Mathematical Programming 36 (1986) 183-209
North-Holland

ON PROJECTED NEWTON BARRIER METHODS FOR LINEAR PROGRAMMING AND AN EQUIVALENCE TO KARMARKAR'S PROJECTIVE METHOD

Philip E. GILL, Walter MURRAY, Michael A. SAUNDERS
Department of Operations Research, Stanford University, Stanford, CA 94305, USA

J.A. TOMLIN
Ketron Incorporated, Mountain View, California 94040, USA

Margaret H. WRIGHT
Department of Operations Research, Stanford University, Stanford, CA 94305, USA

Received 12 October 1985
Revised manuscript received 20 June 1986

Interest in linear programming has been intensified recently by Karmarkar's publication in 1984 of an algorithm that is claimed to be much faster than the simplex method for practical problems. We review classical barrier-function methods for nonlinear programming based on applying a logarithmic transformation to inequality constraints. For the special case of linear programming, the transformed problem can be solved by a "projected Newton barrier" method. This method is shown to be equivalent to Karmarkar's projective method for a particular choice of the barrier parameter. We then present details of a specific barrier algorithm and its practical implementation. Numerical results are given for several non-trivial test problems, and the implications for future developments in linear programming are discussed.

Key words: Linear programming, Karmarkar's method, barrier methods.

1. Introduction

Interest in linear programming methods arises in at least two different contexts: theoretical and practical. The long-established simplex method, developed by G.B. Dantzig in the late 1940's, has been known from the beginning to be of combinatorial complexity in the worst case. However, in practice it tends to require a number of iterations that is approximately *linear* in the problem dimension. The linear algebraic work associated with the simplex method typically involves an LU factorization of a square matrix (the *basis*). Each iteration involves the solution of two linear systems, followed by an update of the basis factors (to reflect replacement of one column). The factorization is periodically recomputed rather than updated, to preserve accuracy and to condense the factors. Although linear programs are often very large, the constraint matrix is normally very *sparse*. Sparse-matrix techniques

The research of the Stanford authors was supported by the U.S. Department of Energy Contract DE-AA03-76SF00326, PA No. DE-AS03-76ER72018; National Science Foundation Grants DCR-8413211 and ECS-8312142; the Office of Naval Research Contract N00014-85-K-0343; and the U.S. Army Research Office Contract DAAG29-84-K-0156.

The research of J.A. Tomlin was supported by Ketron, Inc. and the Office of Naval Research Contract N00014-85-C-0338.

have developed to the point where the factorization and updates required in the simplex method can be performed not only rapidly, but also with assured numerical stability (see the survey by Gill et al., 1984). From a practical viewpoint, these two features—a typically linear number of iterations, and fast methods for performing each iteration—imply that the simplex method is an effective and reliable algorithm for linear programming, despite its seemingly unfavorable complexity.

Many researchers, beginning with Dantzig himself, have observed the apparently unsatisfactory feature that the simplex method traverses the boundary of the feasible region. From the outset, attempts have been made to develop practical linear programming methods that cross the *interior* of the feasible region—for example, von Neumann (1947), Hoffman et al. (1953), Tompkins (1955, 1957) and Frisch (1957). Such methods have sometimes involved the application of *nonlinear* techniques to linear programs. However, none of these methods has previously been claimed, even by its developers, to be competitive in speed with the simplex method for general linear programs.

On the theoretical side, researchers attempted for many years to develop a linear programming algorithm with only polynomial complexity. In 1979, to the accompaniment of wide publicity, this issue was resolved when Khachiyan (1979) presented a worst-case polynomial-time method based on a nonlinear geometry of shrinking ellipsoids. Although initially it was thought that the ellipsoid methods might be as fast in practice as the simplex method, these hopes have not been realized. Broadly speaking, there are two major difficulties: first, the number of iterations tends to be very large; second, the computation associated with each iteration is much more costly than a simplex iteration because sparse-matrix techniques are not applicable.

Within the past two years, interest in linear programming has been intensified by the publication (Karmarkar, 1984a, b) and discussion of a linear programming algorithm that is not only polynomial in complexity, but also is claimed to be *much faster* than the simplex method for practical problems.

In Section 2, we first examine the well known barrier-function approach to solving optimization problems with inequality constraints, and derive a representation for the Newton search direction associated with the subproblem. In Section 3, we show a formal equivalence between the Newton search direction and the direction associated with Karmarkar's (1984a, b) algorithm. Section 4 describes a complete interior-point method for linear programming based on the barrier transformation, and Section 5 gives some numerical results obtained with a preliminary implementation of that method. The implications of these results and directions for future research are discussed in Section 6.

1.2. Notation

The term *projective method* will denote the algorithm given by Karmarkar (1984a, b) for the special linear program (3.1); see below. The term *barrier method* will often be used as an abbreviation for *projected Newton barrier method.* The vector norm $\|\cdot\|$ will always denote the Euclidean norm $\|v\|_2 = (v^T v)^{1/2}$.

2. A barrier-function approach

2.1. Applying a barrier transformation to a linear program

Barrier-function methods treat *inequality* constraints by creating a *barrier function*, which is a combination of the original objective function and a weighted sum of functions with a positive singularity at the constraint boundary. (Many barrier functions have been proposed; we consider only the logarithmic barrier function, first suggested by Frisch, 1955.) As the weight assigned to the singularities approaches zero, the minimum of the barrier function approaches the minimum of the original constrained problem. Barrier-function methods require a strictly feasible starting point for each minimization, and generate a sequence of strictly feasible iterates. (For a complete discussion of barrier methods, see Fiacco, 1979; both barrier and penalty function methods are described in Fiacco and McCormick, 1968. Brief overviews are given in Fletcher, 1981; and Gill, Murray and Wright, 1981.)

We consider linear programs in the following standard form:

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad c^{\mathrm{T}}x \\ &\text{subject to} \quad Ax = b, \ x \geq 0, \end{aligned} \tag{2.1}$$

where A is an $m \times n$ matrix with $m \leq n$. Let x^* denote the solution of (2.1), and note that

$$c = A^{\mathrm{T}}\pi^* + \eta^*, \tag{2.2a}$$

$$\eta^* \geq 0, \tag{2.2b}$$

$$\eta_j^* x_j^* = 0, \quad j = 1, \ldots, n, \tag{2.2c}$$

where π^* and η^* are Lagrange multipliers associated with the constraints $Ax = b$ and $x \geq 0$ respectively. The problem is said to be *primal nondegenerate* if exactly m components of x^* are strictly positive, and *dual nondegenerate* if exactly $n - m$ components of η^* are strictly positive.

When applying a barrier-function method to (2.1), the subproblem to be solved is:

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad F(x) \equiv c^{\mathrm{T}}x - \mu \sum_{j=1}^{n} \ln x_j \\ &\text{subject to} \quad Ax = b, \end{aligned} \tag{2.3}$$

where the scalar μ ($\mu > 0$) is known as the *barrier parameter* and is specified for each subproblem. The equality constraints cannot be treated by a barrier transformation, and thus are handled directly.

If $x^*(\mu)$ is the solution of (2.3), then $x^*(\mu) \to x^*$ as $\mu \to 0$ (see, e.g., Fiacco and McCormick, 1968). Very strong order relations can be derived concerning $x^*(\mu)$ and $c^{\mathrm{T}}x^*(\mu)$ (see, e.g., Mifflin, 1972, 1975; Jittorntrum, 1978; Jittorntrum and Osborne, 1978). In particular, when (2.1) is primal nondegenerate,

$$\|x^*(\mu) - x^*\| = O(\mu) \tag{2.4a}$$

for sufficiently small μ. When (2.1) is primal degenerate, the corresponding relation is

$$\|x^*(\mu) - x^*\| = O(\sqrt{\mu}). \tag{2.4b}$$

The optimality conditions for (2.3) imply that at $x = x^*(\mu)$, there exists a vector $\pi(\mu)$ such that

$$c = A^T\pi(\mu) + \mu D^{-1}e, \tag{2.5}$$

where

$$D = \text{diag}(x_j), \quad j = 1, \ldots, n, \tag{2.6}$$

and $e = (1, 1, \ldots, 1)^T$. Comparing (2.5) and (2.2), we see that if (2.1) is primal nondegenerate, $\pi(\mu) \to \pi^*$ as $\mu \to 0$, and

$$\lim_{\mu \to 0} \frac{\mu}{x_j^*(\mu)} = \eta_j^*. \tag{2.7}$$

2.2. Solution of the subproblem

Given a linearly constrained problem of the form

$$\text{minimize } F(x) \quad \text{subject to } Ax = b, \tag{2.8}$$

a standard approach is to use a *feasible-point descent method* (see, e.g., Gill, Murray and Wright, 1981). The current iterate x always satisfies $Ax = b$, and the next iterate $\bar{x}$ is defined as

$$\bar{x} = x + \alpha p, \tag{2.9}$$

where p is an n-vector (the *search direction*), and α is a positive scalar (the *steplength*). The computation of p and α must ensure that $A\bar{x} = b$ and $F(\bar{x}) < F(x)$.

The *Newton search direction* associated with (2.8) is defined as the step to the minimum of the quadratic approximation to $F(x)$ derived from the local Taylor series, subject to retaining feasibility. Thus, the Newton search direction p_N is the solution of the following quadratic program:

$$\begin{aligned} &\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad g^Tp + \tfrac{1}{2}p^THp \\ &\text{subject to} \quad Ap = 0, \end{aligned} \tag{2.10}$$

where $g \equiv \nabla F(x)$ and $H \equiv \nabla^2 F(x)$. If π_N is the vector of Lagrange multipliers for the constraints in (2.10), then the required solution satisfies the linear system

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -p_N \\ \pi_N \end{pmatrix} = \begin{pmatrix} g \\ 0 \end{pmatrix}. \tag{2.11}$$

Note that π_N converges to the Lagrange multipliers for the constraints $Ax = b$ in the original problem (2.8).

2.3. The projected Newton search direction

When $F(x)$ is the barrier function in (2.3), its derivatives are

$$g(x) = c - \mu D^{-1} e \quad \text{and} \quad H(x) = \mu D^{-2},$$

where D is defined by (2.6). Note that g and H are well defined only if $x_j \neq 0$ for all j.

Let p_B (the *projected Newton barrier direction*) denote the Newton search direction defined by (2.11) when F is the barrier function of (2.3). The associated Lagrange multipliers will be denoted by π_B. Since $H(x)$ is positive definite when $x > 0$, p_B is finite and unique, and is a *descent direction* for $F(x)$, i.e., $(c - \mu D^{-1} e)^{\mathrm{T}} p_B < 0$.

It follows from (2.11) that p_B and π_B satisfy the equation

$$\begin{pmatrix} \mu D^{-2} & A^{\mathrm{T}} \\ A & 0 \end{pmatrix} \begin{pmatrix} -p_B \\ \pi_B \end{pmatrix} = \begin{pmatrix} c - \mu D^{-1} e \\ 0 \end{pmatrix}. \tag{2.12}$$

Rewriting (2.12) in terms of a vector r_B defined by $Dr_B = -\mu p_B$, we see that r_B and π_B satisfy

$$\begin{pmatrix} I & DA^{\mathrm{T}} \\ AD & 0 \end{pmatrix} \begin{pmatrix} r_B \\ \pi_B \end{pmatrix} = \begin{pmatrix} Dc - \mu e \\ 0 \end{pmatrix}. \tag{2.13}$$

It follows that π_B is the solution and r_B the optimal residual of the following linear least-squares problem:

$$\underset{\pi}{\text{minimize}} \quad \| Dc - \mu e - DA^{\mathrm{T}} \pi \|. \tag{2.14}$$

The projected Newton barrier direction is then

$$p_B = -(1/\mu) D r_B. \tag{2.15}$$

For a given positive μ, Newton's method will eventually reach a domain in which the "ideal" unit step along the direction p_B will be feasible and reduce the barrier function. The iterates can thereafter be expected to converge quadratically to $x^*(\mu)$. In general, the smaller μ, the smaller the attractive domain. The algorithm remains well defined as μ tends to zero. (The limiting case can be safely simulated in practice by using a very small value of μ.)

Note that feasible-direction methods can be made independent of the scaling of the search direction by appropriate re-scaling of the steplength α. We could therefore define the barrier search direction as

$$p_B = -D r_B \tag{2.16}$$

for any $\mu \geqq 0$. The "ideal" step would then be $\alpha = 1/\mu$.

The barrier search direction (2.16) with $\mu = 0$ in (2.14) is used in an algorithm proposed by Vanderbei, Meketon and Freedman (1985). From the above comments, we see that such an algorithm has no domain of quadratic convergence.

2.4. Upper bounds

The barrier transformation and the associated Newton search direction can also be defined for linear programs with upper and lower bounds on the variables, of the form:

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} && c^{\mathrm{T}}x \\ &\text{subject to} && Ax = b,\ l \leqslant x \leqslant u. \end{aligned}$$

The subproblem analogous to (2.3) is

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} && c^{\mathrm{T}}x - \mu \sum_{j=1}^{n} \ln(x_j - l_j) - \mu \sum_{j=1}^{n} \ln(u_j - x_j) \\ &\text{subject to} && Ax = b. \end{aligned}$$

The Hessian of the associated barrier function will be positive definite only if at least one of l_j or u_j is finite for every j. In this case, the least-squares problem analogous to (2.14) is

$$\underset{\pi}{\text{minimize}} \quad \|Dc - \mu\bar{D}e - DA^{\mathrm{T}}\pi\|.$$

Here, the matrices D and $\bar{D}$ are defined by $D = \operatorname{diag}(\delta_j)$ and $\bar{D} = \operatorname{diag}(\bar{\delta}_j)$, where

$$\delta_j = 1/(1/s_j^2 + 1/t_j^2)^{1/2} \quad \text{and} \quad \bar{\delta}_j = \delta_j(1/s_j - 1/t_j),$$

with $s_j = x_j - l_j$ and $t_j = u_j - x_j$. For simplicity, the remainder of the discussion will assume that the bounds are of the form in (2.1), i.e., $0 \leqslant x_j \leqslant \infty$, except for the artificial variable discussed in Section 4.2.

3. Relationship with Karmarkar's projective method

In this section, we show the connection between the barrier and projective methods when both are applied to a specialized linear program. We assume that the reader is familiar with the projective method; a good description is given in Todd and Burrell (1985).

3.1. Summary of the projective method

In the projective method, the linear program is assumed to be of the special form

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} && c^{\mathrm{T}}x \\ &\text{subject to} && Cx = 0, \qquad e^{\mathrm{T}}x = 1,\ x \geqslant 0. \end{aligned} \tag{3.1}$$

Let x_K^* denote a solution of (3.1). It is also assumed that

$$c^{\mathrm{T}}x_K^* = 0, \tag{3.2}$$

and that $Ce = 0$. (These assumptions can always be assured by transforming the problem.)

The optimality conditions for (3.1) imply that

$$c = C^{\mathrm{T}}\lambda_C + e\lambda_e + \eta, \tag{3.3}$$

where η is the Lagrange multiplier vector for the bound constraints of (3.1). The complementarity conditions at x_K^* imply that

$$x_j^*\eta_j = 0, \quad j = 1, \ldots, n, \tag{3.4}$$

where x_j^* denotes the jth component of x_K^*. Taking the inner product of c and x_K^* and using (3.2)-(3.4), we obtain $\lambda_e e^{\mathrm{T}}x_K^* = 0$. Since $e^{\mathrm{T}}x_K^* = 1$, it follows that $\lambda_e = 0$.

Any strictly positive diagonal matrix D defines the following projective transformations, which relate any strictly feasible point x and the transformed point x':

$$x' = \frac{1}{e^{\mathrm{T}}D^{-1}x} D^{-1}x, \quad x = \frac{1}{e^{\mathrm{T}}Dx'} Dx'. \tag{3.5}$$

In the projective method, given an iterate x, D is defined as $\mathrm{diag}(x_1, \ldots, x_n)$. (Note that D is the same as the diagonal matrix (2.6) associated with the barrier method, and that $De = x$.) The next iterate in the *transformed* space is given by

$$\bar{x}' = x' - \alpha' r_K, \tag{3.6}$$

where $r_K = Dc - DC^{\mathrm{T}}\pi_K - \phi e$ is the optimal residual of the linear least-squares problem

$$\underset{\pi,\phi}{\text{minimize}} \quad \left\| Dc - (DC^{\mathrm{T}} \; e)\begin{pmatrix}\pi\\ \phi\end{pmatrix}\right\|. \tag{3.7}$$

We assume henceforth that $(DC^{\mathrm{T}} \; e)$ has full rank, so that the solution of (3.7) is unique. (Note that $(DC^{\mathrm{T}} \; e)$ and $(C^{T} \; e)$ have full rank if and only if C has full rank; the last column is orthogonal to the others in both cases.) The steplength α' in (3.6) is chosen to ensure strict feasibility of $\bar{x}'$ as well as descent in the transformed "potential function" (see Karmarkar, 1984a, b, for details).

The new iterate $\bar{x}_K$ in the original parameter space is obtained by applying the transformation (3.5) to $\bar{x}'$, so that

$$\bar{x}_K = \frac{1}{e^{\mathrm{T}}D(x' - \alpha' r_K)} D(x' - \alpha' r_K) = \gamma(x - \tilde{\alpha} D r_K),$$

where γ is chosen to make $e^{\mathrm{T}}\bar{x}_K = 1$.

3.2. *Properties of the projective method*

Let π_C denote the solution of the least-squares problem

$$\underset{\pi}{\text{minimize}} \quad \| Dc - DC^{\mathrm{T}}\pi \|. \tag{3.8}$$

Using the normal equations associated with (3.8), observe that π_C satisfies

$$CD^2C^T\pi_C = CD^2c. \tag{3.9}$$

We also define

$$r_C = Dc - DC^T\pi_C, \tag{3.10a}$$

$$\mu_C = x^T r_C. \tag{3.10b}$$

We now state some properties of various quantities appearing in the projective method.

Lemma 3.1. *The solution of* (3.7) *is* $\pi_K = \pi_C$ *and* $\phi = c^T x/n$.

Proof. From the normal equations associated with (3.7) and the relation $Cx = 0$, we see that π_K and ϕ satisfy

$$CD^2C^T\pi_K = CD^2c, \qquad e^T e\phi = c^T x. \tag{3.11}$$

Since the solution of (3.7) is unique, comparison of (3.9) and (3.11) gives the result directly. □

Lemma 3.2. *In the projective method,*

$$r_K = r_C - \phi e, \qquad \tilde{\alpha} = n\alpha', \qquad \gamma = \frac{1}{1 + \tilde{\alpha}(\phi - \mu_C)},$$

and the new iterate may be written as

$$p_K = \mu_C x - Dr_C, \qquad \bar{x}_K = x + \tilde{\alpha}\gamma p_K.$$

Proof. The result follows from algebraic manipulation of the relevant formulae. □

3.3. Relationship with the barrier search direction

When the barrier method is applied to problem (3.1), it follows from (2.14) and (2.15) that the search direction is given by

$$p_B = -\frac{1}{\mu} Dr_B,$$

where

$$r_B = Dc - \mu e - DC^T\pi_B - \theta_B De,$$

with π_B and θ_B taken as the solution of

$$\underset{\pi,\theta}{\text{minimize}} \quad \left\| Dc - \mu e - D(C^T \; e)\begin{pmatrix}\pi \\ \theta\end{pmatrix}\right\|. \tag{3.12}$$

The new iterate is then defined as

$$\bar{x}_B = x + \alpha p_B,$$

for some steplength α. We assume now that $(C^T \; e)$ has full rank, so that the solution of (3.12) is unique.

Lemma 3.3. *If the barrier parameter is chosen to be $\mu = \mu_C$, then $\theta_B = 0$ and $\pi_B = \pi_C$.*

Proof. The normal equations associated with (3.12) and the relation $Cx = 0$ imply that π_B and θ_B satisfy

$$CD^2C^T\pi_B + CD^2e\theta_B = CD^2c, \qquad x^TDC^T\pi_B + (x^Tx)\theta_B = x^TDc - \mu. \tag{3.13}$$

When $\mu = \mu_C$, direct substitution using (3.9) and (3.10) shows that (3.13) is satisfied for $\pi_B = \pi_C$ and $\theta_B = 0$. Since the solution of (3.13) is unique, this proves the lemma. □

Lemma 3.4. *If $\mu = \mu_C$, then*

$$r_B = r_C - \mu_C e, \qquad p_B = \frac{1}{\mu_C}(\mu_C x - Dr_C), \qquad \bar{x}_B = x + \alpha p_B.$$

Proof. As in Lemma 3.2, the result is obtained by algebraic manipulation. □

Comparing Lemmas 3.2 and 3.4, the main result now follows.

Theorem 3.1. *Suppose that the projective method and the barrier method are applied to problem* (3.1), *using the same initial point. If the barrier parameter is $\mu = \mu_C$, the search directions p_B and p_K are parallel. Further, if the steplengths satisfy $\alpha = \tilde{\alpha}\gamma\mu_C$, the iterates $\bar{x}_B$ and $\bar{x}_K$ are identical.*

Theorem 3.1 is an existence result, showing that a special case of the barrier method would follow the same path as the projective method. This does not mean that the barrier method *should* be specialized. For example, the value μ_C is an admissible barrier parameter only if it is positive. Note that μ_C is positive initially, if the starting point x_0 is a multiple of e. Furthermore, μ_C tends to zero as the iterates converge to x_K^*, and could therefore be a satisfactory choice for the barrier algorithm as the solution is approached.

Similarly, as the barrier method converges to a solution of the original problem for any suitable sequence of barrier parameters, θ_B will converge to λ_e, which is zero. This is consistent with the choice $\mu = \mu_C$, which gives $\theta_B = 0$ directly.

4. A projected Newton barrier algorithm

In this section, we give details of the barrier algorithm used to obtain the numerical results of Section 5. The algorithm applies to problems of the standard form (2.1). Each iteration is of the form $\bar{x} = x + \alpha p$ (2.9), where the search direction is p_B defined by (2.14)-(2.15), and *the barrier parameter may be changed at every iteration.* Any

tolerances described are intended to be suitable for machines whose relative precision is about 10^{-15}.

Alternative approaches to certain parts of the algorithm are discussed in Section 6.

4.1. The iteration

At the start of each iteration, the quantities μ, x, π and η are known, where $\mu > 0$, $x > 0$, $Ax = b$, and $\eta = c - A^{\mathrm{T}}\pi$. For computational reasons we compute a *correction* to π at each stage, since a good estimate is available from the previous iteration. The main steps of the iteration then take the following form.

1. Define $D = \operatorname{diag}(x_j)$ and compute $r = D\eta - \mu e$. Note that r is a scaled version of the residual from the optimality condition (2.5) for the barrier subproblem, and hence that $\|r\| = 0$ if $x = x^*(\mu)$.
2. Terminate if μ and $\|r\|$ are sufficiently small.
3. If appropriate, reduce μ and reset r.
4. Solve the least-squares problem

$$\underset{\delta\pi}{\text{minimize}} \|r - DA^{\mathrm{T}}\delta\pi\|. \tag{4.1}$$

5. Compute the updated vectors $\pi \leftarrow \pi + \delta\pi$ and $\eta \leftarrow \eta - A^{\mathrm{T}}\delta\pi$. Set $r = D\eta - \mu e$ (the updated scaled residual) and $p = -(1/\mu)Dr$ (cf. (2.14) and (2.15)).
6. Find α_M, the maximum value of α such that $x + \alpha p \geq 0$.
7. Determine a steplength $\alpha \in (0, \alpha_M)$ at which the barrier function $F(x + \alpha p)$ is suitably less than $F(x)$.
8. Update $x \leftarrow x + \alpha p$.

All iterates satisfy $Ax = b$ and $x > 0$. The vectors π and η approximate the dual variables π^* and reduced costs η^* of the original linear program (cf. (2.2) and (2.5)).

4.2. The feasibility phase

In order to apply the barrier algorithm to (2.1), a strictly feasible starting point is necessary. Such a point may be found by the following "textbook" phase 1 procedure in which a barrier method is applied to a modified linear program. For any given initial point $x_0 > 0$, we define $\xi_0 s = b - Ax_0$ with $\|s\| = 1$, and solve the modified linear program

$$\begin{aligned} &\underset{x,\xi}{\text{minimize}} \quad \xi \\ &\text{subject to} \quad (A \;\; s)\begin{pmatrix} x \\ \xi \end{pmatrix} = b, \;\; x \geq 0, \; \xi \geq 0, \end{aligned} \tag{4.2}$$

using the feasible starting point $x_0 > 0$, $\xi_0 = \|b - Ax_0\|$. (Note that, even if A is sparse, the additional column s in (4.2) will in general be *dense*.) In our experiments we have used $x_0 = \|b\|e$.

When $\xi = 0$, a suitable point has been found. Since the barrier transformation will not allow ξ to reach the desired value of zero, ξ must be treated differently

from the other variables in solving (4.2) with a barrier algorithm. In our implementation, the search direction p and the maximum step α_M are computed as if the variable ξ were subject to the bound $\xi \geq -1$. If the step α causes ξ to become negative, an appropriate shorter step is taken and phase 1 is terminated. The original linear program is presumed to be infeasible if the final ξ is positive for a sufficiently small value of μ.

As an alternative, we note that the convergence of the barrier method appears to be moderately insensitive to the choice of linear objective function. This suggests a single-phase algorithm in which an objective function of the form $\omega c^T x + \xi$ is used in (4.2), for some positive value of the scalar ω. When ξ reaches zero, it is thereafter excluded from the problem. If a single value of ω can be retained at every iteration, only a slight change in the definition of the linear program is required after a feasible point is found. Some preliminary results with ω fixed at $0.1/\|c\|$ seem promising; see Section 5. In general, a sequence of decreasing values of ω may be needed to ensure that a feasible point is always obtained if one exists.

4.3. *Solution of the least-squares subproblems*

For problems of even moderate size, the time required to perform an iteration will be dominated by solution of the least-squares problem (4.1). The widespread interest in interior-point methods has arisen because of their reported speed on large-scale linear programs. Consequently, problem (4.1) must be solved when A is large and sparse. Fortunately, methods for sparse least-squares problems have improved dramatically in the past decade. (For a recent survey, see Heath, 1984.)

An obvious approach to minimizing $\|r - DA^T\delta\pi\|$ is to solve the associated normal equations

$$AD^2A^T\delta\pi = ADr \tag{4.3}$$

using the Cholesky factorization $AD^2A^T = R^TR$ with R upper triangular. Reliable software exists for factorizing symmetric definite systems, notably SPARSPAK-A (George and Liu, 1981), MA27 (Duff and Reid, 1982, 1983), and YSMP (Eisenstat et al., 1982). If the original linear program (2.1) is primal nondegenerate, the matrix AD^2A^T will be non-singular even at the solution. However, for a degenerate problem, AD^2A^T becomes increasingly ill-conditioned as the solution is approached, and the accuracy of the computed version of AD^2A^T correspondingly deteriorates. Furthermore, any dense columns in A (such as s in phase 1) degrade the sparsity of R.

To alleviate these difficulties, we have used a "hybrid" method in which the least-squares problems are solved by a conjugate-gradient method (LSQR; Paige and Saunders, 1982a, b) with a triangular preconditioner R. Thus, an iterative method is applied to solve

$$\underset{z}{\text{minimize}} \quad \|r - (DA^TR^{-1})z\|, \tag{4.4}$$

and the correction $\delta\pi$ is recovered by solving $R\delta\pi = z$.

The preconditioner R comes from the Cholesky factorization of a sparse matrix that approximates AD^2A^T. Thus,

$$AD^2A^T \approx \tilde{A}D^2\tilde{A}^T = R^TR, \tag{4.5}$$

where $\tilde{A}$ and R are obtained as follows.

1. Before beginning the barrier algorithm, a preliminary row permutation is obtained from the symbolic factorization of a matrix $\tilde{A}\tilde{A}^T$, where $\tilde{A}$ is A with certain columns replaced by zero. For the results of Section 5, we excluded the artificial vector s in (4.2) and any columns of A containing 50 or more nonzeros. Subroutines GENQMD and SMBFCT of George and Liu (1981) were used to obtain a minimum-degree ordering P and to set up appropriate data structures for the subsequent numerical factorizations.

2. At each iteration of the barrier algorithm, further columns and/or rows of $\tilde{A}$ may be replaced by zero: columns for which $x_j \leq 10^{-6}$, and rows that have been marked for exclusion during earlier iterations. Subroutine GSFCT of George and Liu (1981) is used to obtain the Cholesky factorization

$$P\tilde{A}D^2\tilde{A}^TP^T = U^TU, \quad U \text{ upper triangular},$$

with the proviso that if a diagonal element of U satisfies $u_{ii}^2 \leq 10^{-12}$, the ith row of U is replaced by e_i^T, and the ith row of $P\tilde{A}$ is marked for exclusion in later iterations. The preconditioner for (4.4) is defined as $R = UP$.

3. After each iteration, any variables satisfying $x_j \leq 10^{-8}$ are changed to zero for the remaining iterations. This (conservative) test is unlikely to remove the "wrong" variables from the problem, but it allows some economy in computing R and solving the least-squares problems.

The performance of LSQR is strongly affected by the quality of the preconditioner, and by the specified convergence tolerance ATOL (see Paige and Saunders, 1982a). With the present implementation, we have $AD^2A^T = R^TR + E_1 + E_2$, where E_1 has low rank and $\|E_2\|$ is small; the value of ATOL is taken as 10^{-10}. In this situation, LSQR typically requires only one or two iterations to achieve acceptable accuracy in phase 2, and only two or three iterations in phase 1.

There is scope in future work for degrading the approximation (4.5) to obtain a sparser R more quickly, at the expense of further iterations in LSQR. In fact, Gay (1985) has reported considerable success in the analogous task of preconditioning the symmetric conjugate-gradient method in order to solve the normal equations (4.3). We discuss this further in Section 6.1.

4.4. Determination of the steplength

The steplength α in (2.9) is intended to ensure a reduction in the barrier function $F(x)$ in (2.3) at every iteration. Let $f(\alpha)$ denote $F(x+\alpha p)$, treated as a function of α, and let α_M be the largest positive feasible step along p. When $p = p_B$, $f'(0) < 0$; by construction of a positive singularity at the boundary of the feasible region, $f'(\alpha_M) = +\infty$. Thus, there must exist a point α^* in the interval $(0, \alpha_M)$ such that

$f'(\alpha^*)=0$. Because of the special form of f, α^* is unique and is the univariate minimizer of $f(\alpha)$ for $\alpha \in [0, \alpha_M]$.

In our algorithm, α is an approximation to a zero of the function $f'(\alpha)$. In order to obtain a "sufficient decrease" in F (in the sense of Ortega and Rheinboldt, 1970), an acceptable α is any member of the set

$$\Gamma = \{\alpha: |f'(\alpha)| \leq -\beta f'(0)\},$$

where β is a number satisfying $0 \leq \beta < 1$. (The smaller the value of β, the closer the approximation of α to a zero of f'.)

The computation of an acceptable steplength involves an iterative procedure for finding a zero of f'. Many efficient algorithms have been developed for finding the zero of a general univariate function (see, e.g., Brent, 1973), based on iterative approximation by a low-order polynomial. However, such methods tend to perform poorly in the presence of singularities. In order to overcome this difficulty, special steplength algorithms have been devised for the logarithmic barrier function (e.g., Fletcher and McCann, 1969; Murray and Wright, 1976). These special procedures are based on approximating $f(\alpha)$ by a function with a similar singularity.

Given an interval I such that $\alpha^* \in I$ and $\Gamma \subset I$, a new interval $\bar{I}$ ($\bar{I} \subset I$) is generated using α_ϕ, the zero of a simple monotonic function $\phi(\alpha)$ that approximates $f'(\alpha)$. Let $\alpha_B \in I$ be the current best estimate of α^*. Define the function $\phi(\alpha)$ to be

$$\phi(\alpha) = \gamma_1 + \frac{\gamma_2}{\alpha_M - \alpha},$$

where the coefficients γ_1 and γ_2 are chosen such that $\phi(\alpha_B) = f'(\alpha_B)$ and $\phi'(\alpha_B) = f''(\alpha_B)$. The new estimate of the zero of $f'(\alpha)$ is then given by

$$\alpha_\phi = \alpha_M + \gamma_2/\gamma_1.$$

Using this prescription, a *sequence* of intervals $\{I_j\}$ is generated such that $I_0 = [0, \alpha_M]$, $I_j \subset I_{j-1}$ and $\Gamma \subset I_j$. (For additional details, see Murray and Wright, 1976.) The first point α_ϕ that lies in Γ is taken as α.

In practice, a close approximation to the minimum of $F(x+\alpha p)$ can be obtained after a small number (typically 1-3) of estimates α_ϕ. Since the minimum is usually *very close to* α_M, at least one variable will become very near to its bound if an accurate search is performed. Although this may sometimes be beneficial, the danger exists—particularly in phase 1—that the optimal value of that variable could be far from its bound. Thus, performing an accurate linesearch may temporarily degrade the speed of convergence. To guard against this, we use $\alpha = 0.9\alpha_M$ in phase 1 (if $\omega = 0$ in the objective function). Otherwise, we set $\beta = 0.999$ in the linesearch and use $0.9\alpha_M$ as an initial step, which is normally accepted. If necessary, we compute the sequence of estimates α_ϕ as described.

4.5. *Choice of the barrier parameter*

In a "classical" barrier-function method (e.g., as described in Fiacco and McCormick, 1968), the usual procedure is to choose an initial value of μ, solve the

subproblem (2.3), and then decrease μ (say, by multiplying by a constant). In order for $x^*(\mu)$ to converge to x^*, it is essential that $\mu \to 0$. If the barrier search direction and a steplength as defined in Section 4.4 are used to solve (2.3) with a fixed μ, standard proofs for descent methods (see, e.g., Ortega and Rheinboldt, 1970) can be applied to guarantee convergence to $x^*(\mu)$. When (2.1) is primal nondegenerate and μ is "sufficiently small" (say, $\mu = \mu_{\min}$) it follows from (2.4a) that the final iterate of the barrier method will approximate the solution of the linear program (2.1) to within the accuracy specified by $\mu_{\min}$. If the problem is degenerate, (2.4b) implies that the solution will be less accurate.

Various strategies for changing μ can be devised. The main aim is to reduce μ as quickly as possible, subject to ensuring steady progress toward the solution. For example, only a *single step* of Newton's method could be performed for each of a decreasing sequence of μ-values. Alternatively, each value of μ could be retained until the new iterate satisfies some convergence criterion for the subproblem. We have not experimented with the values $\mu = \mu_C$ of Theorem 3.1 because of the difficulty (and artificiality) of converting general problems to the special form (3.1).

As indicated in the description of the algorithm, the vector $r = D\eta - \mu e$ is used to measure convergence for the current subproblem. The size of $\|r\|$ is monitored in our implementation, and the reduction of μ is controlled by two parameters as follows.

1. An initial "target level" for $\|r\|$ is defined to be $\tau = \|r_0\| * \text{RGFAC}$.
2. Whenever $\|r\| \leq \tau$, the barrier parameter is reduced to $\mu * \text{MUFAC}$, r is recomputed, and a new target level is defined to be $\tau = \|r\| * \text{RGFAC}$.

The parameters RGFAC and MUFAC should lie in the range (0, 1) to be meaningful. For example, the values RGFAC = 0.99, MUFAC = 0.25 allow a moderate reduction in μ almost every iteration, while RGFAC = MUFAC = 0.001 requests more discernible progress towards optimality for each subproblem, with a substantial reduction in μ on rare occasions.

4.6. Convergence tests

Two other parameters, μ_0 and $\mu_{\min}$, are used to define the initial and final values of the barrier parameter, and the degree of optimality required for the final subproblem. In the feasibility phase, μ is initialized to $\mu_0(1+\xi)/n$ and progressively reduced as described above until ξ reaches zero. In the optimality phase, μ is reset to $\mu_0(1+|c^Tx|)/n$ (except if $\omega > 0$ in the objective function) and again progressively reduced.

Whenever a reduction is about to take place (in Step 3 of the algorithm), a "final" barrier parameter is defined by

$$\mu_F = \mu_{\min}(1+\xi)/n \quad \text{or} \quad \mu_F = \mu_{\min}(1+|c^Tx|)/n,$$

depending on the phase. If the newly reduced μ is less than μ_F, the barrier parameter and the target level for $\|r\|$ are fixed for the remaining iterations at $\mu = \mu_F$ and

$\tau = \sqrt{m}\,\mu_F$ respectively. Termination then occurs in Step 2 of the algorithm when $\|r\| \leq \tau$.

In our experiments we have used $\mu_0 = 0.1$ and $\mu_{\min} = 10^{-6}$. If μ_0 is too large, a danger exists on problems for which the feasible region $Ax = b$, $x > 0$ is unbounded; since the barrier function is then unbounded below, the iterates can diverge in phase 1 before the artificial variable ξ reaches zero.

5. Numerical results

5.1. Performance of the barrier method on a standard test set

In this section we summarize the performance of the barrier algorithm described in Section 4 on problems from an LP test set in use at the Systems Optimization Laboratory. The first nine problems, which are available through *Netlib* (Dongarra and Grosse, 1985),[1] are listed in order of the number of rows. Problem SCSD6 was obtained from Ho and Loute (1981), and NZFRI is a model developed by the New Zealand Forestry Research Institute (Garcia, 1984).

All problems are in the form (2.1). To obtain constraints of the form $Ax = b$, any general inequality constraints are converted to equalities using slack variables. Details of the problems are given in Table 1. The value of "rows" refers to the number of general constraints, and "columns" to the number of variables, excluding slacks. The number "slacks" is defined above. The column "A" gives the number of nonzeros in the problem. This figure includes one for each slack but excludes the nonzeros in b and c.

The runs summarized in Tables 2–5 were made in double precision on an IBM 3081K (relative precision 2.2×10^{-16}). The source code was compiled with the IBM

Table 1
Problem statistics

Problem	Rows	Slacks	Columns	A	$\|x^*\|$	$\|\pi^*\|$
Afiro	27	19	32	102	9.7×10^{2}	3.9×10^{1}
ADLittle	56	41	97	424	6.1×10^{2}	6.2×10^{3}
Share2b	96	83	79	777	1.8×10^{2}	3.8×10^{2}
Share1b	117	28	225	1179	1.3×10^{6}	7.7×10^{1}
Beaconfd	173	33	262	3408	1.6×10^{5}	1.2×10^{2}
Israel	174	174	142	2443	9.1×10^{5}	5.6×10^{2}
BrandY	220	54	249	2202	6.5×10^{4}	8.7×10^{1}
E226	223	190	282	2768	9.6×10^{2}	4.1×10^{1}
BandM	305	0	472	2494	1.5×10^{3}	3.0×10^{1}
SCSD6	147	0	1350	4316	4.5×10^{0}	7.9×10^{1}
NZFRI	623	40	3521	12840	4.3×10^{5}	3.4×10^{5}

[1] For details, send electronic mail to *netlib@anl-mcs* or to *research!netlib* saying "*send index from lp/data*".

Fortran 77 compiler VS Fortran, using NOSDUMP, NOSYM and OPT(3). All times given are for a complete run, including model input and solution output.

Table 2 gives the number of iterations and CPU-seconds required by the primal simplex method, as implemented in the Fortran code MINOS 5.0 (May 1985), which maintains sparse LU factors of the basis matrix as described in Gill et al. (1986b). The default values of the parameters were used throughout (see Murtagh and Saunders, 1983), except that PARTIAL PRICE 10 was specified for the last two problems. Results are also given in the case where the constraints are scaled by an iterative procedure that makes the matrix coefficients as close as possible to one (Fourer, 1982).

Table 2
Results from the primal simplex code MINOS 5.0

	Optimal objective	No scaling			With scaling		
		Phase 1	Total	Time	Phase 1	Total	Time
Afiro	−464.75314	2	6	0.5	2	6	0.5
ADLittle	225494.96	28	123	1.3	30	98	1.1
Share2b	−415.73224	59	91	1.3	74	121	1.4
Share1b	−76589.319	135	296	3.4	144	260	2.8
Beaconfd	33592.486	8	38	1.9	6	39	1.8
Israel	−896644.82	109	345	5.0	41	231	3.7
BrandY	1518.5099	176	292	4.9	216	377	5.9
E226	−18.751929	109	570	9.4	101	471	7.5
BandM	−158.62802	167	362	7.6	280	534	10.0
SCSD6	50.500000	172	521	7.0	180	1168	14.4
NZFRI	−27892262.	2166	4131	146.0	942	2371	65.4

Many experiments were made during development of the barrier code, incorporating different choices for the parameters RGFAC and MUFAC (Section 4.5), which specify the accuracy of a given subproblem and the rate at which the barrier parameter is reduced. One aim was to find a set of values that could be used reliably on *all* problems. It was found that RGFAC = 0.1 and MUFAC = 0.1 gave the most consistent results.

Table 3 summarizes the performance of the barrier method with these values. The second and third columns of the table give the number of iterations to obtain a feasible point and the total iterations required to satisfy the convergence tests of Section 4.6. The fourth column gives the total CPU time (in seconds). The objective function values found by MINOS 5.0 were used to judge the accuracy of the final objective in the barrier runs. The underlined digits in the fifth column show the correct figures in the objective function on termination. The final two columns indicate the degree of feasibility and optimality of the final point.

Table 4 gives the results of applying the barrier algorithm with the same scaling procedure as in MINOS 5.0. Note that scaling alters the starting point $\|b\|e$ (and

Table 3

Barrier method
No scaling, RGFAC = 0.1, MUFAC = 0.1

Problem	Phase 1	Total	Time	Objective	$\frac{\|b-Ax\|}{\|b\|}$	$\frac{\|D(c-A^{T}\pi)\|}{\|x\|\|\pi\|}$
Afiro	4	18	0.4	−464.75314	10^{-11}	10^{-8}
ADLittle	14	35	1.1	225494.96	10^{-9}	10^{-8}
Share2b	7	22	1.3	−415.73224	10^{-8}	10^{-9}
Share1b	10	54	3.8	−76589.319	10^{-7}	10^{-10}
Beaconfd	21	39	9.0	33592.486	10^{-8}	10^{-9}
Israel	17	49	18.7	−896644.82	10^{-6}	10^{-10}
BrandY	19	39	7.8	1518.5099	10^{-7}	10^{-10}
E226	19	44	9.0	−18.751929	10^{-6}	10^{-10}
BandM	19	42	9.0	−158.62802	10^{-6}	10^{-10}
SCSD6	1	20	5.4	50.500000	10^{-8}	10^{-8}
NZFRI	24	54	53.7	−27892262.	10^{-5}	10^{-12}

Table 4

Barrier method
With scaling, RGFAC = 0.1, MUFAC = 0.1

Problem	Phase 1	Total	Time	Objective	$\frac{\|b-Ax\|}{\|b\|}$	$\frac{\|D(c-A^{T}\pi)\|}{\|x\|\|\pi\|}$
Afiro	4	18	0.4	−464.75314	10^{-11}	10^{-8}
ADLittle	13	32	1.1	225494.96	10^{-9}	10^{-9}
Share2b	8	23	1.4	−415.73224	10^{-8}	10^{-9}
Share1b	7	33	2.6	−76589.319	10^{-6}	10^{-11}
Beaconfd	24	44	9.9	33592.486	10^{-5}	10^{-10}
Israel	10	48	18.6	−896644.82	10^{-6}	10^{-10}
BrandY	19	42	8.5	1518.5099	10^{-7}	10^{-11}
E226	18	43	9.0	−18.751929	10^{-7}	10^{-10}
BandM	20	42	9.0	−158.62802	10^{-6}	10^{-10}
SCSD6	1	21	5.7	50.500000	10^{-8}	10^{-8}
NZFRI	23	53	53.7	−27892262.	10^{-7}	10^{-12}

all subsequent iterates), but the results are essentially the same as without scaling. In this regard, the barrier algorithm appears to be more robust than the simplex method.

Table 5 illustrates the performance of a single-phase barrier method in which a composite objective function of the form $\omega c^{T}x + \xi$ was used throughout (see Section 4.2). The number of phase 1 iterations is sometimes greater than that for $\omega = 0$ (cf. Table 4), but the total number of iterations is generally less.

In all cases, the number of iterations required by the barrier algorithm appears to be qualitatively similar to that reported for various implementations of the projective method (cf. Tomlin, 1985; and Lustig, 1985).

Table 5

Barrier method with composite objective function
With scaling, RGFAC = 0.1, MUFAC = 0.1, $\omega = 0.1/\|c\|$

Problem	Phase 1	Total	Time	Objective	$\frac{\|b-Ax\|}{\|b\|}$	$\frac{\|D(c-A^{T}\pi)\|}{\|x\|\,\|\pi\|}$
Afiro	4	19	0.4	−464.75314	10^{-11}	10^{-8}
ADLittle	20	26	1.0	225494.96	10^{-8}	10^{-9}
Share2b	8	23	1.4	−415.73224	10^{-8}	10^{-9}
Share1b	9	35	2.9	−76589.319	10^{-6}	10^{-11}
Beaconfd	27	29	8.1	33592.486	10^{-5}	10^{-10}
Israel	10	41	15.9	−896644.82	10^{-6}	10^{-10}
BrandY	25	28	6.4	1518.5099	10^{-5}	10^{-11}
E226	23	37	8.5	−18.751929	10^{-6}	10^{-10}
BandM	26	33	7.9	−158.62802	10^{-6}	10^{-10}
SCSD6	1	20	5.7	50.500000	10^{-8}	10^{-8}
NZFRI	39	41	51.4	−27892262.	10^{-5}	10^{-12}

Some statistics concerning the matrix factorizations used in MINOS 5.0 and the barrier method are provided in Table 6. As in Table 1, column "A" gives the number of nonzeros in the problem. The columns "B" and "L+U" give the number of nonzeros in the simplex basis and its LU factors after the last refactorization, which typically produces the most dense factors. Finally, the column "R" contains the number of nonzeros in the Cholesky factorization (4.5) required by the barrier method.

5.2. *Performance on a degenerate test set*

Table 7 gives statistics for a graduated set of three models from a single application. The models are notable for their severe degeneracy.

Table 6

Factorization statistics

Problem	A	B	L+U	R
Afiro	102	67	67	80
ADLittle	424	261	275	355
Share2b	777	564	597	925
Share1b	1179	579	636	1345
Beaconfd	3408	1546	1546	2727
Israel	2443	1644	1664	3533[a]
BrandY	2202	1318	1485	3251
E226	2768	1440	1620	3416
BandM	2494	2016	2372	4355
SCSD6	4316	536	581	2398
NZFRI	12840	2290	2400	18029

[a] 11259 if six dense columns are included.

Table 7
Model statistics—degenerate problem set

Problem	Rows	Slacks	Columns	A
Degen1	66	15	72	296
Degen2	444	223	534	4894
Degen3	1503	786	1818	25432

These problems were solved using an early version of the barrier algorithm with a more relaxed termination criterion (i.e., the final iterate was an "approximate" solution of the linear program). Specifically, the barrier algorithm terminated when $\max_j |x_j \eta_j| \leq 10^{-7} \|c\| \|x\|$, i.e., when complementarity was approximately achieved. Problems Degen1 and Degen2 were solved on an IBM 3033N, and Degen3 was run on an IBM 3081K.

The primal simplex code used was Ketron's WHIZARD optimizer, which was called from MPSIII in all cases except for Degen3, where the host was MPSX/370. The first two columns of Table 8 give the number of simplex iterations required to reach optimality, and the CPU time in seconds.

The next three columns of Table 8 give the results for the barrier algorithm. Because WHIZARD is written in assembly language, a factor γ is included in the times for the barrier method to represent the comparison of Fortran to assembly language. In many applications, it is accepted that a factor of two in speed (corresponding to $\gamma = \frac{1}{2}$) can be gained by programming in assembly language (see, e.g., Bentley, 1982). However, for inner loop optimization of dense floating-point scalar products, the IBM Fortran compiler with the highest level of optimization (used in all the runs reported in this paper) produces such efficient code that little scope remains for improvement by using assembly language, i.e., $\gamma \approx 1$ (Moler, 1985). Because of this uncertainty, no absolute conclusions about the speeds of the simplex and barrier methods can be drawn from Table 8.

However, one trend clearly emerges from the final column of Table 8, which contains the ratio of the time required by the barrier algorithm to the time required by the simplex method. For the degenerate problems, this ratio *increases* with problem size, so that the barrier algorithm becomes relatively less efficient for the larger problems.

Table 8
Degenerate problem set

	Simplex		Barrier			Ratio
	Iterations	Time	Phase 1	Total	Time	(B/S)
Degen1	23	0.7	2	15	0.9γ	1.28γ
Degen2	2650	31.2	13	26	54.9γ	1.76γ
Degen3	8889	226.0	11	25	528.0γ	2.34γ

A reason for this trend is suggested by the statistics in Table 9. The first three columns give the number of non-zeros in a typical basis B, the number of non-zeros in the LU factorization, and their ratio; the last three columns give the number of non-zeros in the original matrix A, the number of non-zeros in the Cholesky factor R of AD^2A^T, and their ratio.

Comparing columns 3 and 6, we see that the relative increase in density of the LU factorization is approximately constant as problem size increases, while the relative density of the Cholesky factor *increases* with problem size. The resulting increased cost of solving the least-squares subproblem in the barrier method provides some explanation for the trend noted in Table 8. However, it is obvious that for certain structures in A, the relative increase in density of the Cholesky factor will remain constant as problem size increases, and that the performance of the barrier algorithm will consequently *improve* relative to the simplex method on such problems as size increases.

An important feature that does not appear in the tables is the substantial time needed for the single execution of the symbolic ordering subroutine GENQMD: 13.4 seconds for Degen2 (24% of the total) and 237 seconds for Degen3 (43%). Clearly, a more efficient means of preprocessing must be found for large problems. (An improved procedure has recently been given by Liu, 1985.)

5.3. *Early termination of the barrier algorithm*

Because of relations (2.4) and (2.5), a "nearly optimal" solution can be obtained by early termination of the barrier algorithm. In contrast, it is well known that early termination of the simplex method does not necessarily produce a "good approximation" to the optimal solution. (This observation emphasizes the fundamental difference in the iterative sequences generated by a combinatorial algorithm like the simplex method and a nonlinear algorithm like the barrier method.)

Table 10 gives the numbers of iterations when the barrier algorithm of Table 4 was terminated "early"—with two figures, three figures and "optimal" accuracy (approximately six correct digits) in the objective function. The results indicate that almost one-half the work of the barrier algorithm can be saved by terminating early, *if an inaccurate solution is acceptable.* As many authors have noted, this suggests the possibility of using a barrier algorithm to identify the correct active set, and then switching to the simplex method (say) to obtain the final solution.

Table 9

Factorization statistics—degenerate problem set

Problem	B	L+U	(L+U)/B	A	R	R/A
Degen1	249	251	1.0	296	514	1.7
Degen2	3076	3718	1.2	4894	16243	3.3
Degen3	18468	20322	1.1	25432	119373	4.7

Table 10
Early termination of barrier algorithm

Problem	Two digits	Three digits	"Optimal" accuracy
Afiro	10	12	18
ADLittle	22	24	32
Share2b	13	15	23
Share1b	19	23	33
Beaconfd	32	37	44
Israel	29	34	48
BrandY	27	35	42
E226	29	34	43
BandM	31	35	42
SCSD6	13	15	21
NZFRI	33	39	53

5.4. *Obtaining an optimal basic solution*

By its very nature, a barrier method can at best terminate somewhere "close" to an optimum. We must then ask: how close is "close", and which of the several characterizations of LP optimality are we close to achieving?

In practice, LP users (and their report-writing programs) expect alleged optimal solutions to be both primal and dual feasible, thus exhibiting complementary slackness. The last columns in Tables 3–5 show that the barrier algorithm can attain complementary slackness to high precision. However, LP users also expect their solutions to be basic. A basic solution can be achieved by taking the final solution from the barrier algorithm and processing it through the BASIC procedure common to most mathematical programming systems.

The BASIC procedure (sometimes known as INSERT-by-value; see Benichou et al., 1977) takes a set of variable names and values and produces a basic solution that has at least as good an objective value or sum of infeasibilities. The simplex method may then be applied to reach optimality. The time required by the BASIC procedure and the post-BASIC simplex iterations provides a practical measure of closeness to optimality of the barrier solution.

Some experiments of this kind were performed on all of the test problems, using the near-optimal solutions obtained by the barrier algorithm and treating components less than 10^{-6} as zero. With $\hat{x}$ denoting the basic solution obtained by BASIC, the quantities $\|b - A\hat{x}\|/\|b\|$ were less than 10^{-5} in all cases, and the values of $|c^{\mathrm{T}}\hat{x} - c^{\mathrm{T}}x^*|/|c^{\mathrm{T}}x^*|$ were all less than 10^{-3}. Clearly, the primary effect of the post-BASIC simplex iterations is to remove dual infeasibilities.

The number of post-BASIC simplex iterations appears to be a function of size and degeneracy. For the test set in Table 1, only a small number of post-BASIC simplex iterations were required to reach optimality: at most 6 for the first nine problems, 61 for SCSD6, and 37 for NZFRI. (Note that PRESOLVE was applied to NZFRI prior to BASIC; see Section 5.5.)

For Degen2 and Degen3, the post-BASIC simplex iterations comprised 11% and 7% of the simplex iterations required when starting from scratch. Thus, the relative number of post-BASIC simplex iterations required for these problems appears to decline with problem size, compared to the number required by WHIZARD to solve the problems from scratch. The total *time* required by BASIC and post-BASIC simplex iterations was about 25% of the total time required when starting from scratch.

It would be of interest to perform further experiments with BASIC, starting from the "early termination" points referred to in Table 10. The results already obtained suggest that the combined barrier/BASIC/simplex approach would often be more effective than either the barrier or simplex algorithms alone.

5.5. Null variables

In practice, linear programs are often *structurally* degenerate, in the sense that certain variables (called null variables) must be zero in any feasible solution. For example, the problems BrandY, E226, BandM and NZFRI have 23, 20, 21 and 1477 null variables respectively, as determined by the PRESOLVE procedure in WHIZARD. Ideally, such variables should be removed before a solution procedure is called. They can then be restored and dual feasibility attained by the POSTSOLVE procedure (see Tomlin and Welch, 1983).

As an example, when WHIZARD was applied to the problem NZFRI, only 552 simplex iterations were required to solve the reduced problem—a substantial improvement over the results in Table 2. (The total time, including PRESOLVE and POSTSOLVE, was 9.5 seconds on an IBM 3081K.)

In our experience, failure to remove large numbers of null variables usually results in many iterations by the simplex method, but not for our particular implementation of the barrier algorithm. This is another area (as with scaling) where the barrier approach appears to be more robust than the simplex method.

6. Future developments and conclusions

6.1. Solving the least-squares subproblem

The present implementation, as in Gay (1985), uses a preconditioned conjugate-gradient method to solve the relevant least-squares subproblems. This approach allows the use of existing software for computing Cholesky factors, and provides a convenient way of dealing with a few dense columns of A that would degrade the sparsity of those factors. Perhaps further efficiency could be gained by discarding small nonzeros in the product AD^2A^T (not just rows and columns of A) before computing its Cholesky factors. However, a new symbolic factorization would then be required at every stage, not just once.

Our experiments indicate that the preconditioner must be of high quality throughout in order to retain efficiency in the conjugate-gradient method. An iteration of LSQR or the conjugate-gradient method requires two matrix-vector products involving A and two solves with the preconditioner R, and is therefore as expensive (typically) as two iterations of the simplex method. To see the relevant tradeoffs, assume that R could be obtained with minimal effort, but that LSQR required an average of 20 iterations to converge; then the barrier method would be similar in speed to the simplex method if it terminated in about $\frac{1}{40}$th the number of iterations.

The effect on speed of excluding dense columns from the preconditioner can be seen in the test problem Israel, which has six dense columns in A. With these excluded from the computation of R, LSQR required an average of 10 iterations to converge, indicating inaccuracy in the preconditioner. On the other hand, retaining all columns decreased the number of LSQR iterations, but produced an R with three times as many nonzeros, and hence doubled the execution time.

For reasons such as these, much research is needed concerning the computation of good preconditioners for arbitrary sparse matrices DA^T, i.e., for arbitrary linear programs. (For some suggested approaches based on the LU factorization, see Gill et al., 1986a.) There is reason to be optimistic for certain problems—for example, those exhibiting a block-triangular structure with many small diagonal blocks.

In place of the iterative methods just described, one can employ a sparse orthogonal factorization of the form

$$DA^T = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad Q^TQ = I, \qquad R \text{ upper triangular} \tag{6.1}$$

to solve the least-squares problems directly, where R is analytically the same as the Cholesky factor of AD^2A^T. General-purpose software exists for this computation, in particular SPARSPAK-B (George and Ng, 1984), which has excellent numerical properties and is able to treat dense rows of DA^T specially in order to preserve the sparsity of R. Its use in this context merits future investigation.

A further direct approach is to apply a sparse indefinite solver to the symmetric system (2.11). The MA27 package of Duff and Reid (1982, 1983) is applicable, and as with the sparse QR (6.1), a single symbolic factorization serves all iterations. The dense artificial column can be excluded from the factorization and treated by partitioning. Unfortunately, on the problems of Table 1, the symbolic factors have proved to be twice as dense as R in (4.5), and the severe indefiniteness of (2.11) leads to numerical factors that are 3 to 10 times as dense as R. Hence, our initial experience with this approach has been unpromising.

6.2. Adjusting the barrier parameter

Numerous authors have suggested extrapolation techniques in connection with barrier functions (see, e.g., Fiacco and McCormick, 1968; Fletcher and McCann, 1969). (Note that an extrapolation strategy would need to be applied to both phases.)

In a small number of our experiments, extrapolation was performed after a reasonably accurate minimization of the barrier function for two quite large values of $\mu(\|c\|/n$ and $0.1\|c\|/n)$. The resulting solutions were typically accurate to about five figures. However, it is difficult to evaluate the practical merits of this approach without further study.

A number of suggestions have been made for automating the choice of μ. The method of centers (see Fiacco and McCormick, 1968; Huard, 1967) is in essence a barrier-function method in which a transformation is also applied to the objective function. Although an explicit barrier parameter is thereby avoided, another parameter must be chosen in order for the procedure to be effective. See Todd and Burrell (1985) for further discussion of this approach.

In our view, the freedom to choose μ may well be an asset, especially in solving linear programs. This is confirmed by our experience with the conservative strategy of allowing μ to be reduced only occasionally. Considerable progress is then often achieved before the least-squares problems become unduly ill-conditioned.

6.3. *Use of the entropy function*

Because of the similarities, we note the work of many authors on incorporating the entropy function into linear programming models. In place of subproblem (2.3), one can consider the subproblem

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad c^{\mathrm{T}}x+\mu\sum_{j=1}^{n} x_j \ln x_j \\ &\text{subject to} \quad Ax=b, \end{aligned}$$

where the scalar μ ($\mu>0$) is again specified for each subproblem. Erlander (1977) reviews problems of this kind and suggests Newton-type methods for their solution. Computational algorithms have been developed by Eriksson (1980, 1981, 1985).

If a feasible-point descent method is applied as in Section 2, the Newton search direction and Lagrange-multiplier estimates satisfy the system

$$\begin{pmatrix} \mu D^{-1} & A^{\mathrm{T}} \\ A & 0 \end{pmatrix}\begin{pmatrix} -p \\ \pi \end{pmatrix}=\begin{pmatrix} c+\mu v \\ 0 \end{pmatrix}$$

in place of (2.12), where $D=\operatorname{diag}(x_j)$ and v has components $v_j=1+\ln x_j$. A least-squares subproblem follows as before. In the algorithm of Section 4.1, r becomes $D^{1/2}(\eta+\mu v)$ in steps 1, 3 and 5, D becomes $D^{1/2}$ in the least-squares problem (4.1), and $p=-(1/\mu)D^{1/2}r$ in step 5.

The entropy function is convex and (unlike the logarithmic barrier function) bounded below. Since its Hessian is μD^{-1} rather than μD^{-2}, the least-squares problems are better conditioned as the LP solution is approached. Further computational work therefore seems to be justified, either as in Eriksson (1980, 1981, 1985) or along the lines suggested here.

6.4. *Conclusions*

Our experience with the barrier method suggests several conclusions. On the

positive side:

- A significant body of computational evidence indicates that for general non-trivial linear programs, a general barrier method can be comparable in speed to the simplex method;
- In some cases the barrier method will be faster than the simplex method (even substantially so), and its advantage will increase on problems in which the least-squares subproblems can be solved rapidly. Furthermore, since we deliberately used the same parameters on all test problems, there is much scope for "tuning" the algorithm on particular problem classes;
- The mathematical and qualitative relationship between the projective and barrier methods places this approach to linear programming in a well understood context of nonlinear programming, and provides an armory of known theoretical and practical techniques useful in convergence analysis and implementation.

On the negative side:

- The barrier method has not been consistently faster than the simplex method on general unstructured problems, and has been considerably slower on certain examples. Furthermore, its efficiency relative to the simplex method may decrease with size on problems for which the density of the Cholesky factor increases more rapidly than that of the LU factorization;
- "Nonlinearizing" a linear problem makes the development of a robust general-purpose algorithm more difficult. For example, extreme nonlinearity near the boundary of the feasible region can lead to poor performance if variables migrate prematurely toward their bounds, or if a "good" starting point is available from an earlier run on a similar problem. Furthermore, nonlinear algorithms typically display wide variations in performance, depending on the selection of various parameters.

Finally, we make the following general observations:

- Most qualitative aspects of Karmarkar's projective method can be found in the projected Newton barrier method;
- No attempt has been made to date to obtain a proof of polynomial complexity for any version of the barrier algorithm;
- The efficiency of the projective and barrier methods depends critically on fast, stable techniques for solving large-scale least-squares problems. Modern sparse-matrix technology is absolutely crucial in any further development of these methods for large-scale linear programming;
- There is much promise for interior-point approaches to linear programming, particularly for specially-structured problems.

Acknowledgements

The authors are grateful to George B. Dantzig for his encouragement and bibliographical assistance, and to Irvin Lustig for his helpful and timely suggestions. We also thank the referees for their comments.

References

M. Benichou, J.M. Gauthier, G. Hentges and G. Ribière, "The efficient solution of large-scale linear programming problems—some algorithmic techniques and computational results," *Mathematical Programming* 13 (1977) 280–322.

J.L. Bentley, *Writing Efficient Programs* (Prentice-Hall, Englewood Cliffs, NJ, 1982).

R.P. Brent, *Algorithms for Minimization without Derivatives* (Prentice-Hall, Englewood Cliffs, NJ, 1973).

G.B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ, 1963).

J.J. Dongarra and E. Grosse, "Distribution of mathematical software via electronic mail," *SIGNUM Newsletter* 20 (1985) 45–47.

I.S. Duff and J.K. Reid, "MA27—a set of Fortran subroutines for solving sparse symmetric sets of linear equations," Report AERE R-10533, Computer Science and Systems Division, AERE Harwell (Harwell, England, 1982).

I.S. Duff and J.K. Reid, "The multifrontal solution of indefinite sparse symmetric linear equations," *ACM Transactions on Mathematical Software* 9 (1983) 302–325.

S.C. Eisenstat, M.C. Gursky, M.H. Schultz and A.H. Sherman, "Yale sparse matrix package I: The symmetric codes," *International Journal of Numerical Methods in Engineering* 18 (1982) 1145–1151.

J. Eriksson, "A note on solution of large sparse maximum entropy problems with linear equality constraints," *Mathematical Programming* 18 (1980) 146–154.

J. Eriksson, "Algorithms for entropy and mathematical programming," Ph.D. Thesis, Linköping University (Linköping, Sweden, 1981).

J. Eriksson, "An iterative primal-dual algorithm for linear programming," Report LiTH-MAT-R-1985-10, Department of Mathematics, Linköping University (Linköping, Sweden, 1985).

S. Erlander, "Entropy in linear programs—an approach to planning," Report LiTH-MAT-R-77-3, Department of Mathematics, Linköping University (Linköping, Sweden, 1977).

A.V. Fiacco, "Barrier methods for nonlinear programming," in: A. Holzman, ed., *Operations Research Support Methodology* (Marcel Dekker, New York, NY, 1979) pp. 377–440.

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

R. Fletcher, *Practical Methods of Optimization, Volume 2* (John Wiley and Sons, Chichester, 1981).

R. Fletcher and A.P. McCann, "Acceleration techniques for nonlinear programming," in: R. Fletcher, ed., *Optimization* (Academic Press, London, 1969) pp. 203–213.

R. Fourer, "Solving staircase linear programs by the simplex method, 1: Inversion," *Mathematical Programming* 23 (1982) 274–313.

K.R. Frisch, "The logarithmic potential method of convex programming," University Institute of Economics (Oslo, Norway, 1955).

K.R. Frisch, "Linear dependencies and a mechanized form of the multiplex method for linear programming," University Institute of Economics (Oslo, Norway, 1957).

O. Garcia, "FOLPI, a forestry-oriented linear programming interpreter," Reprint 1728, New Zealand Forest Service (Christchurch, New Zealand, 1984).

D.M. Gay, "Solving sparse least-squares problems," Presentation, Department of Operations Research, Stanford University (Stanford, CA, 1985).

J.A. George and J.W.H. Liu, *Computer Solution of Large Sparse Positive Definite Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1981).

J.A. George and E. Ng, "A new release of SPARSPAK—the Waterloo sparse matrix package," *SIGNUM Newsletter* 19 (1984) 9–13.

P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, "Sparse matrix methods in optimization," *SIAM Journal on Scientific and Statistical Computing* 5 (1984) 562–589.

P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, "A note on nonlinear approaches to linear programming," Report SOL 86-7, Department of Operations Research, Stanford University (Stanford, CA, 1986a).

P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, "Maintaining LU factors of a general sparse matrix," Report SOL 86-8, Department of Operations Research, Stanford University (Stanford, CA, 1986b). [To appear in *Linear Algebra and its Applications.*]

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

M.T. Heath, "Numerical methods for large sparse linear least squares problems," *SIAM Journal on Scientific and Statistical Computing* 5 (1984) 497-513.

J.K. Ho and E. Loute, "A set of staircase linear programming test problems," *Mathematical Programming* 20 (1981) 245-250.

A.J. Hoffman, M. Mannos, D. Sokolowsky, and N. Wiegmann, "Computational experience in solving linear programs," *Journal of the Society for Industrial and Applied Mathematics* 1 (1953) 17-33.

P. Huard, "Resolution of mathematical programming with nonlinear constraints by the method of centres," in: J. Abadie, ed., *Nonlinear Programming* (North-Holland, Amsterdam, 1967) pp. 207-219.

K. Jittorntrum, *Sequential Algorithms in Nonlinear Programming*, Ph.D. Thesis, Australian National University (Canberra, Australia, 1978).

K. Jittorntrum and M.R. Osborne, "Trajectory analysis and extrapolation in barrier function methods," *Journal of Australian Mathematical Society Series B* 20 (1978) 352-369.

N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Proceedings of the 16th Annual ACM Symposium on the Theory of Computing* (1984a) 302-311.

N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica* 4 (1984b) 373-395.

L.G. Khachiyan, "A polynomial algorithm in linear programming," *Doklady Akademiia Nauk SSSR Novaia Seriia* 244 (1979) 1093-1096. [English translation in *Soviet Mathematics Doklady* 20 (1979) 191-194.]

J.W.H. Liu, "Modification of the minimum-degree algorithm by multiple elimination," *ACM Transactions on Mathematical Software* 11 (1985) 141-153.

I.J. Lustig, "A practical approach to Karmarkar's algorithm," Report SOL 85-5, Department of Operations Research, Stanford University (Stanford, CA, 1985).

R. Mifflin, "On the convergence of the logarithmic barrier function method," in: F. Lootsma, ed., *Numerical Methods for Non-Linear Optimization* (Academic Press, London, 1972) pp. 367-369.

R. Mifflin, "Convergence bounds for nonlinear programming algorithms," *Mathematical Programming* 8 (1975) 251-271.

C.B. Moler, Private communication (1985).

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

B.A. Murtagh and M.A. Saunders, "MINOS 5.0 user's guide," Report SOL 83-20, Department of Operations Research, Stanford University (Stanford, CA, 1983).

J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables* (Academic Press, New York, NY, 1970).

C.C. Paige and M.A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least-squares," *ACM Transactions on Mathematical Software* 8 (1982a) 43-71.

C.C. Paige and M.A. Saunders, "Algorithm 583. LSQR: Sparse linear equations and least squares problems," *ACM Transactions on Mathematical Software* 8 (1982b) 195-209.

M.J. Todd and B.P. Burrell, "An extension of Karmarkar's algorithm for linear programming using dual variables," Report 648, School of Operations Research and Industrial Engineering, Cornell University (Ithaca, NY, 1985).

J.A. Tomlin, "An experimental approach to Karmarkar's projective method for linear programming," Manuscript, Ketron Inc. (Mountain View, CA, 1985). [To appear in *Mathematical Programming Studies.*]

J.A. Tomlin and J.S. Welch, "Formal optimization of some reduced linear programming problems," *Mathematical Programming* 27 (1983) 232-240.

C.B. Tompkins, "Projection methods in calculation," in: H.A. Antosiewicz, ed., *Proceedings of the Second Symposium in Linear Programming* (United States Air Force, Washington, DC, 1955) pp. 425-448.

C.B. Tompkins, "Some methods of computational attack on programming problems, other than the simplex method," *Naval Research Logistics Quarterly* 4 (1957) 95-96.

R.J. Vanderbei, M.S. Meketon and B.A. Freedman, "A modification of Karmarkar's linear programming algorithm," Manuscript, AT&T Bell Laboratories (Holmdel, NJ, 1985).

J. von Neumann, "On a maximization problem," Manuscript, Institute for Advanced Study (Princeton, NJ, 1947).