for the corresponding null space, so that

$$A^T Z = 0, \qquad Z^T Z = I \ .$$

If $\hat{A}$ has full column rank, and if the matrix $Z^T S Z$ is positive definite, then the solution of QP1, $\overset{*}{p}$, can be uniquely expressed as the sum of two orthogonal components: $\overset{*}{p} = Y p_Y + Z p_Z$. Premultiplying the expression for $\overset{*}{p}$ by $\hat{A}^T$ gives:

$$\hat{A}^T \overset{*}{p} = \hat{A}^T Y p_Y = -\hat{c} - \frac{1}{\bar{\rho}}\lambda \ ;$$

hence, p_y is entirely determined by the linear constraints of QP1. The vector p_Z is given by the solution of:

$$Z^T S Z p_Z = -Z^T (g + S Y p_Y) \ . \tag{5}$$

For sufficiently large $\bar{\rho}$, the search direction so constructed will always be a descent direction for the quadratic penalty function; the step to be taken along the search direction is then chosen to achieve an acceptable decrease in the penalty function, which serves as a measure of progress toward $\overset{*}{x}$.

2.3. Description of Algorithm

At the beginning of the k-th iteration, the following vectors and matrices are assumed to be available:

--$x^{(k)}$, an approximation to $\overset{*}{x}$;

--$c^{(k)}$, the vector of values of $\{c_j(x)\}$ evaluated at $x^{(k)}$;

--$g^{(k)}$, the gradient vector of $F(x)$ evaluated at $x^{(k)}$;

--$A^{(k)}$, the matrix whose columns are the gradients of $\{c_j(x)\}$ evaluated at $x^{(k)}$;

--$S^{(k)}$, an approximation to the Hessian matrix of the Lagrangian function.

The procedures followed to compute the next iterate $x^{(k+1)}$ are:

Step 1. An "active set" of constraints is determined. Currently a constraint is included in the active set if its value is less than a small positive number related to the machine wordlength. If the active set contains more than n elements, a special procedure, given in Section 5.1, is carried out to complete the iteration. It will therefore be assumed for the remaining steps that the number of elements in the active set does not exceed n.

The vector of active constraints will be denoted by $\hat{c}$, and the matrix whose columns are the gradients of those constraints will be denoted by $\hat{A}$.

Step 2. The matrix $\hat{A}$ is reduced to upper triangular form by application of a sequence of orthogonal transformations on the left; column interchanges are carried out, to take care of any possible rank deficiency. The result is

$$Q\hat{A}P = \left[\frac{R}{0}\right] , \tag{6}$$

where Q is orthogonal, P is a permutation matrix, and R is upper triangular (see Lawson and Hanson, 1974, for details).

Define the matrices Y and Z by partitioning Q as

$$Q = \left[\frac{Y^T}{Z^T}\right] .$$

Step 3. Determine λ, an estimate of the Lagrange multiplier vector, which is a solution of the linear least-squares problem

$$\min \|\hat{A}\lambda - g^{(k)}\|_2^2 .$$

If $\hat{A}$ has full column rank, λ is unique, and is given by the solution of the triangular system

$$R\lambda = Y^T g^{(k)} .$$

If $\hat{A}$ is rank-deficient, so that R is singular, λ is taken as the minimum-length least-squares solution, which is computed by extending the factorization (6) to the complete orthogonal factorization of $\hat{A}$, namely

$$Q\hat{A}V = \left[\begin{array}{c:c} \bar{R} & 0 \\ \hdashline 0 & \end{array}\right], \tag{7}$$

where $\bar{R}$ is a non-singular, upper triangular matrix, and V is orthogonal. Further details of this procedure are given in Lawson and Hanson (1974).

Step 4. Determine an appropriate value of the penalty parameter $\bar{\rho}$ (see Section 5.3).

Step 5. Compute the vector p_Y, as follows. If $\hat{A}$ has full rank, p_Y is obtained by solving the linear system implied by the linear constraints of QP1, i.e.,

$$\hat{A}^T p = \hat{A}^T Y p_Y = R^T p_Y = -\hat{c} - \frac{1}{\bar{\rho}}\lambda \ . \tag{8}$$

If $\hat{A}$ is rank-deficient, p_Y is taken as a least-squares solution of (8), computed using the factorization (7); the linear constraints of QP1 will then not in general be satisfied exactly.

Step 6. Determine the modified Cholesky factorization, LDL^T, of the matrix $Z^T S^{(k)} Z$, where L is unit lower-triangular, and D is a diagonal matrix with strictly positive elements. In this procedure, the matrix is augmented (if necessary) as the factorization is formed by a positive diagonal matrix E, chosen to make the matrix $(Z^T S^{(k)} Z + E)$ numerically positive definite; E is identically zero if the original matrix is sufficiently positive definite (Gill and Murray, 1974a). A detailed discussion of the various approaches to computing $S^{(k)}$ is given in Section 4.

Step 7. Compute the vector $\bar{p}_Z$ by solving:

$$LDL^T \bar{p}_Z = -Z^T g^{(k)} .$$

Step 8. Let $\gamma_1 = \|\hat{c}\| \|\bar{p}_z\|$, and $\gamma_2 = \|p_y\| \|Z^T g\|$. Test whether:

$$\gamma_1 \leq M \gamma_2 \qquad \text{and} \qquad \gamma_2 \leq M \gamma_1 \tag{9}$$

for M a reasonably large positive number (say, 10^3).

(a) If the test (9) is satisfied (as it almost always is in practice), compute p_Z by solving:

$$LDL^T p_Z = Z^T (g^{(k)} + S^{(k)} Y p_Y) ;$$

then form the search direction as

$$p = Yp_Y + Zp_Z \ .$$

(b) If the test (9) is not satisfied, there is a danger that the two portions of the search direction are not well-scaled, and the following re-scaling procedure is used to correct for a possible imbalance.

If $\|\gamma_1\| > M \|\gamma_2\|$, define the scaling factor $\beta_1 = M \gamma_2/\gamma_1$, and let $p = Yp_Y + \beta_1 Z\bar{p}_Z$; otherwise, define the factor $\beta_2 = M \gamma_1/\gamma_2$, and let $p = \beta_2 Yp_Y + Z\bar{p}_Z$.

If p is not a descent direction for $P(x,\bar{\rho})$, then the penalty parameter $\bar{\rho}$ should be increased (by a factor γ, say, currently set at 10), and the search direction is re-computed, starting with Step 5. This procedure involves solving the relevant linear systems with altered right-hand sides, but does not require any further matrix factorizations.

It can be shown that for sufficiently large $\bar{\rho}$, the computed search direction is guaranteed to be a descent direction for $P(x,\bar{\rho})$ (see Wright, 1976).

Step 9. Determine a positive step, α, that generates an acceptable reduction in the penalty function $P(x,\bar{\rho})$, using a safeguarded cubic or parabolic step length algorithm (e.g., the procedure described in Gill and Murray, 1974b). Special care must be exercised in the step length algorithm to avoid difficulties if $P(x,\bar{\rho})$ is unbounded below along the given search direction (see Section 5).

Step 10. Set $x^{(k+1)}$ to $x^{(k)} + \alpha p$, and return to Step 1.

3. The Barrier Trajectory Algorithm

3.1. Derivation

The barrier trajectory algorithm is based on the logarithmic barrier function, and hence is a feasible-point method -- i.e., the starting point must be strictly feasible, and the singularity at the boundary of the feasible region prevents subsequent iterates from becoming infeasible.

Feasible-point methods are often useful in solving practical optimization problems, for two reasons:

1. Some of the problem functions (objective and/or constraints) may be undefined or ill-defined outside the feasible region. The former situation is common in physical applications -- for example, certain phenomena do not occur outside a particular layer of the ionosphere. In such instances, attempts to extend the definitions artificially run a significant risk of numerical difficulties or physically impossible results. The problem functions can be "ill-defined" in circumstances typified by the case when an optimization problem arises from data fitting, and the objective function is, say, a generalized polynomial. Although this function is theoretically defined everywhere, in reality it is well-behaved and meaningful only in the region of the known data points.
2. In some applications, only a rough approximation to the solution of an optimization problem is required. For example, a typical procedure in model-building is to formulate an initial model of

the desired process, to estimate (by optimization) rough values for key parameters, and then to re-formulate the model. Thus, although only a low level of accuracy is needed in the solution of the optimization problem, it is essential that the approximate solution be feasible. Since non-feasible algorithms nearly always generate non-feasible iterates, and feasibility is achieved only in the limit, it is usually not possible to terminate such a method prematurely at a feasible point.

A detailed derivation of the barrier trajectory algorithm has been given in Wright (1976), and only an abbreviated description will be presented here.

It will be assumed that: the first-and second-order Kuhn-Tucker conditions hold at $\overset{*}{x}$; $\hat{A}(\overset{*}{x})$ has full rank; $\overset{*}{\lambda}_j \neq 0$, $j = 1, 2, \ldots, t$; $\|G\|$ and $\|G_j\|$ are bounded at $\overset{*}{x}$; and $\lim_{r\to 0} \overset{*}{x}(r) = \overset{*}{x}$, where $\overset{*}{x}(r)$ will be taken in this section to mean $\overset{*}{x}_B(r)$.

At $\overset{*}{x}(r)$, by definition,

$$\nabla B(\overset{*}{x}(r),\ r) = g - rA \begin{bmatrix} \frac{1}{c_1} \\ \vdots \\ \frac{1}{c_m} \end{bmatrix} = g - rAd = 0 \ ,$$

where g, A, and c are evaluated at $\overset{*}{x}(r)$, and the function $d(x)$ is defined as the vector $(1/c_1(x) \cdots\cdots 1/c_m(x))^T$. The notation $\hat{d}(x)$ will denote the vector $(1/\hat{c}_1(x) \cdots\cdots 1/\hat{c}_t(x))^T$, which includes only the active constraints.

Consider the step Δx from $\overset{*}{x}(r)$ to $\overset{*}{x}(\bar{r})$, where $\bar{r} < r$, and $(r - \bar{r})$ is small relative to r and $\bar{r}$. By definition of $\overset{*}{x}(\bar{r})$:

$$g(\overset{*}{x}(\bar{r})) = \bar{r}A(\overset{*}{x}(\bar{r}))d(\overset{*}{x}(\bar{r})) \ .$$

Expanding g and A in their Taylor series about $\overset{*}{x}(r)$ yields:

$$g + G\Delta x = \bar{r}Ad(\overset{*}{x}(\bar{r})) + \sum_{j=1}^{m} \bar{r}d_j(\overset{*}{x}(\bar{r}))G_j\Delta x + \mathcal{O}(\|\Delta x\|^2) ,$$

where g, G, A and G_j are evaluated at $\overset{*}{x}(r)$.

We now estimate the size of each term. Because of the properties of the trajectory of the logarithmic barrier function, $\|\Delta x\| = \mathcal{O}(r - \bar{r})$. The quantities $\|G\|$ and $\|G_j\|$ are bounded at $\overset{*}{x}$, so that they are guaranteed by continuity to be bounded for r sufficiently small; however, these quantities may be bounded for any value of r, so that "sufficiently small" need not imply that r is close to zero. For $\bar{r}$ small enough, the components of d corresponding to inactive constraints are strictly bounded, and thus the term $\bar{r}d_j(\overset{*}{x}(\bar{r}))$ is of order $\bar{r}$ if c_j is inactive at $\overset{*}{x}$. Because $\|\overset{*}{\lambda}\|$ is bounded, and $\lim_{r\to 0} \hat{rd}_j(\overset{*}{x}(r)) = \overset{*}{\lambda}_j$, it follows that for $\bar{r}$ small enough, a component $\bar{r}d_j(\overset{*}{x}(\bar{r}))$ corresponding to an active constraint is also bounded. Therefore, all elements of the sum $\sum_{j=1}^{m} \bar{r}d_j(\overset{*}{x}(\bar{r}))G_j\Delta x$ are of order at most Δx, i.e., $\mathcal{O}(r - \bar{r})$. Grouping together terms of order $(r - \bar{r})$ or higher, the result is:

$$g = \bar{r}Ad(\overset{*}{x}(\bar{r})) + \mathcal{O}(r - \bar{r}) .$$

Substituting for g the value $rAd(\overset{*}{x}(r))$, we obtain:

$$rAd(\overset{*}{x}(r)) = \bar{r}Ad(\overset{*}{x}(r)) + \mathcal{O}(r - \bar{r}),$$

or

$$rA\begin{bmatrix} \frac{1}{c_1(\overset{*}{x}(r))} \\ \vdots \\ \frac{1}{c_m(\overset{*}{x}(r))} \end{bmatrix} = \bar{r}A\begin{bmatrix} \frac{1}{c_1(\overset{*}{x}(\bar{r}))} \\ \vdots \\ \frac{1}{c_m(\overset{*}{x}(\bar{r}))} \end{bmatrix} + \mathcal{O}(r - \bar{r}) . \qquad (10)$$

The relationship (10) may hold anywhere along the trajectory if $(r - \bar{r})$ is sufficiently small, and the assumptions inherent in the bounding process are satisfied. However, because $A(\overset{*}{x}(r))$ is not necessarily of full rank, it is not possible to draw any meaningful conclusion about the variation in the constraint values. To refine (10) to correspond to the result (3) in the penalty case, it is necessary to assume that the value of r is sufficiently small so that an $\mathcal{O}(r)$ term is negligible with respect to unity; then the active and inactive constraints may be considered separately. Since the inactive constraints are bounded away from zero in a neighborhood of $\overset{*}{x}$, the components $\{r/c_j(\overset{*}{x}(r))\}$, $\{\bar{r}/c_j(\overset{*}{x}(\bar{r}))\}$ corresponding to inactive constraints may be included in an order r term, leaving a relationship that holds for the active constraints:

$$r\hat{A}\begin{bmatrix} \frac{1}{\hat{c}_1(\overset{*}{x}(r))} \\ \vdots \\ \frac{1}{\hat{c}_t(\overset{*}{x}(r))} \end{bmatrix} = \bar{r}\hat{A}\begin{bmatrix} \frac{1}{c_1(\overset{*}{x}(\bar{r}))} \\ \vdots \\ \frac{1}{\hat{c}_t(\overset{*}{x}(\bar{r}))} \end{bmatrix} + \mathcal{O}(r) .$$

Since $\hat{A}(\overset{*}{x})$ has full rank, $\hat{A}(\overset{*}{x}(r))$ is guaranteed by continuity to be of full rank for sufficiently small r; it may accordingly be cancelled, yielding:

$$r\begin{bmatrix} \frac{1}{\hat{c}_1(\overset{*}{x}(r))} \\ \vdots \\ \frac{1}{\hat{c}_t(\overset{*}{x}(r))} \end{bmatrix} = \bar{r}\begin{bmatrix} \frac{1}{\hat{c}_1(\overset{*}{x}(\bar{r}))} \\ \vdots \\ \frac{1}{\hat{c}_t(\overset{*}{x}(\bar{r}))} \end{bmatrix} + \mathcal{O}(r) . \qquad (11)$$

All elements in the denominators are bounded away from zero for nonzero r, $\bar{r}$, and hence the j-th row of each vector can be multiplied by the factor $\hat{c}_j(\overset{*}{x}(r))\hat{c}_j(\overset{*}{x}(\bar{r}))/r$, which is of order $\bar{r}$, giving:

$$\hat{c}(\overset{*}{x}(\bar{r})) = \frac{\bar{r}}{r}\hat{c}(\overset{*}{x}(r)) + \mathcal{O}(r \cdot \bar{r}) .$$

Expanding $\hat{c}$ in a Taylor series about $\overset{*}{x}(r)$, we obtain:

$$\hat{c} + \hat{A}^T\Delta x = \frac{\bar{r}}{r}\hat{c} + \mathcal{O}(r \cdot \bar{r}) .$$

It can be shown that the approach of the barrier trajectory to $\overset{*}{x}$ does not lie in a tangent plane for any constraint with a finite Lagrange multiplier. Let $y = \lim_{r \to 0} \frac{d\overset{*}{x}(r)}{dr}$; then

$$\hat{A}(\overset{*}{x})^T y = \begin{bmatrix} \frac{1}{\overset{*}{\lambda}_1} \\ \vdots \\ \frac{1}{\overset{*}{\lambda}_t} \end{bmatrix}$$

(see Wright, 1976, for additional details). Because of this non-tangential approach, the term $\hat{A}^T \Delta x$ is known to be of order Δx, i.e., order $(r - \bar{r})$. After re-arrangement, we obtain the desired characterization of Δx:

$$\hat{A}^T \Delta x = -(1 - \frac{\bar{r}}{r})\, \hat{c} + \mathcal{O}(r \cdot \bar{r}) \,, \tag{12}$$

which is similar to that given for the penalty function trajectory in Section 2.1.

An alternative derivation can be given by requiring the Lagrange multiplier estimates for active constraints at $\overset{*}{x}(r)$ and $\overset{*}{x}(\bar{r})$ to agree within order r. At $\overset{*}{x}(r)$, the j-th multiplier estimate is $r/\hat{c}_j(\overset{*}{x}(r))$, so that the requirement is

$$\frac{r}{\hat{c}_j(\overset{*}{x}(r))} = \frac{\bar{r}}{\hat{c}_j(\overset{*}{x}(\bar{r}))} + \mathcal{O}(r) \,, \qquad j = 1, 2, \ldots, t \,.$$

This relationship is the same as (11), so that (12) is again the result. As with the penalty function, it is possible to characterize a step along the trajectory of approach because the first-order variation of the constraints is controlled by the barrier parameter.

As in the penalty case, an algorithm could be based on requiring the direction of search to satisfy the linear constraints (12). However, any direction that satisfies these constraints is a local direction of descent with respect to $\hat{c}_j^2$ for every active constraint. In a general feasible algorithm, it may be desirable at times to take a step that will increase the values of some of the active constraints; furthermore, the current prediction of the active set may be incorrect. A set of linear constraints that characterize a step toward the barrier trajectory in an alternative way can be derived if the current point, x, is "close" to the trajectory, and an estimate of the Lagrange multiplier vector is available. The condition desired for the search direction, p, is that $x + p$ be a good approximation to $\overset{*}{x}(\bar{r})$. At $\overset{*}{x}(\bar{r})$, the vector $(\bar{r}/\hat{c}_1, \ldots, \bar{r}/\hat{c}_t)^T$ is an estimate of the optimal multipliers, with accuracy related to $\bar{r}$. Thus, for each active constraint, it is required that

$$\frac{\bar{r}}{\hat{c}_j(x + p)} \approx \lambda_j ,$$

or

$$\bar{r} \approx \lambda_j \hat{c}_j(x + p) , \qquad j = 1, 2, \ldots, t .$$

Ignoring all but first-order terms, the resulting condition is:

$$\bar{r} = \lambda_j \hat{c}_j + \lambda_j \hat{a}_j^T p \ ,$$

so that

$$\hat{a}_j^T p = -\hat{c}_j + \frac{\bar{r}}{\lambda_j} \ ,$$

or

$$\hat{A}^T p = -\hat{c} + \bar{r} \begin{bmatrix} \frac{1}{\lambda_1} \\ \vdots \\ \frac{1}{\lambda_t} \end{bmatrix} \ , \qquad (13)$$

where $\hat{c}$ and $\hat{A}$ are evaluated at the current point. The linear equality constraints (13) are thus based on the relationship that should hold along the barrier trajectory among the barrier parameter, the active constraint values, and the multiplier estimates.

3.2. Properties of the Search Direction

The relationship (13) to be satisfied by the search direction determines the portion of the search direction in the range of the gradients of the active constraints; the aim is to choose the remainder to minimize a quadratic approximation to the Lagrangian function. These two properties imply that the search direction of the barrier trajectory algorithm should be constructed at each iteration as the solution of the following quadratic program:

$$\text{minimize} \quad \tfrac{1}{2}p^T S p + p^T g$$

QP2:

$$\text{subject to} \quad \hat{A}^T p = d,$$

where $d_j = -\hat{c}_j + \bar{r}/\lambda_j$. The variables are defined as for QP1 in Section 2.2, except that $\bar{r}$ is the current value of the barrier parameter. Exactly as in the penalty case, the solution of QP2 can be written in terms of two orthogonal portions, Yp_Y and Zp_Z, which lie respectively in the range and null space of the columns of $\hat{A}$; the vector p_Y is determined by the linear constraints, and p_Z is given by the solution of

$$Z^T S Z p_Z = -Z^T(g + SYp_Y) \; . \tag{14}$$

At each iteration, it will be required to achieve an acceptable reduction in the logarithmic barrier function, which serves as a convenient merit function to measure progress toward $\overset{*}{x}$. However, the search direction given by the solution of QP2 may not be a descent direction for $B(x,r)$ for any value of r, in contrast to the guaranteed descent properties of the search direction in the penalty trajectory algorithm. This situation exists because of the reversed roles of the objective function and the constraints in the penalty and barrier functions as the relevant parameters approach the limit. As the penalty parameter increases, the squared penalty term (representing the constraint violations)

dominates the penalty function; thus, the search direction of the penalty trajectory algorithm must be a descent direction for the penalty function if ρ is sufficiently large, because the linear constraints of QP1 assure that the search direction is a descent direction for the squared penalty term. On the other hand, the approach to the limit of the barrier parameter causes the objective function to dominate the barrier function locally, since the effect of the singularities induced by active constraints is reduced; the linear constraints of QP2 do not assure a descent direction for the objective function, since they pertain only to the constraints.

In order to assure a decrease in a barrier function at every iteration, an alternative procedure for obtaining the search direction in the barrier trajectory algorithm is based on computing p_Z to minimize a quadratic approximation to the Lagrangian function, independent of the vector p_Y. With this definition, p_Z is given by the solution of the linear system:

$$Z^T S Z p_Z = -Z^T g \ , \tag{15}$$

and the desired descent property will hold if $\bar{r}$ is sufficiently small (full details are given in Wright, 1976).

3.3. Description of Algorithm

The iterates generated by the barrier trajectory algorithm necessarily lie within the strict interior of the feasible region. This algorithm is intended for use on problems where some or all of the problem

functions may be ill-defined or undefined outside the feasible region, and it requires a strictly feasible starting point.

At the beginning of the k-th iteration of the barrier trajectory algorithm, the same vectors and matrices are available as for the penalty trajectory algorithm. The computational procedures followed during the k-th iteration are:

Step 1. Determine the set of "active" constraints (see Section 5.2); the vector $\hat{c}$ will denote the vector of these values. Form the matrix $\hat{A}$, whose columns are the columns of $A^{(k)}$ corresponding to the active set. By construction, $\hat{A}$ has no more than n columns.

Step 2. Factorize $\hat{A}$ into upper triangular form, using orthogonal transformations and column interchanges, so that

$$Q\hat{A}P = \begin{bmatrix} R \\ \hline 0 \end{bmatrix},$$

as before.

Step 3. Determine the Lagrange multiplier estimate λ, exactly as in the penalty trajectory algorithm. If one or more components of λ are negative, the constraint corresponding to the most negative multiplier is deleted from the active set; the modified $\hat{A}$ is then factorized, and the new λ is calculated for the re-defined active set. Since the

modified $\hat{A}$ is simply the previous $\hat{A}$ with one column deleted, the new factorization can be obtained by a simple updating scheme (Gill, Golub, Murray and Saunders, 1974).

Step 4. Determine the barrier parameter, $\bar{r}$ (see Section 5.3).

Step 5. Construct the vector d according to the following rule:

for $i = 1, 2, \ldots, m$:

(a) let $\beta_i = \max\ (10, [\|\hat{c}\|/\sqrt{m}|\hat{c}_i|])$ (β_i is an integer between 0 and 10 that reflects the "smallness" of $|\hat{c}_i|$);

(b) let $\gamma = \max\ (\lambda_i, \bar{r}/((\beta_i + 1)\hat{c}_i))$;

(c) $d_i = -\hat{c}_i + \frac{\bar{r}}{\gamma}$.

With this definition, d_i can not exceed $\beta_i \hat{c}_i$.

Step 6. Compute the vector p_Y, which is the solution of the linear system:

$$\hat{A}^T p = \hat{A}^T Y p_Y = R^T p_Y = d\ .$$

In this way, the search direction satisfies the desired linear equality constraints of QP2 for those active constraints corresponding to sufficiently positive multiplier estimates; an alternative relationship is satisfied for any other active constraints.

The computation of p_Y in the rank-deficient case is carried out as for the penalty trajectory algorithm.

Step 7. Compute the modified Cholesky factorization of $Z^T S^{(k)} Z$, which will be denoted by LDL^T.

Step 8. Determine $\bar{p}_Z$ by solving:

$$LDL^T \bar{p}_Z = -Z^T g^{(k)} .$$

Let $\gamma_1 = \|\hat{c}\| \|\bar{p}_Z\|$, and $\gamma_2 = \|p_Y\| \|Z^T g\|$; test whether

$$\gamma_1 \leq M \gamma_2 \qquad \text{and} \qquad \gamma_2 \leq M \gamma_1 , \tag{16}$$

for M a reasonably large positive number (currently, $M = 10^3$).

(a) If the test (16) is satisfied, obtain p_Z by solving

$$LDL^T p_Z = -Z^T (g^{(k)} + S^{(k)} Y p_Y) ,$$

and define the trial search direction as:

$$p = Y p_Y + Z p_Z .$$

If p is not a descent direction for $B(x,\bar{r})$, re-define p as:

$$p = Yp_Y + Zp_Z \ .$$

(b) If the test (16) is not satisfied, then adjust the scaling, as in the penalty trajectory algorithm.

If p is not a descent direction for $B(x,\bar{r})$, then the barrier parameter $\bar{r}$ should be decreased, and the search direction is re-computed, starting with Step 5.

It can be shown that for sufficiently small $\bar{r}$, the search direction so constructed is guaranteed to be a descent direction for $B(x,\bar{r})$.

Step 9. Determine a step length, α, that accomplishes a suitable reduction in $B(x,\bar{r})$ using special procedures designed for one-dimensional minimization with respect to the logarithmic barrier function (see Murray and Wright, 1976). During the search procedure, record whether the step length is restricted because a larger step would violate a constraint currently considered "inactive"; if so, the constraint corresponding to the smallest step that caused a violation is added to the active set at the next iteration.

Step 10. Set $x^{(k+1)}$ to $x^{(k)} + \alpha p$, and return to Step 1.

4. Approximation of the Hessian of the Lagrangian Function

At each iteration of the trajectory algorithms, the matrix $S^{(k)}$ is intended to serve as an approximation to the Hessian matrix of the Lagrangian function,

$$W(x, \lambda) = G(x) - \sum_{i \in I} \lambda_i G_i(x) ,$$

evaluated at $(\overset{*}{x}, \overset{*}{\lambda})$. In this section we consider some possible approaches to computing $S^{(k)}$, depending on the available information; it is important to note that the full matrix is not required, but only certain projections of it.

4.1. Exact second derivatives

When the Hessian matrices of F and $\{c_i\}$ can be evaluated at each point, the full matrix $S^{(k)}$ is given by $W^{(k)}$, defined as:

$$W^{(k)} = G^{(k)} - \sum_{i \in I} \lambda_i^{(k)} G_i^{(k)} ,$$

where the index set I contains those constraints considered active, and the superfix k denotes the relevant quantities evaluated at $x^{(k)}$. With this choice of $S^{(k)}$, the trajectory algorithms display a local quadratic rate of convergence to $\overset{*}{x}$, even with only first-order multiplier estimates (see Wright, 1976).

4.2. Two Finite-Difference Approaches

When exact second derivatives are not available, but gradients of F and $\{c_i\}$ can be computed, the full matrix $S^{(k)}$ can be given as a standard finite-difference approximation to $W^{(k)}$. Let $\tilde{S}$ be the matrix whose i-th column is given by:

$$\tilde{s}_i = \frac{1}{h}[g(x^{(k)} + he_i) - \hat{A}(x^{(k)} + he_i)\lambda - (g(x^{(k)}) - \hat{A}(x^{(k)})\lambda)] ,$$

where e_i is the i-th column of the identity matrix, and h is a small scalar, which should be chosen to balance truncation and cancellation error (for a well-scaled problem, the optimal h is of the order of the square root of machine precision). The matrix $S^{(k)}$ is then given by:

$$S^{(k)} = \frac{1}{2}(\tilde{S} + \tilde{S}^T) .$$

This scheme requires n evaluations of the relevant gradients at each iteration, and may therefore be inefficient for moderate or large values of n. (If W is known to be sparse, it is possible to exploit the sparsity pattern to effect considerable economies in the number of gradient evaluations required, by careful choice of the finite-difference vectors; see Gill and Murray, 1974c).

An alternative procedure that typically requires substantially fewer gradient evaluations, and that takes explicit advantage of the linear constraints of the sub-problem, can be devised by noting that

the matrix $S^{(k)}$ itself is not required in the computations, but only the matrix $S^{(k)}Z$ (or $Z^TS^{(k)}$). Let V be the n by $(n-t)$ matrix whose i-th column is given by

$$v_i = \frac{1}{h}[g(x^{(k)} + hz_i) - \hat{A}(x^{(k)} + hz_i)\lambda - (g(x^{(k)}) - \hat{A}(x^{(k)})\lambda)] ,$$

where z_i is the i-th column of Z. The vector v_i is a direct finite-difference approximation to $W^{(k)}z_i$, and thus the matrix V is an $\mathcal{O}(h)$ approximation to $W^{(k)}Z$, which may be computed with only $(n-t)$ evaluations of the relevant gradients at each iteration. The matrix $Z^TS^{(k)}Z$ is then given by $\frac{1}{h}(Z^TV + V^TZ)$, and the vector $Z^TS^{(k)}Yp_Y$ is given by V^TYp_Y.

The advantage of these finite-difference techniques is that the algorithm can achieve essentially the same local convergence properties as for the case when second derivatives are available, but using only first derivatives. A further advantage of using either exact or finite-difference approximations to second derivatives is that convergence to a local minimum can be confirmed, except in rare cases.

4.3. Quasi-Newton Approximations

The obvious next step in computing $S^{(k)}$ is to use a quasi-Newton approximation to $W(x, \lambda)$; this idea was presented in Murray's original algorithm (1969a,b), and has recently been considered by other

authors in the context of various projected Lagrangian methods (see, for example, Han, 1976, 1977; Powell, 1977a,b).

There are several complications inherent in the use of a quasi-Newton approximation to the Hessian of the Lagrangian function. First, since the estimate of the Lagrange multipliers is changed at each iteration, the properties derived in the unconstrained case do not carry over in a straightforward way -- for example, even if F and $\{c_i\}$ are quadratic functions, the Lagrangian function is not quadratic, since the Lagrange multiplier estimates are nonlinear functions of x. A second complication is that the good properties of some of the best-known quasi-Newton updates are dependent on positive-definiteness of the underlying Hessian matrix. However, the Hessian of the Lagrangian function need <u>not</u> be positive definite at the solution, so that care must be taken in applying the standard update formulas. Nonetheless, it is evident that quasi-Newton techniques can be used effectively in this context, and that further investigation and extensive numerical experimentation will be worthwhile.

In the remainder of this section we give two alternative quasi-Newton approaches which have been tried with some success, but no claim is made that the given methods will prove to be the best for the case of nonlinearly constrained optimization. A full discussion of quasi-Newton methods in the unconstrained context is given in Dennis and More'(1977).

4.3.1. Approximation of the full matrix

For unconstrained optimization, the BFGS formula (see, for example, Broyden, 1970) is widely considered the most effective update procedure (Dennis and Moré, 1977). Let s denote the change in x during the previous iteration; let y denote the change in gradient of the function whose Hessian is to be approximated; and let B denote the current Hessian approximation. The new approximation $\bar{B}$ is given by:

$$\bar{B}_{BFGS} = B + \frac{yy^T}{y^T s} - \frac{Bss^T B}{s^T Bs} .$$

If B is positive definite, $\bar{B}$ is also positive definite if and only if $y^T s > 0$. This latter condition almost invariably holds in the case of unconstrained optimization, where the step in x is related in a special way to the matrix B and the gradient of the objective function. In the present algorithms, however, when y represents the change in the gradient of the Lagrangian function, the quantity $y^T s$ may be negative (which would mean that $\bar{B}$ could become indefinite), or arbitrarily small (so that the elements of $\bar{B}$ would be unbounded). (A similar situation can theoretically occur even in the unconstrained case with the symmetric rank-one update, but in practice the difficulty has not proved to be serious.) A second update formula, for which the elements of the updated matrix remain bounded, is the PSB update (Powell, 1970):

$$\bar{B}_{PSB} = B + \frac{(y - Bs)s^T + s(y - Bs)^T}{s^T s} - \frac{[(y - Bs)^T s]ss^T}{(s^T s)^2} ;$$

however, hereditary positive definiteness cannot be guaranteed with the PSB update, which consequently has not been as effective as the BFGS formula on unconstrained problems. Powell (1977b) has recently given a quasi-Newton update for the Hessian of the Lagrangian function in which the approximating matrix is always positive definite, and for which local superlinear convergence can be attained.

When $S^{(k)}$ is a quasi-Newton approximation to $W(x, \lambda)$, the required matrix $Z^T S^{(k)} Z$ and vector $Z^T S^{(k)} Y p_Y$ are obtained using the Z and Y matrices corresponding to the current iteration. A modified Cholesky factorization of $Z^T S^{(k)} Z$ is then computed, so that a positive definite matrix is always used when solving the linear system for the null-space portion of the search direction. It should be noted that, even if a positive definite update is used, rounding errors may have introduced indefiniteness in the computed $Z^T S^{(k)} Z$.

4.3.2. Approximation of the projected matrix

An alternative way to use a quasi-Newton update is based on techniques from the linearly constrained case, where it is possible to maintain a quasi-Newton approximation to the <u>projected</u> Hessian matrix, $Z^T G(x) Z$. This approach, suggested by Gill and Murray (1974a), has two closely related advantages: (1) because the optimality conditions require that $Z^T G(x^*) Z$ be positive semi-definite, it is

reasonable to maintain a positive definite approximation (even though the full Hessian at x^* may be indefinite); (2) the dimension of the projected Hessian is $(n-t)$ by $(n-t)$, so that only a matrix of reduced size needs to be stored. An additional complication with such an approach applied to nonlinear constraints arises because the matrix Z changes at every iteration. Thus, it is necessary for the updated matrix to reflect the variation in Z as well as the accumulated information about the curvature of the Lagrangian function. Several techniques for approximating the projected Hessian are currently under investigation.

5. Algorithmic Details

5.1. Selection of the Active Set for the Penalty Trajectory Algorithm

The "active set" of constraints at a given iteration of the penalty trajectory algorithm includes those constraints whose values are less than a specified tolerance (currently defined for the i-th constraint as a scaled multiple of the square root of machine precision). With this definition, the active set is essentially equivalent to the set of violated constraints, and can be determined in a straightforward manner.

Such a strategy is reasonable because the algorithm was originally motivated by properties of the quadratic penalty function, and the violated set at $x_p^*(\rho)$ is equivalent to the active set at x^* for sufficiently large ρ. This convenient property does not hold for augmented Lagrangian methods, since there is no a priori knowledge that an active constraint will be violated as the solution is approached.

It was noted in the definition of the algorithm that a special procedure is used to define the search direction when more than n constraints are violated at the beginning of an iteration. In this case, we seek to reduce $\|\hat{c}(x + p)\|_2^2$ by choosing p as the solution of the linear least-squares problem $\min \|\hat{c} + \hat{A}^T p\|_2^2$. The search direction is calculated by the following procedure, where we assume for simplicity that $\text{rank}(\hat{A}) = n$:

1) Factorize $\hat{A}^T$ in the form

$$Q\hat{A}^T = \begin{bmatrix} R \\ \cdots \\ 0 \end{bmatrix}, \quad Q^T Q = I$$

where R is upper triangular.

2) Solve $Rp = -Y^T \hat{c}$, where the columns of Y are the first n rows of Q.

This procedure is similar to the calculation of Lagrange multiplier estimates in the usual iteration, and can be extended in an obvious way if $\text{rank}(\hat{A}) < n$.

Normally, the condition that more than n constraints are violated occurs because the current point is a poor estimate of the solution, and does not hold at the next iteration, when the estimate improves. However, it is conceivable that this condition could hold even at x^*, so that possibly every iteration might be special. In such a case, the Hessian matrix of the penalty function is not ill-conditioned as ρ approaches its limit. The choice of search direction given above has the same effect as choosing $\rho = \infty$ in the usual definition of the algorithm, and is equivalent to the Gauss-Newton method.

5.2. Selection of the Active Set for the Barrier Trajectory Algorithm

The criteria for selecting the active set in the barrier trajectory algorithm are not so straightforward as in the penalty case. Because all constraints are strictly satisfied at the beginning of every iteration, the constraints to be considered "active" must be determined by analysis of the behavior of the constraints and multiplier estimates as the computation proceeds.

At the starting point, the active set is determined by the following procedure, where I denotes an index set which is initialized to the null set (a first approximation to the active index set):

Step 1. Compute the search direction, p, with I as the set of active constraints.

Step 2. Determine j, the index for which

$$\alpha_j = \min_{i \notin I} \{\frac{-c_i}{a_i^T p} \mid a_i^T p < 0\} .$$

Step 3. If $a_i^T p \geq 0$ for all $i \in I$, or $\alpha_j < 1$, the process terminates with I as the index set. Otherwise, the index j is added to I, and the process is repeated, providing that the number of elements in I is less than n; if I contains n elements, the process terminates.

At the beginning of each subsequent iteration, the active set is modified according to the following rules:

1. If the steplength algorithm during the previous iteration was restricted because a constraint was violated, the constraint for which violation occurred at the smallest step is added to the

active set at the beginning of the next iteration. Such a constraint will not be deleted during the next iteration regardless of its size or the sign of its multiplier estimate. If the number of active constraints would exceed n following such an addition, the active constraint with the largest value is deleted.

2. The constraint corresponding to the most negative Lagrange multiplier estimate (if one exists) is deleted from the active set, and the remaining multipliers are modified accordingly.
3. If the largest constraint exceeds $r/\varepsilon^{1/4}$, it is deleted from the active set. The aim of this test is to remove "active" constraints that appear to be bounded away from zero as the solution is approached.

The above procedure has been satisfactory on the examples tested. The active set tends to be altered only during the early iterations, because of misleading local indications that certain constraints are active. The decisions based on "size" are obviously dependent on scaling; this topic is discussed further in Section 5.5.

5.3. Adjustment of the Penalty and Barrier Parameters

The motivation behind the rules for determining the value of the penalty or barrier parameter is to choose a value closer to the limit than the value corresponding to the nearest point on the trajectory. In general, we can only approximate the parameter value corresponding to the closest point on the trajectory; the parameter is

then altered if it appears that the current iterate is sufficiently near the trajectory for the estimates inherent in the derivation to be valid.

Because a penalty or barrier function is decreased at every iteration, and the penalty or barrier parameter can be adjusted both between and within iterations, a poor choice can be quickly improved. The uncertainties of the choice are inevitable with any attempt to use information at an arbitrary point to reveal properties of the solution.

The initial value of the penalty parameter is selected by the following procedures:

Step 1. Given $x^{(0)}$, let g denote $g(x^{(0)})$, $\hat{c}$ denote $\hat{c}(x^{(0)})$, and so on. If $\|g\| > \varepsilon^{1/2}$, let $\hat{\rho}$ be the least-squares solution of $\min \|g + \rho \hat{A} \hat{c}\|_2^2$; if $x^{(0)}$ were exactly $x^*_p(\rho)$, then $\|g + \rho \hat{A} \hat{c}\| = 0$. Let $b = \hat{A} \hat{c}$; $\hat{\rho}$ is given by the explicit formula:

$$\hat{\rho} = \frac{-b^T g}{b^T b} .$$

Step 2. If $\hat{\rho} > 0$, compute $\beta = \|g + \hat{\rho} \hat{A} \hat{c}\|/\|g\|$. If β is "small" (say, $<.25$), the initial point is "close to" the penalty trajectory. In this case:

a. if $\hat{\rho} > M$ (say, $M = 100$), let $\bar{\rho} = \hat{\rho}^2$;

b. otherwise, $\bar{\rho} = \tau\hat{\rho}$ (currently, $\tau = 10$).

Step 3. If $\hat{\rho} < 0$ or β is not small, let $\mu = \|Z^T g\|/(1 + \|g\|) + \|\hat{c}\|/t$, where t is the number of active constraints (the last term is omitted if $t = 0$).

The initial $\bar{\rho}$ is defined as $\max (1/\mu^2, 1)$.

At each subsequent iteration, the penalty parameter is altered according to the following rules:

1. If $\|Z^T g\|/(1 + \|g\|) + \|\hat{c}\|/t < \varepsilon^{1/4}$, let $\bar{\rho}$ be replaced by $\bar{\rho}^2$.
2. Replace $\bar{\rho}$ by $\tau\bar{\rho}$ (currently, $\tau = 10$) if any of the following is true:

   a. $\sum_i |\lambda_i + \bar{\rho}\, \hat{c}_i|/|\lambda_i|$, or $\sum_i |\lambda_i + \bar{\rho}\, \hat{c}_i|$, is "small" (say $< \varepsilon^{1/4}$);
   b. $\bar{\rho} < \hat{\rho}$, where $\hat{\rho}$ is the least-squares solution of $\min \|g + \rho\, \hat{A}\, \hat{c}\|_2^2$;
   c. $\bar{\rho} < \|\lambda\|/\|\hat{c}\|$.

3. If $\hat{\rho} > 0$ and $\bar{\rho} > 100\hat{\rho}$, $\bar{\rho}$ is replaced by $\bar{\rho}/\tau$.
4. Otherwise, $\bar{\rho}$ is unchanged.

An exactly analogous procedure is used for the barrier parameter, using the relationships that hold along its trajectory.

5.4. Detection and Correction of Unbounded Decrease of Penalty Function

Even when the problem P1 has a well-defined solution, the corresponding penalty function may be unbounded below, even for arbitrarily large values of the penalty parameter, because the effect of a penalty transformation is in general only to create local minima in a neighborhood of the solution (see the discussion in Powell, 1972). (The same danger is even more acute with augmented Lagrangian functions.) Such unboundedness causes extreme difficulties when executing a one-dimensional minimization with respect to a penalty function: either the next iterate is so large as to be meaningless, or an excessive number of function evaluations are required before the procedure terminates. Many steplength algorithms offer no protection from unboundedness, since they are usually designed in the context of unconstrained optimization, where the function to be minimized can reasonably be assumed to be bounded below.

The safeguarded cubic or quadratic steplength algorithms (Gill and Murray, 1974b) used in the penalty trajectory algorithms allow protection against unboundedness by requiring specification of an upper bound on the step to be taken during a given iteration. In the implementation of the penalty trajectory algorithm, the upper bound is set to correspond to a "reasonable" value. In some cases, this restriction may impose an unnecessary limit on the stepsize; nonetheless, there is generally no serious loss of efficiency for the overall computation, and the conservative strategy is considered to be justified by the difficulties that would otherwise occur.

If the step taken at a given iteration is equal to the specified upper bound, it is assumed that the penalty function may be unbounded along the given direction. Almost always, the indicated unboundedness can be eliminated simply by increasing the penalty parameter, so that eventually the penalty function becomes dominated by the squared constraint violations.

5.5. Scaling

No procedure for scaling the variables is included in the current implementation. This omission does not reflect a lack of concern with scaling, which is an extremely important aspect of practical optimization algorithms; rather, in our experience, scaling by the knowledgable user seems to be superior to any automatic procedure available today.

The only form of scaling in the existing trajectory algorithms is that each constraint function is multiplied by a scalar weighting factor at every iteration; the decisions as to whether certain quantities are "small" reflect this scaling. The constraint functions are scaled so that all columns of $A^{(k)}$ are of unit length (unless $\|a_i^{(k)}\|$ is exactly zero, in which case the i-th scaling factor is unity). The motivation for this strategy is that a unit change in x along a constraint normal should produce (to first order) a similar change in the value of each constraint. Furthermore, this column scaling is considered beneficial when carrying out the QR decomposition with column interchanges (Golub, Klema and Stewart, 1976). The weight

corresponding to a constraint might alternatively be chosen to make the Lagrange multipliers of the scaled problem unity, or to make the active constraints approach zero at a uniform rate, i.e.,

$$\lim_{k \to \infty} (\hat{c}_i^{(k)} / \hat{c}_j^{(k)}) = 1.$$

5.6. Lagrange Multiplier Estimates

The description of the algorithms given here provides for computation of only first-order estimates of the Lagrange multipliers. It has been shown (Gill and Murray, 1977) that the use of first-order estimates in projected Lagrangian algorithms does not inhibit a higher-order rate of convergence, as it does for methods based on unconstrained minimization of augmented Lagrangian functions. However, little can be lost by employing higher-order multiplier estimates if they are available. The implementation of the algorithms includes the scheme suggested by Gill and Murray (1977) of computing and comparing first-order and higher-order estimates (when appropriate).

5.7. Quadratic Programming Sub-Problem

At $\overset{*}{x}$, the matrix $Z^T W Z$ must be positive semi-definite; in practice, it is nearly always strictly positive definite. Consequently, in the neighborhood of $\overset{*}{x}$ one would expect $Z^T W Z$ (and hence $Z^T S Z$) to be positive definite, which means that the quadratic program defining the trajectory search direction is well-posed. At points far from the solution, however, it cannot be assumed that $Z^T S Z$ is positive

definite, or even semi-definite. If Z^TSZ is indefinite but non-singular, p defines the step to a saddle point or possibly even a maximum of the projected quadratic form. If Z^TSZ is singular, p is no longer defined. In either case, a modification to the definition of the search direction is clearly necessary (this remark applies equally to any projected Lagrangian method).

The modification required is directly analogous to that in Newton's method for unconstrained optimization: if the Hessian matrix G is indefinite, the search direction is not defined by $-G^{-1}g$, but rather as $-\bar{G}^{-1}g$ (where $\bar{G}$ is guaranteed to be positive definite), or as a direction of negative curvature. In the current implementation, the matrix Z^TSZ is factorized by a modified Cholesky algorithm (Gill and Murray, 1974a). If Z^TSZ is sufficiently positive definite, this procedure is identical to the standard Cholesky algorithm; otherwise, it is equivalent to applying the standard method to the matrix $Z^TSZ + E$, where E is a non-negative diagonal matrix, which is determined during the factorization.

The importance of altering the sub-problem if S is unsatisfactory emphasizes the necessity to solve the quadratic program by a procedure that will determine whether Z^TSZ is positive definite. This information would not be known, for example, if the search direction were computed using the theoretically equivalent formula (when S is non-singular):

$$p = S^{-1}(\hat{A}(\hat{A}^T S^{-1} \hat{A})^{-1}(\hat{A}^T S^{-1} g + d) - g) ,$$

which is often advocated in algorithms that solve a quadratic programming sub-problem. There are, in addition, serious numerical objections to the above formulation of p. By contrast, the procedures given in Sections 2 and 3 for calculating p are numerically stable, and also allow an independent check to be made to determine whether the two portions of the search direction are reasonably scaled.

5.8. Special Treatment of Linear Constraints

Many of the features of the trajectory algorithms are included specifically to cater for the nonlinearity of the constraints. For example, the penalty or barrier merit function enables a sensible balance to be struck between reducing the objective function and satisfying the constraints. With exclusively linear constraints, such a merit function is no longer required. If all the constraints of problem P1 were linear, the solution could be obtained by algorithms specifically designed for such problems (see, e.g., Gill and Murray, 1974c; Murtagh and Saunders, 1978). It is typical of algorithms for linear constraints that all iterates are feasible; in addition, a certain subset of constraints is exactly satisfied at each estimate of the solution.

If both linear and nonlinear constraints are included in the problem to be solved, the linear constraints should be treated separately, and a trajectory algorithm will deal only with the nonlinear constraints, as described below. The problem of concern is of the form

$$\begin{aligned} \min \quad & F(x) \\ \text{subject to} \quad & c(x) \geq 0 \\ & A^T x \geq b, \end{aligned}$$

where A^T is an $\ell \times n$ matrix, and b is an ℓ-vector. It should be noted that bounds on the variables are usually distinguished from general linear constraints, because of the simplifications that result from the special form. In the present context, however, this distinction does not influence the sequence of iterates, and hence bounds are not considered separately. Linear equality constraints are not included in this problem statement, but their treatment is completely straightforward.

It is assumed that $x^{(k)}$ satisfies exactly a subset of the linear constraints, and is strictly feasible with respect to the remaining linear constraints. Let $\bar{A}^{(k)}$ denote the matrix whose columns are the coefficients of the active set of linear constraints at $x^{(k)}$, and let $\tilde{A}^{(k)}$ be the Jacobian matrix of the "active" set of nonlinear constraints (see Sections 5.1 and 5.2). In terms of the previous notation,

$$\hat{A}^{(k)} = [\bar{A}^{(k)} \vdots \tilde{A}^{(k)}] .$$

In order to remain "on" the active linear constraints, the search direction must satisfy:

$$\bar{A}^{(k)^T} p = 0 .$$

The quadratic programming sub-problem to be solved by the search direction $p^{(k)}$ is accordingly given by:

$$\min \ \frac{1}{2} p^T S^{(k)} p + p^T g^{(k)}$$

$$\text{subject to} \ \bar{A}^{(k)^T} p = 0$$

$$\tilde{A}^{(k)^T} p = d,$$

where the vector d is defined in Section 2.3 for the penalty algorithm and in Section 3.3 for the barrier algorithm. The linear constraints of this quadratic program can be written as:

$$\hat{A}^{(k)^T} p = \hat{d} \ ,$$

where:

$$\hat{d} = \begin{bmatrix} 0 \\ \cdots \\ d \end{bmatrix} \ .$$

Therefore, the modification of the trajectory algorithms to include linear constraints is quite straightforward, since the only difference between the old and new sub-problems is the definition of the right-hand side of the linear constraints.

Only the nonlinear constraints are included in choosing the steplength to reduce a penalty or barrier function. The techniques for modifying the active set of linear constraints are given in Gill and Murray (1974c).

6. Numerical Results

6.1. Background

It is increasingly recognized that the evaluation of optimization algorithms is a complicated process, for a variety of reasons -- for example, there is no universal agreement on a "merit function" to measure algorithm performance, and it is often difficult to separate the influences of the theoretical aogorithm and the details of implementation. Initial efforts have been made to apply a systematic technique to performance evaluation (Lyness and Greenwell, 1977; Moré, Garbow, and Hillstrom, 1978), but at present this field remains in a primitive slate.

The above remarks indicate our view that the presentation of selected numerical results, to display the superiority of a proposed algorithm compared to other previously published techniques, cannot be taken as definitive, since the relative performance of two methods depends very strongly on the chosen example, the starting point, the measure of quality, and even on termination criteria. Nonetheless, some numerical results will be given for a few examples, to indicate that the suggested algorithms have been successful in practice; details of the implementations and of numerous additional examples are given in Wright (1976).

6.2. Test Examples

The following examples are given in full in the cited references. The iterations were terminated when $\|Z^T g\|$ and $\|\hat{c}\|$ were both less

than 10^{-6}; other details of implementation, such as the line search tolerance, etc., are given in Wright (1976). The calculation of Z^TWZ by "finite differences" refers to the scheme discussed in Section 4.2, where the gradient of the Lagrangian function is differenced along the columns of Z. In the tables of results, "evaluations" refers to the number of evaluations of the objective and constraint functions, and their gradients (except for the barrier trajectory algorithm, where the additional constraint evaluations required to maintain feasibility are indicated separately).

Example 1: (Powell, 1969)

This example has five variables, and three nonlinear equality constraints. The problem was posed as originally given by Powell, i.e., with $F(x) = x_1 x_2 x_3 x_4 x_5$. An exponential transformation of $F(x)$ is sometimes used to avoid unboundedness when solving this problem with penalty function or augmented Lagrangian methods; however, this transformation drastically alters the scaling of the problem. With the present algorithms, the modification of $F(x)$ is unnecessary, and the relative scaling of the objective and constraint functions is more evenly balanced.

(a) Starting point = $(-2, 2, 2, -1, -1)^T$

	Iterations	Evaluations
Penalty, Z^TWZ by finite differences	6	18
Penalty, BFGS update of W	7	8
Penalty, PSB update of W	8	8

(b) Starting point = $(-2, -2, -2, -2, -2)^T$

	Iterations	Evaluations
Penalty, Z^TWZ by finite differences	8	24
Penalty, BFGS update to W	12	16
Penalty, PSB update to W	11	13

Example 2: Rosen-Suzuki (Rosen and Suzuki, 1965)

This example has four variables and three nonlinear inequality constraints. Two constraints are active at the solution.

(a) Starting point = $(0, 0, 0, 0,)^T$

	Iterations	Evaluations
Penalty, Z^TWZ by finite differences	10	35
Penalty, BFGS update to W	14	21
Penalty, PSB update to W	14	19
Barrier, Z^TWZ by finite differences	13	44 (+ 6 constraint evaluations)

(b) Starting point = $(3, 3, 3, 3)^T$

	Iterations	Evaluations
Penalty, Z^TWZ by finite differences	14	37
Penalty, BFGS update to W	21	32
Penalty, PSB update to W	20	30

Example 3: Example 9 from Wright (1976)

This example has five variables and three nonlinear inequality constraints. Two constraints are active at the solution.

Because this extremely difficult problem has been published in only one reference, it will be stated in full:

$$\text{minimize} \quad 10x_1x_4 - 6x_3x_2^2 + x_2x_1^3 + 9\sin(x_5 - x_3) + x_5^4x_4^2x_2^3$$

$$\text{subject to} \quad x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \leq 20$$

$$x_1^2x_3 + x_4x_5 + 2 \geq 0$$

$$x_2^2x_4 + 10x_1x_5 - 5 \geq 0$$

(a) Starting point = $(1, 1, 1, 1, 1)^T$

	Iterations	Evaluations
Penalty, Z^TWZ by finite differences	13	66
Penalty, BFGS update to W	28	57
Penalty, PSB update to W	22	48
Barrier, Z^TWZ by finite differences	20	84 (+ 15 constraint evaluations)

Example 4: "Optimal hexagon" (Murray, 1969a; Wright, 1976)

This problem involves maximizing the area of a hexagon, such that no two of the vertices are farther than one unit apart; this specification produces 9 variables, and 25 constraints (9 linear, 16 nonlinear). Six nonlinear constraints are active at the solution.

The problem can be generalized to any number of vertices, and is of special interest because the iterates can be represented geometrically in two dimensions.

(a) Starting point = $(\frac{1}{3}, \frac{2}{3}, \frac{11}{10}, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{2}{3}, -\frac{1}{3}, -\frac{2}{3})^T$

	Iterations	Evaluations
Penalty, Z^TWZ by finite differences	9	40
Penalty, BFGS update to W	13	20
Penalty, PSB update to W	10	10

(b) Starting point = $(\frac{1}{10}, \frac{1}{8}, \frac{1}{6}, \frac{1}{9}, \frac{1}{7}, \frac{1}{5}, \frac{1}{4}, -\frac{1}{4}, -\frac{1}{5})^T$

	Iterations	Evaluations
Penalty, Z^TWZ by finite differences	12	73
Penalty, BFGS update to W	17	30
Penalty, PSB update to W	17	36
Barrier, Z^TWZ by finite differences	15	78 (+ 14 constraint evaluations)

6.3. Assessment of Results

The notable features of the results given in Section 6.2, and numerous other numerical experiments with the trajectory algorithms, are the following:

1. In terms of function/gradient evaluations, the methods are competitive with the best previously published figures (in many instances, the present results are much superior).
2. The trajectory methods are able to converge efficiently even when the initial point is far from optimal.
3. The computation of the search direction requires a fixed number of arithmetic operations (depending on the number of variables and the number of active constraints), and can thus be expected to require less work than in some other projected Lagrangian methods (for example, those that obtain the search direction by solving a complete linearly constrained problem).
4. The barrier algorithm is generally quite successful at avoiding the danger common with feasible-point methods of becoming "trapped" near the boundary of the feasible region, and thereby tending to approach the optimum in painfully short steps (see Avriel, 1976, Chapter 13).

7. Conclusion

Robust algorithms for nonlinearly constrained optimization are by necessity complex. The strategy underlying any such algorithm inevitably depends on using local information to deduce global properties that lead to an improved estimate of the solution; however, a successful strategy should not depend on conditions that hold only for special cases, or in a close neighborhood of the optimum.

The algorithms described in this paper can achieve an excellent local rate of convergence near the solution. In addition, they are able to adapt to difficult circumstances, so that progress can be guaranteed far from the solution, even when local indications are misleading. The current implementations have proved thus far to be successful and efficient in solving a large selection of non-trivial problems.

The trajectory algorithms can be applied to problems with varying levels of derivative information; the barrier algorithm has the additional unusual feature of retaining strict feasibility. The implementations include state-of-the-art, numerically stable techniques for matrix factorization, univariate search, and computation of Lagrange multiplier estimates. However, much work remains to be done before certain questions are fully resolved -- particularly with respect to the use of quasi-Newton techniques for approximating the Hessian matrix of the Lagrangian function.

8. References

Avriel, M. (1976). Nonlinear Programming, Analysis and Methods, Prentice-Hall, New Jersey.

Biggs, M.C. (1972). "Constrained minimization using recursive equality quadratic programming", in Numerical Methods for Non-linear Optimization (F.A. Lootsma, ed.), pp. 411-428, Academic Press, London and New York.

Biggs, M.C. (1974). The Development of a Class of Constrained Optimization Algorithms and Their Application to the Problem of Electric Power Scheduling, Ph.D. Thesis, University of London.

Broyden, C.G. (1970). The convergence of single-rank quasi-Newton methods, Math. Comp., vol 24, pp. 365-382.

Colville, A.R. (1968). A comparative study on nonlinear programming codes, Report 320-2949, IBM New York Scientific Center.

Dennis, J.E. and J.J. Moré (1977). Quasi-Newton methods, motivation and theory, SIAM Review, vol. 19, pp. 46-89.

Fiacco, A.V., and G.P. McCormick (1968). Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley and Sons, New York.

Garcia-Palomares, U.M. and O.L. Mangasarian (1976). Superlinearly convergent quasi-Newton algorithms for nonlinearly constrained optimization problems, Math. Prog. vol. 11, pp. 1-13.

Gill, P.E., Golub, G.H., Murray, W., and M.A. Saunders (1974). Methods for modifying matrix factorizations, Math. Comp., vol. 28, pp. 505-535.

Gill, P.E. and W. Murray (1974a). Newton-type methods for unconstrained and linearly constrained optimization, Math. Prog., Vol. 7, pp. 311-350.

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

Gill, P.E. and W. Murray (eds.) (1974c). Numerical Methods for Constrained Optimization, Academic Press, London and New York.

Gill, P.E. and W. Murray (1977). The computation of Lagrange multiplier estimates for constrained minimization, Report NAC 77, National Physical Laboratory, England.

Golub, G.H., Klema, V. and G.W. Stewart (1976). Rank degeneracy and least-squares problems, Report 76-559, Computer Science Department, Stanford University.

Han, S.P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming problems, Math. Prog., vol. 11, pp. 263-282.

Han, S.P. (1977). A globally convergent method for nonlinear programming, J. Opt. Theory Appl., vol. 22, pp. 297-310.

Lawson, C.L. and R.J. Hanson (1974). Solving Least Squares Problems, Prentice-Hall, New Jersey.

Lyness, J.N. and C. Greenwell (1977). A pilot scheme for minimization software evaluation, Tech. Memo. 323, Argonne National Laboratory, Illinois.

Moré, J.J., Garbow, B.S. and K.E. Hillstrom (1978). Testing unconstrained optimization software, Tech. Memo. 324, Argonne National Laboratory, Illinois.

Murray, W. (1969a). Constrained optimization, Report MA79, National Physical Laboratory, Teddington, England.

Murray, W. (1969b). "An algorithm for constrained minimization", in Optimization (R. Fletcher, ed.), pp. 247-258, Academic Press, London and New York.

Murray, W. (1971). Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions, J. Opt. Theory Appl., vol 7, pp. 189-196.

Murray, W. and M.H. Wright (1976). Efficient linear search algorithms for the logarithmic barrier function, Report SOL 76-18, Department of Operations Research, Stanford University.

Murtagh, B.A. and M.A. Saunders (1978). Large-scale linearly constrained optimization, Math. Prog., vol. 14, pp. 41-72.

Powell, M.J.D. (1969). "A method for nonlinear constraints in minimization problems", in Optimization (R. Fletcher, ed.), pp. 283-298, Academic Press, London and New York.

Powell, M.J.D. (1970). "A new algorithm for unconstrained optimization", in Nonlinear Programming (J.B. Rosen, O.L. Mangasarian, K. Ritter, eds.), pp. 31-65, Academic Press, London and New York.

Powell, M.J.D. (1972). "Problems related to unconstrained optimization", in Numerical Methods for Unconstrained Optimization (W. Murray, ed.), pp. 29-55, Academic Press, London and New York.

Powell, M.J.D. (1977a). A fast algorithm for nonlinearly constrained optimization calculations, Report DAMTP 77/NA 2, University of Cambridge, England.

Powell, M.J.D. (1977b). Variable metric methods for constrained optimization, Report DAMTP 77/NA 5, University of Cambridge, England.

Robinson, S.M. (1972). A quadratically convergent algorithm for general nonlinear programming problems, Math. Prog., Vol. 3, pp. 145-156.

Rosen, J.B. and J. Kreuser (1972). "A gradient projection algorithm for nonlinear constraints", in Numerical Methods for Non-linear Optimization (F.A. Lootsma, ed.), pp. 297-300, Academic Press, London and New York.

Rosen, J.B. and S. Suzuki (1965). Construction of nonlinear programming test problems, Comm. ACM, vol. 8, p. 113.

Rosen, J.B. (1977). Two-phase algorithm for nonlinear constraint problems, Report 77-22, Department of Operations Research, Stanford University.

Ryan, D.M. (1974). "Penalty and barrier functions", in Numerical Methods for Constrained Optimization (P.E. Gill and W. Murray, eds.), pp. 175-190, Academic Press, London and New York.

Wilson, R.B. (1963). A Simplicial Algorithm for Concave Programming, Ph.D. Thesis, Graduate School of Business Administration, Harvard University.

Wright, M.H. (1976). Numerical methods for nonlinearly constrained optimization, Report 193, Stanford Linear Accelerator Center.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 78-23	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PROJECTED LAGRANGIAN METHODS BASED ON THE TRAJECTORIES OF PENALTY AND BARRIER FUNCTIONS		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER SOL 78-23
7. AUTHOR(s) Walter Murray and Margaret H. Wright		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research -- SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Operations Research Program -- ONR Department of the Navy 800 N. Quincy St., Arlington, VA 22217		12. REPORT DATE November 1978 13. NUMBER OF PAGES 67
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

nonlinear programming
penalty function
Lagrangian method
nonlinearly constrained optimization
barrier function
feasible-point method

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED

OVER

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 78-23 PROJECTED LAGRANGIAN METHODS BASED ON THE TRAJECTORIES OF PENALTY AND BARRIER FUNCTIONS

by Walter Murray and Margaret H. Wright

This report contains a complete derivation and description of two algorithms for nonlinearly constrained optimization which are based on properties of the solution trajectory of the quadratic penalty function and the logarithmic barrier function. The methods utilize the penalty and barrier functions only as merit functions, and do not generate iterates by solving a sequence of ill-conditioned problems. The search direction is the solution of a simple well-posed quadratic program (QP), where the quadratic objective function is an approximation to the Lagrangian function; the steplength is based on a sufficient decrease in a penalty or barrier function, to ensure progress toward the solution.

The penalty trajectory algorithm was first proposed by Murray in 1969; the barrier trajectory algorithm, which retains feasibility throughout, was given by Wright in 1976. Here we give a unified presentation of both algorithms, and indicate their relationship to other QP-based methods. Full details of implementation are included, as well as numerical results that display the success of the methods on non-trivial problems.