A Primal Null-Space Affine-Scaling Method

K. KIM
Kei-Myung University
and
J. L. NAZARETH
Washington State University

This article develops an affine-scaling method for linear programming in standard primal form. Its descent search directions are formulated in terms of the null-space of the linear programming matrix, which, in turn, is defined by a suitable basis matrix. We describe some basic properties of the method and an experimental implementation that employs a periodic basis change strategy in conjunction with inexact computation of the search direction by an iterative method, specifically, the conjugate-gradient method with diagonal preconditioning. The results of a numerical study on a number of nontrivial problems representative of problems that arise in practice are reported and discussed.

A key advantage of the primal null-space affine-scaling method is its compatibility with the primal simplex method. This is considered in the concluding section, along with implications for the development of a more practical implementation.

Categories and Subject Descriptors: G.1.6 [**Numerical Analysis**]: Optimization—*linear programming*

General Terms: Algorithms

Additional Key Words and Phrases: Conjugate gradients, diagonal preconditioning, interior-point algorithm, null-space affine scaling, primal method

1. INTRODUCTION

Since the development of the simplex method by Dantzig in 1947, research in linear programming has been very intensive in developing mathematical theory, new algorithms, and efficient and practical implementations. In 1984, Karmarkar discovered a new method for linear programming called the projective-scaling method. In contrast to the vertex-following simplex method, which relies on the combinatorial structure of a linear program, the projective method moves through the interior of the feasible polytope. Although cur-

This research was supported in part by National Science Foundation grant DMS-8815513.
Authors' addresses: K. Kim, Department of Mathematics, Kei-Myung University, Daegu, Republic of Korea, 704-701; J. L. Nazareth, Department of Pure and Applied Mathematics, Washington State University, Pullman, WA 99164-3113; email: nazareth@amath.washington.edu.

rently less evolved than the simplex method, it has the theoretical advantage of polynomial-time convergence.

Many offshoots of Karmarkar's method have been proposed and studied by researchers since the original discovery. Three main categories are projective-scaling methods, affine-scaling methods, and path-following methods. Affine scaling was originally proposed by Dikin [1967] and rediscovered by several researchers, and the path-following method was originated by Megiddo [1989] in a pioneering paper using a logarithmic barrier approach.

This article centers on the *affine-scaling method*. We define $T(x) = y + Qx$ to be an *affine transformation* if both $y \in R^n$ and Q is an $n \times n$ nonsingular matrix. In an affine-scaling method, the variables are scaled by an $n \times n$ nonsingular diagonal matrix through the affine transformation. This method is further classified into two main categories: *range-space* and *null-space* affine-scaling methods. Barnes [1986] and Vanderbei et al. [1986] proposed a range-space version of the affine-scaling method in *primal* form. Adler et al. [1986] proposed and studied an affine-scaling method in *dual* form, which can be shown to be of the null-space[1] type (see Nazareth [1987b]). Under nondegeneracy assumptions, Barnes and Vanderbei et al. established convergence results independently, and the viability of their methods was tested on small dense problems. For the dual affine-scaling method, Adler et al. [1986] and Mehrotra [1989] showed encouraging computational results on large sparse practical problems.

Nazareth [1987b] proposed a null-space version of the affine-scaling method in *primal* form and tested its viability on small dense problems. (In the rest of our discussion we shall call this method "the null-space affine-scaling method.") Its descent direction is defined in the null-space of the matrix A. This method possesses several distinguishing properties and certain advantages over the other affine-scaling methods. One significant advantage of the null-space version over the range-space version of the affine-scaling method is that the descent direction can be approximated by an iterative method, i.e., it does not have to be computed exactly.

The merits of using the primal form versus the dual form, Adler et al. [1986], also require consideration. The primal null-space affine-scaling method is applied directly to the standard primal form of linear program, whereas the dual affine method requires that problems be reformulated (implicitly) into the dual form. Most practical problems are formulated and solved on the primal form. Frequently, the solution sought needs not be an optimal solution but rather a "good" approximate solution to the problem. Primal algorithms are certainly to be preferred in this situation. In contrast, dual algorithms must accurately find a dual optimal solution in order for its dual to be a primal feasible optimal solution. An advantage of the dual affine-scaling method over the primal is that an initial starting point is more easily obtained, and the method seems to be less sensitive to the choice of starting

[1]Note that the range-space version of the affine-scaling method on the standard *dual* form does not exist; see Nazareth [1993].

point, in contrast to the primal affine-scaling method, as will be discussed in a later section. In the dual affine-scaling algorithm, a fixed-basis matrix corresponding to the identity matrix is used, making it possible to use direct factorization methods to obtain the search direction. The primal null-space affine-scaling algorithm is more reliant on iterative methods (when the basis is varied). Finally, recall that the primal and dual simplex algorithms are *not* really competitors because they complement one another, i.e., both are needed in the mathematical programming repertoire, and the same holds true for primal and dual affine-scaling algorithms.

The purposes of this article are to describe some basic properties of the null-space affine-scaling method and to examine its applicability to practical problems. As mentioned above, the method works directly on the standard primal form of a linear program and computes the descent search direction inexactly using a conjugate-gradient method with diagonal preconditioning. Additionally, a "basis change strategy" is employed to form a new basis (like a basis in the simplex method [Dantzig 1963]) periodically. One of our main objectives here is to study the practicality of this combination of very basic strategies. In these level-2 experiments (see Nazareth [1985] for terminology), the experimental implementation is tested on a number of nontrivial publicly available test problems; see Gay [1986]; and our results are discussed in detail.

A key advantage of the null-space affine-scaling method is its compatibility with the primal simplex method. This is discussed in general terms in the concluding section within the context of implications of our results for the development of a more practical (level-3) implementation.

This article is organized as follows. We review briefly the null-space affine-scaling method in Section 2 and give some important properties of this method in Section 3. In Section 4, the null-space affine-scaling algorithm and the conjugate-gradient algorithm are given. Section 5 describes details of the experimental implementation. Computational results of the null-space affine-scaling method on a number of practical problems are presented in Section 6. Compatibility with simplex techniques and conclusions are discussed in Section 7.

2. THE NULL-SPACE AFFINE-SCALING METHOD

We begin by reviewing the null-space affine-scaling method given in Nazareth [1987b]. Consider the following linear program (LP) in standard form:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{1}$$

where c and x are n-dimensional vectors; b is an m-dimensional vector; and A is an $m \times n$ matrix of full row rank with $m < n$. We define the set $\{x > 0 : Ax = b\}$ to be the *interior* of the set $\{x \geq 0 : Ax = b\}$ relative to the affine space $\{x : Ax = b\}$. We shall assume that the linear program (1) is bounded with a nonempty interior.

Let $x^\circ > 0$ be any interior point such that $Ax^\circ = b$. We define $z^\circ \equiv c^T x^\circ$ and $\bar{n} \equiv n - m$. The linear program (1) can be rewritten as

$$\begin{aligned} \text{minimize} \quad & c^T(x - x^\circ) + z^\circ \\ \text{s.t.} \quad & A(x - x^\circ) = 0 \\ & x \geq 0. \end{aligned} \tag{2}$$

One approach to defining a suitable local (reduced) model, when dropping inactive bounds at x°, is to augment the objective function by a quadratic regularizing term with a metric that defines a measure of distance relative to x°. By employing this, we obtain

$$\begin{aligned} \text{minimize} \quad & c^T(x - x^\circ) + \tfrac{1}{2}(x - x^\circ)^T D_0^{-2}(x - x^\circ) + z^\circ \\ \text{s.t.} \quad & A(x - x^\circ) = 0, \end{aligned} \tag{3}$$

where $D_0 \equiv \text{diag}[x_1^\circ, \ldots, x_n^\circ] > 0$. An alternative way to characterize (3) is to say that it defines a suitable local quadratic approximating model. Note that it has only *equality* constraints and can therefore be solved explicitly.

The solution of the equality-constrained quadratic problem (3) is given by a Karush-Kuhn-Tucker (KKT)-type system, namely,

$$\begin{bmatrix} -D_0^{-2} & A^T \\ A & \end{bmatrix} \begin{bmatrix} x - x^\circ \\ \pi \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix}, \tag{4}$$

and this KKT-type system can be solved by various means. The null-space approach discussed in this article is one such means.

Let Z be an $n \times \bar{n}$ matrix of *full* column rank for which $AZ = 0$, i.e., Z spans the null space of A, and make a transformation of variables

$$x = x^\circ + Z\Delta x_S, \tag{5}$$

where $\Delta x_S \in R^{n-m}$.

By making the transformation of variables given by (5) in (3), we obtain the following *quadratic local reduced model*

$$\text{minimize}_{\Delta x_S \in R^{n-m}} (Z^T c)^T \Delta x_S + \tfrac{1}{2}\Delta x_S^T Z^T D_0^{-2} Z \Delta x_S + z^\circ, \tag{6}$$

which is an unconstrained quadratic problem. Note that the dimension of the variable Δx_S in this reduced model is $(n - m) \times 1$. The *minimizing* point is given by

$$(Z^T D_0^{-2} Z)\Delta x_S = -Z^T c, \tag{7}$$

and the associated *descent* direction in the original space is

$$\Delta x = Z\Delta x_S = -Z(Z^T D_0^{-2} Z)^{-1} Z^T c. \tag{8}$$

When a feasible step is taken in the direction given by (8), a better solution is determined. Basically this constitutes one cycle of the null-space affine-scaling method.

It is easily shown that the direction Δx defined by (8) can be *equivalently* expressed in *range-space* form, namely,

$$\Delta x = -D_0\left[I - D_0 A^T (AD_0^2 A^T)^{-1} AD_0\right] D_0 c. \tag{9}$$

This is the form of the affine-scaling direction proposed by Vanderbei et al. [1986].

Note that when $(Z^T D_0^{-2} Z)$ in (8) is replaced by the identity matrix, one obtains the simpler so-called Wolfe reduced-gradient direction. A related reduced-gradient form of Karmarkar's projective scaling direction[2] is the primary focus of a computational study of Shanno and Marsten [1988].[3]

For a broader discussion of the foregoing affine-scaling approaches, see Nazareth [1987b].

3. PROPERTIES OF THE METHOD

In this section we present some important properties of the null-space affine-scaling method. Properties given in Sections 3.2, 3.3, and 3.4 are implicit in Nazareth [1987b]; but our discussion is more systematic, and in particular, it highlights the basis change strategy (discussed in more detail in Section 5.4). Proposition 3.1.1 and the proof of Proposition 3.5.1 are new, and these two propositions hold the key to effective implementation of the null-space affine-scaling method.

3.1 Finite Bound for Δx

As pointed out by Stewart [1988], it is possible to choose D_0^{-2} with $\|D_0^{-2}\| = 1$ such that the factor $(Z^T D_0^{-2} Z)^{-1}$ in (8) is arbitrarily large. However, he has shown that if X is a matrix of full column rank and if D is any diagonal matrix with positive elements, then $\|P_D\|$ and $\|X_D^\dagger\|$ are bounded independently of D, where $X_D^\dagger = (X^T DX)^{-1} X^T D$ and $P_D = XX_D^\dagger = X(X^T DX)^{-1} X^T D$. Here $\|\cdot\|$ is the Euclidean vector norm or the spectral matrix norm. Using these results, we show that Δx in (8) is well behaved if the linear program (1) is bounded.

PROPOSITION 3.1.1. *Assume that the linear program* (1) *is bounded. Then* Δx *in* (8) *is bounded independently of* D_0^{-2}.

PROOF. We have

$$\begin{aligned}\Delta x &= -Z(Z^T D_0^{-2} Z)^{-1} Z^T c \\ &= -Z(Z^T D_0^{-2} Z)^{-1} Z^T D_0^{-2} D_0^2 c.\end{aligned}$$

Since the problem is bounded, $\|D_0^2 c\|$ is bounded by a constant independently of D_0. To show Δx is bounded, it suffices to show that the matrix

[2]This uses Karmarkar's special canonical form of the linear program, and it is not necessarily a direction of descent with respect to c.

[3]The results obtained were not encouraging. A null-space Karmarkar-type projective-scaling algorithm, which uses a "projected-gradient" direction more like (8), was also studied on a single example, almost as an afterthought to the main computational investigation. This seemed to offer more hope, but was not pursued further.

$Z(Z^T D_0^{-2} Z)^{-1} Z^T D_0^{-2}$ is bounded. By construction, Z has full column rank, and D_0^{-2} is a diagonal matrix with positive elements. Hence, by the Stewart result quoted above, $Z(Z^T D_0^{-2} Z)^{-1} Z^T D_0^{-2}$ is bounded if we let $X = Z$ and $D = D_0^{-2}$. Thus, Δx is bounded. □

3.2 Search Direction with Any Choice of Basis

Now let Z be defined to be

$$Z \equiv \begin{bmatrix} -B^{-1}S \\ I_{\bar{n}\times\bar{n}} \end{bmatrix}, \tag{10}$$

where $A \equiv [B \mid S]$ is partitioned as in the simplex algorithm into basic and nonbasic columns. The difference is that the starting basis B is defined to be any arbitrary linearly independent set of m columns of A, and since x° is an interior point, each basic and nonbasic variable is at a positive level. Without loss of generality, we can assume that B consists of the first m columns of A.

Since (8) and (9) define the same direction, and since (9) is independent of the basis, it follows that the search direction given by (8) is independent of the choice of basis. The starting basis B can either be fixed throughout the whole operation or changed as often as one wishes to form a new basis. This valuable freedom enabled us to consider a strategy called the "basis change strategy," which will be discussed in detail in Section 5.4.

3.3 Natural Preconditioner

Just like any other interior-point method, the core of this method is: how efficiently can one solve (7)? Substituting (10) in (7), we obtain the more convenient computational form

$$(S^T B^{-T} D_B^{-2} B^{-1} S + D_S^{-2}) \Delta x_S = -Z^T c,$$

where $D_B \equiv \text{diag}[x_1^\circ, \ldots, x_m^\circ] > 0$ and $D_S \equiv \text{diag}[x_{m+1}^\circ, \ldots, x_n^\circ] > 0$. After a transformation of variables defined by

$$\Delta x_S = D_S \Delta \bar{x}_S, \tag{11}$$

the search direction in the reduced model can be obtained by solving the equivalent system

$$[I + M_S] \Delta \bar{x}_S = -\bar{c}, \tag{12}$$

where

$$\bar{c} = D_S Z^T c$$

and

$$M_S = D_S S^T B^{-T} D_B^{-2} B^{-1} S D_S.$$

In the implementation of the null-space affine-scaling algorithm (NSASA) defined in Section 4, the above system of equations (12) will be solved by using the conjugate-gradient method. This system of equations will be a well-conditioned problem if one can maintain the size of the elements of D_B

large and the size of the elements of D_S small. The basis change strategy in Section 5.4 takes this into consideration; a new basis is formed periodically, and one of the criteria used to form a new basis is the size of the variables. The transformation (11) implies that D_S is acting as a diagonal preconditioner for the conjugate-gradient method. We call this the *natural preconditioner*.

3.4 Approximation of Δx

Δx in (8) defines the direction used in the null-space affine-scaling method, and Δx in (9) defines the direction used in the range-space affine-scaling method. Although the two directions are mathematically identical, one significant advantage of the form (8) over the form (9) is that Δx_S in (7) can be approximated without violating feasibility of the next iterate and without incurring the expense of computing Δx_S exactly. This can be seen as follows. If $\Delta \hat{x}_S$ is an approximation in (7), then the corresponding descent search direction in the original space is given by $\Delta x = Z\Delta \hat{x}_S$ and $A\Delta x = 0$ since $AZ = 0$, i.e., the approximated direction vector Δx is in the space generated by the columns of Z that span the null space of A.

On the other hand, one can easily see that Δx defined by (9) cannot be approximated in the foregoing manner, i.e., it must be computed exactly, suggesting use of a direct technique, e.g., Cholesky or QR factorization. The approximation of Δx_S can be done more advantageously by an iterative method, one of the most attractive being the method of conjugate gradients (CG method).

3.5 Descent Property of Δx with Inexact CG Method

When each step Δx_S is approximated by the conjugate-gradient method, one of the primary concerns is whether the corresponding direction in the original space yields a descent direction. This is indeed the case, and we show it in the following proposition.

PROPOSITION 3.5.1. *If Δx_S in (7) is approximated by the CG method and if $\Delta x_S^\circ = 0$ is the starting point for the CG method, then the corresponding Δx is always a strictly descent direction in the original space. Also, Δx can be expressed in the form $\Delta x = -Z\Omega Z^T c$, where Ω is a positive definite matrix.*

PROOF. Let f be the quadratic function to be minimized in (6), i.e., $f(\Delta x_S) = (Z^T c)^T \Delta x_S + (1/2)\Delta x_S^T Z^T D_0^{-2} Z \Delta x_S + z^\circ$. Let Δx_S° be the starting point of the CG method with $\Delta x_S^\circ = 0$. This gives $f(\Delta x_S^\circ) = z^\circ$ and $\nabla f(\Delta x_S^\circ) = Z^T c$. If $\Delta \hat{x}_S$ is an approximation of Δx_S in (7) by the CG method (i.e., CG is terminated after any $k > 0$ iterations), then by its intrinsic property of descent, we always have $f(\Delta \hat{x}_S) < f(\Delta x_S^\circ)$. Since $Z^T D_0^{-2} Z$ is positive definite, $(1/2)\Delta \hat{x}_S^T Z^T D_0^{-2} Z \Delta \hat{x}_S > 0$. This implies that $(Z^T c)^T \Delta \hat{x}_S < 0$ and $c^T Z \Delta \hat{x}_S = c^T \Delta x < 0$. Thus, Δx is a strictly descent direction in the original space.

Now, let $s = \Delta \hat{x}_S$ and $y = -Z^T c$. Then, $y^T s > 0$. We construct a matrix Ω as follows.

$$\Omega = \left(I - \frac{sy^T}{y^T s}\right)\left(I - \frac{ys^T}{y^T s}\right) + \frac{ss^T}{y^T s}.$$

We claim that Ω is a positive-definite matrix. To show this, let $z \neq 0 \in R^n$. Then, $z^T \Omega z = \bar{z}^T \bar{z} + (z^T s)^2 / y^T s$ where $\bar{z} = (I - (ys^T / y^T s))z$. If $z^T s \neq 0$, then $z^T \Omega z > 0$ since $y^T s > 0$. On the other hand, if $z^T s = 0$, then $\bar{z}^T \bar{z} = z^T z > 0$ since $z \neq 0$. Either way, $z^T \Omega z > 0$. Therefore, Ω is a positive-definite matrix. Also, it is obvious that $\Omega y = s$. Since $s = \Delta \hat{x}_S$ and $y = -Z^T c$, we have $\Delta \hat{x}_S = -\Omega Z^T c$. Thus, $\Delta x = Z \Delta \hat{x}_S = -Z \Omega Z^T c$. □

4. THE NULL-SPACE AFFINE-SCALING ALGORITHM

We present the primal null-space affine-scaling algorithm (NSASA) for linear programming and the conjugate-gradient algorithm (CGA) that solves the system of equations to approximate the descent search direction.

Algorithm A1. Null-Space Affine-Scaling Algorithm (NSASA)

Given $x^\circ > 0$ and $0 < \theta < 1$ and
$\mathcal{B} = \{\beta_1, \ldots, \beta_m\}$, $\mathcal{S} = \{\beta_{m+1}, \ldots, \beta_n\}$
corresponding to the index set of basic and nonbasic variables, respectively.
For $k = 0, 1, 2, \ldots$ until a stopping criterion is satisfied **do**
Let $D_B \leftarrow \text{diag}(x_{\beta_1}, \ldots, x_{\beta_m})$, $D_S \leftarrow \text{diag}(x_{\beta_{m+1}}, \ldots, x_{\beta_n})$
Solve $[I + D_S S^T B^{-T} D_B^{-2} B^{-1} S D_S] \Delta \bar{x}_S = -\bar{c}$
using the inexact conjugate-gradient method, Algorithm A2, defined below.
$\Delta x_S \leftarrow D_S \Delta \bar{x}_S$
$\Delta x \leftarrow Z \Delta x_S$
(compute step length)
Let $\alpha_+ \leftarrow \max[\alpha > 0 \mid x^k + \alpha \Delta x \geq 0]$
(update)
$x^{k+1} \leftarrow x^k + \theta \alpha_+ \Delta x$
(change basis)
Change basis every p iterations based on the value of x^{k+1}
and the sparsity of columns of A.
Update $\mathcal{B}$, $\mathcal{S}$ and B, S.
End

Algorithm A2. Conjugate-Gradient Algorithm (CGA)

Given $\epsilon_{CGA} > 0$ and JMAX
STEP CG0: **Let** $\Delta x_S \leftarrow 0$
$r \leftarrow -D_S Z^T c$
$d \leftarrow r$
(CG iteration; index j)
$j \leftarrow 1$
STEP CG1: **Form** $q \leftarrow [I + D_S S^T B^{-T} D_B^{-2} B^{-1} S D_S] d$ by the following steps.
$w \leftarrow S D_S d$
Solve $Bv = w$
$w \leftarrow D_B^{-2} v$
Solve $B^T v = w$
$q \leftarrow D_S S^T v + d$

STEP CG2: $\beta \leftarrow r^T r / d^T q$
$\Delta x_S \leftarrow \Delta x_S + \beta d$
$r_+ \leftarrow r - \beta q$
If ($\|r_+\| < \epsilon_{CGA}$ **or** $j \geq$ JMAX) **Exit**
$\gamma \leftarrow r_+^T r_+ / r^T r$
$d \leftarrow r_+ + \gamma d$
$r \leftarrow r_+$
$j \leftarrow j + 1$
Go to STEP CG1.

5. THE EXPERIMENTAL IMPLEMENTATION

Some preliminary level-1 (see [Nazareth [1985] for terminology) numerical experiments with the null-space affine scaling on small dense Kuhn-Quandt-type problems are described in Nazareth [1987b]. These problems were of symmetric primal-dual form and permitted a *fixed*-identity or negative-identity basis matrix corresponding to slack variables to be used throughout.

In this section, we discuss a much more practical implementation of algorithm NSASA described in the foregoing section. In particular, we discuss the following: sparse-problem representation, obtaining an initial interior point, finding a starting basis, the basis change strategy, preconditioning and stopping criterion for the conjugate-gradient method, and finally, stopping criterion for the main algorithm.

In these level-2 experiments of a primal null-space affine-scaling method, our main objective is to demonstrate that the simple and natural diagonal preconditioner of Section 5.5 along with the basis change strategy of Section 5.4 make the method applicable to realistic problems. We are also interested in seeing how quickly the method finds an approximate optimal solution.

5.1 Sparse-Problem Representation

The test problems are available in MPS format (see Gay [1986]). In this format, only nonzero elements are stored, and we use the *column list/row index packed-data structure* for the internal representation of an LP matrix. Readers are referred to Nazareth [1987a] for a more detailed discussion.

When solving the system of equations defined in the null-space affine-scaling method, we do not form the matrix $M_S = D_S S^T B^{-T} D_B^{-2} B^{-1} S D_S$ explicitly. What we need are the index sets of basic variables and nonbasic variables. The detailed steps of computing the descent direction without forming M_S explicitly are described in Algorithm A2 of Section 4.

5.2 Initial Solution

Any existing interior-point method requires an initial feasible interior point, which is not readily available in most practical problems. A common practice is to imbed the original problem in an expanded problem for which an interior point is easily provided. Furthermore, it is highly desirable that this interior point be far from the facets of the polyhedral set. We use the above ideas to initiate our algorithm.

In order to find an initial guessed solution $x^\circ > 0$ such that $Ax^\circ \approx b$, we use the following steps. Suppose $\mu e \approx x^\circ$ where e is a vector of ones. Then $A(\mu e) \approx b$ and $\mu(b^TAe) \approx b^Tb$. This implies that $\mu \approx \|b\|^2/|b^TAe|$. Suppose $\mu^\circ = \max(\mu, 1.0)$ and $x^\circ = \mu^\circ e$. If we add an artificial variable v_a to the primal problem, then $x = x^\circ$ and $v_a = \mu^\circ$ gives an interior point of the expanded problem

$$\begin{aligned} &\text{minimize} \quad c^Tx + Mv_a \\ \text{s.t.} \quad &Ax + \left(\frac{(b - Ax^\circ)}{\mu^\circ}\right)v_a = b \\ &x, v_a \geq 0, \end{aligned} \tag{13}$$

where M is a sufficiently large positive number. The variable v_a becomes zero at a certain iteration (end of Phase-1) or approaches zero as an optimal solution is approached.

A simpler approach is to set $\mu^\circ = 1$ in the expanded problem, and a third strategy is to set $\mu^\circ = \|b\|$.

The choices $\mu^\circ = \max(\mu, 1.0)$ and $\mu^\circ = 1$ are used in some interior-point methods. In our method, $\mu^\circ = \|b\|$ is used, because empirical evidence showed that this choice gave slightly better convergence results when compared to the other two.

5.3 Starting Basis

Our method is applied to the form of (1) (and its expanded version (13)), and an initial starting basis must be provided to initiate NSASA.

To find a starting basis, we utilized as many columns corresponding to slack variables (inequality constraints) in the original problem as possible and then augmented these by a set of linearly independent columns using the LUSOL set of subroutines [Gill et al. 1987] described in the next section. Such so-called "crashing" procedures for finding a starting basis are standard in linear programming (see Murtagh and Saunders [1983], for example).

5.4 Basis Change Strategy

We propose to change the basis periodically. The rationale behind our strategy is based on the following observations and experimental results:

—In NSASA, we need to solve the system $[I + D_S S^T B^{-T} D_B^{-2} B^{-1} S D_S]\Delta\bar{x}_S = -\bar{c}$ using the CG method, where $D_B \leftarrow \text{diag}(x_{\beta_1}, \ldots, x_{\beta_m})$ and $D_S \leftarrow \text{diag}(x_{\beta_{m+1}}, \ldots, x_{\beta_n})$. Clearly this system of equations will define a well-conditioned problem if the basis matrix B is chosen so as to maintain the size of elements of D_B large and the size of elements of D_S closer to zero. Here, D_S is acting as a preconditioner for the conjugate-gradient method.

—When a basis that differed greatly from a true optimal basis was kept fixed, preliminary experiments showed that increasing numbers of conjugate-gradient iterations were required to satisfy the stopping criterion. Also, these iteration counts were substantially large. When the basis was closer to the optimal basis, the number of conjugate-gradient steps was drastically reduced.

—If a problem is nondegenerate, then an optimal solution (vertex) consists of basic variables whose values are all strictly positive. When a sequence of points $x^1, x^2, x^3, \ldots$ is generated and converges to an optimal solution, sooner or later the components of x^k corresponding to the indices that are nonbasic at optimality will go to zero. (Note, however, that most problems are highly degenerate.)

—In CGA, two systems of equations involving the basis B need to be solved, which implies that a factorization of B is inevitable for reasons of efficiency. Therefore, we want B to be as sparse as possible.

These points suggest a strategy of changing basis periodically (not too often because of factorization cost) based on the following.

(1) Size of elements of x^k.

(2) Sparsity of the basis matrix.

To obtain a new basis we sort the components of x^k from largest to smallest and form a subset of A, e.g., the first $m + r$ columns, based on this sorting where r is a small positive integer. Then, apply LUSOL to identify m linearly independent columns.

The LUSOL subroutines are similar to those of Reid [1976] for factorizing a sparse square matrix. However, an important feature of LUSOL is that it can factorize a *rectangular* (*possibly singular*) *sparse* matrix and can identify columns that are linearly independent. Additionally, the factorization is performed based on sparsity to avoid substantial growth of nonzero elements, i.e., if there are two or more dependent columns and if one of them must be included in a basis, then LUSOL selects the one with the most sparsity. These new features are crucial to our experiments.

It is possible that the first set given to LUSOL might not contain m linearly independent columns (when a problem is dense, almost always, the first set contained m linearly independent columns). If this happens, we may either keep searching by appending another subset of columns from the remaining set or keep the previous basis for the new iterations. In our initial experiment, we append another subset of columns even though this process is somewhat inefficient.

When either this strategy is used or a starting basis needs to be provided, a certain number of factorizations of subsets of columns of A are inevitable, e.g., every p iterations; see Section 6.1. Although factorization is what one wants to avoid whenever possible, these processes can be rationalized by the following arguments. In most interior-point methods, the number of iterations required to find a good approximate optimal solution seemed negligible with varying problem size, and most of them are within 50 iterations (see, for example, Adler et al. [1986]). If we assume that our interior-point method also takes within 50 iterations for most of the problems and if we let $p = 5$, then the number of factorizations required in our basis change procedure is approximately 10. Now, suppose the simplex method is used to solve a large-scale problem, and the number of iterations required is 5000. If the basis matrix is refactorized every 25 iterations, then the total number of

basis factorizations is 200. In comparison, 10 additional factorizations appear to be insignificant. If the basis change strategy proves to be efficient, then the cost of using additional factorizations will have paid off.

5.5 Preconditioning and Stopping Criterion for the CG Algorithm

When a CG algorithm is used, it is strongly recommended that a good preconditioner be employed in order to enhance convergence. As mentioned in Section 3.3, one of the characteristics of our method is that it possesses a natural preconditioner (i.e., D_S is acting as a diagonal preconditioner). In our implementation, we employ no other preconditioner for the conjugate-gradient algorithm. Thus, we seek to investigate the effect of using the diagonal preconditioner and the basis change strategy together. We use a very simple stopping criterion for the CG algorithm; namely, we terminate the CG algorithm when either $\|r_+\| < \epsilon_{CGA}$ or $j \geq \text{JMAX}$, where j is the number of CG iterations. However, much more sophisticated strategies are possible.

5.6 Stopping Criterion

In the simplex method, an optimal solution is found in a finite number of steps. For an interior-point method the process is infinite. To terminate the main loop of our algorithm, we check the relative improvement of the objective-function value. Given a small positive tolerance ϵ, the null-space affine-scaling algorithm is terminated when

$$\frac{|c^T x^k - c^T x^{k-1}|}{|c^T x^{k-1}|} < \epsilon.$$

Then $x^* \equiv x^k$ will be an approximate optimal solution for the given problem. Note that this stopping criterion is very rudimentary, but adequate for our experiments. In practice, the stopping criterion should be based on the optimality conditions of the linear program.

6. COMPUTATIONAL RESULTS

6.1 Test Problems and Parameters Used

The implementation of NSASA is tested on a number of netlib test problems provided by Gay [1986]. To reduce computational costs of these level-2 experiments we chose the smaller, but nevertheless realistic, problems in this collection. These problems are in the standard form and have no explicit upper-bounded variables. The problems are available in MPS format. Table I displays the details of the problems tested. Problem names are displayed in the first column of the table, and the second column gives the number of rows without including the objective function. The next two columns show the number of less-than-or-equal-to and greater-than-or-equal-to inequality constraints, respectively. The number of variables excluding slacks is given in column 5. Column 6 contains the number of nonzero elements of A, and finally, the last column displays the objective-function value obtained from MINOS version 5.0 [Murtagh and Saunders 1983].

Table I. Problem Statistics

Problem	Rows	L	G	Columns	Nonzeros	Objective
Afiro	27	19	0	32	88	−4.6475314e+02
Adlittle	56	40	1	97	465	2.2549496e+05
Scagr7	129	38	7	140	553	−2.3313893e+06
Share2b	96	83	0	79	730	−4.1573224e+02
Share1b	117	28	0	225	1182	−7.6589319e+04
ScTap1	300	0	180	480	2052	1.4122500e+03
Beaconfd	173	33	0	262	3476	3.3592486e+04
Scsd6	147	0	0	1350	5666	5.0500000e+01

NSASA was implemented in Fortran, and the computational testing was done on a DEC VAX 3500 under the VMS version 5.3 operating system in double-precision arithmetic. The parameter settings are as follows. The step size parameter $\theta = 0.97$ throughout the iterations. It is used to keep the next iterate within the interior of the feasible region (strictly positive). The algorithm is terminated when the relative improvement in the objective function falls below $\epsilon = 1.0 \times 10^{-8}$. The artificial variable cost M is set to be 1.0×10^{6} for all runs. $\epsilon_{CGA} = 1.0 \times 10^{-3}$ and JMAX $= 3\sqrt{n}$ are used to terminate CGA. A new basis is formed based on the two criteria in Section 5.4 every p iterations, and we used the relatively conservative choice $p = 5$. Also, $\mu^{\circ} = \|b\|$ is used for the initial starting point as discussed in Section 5.2.

6.2 Results

Table II summarizes results obtained by running our implementation of NSASA. In the rest of this section, "iteration" refers to the iteration taken by NSASA. When we refer to the iteration of the conjugate-gradient algorithm, we use "CG iteration." Column 3 shows the total number of iterations required to satisfy the stopping criterion in Section 5.5. The average number of CG iterations required is given in column 4, and the approximated optimal objective value is recorded in the last column. The number of Phase 1 iterations in column 2 is obtained by checking if the value of the artificial variable $v_a < 1.0 \times 10^{-14}$ when the step length $\theta = 1.0$. Whenever this happened, v_a is dropped, and Phase 2 is started.

We also present more detailed results for two of these problems, AFIRO in Table III and SHARE2B in Table IV. These tables show how the CG iteration count changes for each NSASA iteration with different choice of JMAX, the upper bound for CG iterations (see Section 6.1). For these two tables, the first column represents the iteration count of NSASA. The three pairs of remaining columns give the number of CG iterations and the objective-function value when no upper bound is imposed on the CG iteration count and

Table II. Null-Space Affine-Scaling Test Statistics

Problem	Phase 1	Total	Avg Num CG	Objective Value
Afiro	3	20	13	−4.6475115e+02
Adlittle	6	40	35	2.2834384e+05
Scagr7	8	39	30	−2.3313898e+06
Share2b	6	24	25	−4.1572770e+02
Share1b	19	81	35	−7.6589315e+04
ScTap1	6	94	78	1.4122679e+03
Beaconfd	37	37	36	3.3592491e+04
Scsd6	1	39	87	5.0512982e+01

Table III. Problem: Afiro

K	CG	Obj.(JMAX = ∞)	CG	Obj.(JMAX = $3\sqrt{n}$)	CG	Obj.(JMAX = $2\sqrt{n}$)
1	32	2.3653822e+08	18	2.3653430e+08	12	2.3676490e+08
2	40	4.8262286e+07	18	4.8358506e+07	12	4.6285100e+07
3	36	−9.3131798e−01	18	−1.7073753e+00	12	1.7121071e+06
4	30	−1.2079128e+02	18	−1.2171494e+02	12	−5.2526923e+00
5	26	−3.0965928e+02	18	−3.0805632e+02	12	−1.1630458e+02
6	13	−3.9866114e+02	10	−3.9904504e+02	12	−3.2044357e+02
7	15	−4.5265653e+02	12	−4.5266613e+02	12	−3.9632306e+02
8	18	−4.5722590e+02	14	−4.5726766e+02	12	−4.5166146e+02
9	17	−4.6131904e+02	14	−4.6141904e+02	12	−4.6006155e+02
10	19	−4.6253329e+02	15	−4.6259895e+02	12	−4.6292963e+02
11	13	−4.6428699e+02	12	−4.6428771e+02	12	−4.6453942e+02
12	17	−4.6462129e+02	13	−4.6462455e+02	10	−4.6465373e+02
13	15	−4.6471190e+02	11	−4.6471946e+02	11	−4.6472550e+02
14	13	−4.6474542e+02	8	−4.6474505e+02	6	−4.6474266e+02
15	9	−4.6474993e+02	5	−4.6475023e+02	6	−4.6475113e+02
16	9	−4.6475068e+02	10	−4.6475094e+02	9	−4.6475166e+02
17	11	−4.6475090e+02	10	−4.6475111e+02	10	−4.6475180e+02
18	9	−4.6475093e+02	12	−4.6475114e+02	11	−4.6475182e+02
19	10	−4.6475093e+02	10	−4.6475114e+02	9	−4.6475183e+02
20			10	−4.6475115e+02	10	−4.6475183e+02
$\sum_{CG}$	352		256		214	

Table IV. Problem: Share2b

K	CG	Obj.(JMAX = ∞)	CG	Obj.(JMAX = $3\sqrt{n}$)	CG	Obj.(JMAX = $2\sqrt{n}$)
1	230	1.5913898e+07	27	1.9647906e+07	18	3.6095128e+07
2	141	5.6078344e+06	27	7.1894776e+06	18	1.8721292e+07
3	107	2.0113111e+06	27	2.2956270e+06	18	1.1384775e+07
4	93	5.0848943e+05	27	7.1160889e+05	18	6.4830991e+06
5	110	1.2090517e+05	27	1.5411370e+05	18	3.8509546e+06
6	111	−3.4433222e+02	27	−3.4389535e+02	18	2.2653503e+06
7	79	−3.6351291e+02	27	−3.6217538e+02	18	1.5069221e+06
8	94	−3.8680363e+02	27	−3.8502709e+02	18	1.1103877e+06
9	110	−4.0268746e+02	27	−3.9242346e+02	18	7.5868944e+05
10	129	−4.1071041e+02	27	−3.9755083e+02	18	6.4458520e+05
11	25	−4.1310748e+02	27	−4.0657880e+02	18	2.8658951e+05
12	24	−4.1501531e+02	27	−4.1232425e+02	18	4.3950269e+04
13	26	−4.1535714e+02	27	−4.1331999e+02	18	−3.4969574e+02
14	25	−4.1563907e+02	27	−4.1400489e+02	18	−3.5789990e+02
15	29	−4.1569468e+02	27	−4.1528013e+02	18	−3.6574079e+02
16	10	−4.1572499e+02	11	−4.1545760e+02	18	−3.8545580e+02
17	9	−4.1572806e+02	13	−4.1563599e+02	18	−3.9579516e+02
18	4	−4.1572818e+02	12	−4.1567479e+02	18	−3.9992117e+02
19	5	−4.1572821e+02	12	−4.1571021e+02	18	−4.0117390e+02
20	4	−4.1572821e+02	10	−4.1572481e+02	18	−4.0246273e+02
21	4	−4.1572844e+02	8	−4.1572753e+02	18	−4.1137699e+02
22	4	−4.1572847e+02	4	−4.1572768e+02	18	−4.1429755e+02
23	4	−4.1572848e+02	4	−4.1572769e+02	18	−4.1513307e+02
24	4	−4.1572848e+02	5	−4.1572770e+02	18	−4.1558688e+02
25					18	−4.1562062e+02
26					12	−4.1570085e+02
27					10	−4.1572464e+02
28					12	−4.1573058e+02
29					2	−4.1573062e+02
30					2	−4.1573063e+02
31					3	−4.1573076e+02
32					2	−4.1573077e+02
33					4	−4.1573077e+02
34					2	−4.1573077e+02
$\sum_{CG}$	1381		484		499	

when an upper bound is imposed with JMAX = $3\sqrt{n}$ and JMAX = $2\sqrt{n}$, respectively.

6.3 Discussion

The results tabulated in Tables III and IV illustrate how the iteration count can change with different settings of parameters and the interplay between terminating the CG early and the total NSASA iteration count. The last lines of Tables III and IV give the total number of CG iterations for each case. An

important feature of NSASA is that the cost of each outer K iteration varies and that when the inner iteration cost is made cheaper (lower value of JMAX) one can expect the NSASA total number of iterations to go up, but not by too much. Also, toward the end, the number of CG iterations can go down dramatically, so that the corresponding outer iteration is much cheaper. This means that it does not matter if the NSASA iteration count goes up a little. (Other problems behaved similarly and are not reported.)

The accuracy we obtained in Table II is *consistent* with the stopping criterion for NSASA given in Section 5.6. Observe that in each case the number of significant digits obtained plus the order of magnitude of the optimal objective value equals or exceeds 7 or 8 (recall choice of $\epsilon = 10^{-8}$). For example, for Afiro, we have 5 digits of accuracy plus 3 (order of magnitude of optimal objective value is about 10^3) giving 8. With Adlittle we have 2 digits of accuracy plus 5 equals 7. Sometimes we do much better, for example, Share1b, Scagr7, and Beaconfd. The one exception is Scsd6. Here we have only 3 digits of accuracy, but the order of magnitude of the optimal objective value is 10^2. For this problem, the accuracy can be increased by adjusting parameters, for instance, decreasing value of ϵ_{CGA}. (See Section 6.1.) However, we do not seek full accuracy, because in any practical implementation, we envision cleaning up with the primal simplex method, as will be discussed in the next section. From the two problems in Tables III and IV, the convergence characteristics of our method show that it has the ability to find an approximate optimal solution quickly. Hence, for the purpose of a more practical approach, the NSASA algorithm could have been terminated much earlier than the number of iterations given in Table II. (In Table II, JMAX = $3\sqrt{n}$.)

For the conjugate-gradient algorithm, one can expect that a good preconditioner can lead to very rapid convergence, often after $O(\sqrt{n})$ CG iterations (see Golub and Van Loan [1989]). For 6 of the 8 problems we tested, excluding the problems Share1b and ScTap1, the average order of CG iterations was $2.33 \times \sqrt{n}$ to obtain approximate optimal values given in Table II. These results imply that the natural diagonal preconditioner along with the basis change strategy worked well on these 6 problems, and the method required a very low number of CG iterations. The total number of iterations of NSASA required for each of these problems was less than or equal to 40.

The strategies we used did not work well all the time and for different reasons. Problems Share1b and ScTap1 illustrate some of the drawbacks of the experimental implementation of our method. Share1b required 81 iterations. The basis change and diagonal-preconditioning strategies worked well for this problem, but we found that our method is sensitive to the starting point for this problem. When $\mu^\circ = 10 * \|b\|$ is used instead of $\mu^\circ = \|b\|$, the number of iterations required was only 52 for the same degree of accuracy of the objective-function value. When the *exact* CG algorithm is used, with varying starting points, the number of iterations required for the same degree of accuracy of the objective-function value ranged from as high as 180 to as low as 41. The case of 180 iterations corresponds to the choice $\mu^\circ = 1$,

and 41 iterations corresponds to the choice $\mu^\circ = 5.0 \times 10^4$ at the starting point. The problem ScTap1 required 94 iterations. When a problem is highly degenerate, the basis change strategy can yield a new basis that is no better or even worse than the old basis, i.e., the new basis remains very different from the true optimal basis. Also, if the system of equations to be solved by the CG algorithm is very ill conditioned, then the use of the diagonal preconditioner may not be good enough. When this happens, a large number of CG iterations is required. This in turn causes more NSASA iterations, when the maximum number of CG iterations is limited to $3\sqrt{n}$.

To summarize, we found:

—The descent direction can be computed inexactly with a relatively small number of CG iterations.

—A good approximation to the optimal solution can be found quickly.

—A good starting point that works for all problems is a subject that remains to be explored.

—A simple combination of strategies is workable. The basis change strategy could be more adroitly employed either when problems are highly degenerate or the system of equations to be solved is ill-conditioned. Employing a better preconditioner such as an incomplete Cholesky factorization [Golub and Van Loan 1989] rather than the natural diagonal preconditioner may enhance the effectiveness of the method. Some interesting connections with preconditioners discussed in Gill et al. [1992] are also worth exploring.

7. COMPATIBILITY WITH THE SIMPLEX METHOD

The initial flurry of excitement and the great strides made in the study of interior-point methods in the eighties, which some felt might even sweep aside much of the earlier algorithmic work in linear programming, have given way to a more balanced view. It is now recognized that the repertoire of computational linear programming has been greatly broadened, but that the simplex method continues to play a key role. Indeed, the development of interior-point methods has helped to stimulate the reexamination and enhancement of simplex techniques.

From the preceding sections of this article, it should be clear that the null-space affine-scaling method and the primal simplex method share much of the machinery of the reduced-gradient approach and are highly compatible with one another. (For additional background, see Nazareth [1987a] and Murtagh and Saunders [1983].) This is a key advantage of the NSASA approach and can be exploited in various ways.

A simple yet effective approach is to pick up from a particular basis found by the NSASA algorithm and clean up with the "pegged-variable" extension of the simplex method, which is able to use an interior point (see Nazareth [1987a] for details).

A much more ambitious approach is explored in Kim and Nazareth [1992], where the two approaches are hybridized within the framework of Dembo

and Sahi [1984]. In effect, the simplex method and NSASA are used in combination in a multiple-pricing strategy. The hybridized method is initiated with a given feasible solution and a partition of the constraint matrix A into three parts that correspond to basic, superbasic, and nonbasic variables (terminology of Murtagh and Saunders [1983]). The partition is done based on a given *feasibility tolerance* $0 < \delta < 1$ and the value of the component of the feasible solution. The simplex computation of reduced costs (pricing), utilizing basic and nonbasic columns, is used to identify suitable candidate columns, and NSASA, utilizing basic and superbasic columns, is used to make rapid progress within a subset of columns selected. Note that the two methods, simplex and NSASA, share exactly the same basis matrix. This kind of approach was explored in a preliminary way and some computational results were obtained; see Kim [1991]. Note that this also has a lot in common with so-called "build-up" and "build-down" strategies; see Dantzig and Ye [1990] and Ye [1990]. Some recent suggestions of Tseng [1992] on "partial affine scaling" are also relevant in this context.

Some of the characteristics and advantages of the hybridized method are as follows.

—The method can be initiated from any point: interior, vertex, or on a facet of the feasible polytope.
—It includes both pure simplex and pure interior-point methods as special cases and employs partial and multiple pricing.
—It exploits the fact that NSASA finds an approximate solution quickly and shares common machinery with the simplex method (same basis matrix).

At this point, the development is very preliminary, and much more needs to be done. However, we think the approach has a great deal of potential.

The experimental implementation described in this article was based on problem setup modules from LPKIT (see Nazareth [1986]), LUSOL routines described in Section 5.3, as well as other homegrown modules. The results are sufficiently encouraging to warrant the development of a much more practical level-3 implementation. We believe that the best way to achieve this would be to *directly extend* MINOS [Murtagh and Saunders 1983], because the code already contains much of the machinery needed to incorporate null-space affine-scaling interior-point techniques. Our research lays the foundation for this substantial and worthwhile further undertaking.

Finally, we note that the preconditioned-CG approach developed here can be adapted to solve Karush-Kuhn-Tucker-type systems arising in other interior methods, for example, the primal-dual predictor-corrector method (see Lustig et al. [1992]).

ACKNOWLEDGMENTS

We are very grateful to the authors of the LUSOL subroutines for making them available for use in our implementation. We also thank the referees for their helpful comments.

REFERENCES

ADLER, I., KARMARKAR, N., RESENDE, M. G. C., AND VEIGA, G. 1986. An implementation of Karmarkar's algorithm for linear programming. *Math. Program. 44*, 297–336.

BARNES, E. R. 1986. A variation on Karmarkar's algorithm for solving linear programming problems. *Math. Program. 36*, 174–182.

DANTZIG, G. B. 1963. *Linear Programming and Extensions*. Princeton University Press, Princeton, N.J.

DANTZIG, G. B., AND YE, Y. 1990. A build-up interior method for linear programming: Affine scaling form. Tech. Rep. SOL 90-4, Systems Optimization Lab., Stanford Univ., Stanford, Calif.

DEMBO, R., AND SAHI, S. 1984. A convergent framework for constrained nonlinear optimization. School of Organization and Management Working Paper, Series B #69, Yale Univ., New Haven, Conn.

DIKIN, I. I. 1967. Iterative solution of problems of linear and quadratic programming. *Sov. Math. Doklady 8*, 674–675.

GAY, D. M. 1986. Electronic mail distribution of linear programming test problems. Numerical Analysis Manuscript 86-0, AT & T Bell Laboratories, Murray Hill, N.J.

GILL, P. E., MURRAY, W., PONCELEÓN, D. B., AND SAUNDERS, M. A. 1992. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix Anal. Appl. 13*, 292–311.

GILL, P. E., MURRAY, W., SAUNDERS, M. A., AND WRIGHT, M. H. 1987. Maintaining LU factors of a general sparse matrix. *Lin. Alg. Appl. 88/89*, 239–270.

GOLUB, G., AND VAN LOAN, C. 1989. *Matrix Computations*. 2nd ed., Johns Hopkins, Baltimore, Md.

KIM, K. 1991. The effective integration of simplex and interior point techniques. Part I: Decomposition. Part II. Null-space affine scaling. Ph.D. thesis, Dept. of Pure and Applied Mathematics, Washington State Univ., Pullman, Wash.

KIM, K., AND NAZARETH, J. L. 1992. Implementation of a primal null-space affine scaling method and its extensions. Tech. Rep. 92-1, Washington State Univ., Pullman, Wash.

LUSTIG, I. J., MARSTEN, R. E., AND SHANNO, D. F. 1992. Computational experience with a globally convergent primal-dual predictor-corrector algorithm for linear programming. Tech. Rep. SOR 92-10, Dept. of Civil Engineering and Operations Research, Princeton Univ., Princeton, N.J.

MEGIDDO, N. 1989. Pathways to the optimal set in linear programming. In *Progress in Mathematical Programming*. Springer, New York, 131–158.

MEHROTRA, S. 1989. Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate-gradient methods. Tech. Rep. 89-04, Dept. of IEMS, Northwestern Univ., Evanston, Ill.

MURTAGH, B. A., AND SAUNDERS, M. A. 1983. MINOS user's guide. Tech. Rep. SOL 83-20, Stanford Optimization Lab., Stanford, Calif.

NAZARETH, J. L. 1993. Quadratic and conic approximating models in linear programming. *MPS-COAL Bull.* 23 (Dec.).

NAZARETH, J. L. 1987a. *Computer Solution of Linear Programs*. Oxford University Press, New York.

NAZARETH, J. L. 1987b. Pricing criteria in linear programming. Rep. PAM-382, Center for Pure and Applied Mathematics, Univ. of California, Berkeley, Calif. Also in *Progress in Mathematical Programming*. Springer-Verlag, New York, 1989, 105–129.

NAZARETH, J. L. 1986. Implementation aids for optimization algorithms that solve sequences of linear programs. *ACM Trans. Math. Softw. 12*, 3, 307–323.

NAZARETH, J. L. 1985. Hierarchical implementation of optimization methods. In *Numerical Optimization*. SIAM, Philadelphia, Penn., 199–210.

REID, J. K. 1976. Fortran subroutines for handling sparse linear programming bases. A.E.R.E.-R.8269, Computer Science and Systems Div., AERE Harwell, Oxfordshire, U.K.

SHANNO, D. F., AND MARSTEN, R. E. 1988. A reduced-gradient version of Karmarkar's algorithm and null-space projections. *J. Optim. Theor. Appl. 57*, 383–397.

STEWART, G. W. 1988. On scaled projections and pseudo-inverses. Computer Science Tech. Rep. Series, TR-2026, Univ. of Maryland, College Park, Md.

TSENG, P. 1992. Partial affine-scaling for linearly constrained minimization. Dept. of Mathematics, Univ of Washington, Seattle (preprint).

VANDERBEI, R. J., MEKETON, M. S., AND FREEDMAN, B. A. 1986. A modification of Karmarkar's algorithm. *Algorithmica 1*, 395–409.

YE, Y. 1990. The "build-down" scheme for linear programming. *Math. Program. 46*, 61–72.

Received March 1992, revised March 1993 and September 1993; accepted October 1993