A QMR-based interior-point algorithm for solving linear programs

Roland W. Freund [a,*], Florian Jarre [b]

[a] *Bell Laboratories, 700 Mountain Avenue, Room 2C–420, Murray Hill, New Jersey 07974–0636, USA*

[b] *Institut für Angewandte Mathematik und Statistik, Universität Würzburg, Am Hubland, D–97074 Würzburg, Federal Republic of Germany*

Abstract

A new approach for the implementation of interior-point methods for solving linear programs is proposed. Its main feature is the iterative solution of the symmetric, but highly indefinite 2×2-block systems of linear equations that arise within the interior-point algorithm. These linear systems are solved by a symmetric variant of the quasi-minimal residual (QMR) algorithm, which is an iterative solver for general linear systems. The symmetric QMR algorithm can be combined with indefinite preconditioners, which is crucial for the efficient solution of highly indefinite linear systems, yet it still fully exploits the symmetry of the linear systems to be solved. To support the use of the symmetric QMR iteration, a novel stable reduction of the original unsymmetric 3×3-block systems to symmetric 2×2-block systems is introduced, and a measure for a low relative accuracy for the solution of these linear systems within the interior-point algorithm is proposed. Some indefinite preconditioners are discussed. Finally, we report results of a few preliminary numerical experiments to illustrate the features of the new approach.

Key words: Linear program; Interior-point method; Symmetric indefinite linear system; Quasi-minimal residual iteration; Indefinite preconditioner

1. Introduction

In this paper, we propose a new version of an interior-point method for solving linear programs. This new version is adapted to support an iterative solver for the linear systems that define the search direction at each step of the interior-point algorithm. Our version is based on inexact search directions, and, in contrast to predictor-corrector methods, it does not rely on the availability of some factorization of the matrix of the linear system.

For solving the linear system that determines the search direction at each step of the algorithm, we derive a simple and stable reduction of the original 3×3-block matrix, which results from the Karush-Kuhn-Tucker optimality conditions, to a 2×2-block matrix. The coefficient matrix of the reduced 2×2-block system is symmetric, but *highly indefinite* in the sense that it has many positive and many negative eigenvalues, and for large-scale linear programs, it is usually sparse. For the

* Corresponding author. *e-mail*: freund@research.bell-labs.com

solution of these symmetric indefinite linear systems, we employ a symmetric variant of the quasi-minimal residual (QMR) algorithm, which was recently proposed by Freund and Nachtigal [12, 13]. The QMR algorithm is a Krylov-subspace iteration for the solution of general, not necessarily symmetric, linear systems. The main features of QMR are iterates defined by a quasi-minimization of the residual norm, and its low work and storage requirements per iteration. The key feature of the symmetric QMR algorithm is that it can be combined with indefinite preconditioners, which is crucial for the efficient solution of highly indefinite linear systems, yet it still fully exploits the symmetry of the matrix of the linear system. We stress that other existing Krylov-subspace iterations for symmetric indefinite systems, such as SYMMLQ and MINRES [28], require the use of symmetric positive definite preconditioners, which is a rather unnatural restriction when the matrix itself is highly indefinite. In this paper, we also discuss the construction of efficient symmetric indefinite preconditioners for the 2×2-block linear systems that arise within our interior-point algorithm.

The standard approach to solving the linear systems in interior-point algorithms is to use direct methods based on a factorization of some matrix. Usually, this is done by first reducing the linear system resulting from the Karush-Kuhn-Tucker optimality conditions to a symmetric positive definite linear system that is then solved by sparse Cholesky factorization; see, e.g., [1, 24] and the references given there. An alternative is to solve the symmetric indefinite 2×2-block system by means of a sparse adaption of the Bunch-Parlett decomposition [3]; see, e.g., [5] and the references therein. In both cases, a substantial amount of fill-in in the factorizations may occur, especially when the constraint matrix of the linear program has dense columns. The problem of fill-in for direct methods is the main motivation for the development of interior-point algorithms based on iterative solvers, such as the one presented in this paper. Another advantage of iterative algorithms over direct methods is the potential reduction of computational costs by working with inexact search directions that are obtained by solving the linear systems only to a low relative accuracy.

The outline of the remainder of the paper is as follows. In Section 2, we formulate the linear program in standard form. In Section 3, we describe our version of an interior-point method. In Section 4, we present a stable reduction of the 3×3-block system for the search direction to a symmetric 2×2-block system. In Section 5, we review the symmetric QMR algorithm for the solution of symmetric indefinite linear systems. In Section 6, we describe various preconditioning strategies for the symmetric QMR algorithm. In Section 7, a few numerical results are reported. Finally, in Section 8, we make some concluding remarks.

Throughout the paper, we use the following notation. Lower-case letters are used for vectors, and upper-case letters are used for matrices. For a vector denoted by a lower-case letter, the upper-case version means the diagonal matrix whose diagonal elements are the components of the vector. For example,

$$X = \operatorname{diag}(x_1, x_2, \ldots, x_n)$$

is the diagonal matrix associated with the vector $x = [\, x_1 \quad x_2 \quad \cdots \quad x_n \,]^T \in \mathbb{R}^n$. The vector norm $\|x\| = \sqrt{x^T x}$ is always the Euclidean norm, and $\|M\| = \max_{\|x\|=1} \|Mx\|$ denotes the corresponding matrix norm. The notation $x > 0$ (respectively, $x \geq 0$) means that all components of the vector x are positive (respectively, nonnegative). The same symbol, 0, will be used to denote the number zero, the zero vector, and the zero matrix. When 0 denotes the zero vector or the zero matrix, then its length or dimension, respectively, will always be clear from the context. The symbol I is used to denote the (square) identity matrix; its size will always be apparent from the context as well.

2. The problem

We consider linear programs in the standard form

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ c^T x \qquad \text{subject to} \quad Ax = b \quad \text{and} \quad x \geq 0. \tag{1}$$

We assume that A is a given real $m \times n$ matrix, and b and c are given real vectors of length m and n, respectively. Without loss of generality, we further assume that the columns of A are scaled to have Euclidean norm 1; this can always be achieved by changing x and c accordingly.

The Karush-Kuhn-Tucker optimality conditions for the problem (1) can be stated as follows:

$$Ax = b, \tag{2}$$

$$A^T y + s = c, \tag{3}$$

$$Xs = 0, \tag{4}$$

$$x \geq 0, \ s \geq 0. \tag{5}$$

Here, the unknown $s \in \mathbb{R}^n$ is a slack vector, and $y \in \mathbb{R}^m$ is the unknown vector of Lagrange multipliers. Equations (2) and (3) are referred to as primal and dual feasibility, respectively, and the nonlinear equation (4) is called the complementarity condition.

For simplicity of presentation, we have chosen the standard form (1) without bounds and ranges. However, we would like to stress that the incorporation of bounds and ranges is rather simple, and it does not require any modifications of our approach (such as splitting of free variables, as in other implementations of interior-point methods).

3. A modified interior-point algorithm

Since the publication [19] of Karmarkar's original interior-point algorithm, numerous variants of the method have been developed; see, e.g., Kojima, Mizuno, and Yoshise [22], Megiddo [25], Monteiro and Adler [27], and Tanabe [29]. We use a modified version of the interior-point algorithm by Lustig, Marsten, and Shanno [23]. In this section, we describe our version of an interior-point method, and we also motivate our modifications.

3.1 The Newton step

Our interior-point method is merely a modification of Newton's method for solving the system of equations consisting of (2), (3), and

$$Xs = \mu e, \tag{6}$$

subject to the *interior-point* condition

$$x > 0 \quad \text{and} \quad s > 0. \tag{7}$$

In (6), $\mu \in \mathbb{R}$ is a small positive number, and $e = [\,1 \quad 1 \quad \cdots \quad 1\,]^T \in \mathbb{R}^n$ is the vector of all ones. Note that (7) is stronger than the original condition (5), and that (6) is an appropriate relaxation of the complementarity condition (4) to avoid violating the strict positivity (7) of x and s.

Applying Newton's method to (2), (3), and (6) yields the system of linear equations

$$\begin{aligned} A\Delta x &= b - Ax, \\ A^T\Delta y + \Delta s &= c - A^T y - s, \\ S\Delta x + X\Delta s &= \mu e - Xs, \end{aligned} \tag{8}$$

for the Newton corrections Δx, Δs, and Δy. We remark that if in the last equation the right-hand side $\mu e - Xs$ were replaced by $\mu e - Xs - \Delta X \Delta s$ (ΔX denotes the diagonal matrix induced by the vector Δx), then the system (8) would actually be equivalent to (2), (3), (6); in (8), the quadratic term $\Delta X \Delta s$ is ignored.

We now denote by

$$r := b - Ax, \quad p := c - A^T y - s, \quad \text{and} \quad q := \mu e - Xs \tag{9}$$

the right-hand sides in (8). Then, after a reordering, the linear system (8) can be written in the form

$$\begin{bmatrix} X & S & 0 \\ I & 0 & A^T \\ 0 & A & 0 \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} q \\ p \\ r \end{bmatrix}, \tag{10}$$

with a 3×3-block matrix.

3.2 The algorithm

Let $x = x^k > 0$, $s = s^k > 0$, $y = y^k \in \mathbb{R}^m$ be the iterate at step k of the interior-point method, and let Δx, Δy, Δs be the solution of the linear system (10). Then, an improved iterate $x^{k+1} = x^+$, $s^{k+1} = s^+$, $y^{k+1} = y^+$ at the next step, $k + 1$, is obtained via

$$\begin{aligned} x^+ &= x + \alpha_x \Delta x, \\ s^+ &= s + \alpha_s \Delta s, \\ y^+ &= y + \alpha_s \Delta y, \end{aligned} \tag{11}$$

where the step lengths α_x and α_s are chosen such that $0 < \alpha_x, \alpha_s < 1$ and x^+, s^+ satisfy the interior-point condition (7), i.e., $x^+ > 0$ and $s^+ > 0$.

As we will motivate below, in our interior-point algorithm the systems (10) are solved iteratively, and the concept of our implementation using an iterative solver with a cheap preconditioner (or one that is easy to update) is based on the following principles.

- No predictor-corrector strategy is applied since the corrector step is approximately as expensive as the predictor step. (This is in contrast to the standard methods based on direct solvers that use the same sparse factorization of some reduced version of the coefficient matrix of (10) for the predictor and the corrector step.) Some *adaptive centering* is used instead.

- *Relatively short steps* (i.e., not the usual 99.995% of the way to the boundary) are taken. This is done for two reasons. First, if the search direction is computed to low relative accuracy, a long step is not meaningful, and secondly, very long steps might worsen the condition number of the linear systems.

The stopping criterion in our interior-point algorithm is based on the error measure

$$\Delta(x,s,y) := \frac{\|Ax-b\|}{\|b\|} + \frac{\|A^T y + s - c\|}{\|c\|} + x^T s.$$

Furthermore, our implementation involves two parameters γ_k and ρ_k, which are chosen at each iteration. Next we outline our algorithm and discuss the choice of these two parameters.

Algorithm 3.1 (Modified interior-point algorithm)

INPUT: *An initial point* $x^0 > 0$, $s^0 > 0$, y^0, *and a stopping tolerance* $\epsilon > 0$.

Set $\mu_{-1} = (x^0)^T s^0/n$.

For $k = 0, 1, \ldots,$ *do* :

1) *Compute* $\Delta(x^k, s^k, y^k)$.
   If $\Delta(x^k, s^k, y^k) \le \epsilon$, *then stop.*
2) *Choose* $\gamma_k \in (0,1)$.
3) *Compute* Δx, Δs, Δy *by solving the linear system* (10), (9) *with*

$$x = x^k, \quad s = s^k, \quad y = y^k, \quad \textit{and} \quad \mu = \mu_k := \min\left\{ \mu_{k-1},\ \gamma_k \frac{(x^k)^T s^k}{n} \right\}. \tag{12}$$

4) *Compute*

$$\bar{\alpha}_x^k := \max\{\, \alpha \le 1 \mid x^k + \alpha\Delta x \ge 0 \,\} \quad \textit{and} \quad \bar{\alpha}_s^k := \max\{\, \alpha \le 1 \mid s^k + \alpha\Delta s \ge 0 \,\}. \tag{13}$$

5) *Choose* $\rho_k \in (0,1)$.
6) *Set*

$$\alpha_x^k = \rho_k \bar{\alpha}_x^k \quad \textit{and} \quad \alpha_s^k = \rho_k \bar{\alpha}_s^k.$$

7) *Set*

$$\begin{aligned} x^{k+1} &= x^k + \alpha_x^k \Delta x, \\ s^{k+1} &= s^k + \alpha_s^k \Delta s, \\ y^{k+1} &= y^k + \alpha_s^k \Delta y. \end{aligned}$$

The quantities γ_k and ρ_k depend on four input parameters χ_1, χ_2, χ_3, and χ_4 of the algorithm, with the default values

$$\chi_1 = 0.1, \quad \chi_2 = 5, \quad \chi_3 = 2, \quad \text{and} \quad \chi_4 = 1/3.$$

The number γ_k at iteration step k, is defined as follows:

$$\gamma_k = \begin{cases} \chi_1, & \text{if } k = 0, \\ \left(\max\{\, 1 - \alpha_x^{k-1},\ 1 - \alpha_s^{k-1} \,\}\right)^{\chi_2}, & \text{if } k > 0. \end{cases} \tag{14}$$

The number ρ_k at iteration step k is chosen as

$$\rho_k = \max \left\{ \min \{ 1 - \gamma_k^{\chi_3},\ 1 - (1 - \bar{\alpha}_x^k)^{\chi_3},\ 1 - (1 - \bar{\alpha}_s^k)^{\chi_3} \},\ \chi_4 \right\}. \tag{15}$$

The motivation for this choice of ρ_k is that γ_k being small implies a large reduction in complementarity, and $\bar{\alpha}_x^k$ and $\bar{\alpha}_s^k$ being close to 1 implies a large reduction of infeasibility by the full Newton step. Thus, if both is the case, a large ρ_k, which brings the iterate close to the boundary of the feasible set, is acceptable.

The motivation for the choice (14) of γ_k is as follows. First note that, in view of (12), the values of α_x^k and α_s^k already depend on γ_k, and therefore, our strategy for choosing γ_k needs to be based on the previous values α_x^{k-1} and α_s^{k-1}. If α_x^{k-1} and α_s^{k-1} were large, then, at iteration step $k-1$, good progress towards optimality was made, provided γ_{k-1} was small at the same time (maybe it was!). We now try to make progress at step k by choosing a small γ_k, and this is achieved by (14). Conversely, if α_x^{k-1} and α_s^{k-1} were small, then little progress was made in step $k-1$, and (14) computes a large γ_k, i.e., the algorithm does some centering at iteration step k.

We remark that a convergence analysis of Algorithm 3.1 is given in [10]. It turns out that, in order to prove convergence, we need to incorporate a safeguard for the choice of γ_k and ρ_k in Algorithm 3.1. The purpose of this safeguard is to ensure that the iterates approximately satisfy the centrality condition, (6), $Xs = \mu e$. Here, closeness to centrality is measured by

$$d(x,s) := \max_{i,j} \left\{ \frac{x_i s_i}{x_j s_j} \right\}.$$

Indeed, in [10], we show that, for any given $\delta \in (0,1)$, γ_k and ρ_k can be chosen such that

$$\liminf_{k\to\infty} d(x^k, s^k) \ < \ \frac{1}{\delta}$$

is guaranteed.

Kojima, Megiddo, and Mizuno [21] proposed a different safeguard, which also turned out to be efficient in an actual implementation; see [24]. The proof in [21], however, does not carry over to the type of inexact search directions used in Algorithm 3.1.

We tested Algorithm 3.1 on those NETLIB test problems [16] that have no bounds and ranges. The result in terms of the number of iterations was by a factor of about 1.5 to 2 worse than the best results reported to date (for example, by Lustig, Marsten, and Shanno [24], Vanderbei and Carpenter [30], or Xu, Hung, and Ye [31]). This iteration count has to be seen in light of our goals of not using any predictor-corrector steps and of using relatively short steps.

In the forthcoming paper [11], we will report results of a complete implementation of a QMR-based interior-point method, and a comparison with an interior-point method that uses the standard approach of solving (10) via a direct factorization of the symmetric positive definite matrix $AXS^{-1}A^T$ (cf. Section 4.4 below).

3.3 Sufficient accuracy

Recall from (11) that, given an iterate $x > 0$, $s > 0$, y, and some approximate solution Δx, Δs, Δy of (10), the step lengths α_x, $\alpha_s \in (0,1)$ in the interior-point method are chosen so that $x^+ = x + \alpha_x \Delta x > 0$ and $s^+ = s + \alpha_s \Delta s > 0$ is fulfilled. Typical values for α_x, α_s that we observed in our implementation are about 0.2 to 0.8 during the first iterations of the interior-point method, reaching larger values, such as 0.99, in the final stages.

If Δx, Δs, Δy solve the linear systems (10) exactly, then, by the linearity of r, the primal residual r^+ at x^+ satisfies

$$\|r^+\| = \|b - Ax^+\| = \|b - A(x + \alpha_x \Delta x)\| = \|(1-\alpha_x)(b - Ax)\| = (1-\alpha_x)\|r\|.$$

Thus, the norm of the primal residual r is reduced by the factor $1-\alpha_x$ in the case of exact solution of (10). If we allow for some tolerance η in the approximate solution of (10) by requiring that Δx satisfies only

$$\|b - Ax - A\Delta x\| \ \le \ \eta(1-\alpha_x)\|b - Ax\| \tag{16}$$

instead of $A\Delta x = b - Ax$, then, by the triangle inequality, we have

$$\begin{aligned}
\|r^+\| &= \|b - A(x + \alpha_x \Delta x)\| \\
&= \|(1-\alpha_x)(b - Ax) + \alpha_x(b - Ax - A\Delta x)\| \\
&\le (1+\eta\alpha_x)(1-\alpha_x)\|b - Ax\| = (1+\eta\alpha_x)(1-\alpha_x)\|r\|.
\end{aligned}$$

Together with $\alpha_x \in (0,1)$, it follows that the norm of r is reduced by a factor of at least

$$(1+\eta\alpha_x)(1-\alpha_x) \ \le \ \min\{\,(1+\eta)(1-\alpha_x),\ 1-(1-\eta)\alpha_x\,\},$$

instead of the factor $1-\alpha_x$ if the linear system (10) were solved exactly. For example, if we choose $\eta = 0.1$, then step (11) with the approximate Δx still yields at least 90% of the maximal possible reduction of $\|r\|$. Indeed, if α_x is close to 1, then $\|r\|$ is reduced by at least $1.1(1-\alpha_x)$, and if α_x is small, then $\|r\|$ is reduced by at least $1 - 0.9\alpha_x$. Note that, given some approximate direction Δx, the criterion (16) is easy to check (for the approximate direction), since $\alpha_x = \alpha_x^k = \rho_k \bar{\alpha}_x^k$ can be computed immediately using the definitions of ρ_k and $\bar{\alpha}_x^k$ in (15) and (13), respectively.

Of course, analogous considerations hold for p. Similarly, one might require $\|q^+\|_\infty \le \eta(1-\alpha)\mu$. We point out, however, that our stable reduction described in Section 4.2 below even guarantees $q^+ = 0$; this follows readily from (28). This is important, since $q^+ \approx 0$ is crucial for the existence of a large feasible step length.

A practical modification of criterion (16) can be obtained by requiring that Δx satisfy

$$\|b - A\Delta x - Ax\| \ \le \ \eta(1-\alpha_x)\max\{\,\|b - Ax\|,\ \mu\,\}.$$

This allows for a moderate accuracy in the primal residual even if the starting point x^0, for example, happened to be primal feasible, i.e., $Ax^0 = b$.

4. A stable reduction of the 3×3-block system

The iterates $x = x^k$, $s = s^k$, $y = y^k$, $k = 1, 2, \ldots$, generated by Algorithm 3.1 will approach feasibility (i.e., (2) and (3)) and optimality (i.e., (4)) simultaneously, if the parameters μ, α_x, α_s are suitably chosen at each step (see, e.g., [21]), and throughout the algorithm one can observe that roughly

$$\|r\| \approx \|p\| \approx \|q\|.$$

This observation—even though the meaning of "$\approx$" depends on the initial point, on the actual algorithm used, and on quantities like $\|x\|$, $\|s\|$—shall motivate our first step towards the solution of (10) by reducing the size of the 3×3-block matrix in (10).

4.1 The standard reductions

The common practice to reduce the size of the linear system (10) is to eliminate either the variable Δs or the variable Δx from the first set of equations ($X\Delta s + S\Delta x = q$), resulting in the equivalent reduced systems

$$\begin{bmatrix} -SX^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} p - X^{-1}q \\ r \end{bmatrix}, \tag{17}$$

or

$$\begin{bmatrix} -SX^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} u \\ \Delta y \end{bmatrix} = \begin{bmatrix} p \\ r - AS^{-1}q \end{bmatrix}, \tag{18}$$

respectively. Here, we have written system (18) with a symmetric matrix by setting $u = -XS^{-1}\Delta s$. Given the solution Δx, Δy of (17), the part Δs of the solution of (10) is obtained via

$$\Delta s = X^{-1}(q - S\Delta x). \tag{19}$$

Similarly, from the solution u, Δy of (18), the parts Δs and Δx of the solution of (10) are obtained by setting

$$\begin{aligned} \Delta s &= -SX^{-1}u, \\ \Delta x &= S^{-1}(q - X\Delta s). \end{aligned}$$

Both reductions (17) and (18) suffer from a severe drawback. Since x approaches an optimal point and s approaches the associated Lagrange multiplier, some components of x and some components of s tend to zero in the course of the interior-point algorithm. Thus, in finite-precision arithmetic, the components of p corresponding to small components of x are "lost" when forming $p - X^{-1}q$ in (17), and likewise, the information of r is "lost" when forming $r - AS^{-1}q$ in (18) if some of the components of s are very small.

To be more precise about this drawback, note that, in practice, the system (17) can only be solved to a certain relative accuracy in the residual. In particular, the first block row of (17) will not hold exactly, but rather satisfy the relation

$$\left\| [\, -SX^{-1} \quad A^T \,] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} - \left(p - X^{-1}q\right) \right\|^2 = \varepsilon^2 \left(\left\| p - X^{-1}q \right\|^2 + \|r\|^2 \right), \tag{20}$$

for some $\varepsilon > 0$ that we may think of as being roughly the size of the machine precision.

Let us assume, for simplicity, that the substitution (19) used to compute Δs from the solution of (17) can be carried out in exact arithmetic. If the small components of x are of order ε and $\|p\| \approx \|q\|$, then, in general, we have

$$\left\| p - X^{-1}q \right\| \approx \frac{\|p\|}{\varepsilon}.$$

As a result, even for a full Newton step, i.e., $\alpha_s = 1$, the new point $s^+ = s + \Delta s$, $y^+ = y + \Delta y$ satisfies the dual-feasibility condition (3) only poorly. In fact, with (9), (19), and (20), one easily verifies that the norm of the corresponding residual vector $p^+ = c - A^T y^+ - s^+$ fulfills

$$\begin{aligned} \|p^+\| &= \left\| c - A^T(y + \Delta y) - (s + \Delta s) \right\| \\ &= \left\| p - X^{-1}q - A^T\Delta y + X^{-1}S\Delta x \right\| \\ &= \varepsilon \left(\left\| p - X^{-1}q \right\|^2 + \|r\|^2 \right)^{1/2} = \Omega(\|p\|). \end{aligned}$$

Here, $a = \Omega(b)$ means that $b = \mathcal{O}(a)$. In other words, the new residual p^+ is of about the same size or larger than the previous residual p. Similarly, if (18) is solved to finite precision, then $x + \Delta x$ may have a large primal residual if some components of s are of order ε.

4.2 A stable reduction

Next, we show that this potential instability can be avoided by simply combining parts of both approaches (17) and (18) to obtain a novel stable reduction of the original 3×3-block system (10) to a linear system with the 2×2-block matrix

$$K := \begin{bmatrix} -SX^{-1} & A^T \\ A & 0 \end{bmatrix}. \tag{21}$$

Since K results from the Karush-Kuhn-Tucker optimality conditions, we refer to it as the *KKT matrix* in the sequel. It is easy to verify that, provided A has full row rank m, the KKT matrix has exactly m positive and n negative eigenvalues. Hence, unless $m \ll n$, the matrix K is highly indefinite.

The basis of our stable reduction is a partition of the vectors x and s into two parts x_1, x_2 and s_1, s_2, respectively, such that $x_1 \geq s_1$ and $x_2 < s_2$ is satisfied[1]. Moreover, without loss of generality, we assume that x_1 and s_1 are the leading entries of x and s, i.e.,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{and} \quad s = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}. \tag{22}$$

Conforming with (22), we write the constraint matrix A as follows:

$$A = [\, A_1 \quad A_2 \,]. \tag{23}$$

Similarly, (22) induces the partition

$$p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \quad \text{and} \quad q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{24}$$

of the residual vectors p and q defined in (9). As usual, we denote by X_1, X_2, S_1, and S_2 the diagonal matrices associated with the vectors x_1, x_2, s_1, and s_2, respectively.

Instead of the original system (10), we propose to solve the equivalent reduced 2×2-block system

$$\begin{bmatrix} -SX^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} u \\ \Delta y \end{bmatrix} = \begin{bmatrix} \tilde{r} \\ \tilde{p} \end{bmatrix}, \tag{25}$$

where

$$\tilde{r} := \begin{bmatrix} p_1 - X_1^{-1} q_1 \\ p_2 \end{bmatrix} \quad \text{and} \quad \tilde{p} := r - A_2 S_2^{-1} q_2. \tag{26}$$

Furthermore, the new variable u is connected with x_1 and s_2 via the relation

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \text{where} \quad u_1 := \Delta x_1 \quad \text{and} \quad u_2 := -X_2 S_2^{-1} \Delta s_2. \tag{27}$$

[1] Throughout Section 4, subscripts, as in x_1 and s_2, always refer to this partition of the vectors x and s, and they do not denote single components of the vectors x and s.

Using (22)–(24), (26), and (27), together with

$$
\begin{aligned}
\Delta x_2 &= S_2^{-1} q_2 + u_2, \\
\Delta s_1 &= X_1^{-1} \left(q_1 - S_1 \Delta x_1 \right),
\end{aligned} \tag{28}
$$

one readily verifies that the reduced 2×2-block system (25) is indeed equivalent to the original system (10).

In our QMR-based interior-point method, we solve the reduced system (25) iteratively to a low relative accuracy, and thus we obtain only an approximate solution. Let u, Δy be such an approximate solution of (25). Then, in view of (27), we set $\Delta x_1 = u_1$. Note that computing Δs_2 from (27) would involve the inverse of the "small" matrix X_2. Instead, we compute Δs_2 by means of the relation

$$
\Delta s_2 = c_2 - A_2^T (y + \Delta y) - s_2,
$$

which is part of the original equations (8). Finally, we obtain Δx_2 and Δs_1 from (28). Observe that, in all these computations to obtain Δx, Δy, and Δs, only the inverses of the "large" matrices X_1 and S_2 are used, but never X_2^{-1} or S_1^{-1}. Furthermore, we remark that the partition of x into x_1 and x_2 has nothing to do with guessing active indices. The partition is merely done to improve the stability for solving the current linear system, it does not involve any additional computational cost, and there is no loss if the partition changes at each iteration. Finally, we point out that most interior-point methods for solving linear programs generate a sequence of iterates that converge to a strictly complementary point, even if the linear program is degenerate. For such methods, our partition above allows a uniform bound for $\|X_1^{-1}\|$ and $\|S_2^{-1}\|$, that is valid for all iterations. However, this bound depends on the given linear program, and it can be very large.

4.3 Connections with matrix factorizations

All three reduced linear systems (17), (18), and (25) are based on certain matrix factorizations that reduce the matrix of the original system (10) to upper 2×2-block-triangular matrices whose $(2,2)$-block is the KKT matrix K defined in (21).

In fact, the reduction (17) is based on the transformation

$$
\begin{bmatrix} I & 0 & 0 \\ -X^{-1} & I & 0 \\ 0 & 0 & I \end{bmatrix} \cdot \begin{bmatrix} X & S & 0 \\ I & 0 & A^T \\ 0 & A & 0 \end{bmatrix} = \begin{bmatrix} X & S & 0 \\ 0 & -SX^{-1} & A^T \\ 0 & A & 0 \end{bmatrix}, \tag{29}
$$

and the reduction (18) corresponds to the decomposition

$$
\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -AS^{-1} & 0 & I \end{bmatrix} \cdot \begin{bmatrix} X & S & 0 \\ I & 0 & A^T \\ 0 & A & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & -SX^{-1} & 0 \\ I & 0 & 0 \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} S & -S & 0 \\ 0 & -SX^{-1} & A^T \\ 0 & A & 0 \end{bmatrix}. \tag{30}
$$

Note that, in both cases, the KKT matrix K appears as the $(2,2)$-block of the matrices (viewed as upper 2×2-block-triangular matrices) on the right-hand side of the factorizations (29) and (30).

Our proposed stable reduction (25) is based on the decomposition

$$\begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ -X_1^{-1} & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & -A_2 S_2^{-1} & 0 & 0 & I \end{bmatrix} \cdot \begin{bmatrix} X_1 & 0 & S_1 & 0 & 0 \\ 0 & X_2 & 0 & S_2 & 0 \\ I & 0 & 0 & 0 & A_1^T \\ 0 & I & 0 & 0 & A_2^T \\ 0 & 0 & A_1 & A_2 & 0 \end{bmatrix} \cdot \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -S_2 X_2^{-1} & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}$$

$$= \begin{bmatrix} X_1 & 0 & S_1 & 0 & 0 \\ 0 & S_2 & 0 & -S_2 & 0 \\ 0 & 0 & -S_1 X_1^{-1} & 0 & A_1^T \\ 0 & 0 & 0 & -S_2 X_2^{-1} & A_2^T \\ 0 & 0 & A_1 & A_2 & 0 \end{bmatrix},$$

which can be viewed as a composition of the factorizations (29) and (30). Again the KKT matrix K appears as a subblock on the right-hand side of the decomposition.

We close this section with a remark on the condition number of the matrix K in (21). It is straightforward to see that even for nondegenerate programs and for points $x = x(\mu)$, $s = s(\mu)$ on the central path (i.e., for points satisfying (2), (3) and (6)), the condition number of K becomes unbounded as $\mu \to 0$. However, a simple symmetric diagonal rescaling—as well as any reasonable preconditioner, see Section 6 below—immediately eliminates this ill-conditioning. For example, with

$$\Lambda := \operatorname{diag}\left(I, S_2^{-1/2} X_2^{1/2}, I\right),$$

the rescaled KKT matrix

$$\Lambda K \Lambda = \begin{bmatrix} -S_1 X_1^{-1} & 0 & A_1^T \\ 0 & -I & X_2^{1/2} S_2^{-1/2} A_2^T \\ A_1 & A_2 S_2^{-1/2} X_2^{1/2} & 0 \end{bmatrix}$$

has a bounded condition number for nondegenerate programs and $x = x(\mu)$, $s = s(\mu)$, as $\mu \to 0$. Such a rescaling was also considered by Gill et al. in [17].

4.4 Reduction to the Schur complement

Of course, the system (25) can be reduced further to the linear system

$$AXS^{-1}A^T \Delta y = \tilde{p} + AXS^{-1}\tilde{r} \tag{31}$$

whose coefficient matrix $AXS^{-1}A^T$ is just the Schur complement of the leading diagonal block $-SX^{-1}$ in the KKT matrix K. However, while the construction of the stable reduction (25) does not create any fill-in, nor a systematic cancellation error as in (17) and (18), further reduction to (31) no longer preserves these properties. Small components of the vector s may numerically cancel parts of $\tilde{r}$, and the sparsity structure of the matrix $AXS^{-1}A^T$ may be destroyed by dense columns of A. Therefore, a high accuracy is needed when solving (31), and dense columns of A need special treatment via rank-1 updates or splitting of dense columns. Furthermore, while the existence of free variables (i.e., variables with no upper or lower bounds) does not cause any conceptual difficulties for the stable reduction (25), the reduction to (31) can only be carried out if the free variables are eliminated, for instance by splitting each free variable into two nonnegative variables.

The advantage of (31) certainly is that the coefficient matrix $AXS^{-1}A^T$ is symmetric positive definite, and sparse Cholesky factorization combined with some ordering heuristics has proved to be very effective in practice. Thus, despite the above mentioned drawbacks, the solution of systems of the form (31) (without the use of any stable reduction, and hence with different right-hand sides) has become the standard in state-of-the-art codes for interior-point methods.

Since $AXS^{-1}A^T$ is symmetric positive definite, one can also use the standard conjugate gradient algorithm [18] to solve systems of the form (31) iteratively; this approach was considered, e.g., in [20, 26], but it has not been widely used since then.

5. The symmetric QMR algorithm

In this section, we review the symmetric QMR algorithm for the iterative solution of symmetric linear systems

$$Kz = d. \tag{32}$$

Recall from (25) that the systems we need to solve within our interior-point algorithm are of the type (32) where K is defined in (21) and z, d are given by

$$z = \begin{bmatrix} u \\ \Delta y \end{bmatrix} \quad \text{and} \quad d = \begin{bmatrix} \tilde{r} \\ \tilde{p} \end{bmatrix}.$$

However, in this section, we consider general systems (32) where K is a nonsingular real symmetric $N \times N$ matrix and d is a real vector of length N.

5.1 Preconditioning

The QMR algorithm is a Krylov-subspace iteration, and such schemes are hardly ever applied directly to the linear system, but instead they are combined with preconditioning. Let M be a nonsingular $N \times N$ matrix with the following properties:

(i) The matrix M "in some sense" approximates the matrix K of (32);

(ii) Linear systems with M as coefficient matrix are much easier to solve than the original system (32).

The matrix M is called the *preconditioner*.

Since K is symmetric, in view of (i), we also require that M be symmetric, and hence we can write M in the form

$$M = U^T D^{-1} U, \quad \text{where} \quad D = D^T. \tag{33}$$

Using (33), the solution of a system with coefficient matrix M requires the solution of one system with U and one with U^T, and a matrix-vector multiplication with D. In view of (ii), these computational tasks should be "easy", compared to the solution of (32). This is the case for the modified SSOR preconditioner that we will discuss in Section 6.2. There, U is a permuted upper block-triangular matrix, and D is a block-diagonal matrix with 1×1 and 2×2 blocks.

Using the decomposition (33), we see that the original system (32) is equivalent to the preconditioned system

$$Bz' = d', \tag{34}$$

where

$$B = U^{-T} K U^{-1} D, \quad z' = D^{-1} U \left(z - z_0 \right), \quad \text{and} \quad d' = U^{-T} \left(d - K z_0 \right). \tag{35}$$

Here, z_0 is an arbitrary initial guess for the solution of (32). The iterative scheme, such as the symmetric QMR algorithm described in Section 5.3 below, is then applied to the preconditioned system (34), using $z'_0 = 0$ as initial vector, and it generates a sequence of approximations

$$z'_j, \quad j = 1, 2, \ldots, \tag{36}$$

to the exact solution $z' = B^{-1} d'$ of (34). In view of (35), the iterates in (36) can be translated back into iterates z_j for the original system (32) by setting

$$z_j = z_0 + U^{-1} D z'_j, \quad j = 1, 2, \ldots .$$

We remark that, in (35), parts of the preconditioner are applied from both the left and the right to the original matrix K, and this is called *mixed* preconditioning. It is also possible to apply the whole preconditioner either from the left or the right, and this is referred to as *left* or *right* preconditioning. However, mixed preconditioning has definite advantages over left or right preconditioning. For example, when M is an SSOR-type preconditioner, then, as we will see in Section 6.3, mixed preconditioning can be implemented very efficiently by using the so-called Eisenstat trick [4]. This trick is not applicable for left or right preconditioning. Furthermore, using mixed preconditioning (instead of left or right preconditioning) typically reduces possible nonnormality of the coefficient matrix B in (34), and for Krylov-subspace algorithms, it is desirable to avoid highly nonnormal B. This is why we consider only mixed preconditioning (35) in this paper.

5.2 Krylov-subspace algorithms and QMR

The set

$$K_j(B, d') := \operatorname{span}\{\, d', Bd', B^2 d', \ldots, B^{j-1} d' \,\} \tag{37}$$

is called the jth *Krylov subspace* of $\mathbb{R}^N$ generated by B and d'. An iterative scheme for the solution of the system (34) is called a *Krylov-subspace algorithm* if its iterates (36) satisfy

$$z'_j \in K_j(B, d'), \quad j = 1, 2, \ldots . \tag{38}$$

Here, in accordance with the setting in Section 5.1, we assume that the initial iterate is $z'_0 = 0$.

The power basis used in the definition (37) of Krylov subspaces is in general ill-conditioned, and hence it is not suited for numerical computations. Instead, for a practical Krylov-subspace algorithm one first needs to specify the choice of suitable basis vectors

$$v_1, v_2, \ldots, v_j, \quad j = 1, 2, \ldots, \tag{39}$$

for $K_j(B, d')$. Secondly, one needs to specify how the iterates (38) are chosen. For a recent survey of Krylov-subspace algorithms, we refer the reader to [8].

The quasi-minimal residual (QMR) method [12, 13] is a Krylov-subspace algorithm for the solution of linear systems (34) with general nonsingular matrices B. The development of QMR was motivated by the desire to have a Krylov-subspace algorithm that uses an inexpensive procedure to compute a suitable sequence of basis vectors (39) and, at the same time, produces good approximations z'_j to the solution of (34).

The QMR method uses the look-ahead Lanczos process [9] to generate the basis vectors (39). In addition to (39), the Lanczos procedure also computes a second sequence of basis vectors

$$\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_j, \quad j = 1, 2, \ldots, \tag{40}$$

that span the jth Krylov subspace $K_j(B^T, \tilde{v}_1)$ generated by B^T and $\tilde{v}_1$. Here, $\tilde{v}_1 \in \mathbb{R}^N$ is an arbitrary nonzero vector that can be chosen freely. In the Lanczos process, the two sequences of vectors (39) and (40) are constructed to be biorthogonal, i.e.,

$$\tilde{v}_l^T v_k = 0 \quad \text{for all} \quad l \neq k = 1, 2, \ldots, j. \tag{41}$$

However, it may not always be possible or numerically advisable to enforce the condition (41), as it would lead to a so-called *breakdown* or *near-breakdown* in the Lanczos process. Fortunately, such breakdowns and near-breakdowns can be avoided by performing *look-ahead steps*, instead of a step of the standard Lanczos process. More precisely, look-ahead steps avoid breakdowns and near-breakdowns by enforcing only a relaxed block-biorthogonality condition, rather than (41).

The Lanczos vectors (39) and (40) can be constructed by means of either three-term recurrences [9, 12] or coupled two-term recurrences [13]. The three-term recurrences for computing the first $j+1$ basis vectors $v_1, v_2, \ldots, v_{j+1}$ can be summarized compactly as follows:

$$BV_j = V_{j+1} T_j. \tag{42}$$

Here, $V_j := [\, v_1 \quad v_2 \quad \cdots \quad v_j \,]$, and T_j is a $(j+1) \times j$ matrix whose entries are the coefficients of the Lanczos recursions. Furthermore, T_j is both a block-tridiagonal matrix and an unreduced upper Hessenberg matrix, and it always has full rank j. The blocks in the block-tridiagonal matrix T_j are almost all of size 1×1, and blocks of size bigger than 1 only occur if look-ahead steps are necessary. We remark that, since T_j is an unreduced upper Hessenberg matrix, the relations summarized in (42) are indeed recurrences for computing the vectors $v_1, v_2, \ldots, v_{j+1}$. For example, consider the last column of the matrix equation (42), and assume that no look-ahead steps are taken, so that T_j is a scalar tridiagonal matrix. Then, by solving the relation in the last column of (42) for the vector v_{j+1} (this is possible since T_j is unreduced, i.e., all its subdiagonal elements $(T_j)_{i+1,i}$ are different from zero), it follows that v_{j+1} can be obtained as a combination of Bv_j and the two previous basis vectors v_j and v_{j-1}. Similarly, if look-ahead steps occur, then v_{j+1} is computed as a combination of Bv_j and two blocks of previous vectors that include at least v_j and v_{j-1}; for details we refer to [9, 12]. The three-term recurrences for computing the first $j+1$ basis vectors $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_{j+1}$ can be summarized in an analogous relation of the form:

$$B^T \tilde{V}_j = \tilde{V}_{j+1} \tilde{T}_j. \tag{43}$$

When coupled two-term recurrences are used for the construction of the Lanczos vectors (39) and (40), then one obtains relations that are mathematically equivalent to (42) and (43); see [13].

Next, we turn to the choice of the iterates (38) in QMR. Since the vectors $v_1, v_2, \ldots, v_j$ span the jth Krylov subspace $K_j(B, d')$, any possible iterate (38) can be written as

$$z'_j = V_j \zeta_j, \quad \text{where} \quad \zeta_j \in \mathbb{R}^j,$$

and ζ_j is still a free parameter vector. In QMR, ζ_j is chosen as the (unique) solution of the least-squares problem

$$\|f_j - T_j \zeta_j\| = \min_{\zeta \in \mathbb{R}^j} \|f_j - T_j \zeta\|, \quad \text{where} \quad f_j := [\, \|d'\| \quad 0 \quad \cdots \quad 0 \,]^T \in \mathbb{R}^{j+1}. \tag{44}$$

Here, for simplicity, we have assumed that the basis vectors are scaled to unit length:

$$\|v_l\| = 1 \quad \text{for all} \quad l = 1, 2, \dots .$$

If the vectors v_l are unscaled, then (44) is replaced by a suitably weighted least-squares problem. We remark that, by (42), the residual vector of the jth QMR iterate z'_j is given by

$$d' - Bz'_j = V_{j+1}\left(f_j - T_j \zeta_j\right). \tag{45}$$

Hence, in view of (44), the jth QMR iterate z'_j is characterized by a minimization of the second factor, $f_j - T_j\zeta_j$, in the representation (45) of its residual vector. This is called a *quasi-minimal residual* (QMR) property, and the resulting iterative scheme for solving linear systems (34) is the QMR method.

For a more detailed description of the QMR method and its properties, we refer the reader to [12, 13, 6]. In particular, in [12, 6], convergence results for QMR are derived. These results show that QMR will converge fast if the eigenvalues of B are clustered around 1.

5.3 A simple implementation of the symmetric QMR algorithm

The QMR method described in the previous section is an iterative solver for linear systems (34) with general coefficient matrices B. At each jth iteration step, QMR requires two matrix-vector products, namely

$$Bv_{j-1} \quad \text{and} \quad B^T \tilde{v}_{j-1}, \tag{46}$$

that are used to extend the Krylov subspaces $K_{j-1}(B, d')$ and $K_{j-1}(B^T, \tilde{v}_1)$ at the previous step $j-1$ to $K_j(B, d')$ and $K_j(B^T, \tilde{v}_1)$, respectively. The matrix-vector multiplications (46) are the main computational work in QMR.

We now turn to the special systems (34), with B given by (35), that result from preconditioning the symmetric linear system (32) using a general symmetric preconditioner M, factored as in (33). If M is symmetric positive definite, then we can choose $D = I$ in (33) and the preconditioned matrix B is also symmetric. If M is indefinite, then B is a nonsymmetric matrix, and at first glance, it looks like one would need to use the general QMR algorithm for the solution of the preconditioned system (34). However, as Freund and Zha [7, 15] observed, it is possible to still exploit the symmetry of the indefinite matrices K and M and to devise a *symmetric* QMR algorithm for which the work and storage requirements are half of those of general QMR. The idea is to take advantage of the freedom in choosing the initial vector $\tilde{v}_1$ for the second sequence of Lanczos vectors (40). Indeed, setting $\tilde{v}_1 = Dd'$, it is easy to see that

$$\tilde{v}_j = \xi_j D v_j \quad \text{for all} \quad j = 1, 2, \dots, \tag{47}$$

where $\xi_j \in I\!R$ are suitable scaling factors. Hence, by (47), one can replace the second matrix-vector multiplication in (46) by one multiplication with D. For all interesting preconditioners M, the matrix D is a block-diagonal matrix with only 1×1 and 2×2 blocks, and multiplications with D are usually negligible compared to the work required to compute products with B^T.

We now state a simple implementation without look-ahead of the resulting symmetric QMR algorithm. This implementation uses coupled two-term recurrences to generate the vectors v_j, and it is a counterpart to the simple "QMR-from-BCG" algorithm for general matrices derived in [14].

Algorithm 5.1 (Symmetric QMR without look-ahead for the solution of symmetric systems (32) with symmetric preconditioner (33))

0) *Choose* $z_0 \in \mathbb{R}^N$, *and set* $z_0' = 0 \in \mathbb{R}^N$.

   Compute the vectors

$$g_0 = U^{-T}(d - Kz_0), \quad v_0 = g_0, \quad \text{and} \quad w_0 = Dv_0.$$

   Compute the scalars

$$\tau_0 = g_0^T g_0 \quad \text{and} \quad \varrho_0 = g_0^T w_0.$$

   Set $c_0 = 0 \in \mathbb{R}^N$ *and* $\vartheta_0 = 0 \in \mathbb{R}$.

For $j = 1, 2, \ldots,$ *do* :

1) *Compute the matrix-vector product*

$$t_{j-1} = Bv_{j-1}, \quad \text{where} \quad B = U^{-T}KU^{-1}D. \tag{48}$$

2) *Compute* $\sigma_{j-1} = w_{j-1}^T t_{j-1}$.

   If $\sigma_{j-1} = 0$, *then stop.*

3) *Set*

$$\omega_{j-1} = \frac{\varrho_{j-1}}{\sigma_{j-1}} \quad \text{and} \quad g_j = g_{j-1} - \omega_{j-1}t_{j-1}.$$

4) *Compute the scalars*

$$\vartheta_j = \frac{g_j^T g_j}{\tau_{j-1}}, \quad \psi_j = \frac{1}{1+\vartheta_j}, \quad \text{and} \quad \tau_j = \tau_{j-1}\vartheta_j\psi_j.$$

   Update the vectors

$$\begin{aligned} c_j &= \psi_j\vartheta_{j-1}c_{j-1} + \psi_j\omega_{j-1}v_{j-1}, \\ z_j' &= z_{j-1}' + c_j. \end{aligned}$$

5) *If* z_j' *has converged, then set*

$$z_j = z_0 + U^{-1}Dz_j',$$

   and stop.

6) *If* $\varrho_{j-1} = 0$, *then stop.*

7) *Set*

$$\begin{aligned} \varrho_j &= g_j^T D g_j, \quad \beta_j = \frac{\varrho_j}{\varrho_{j-1}}, \\ v_j &= g_j + \beta_j v_{j-1}, \\ w_j &= Dv_j. \end{aligned}$$

We note that, as a by-product, Algorithm 5.1 generates the upper bound

$$\left\| d' - Bz'_j \right\| \;=\; \left\| U^{-T}\left(d - Kz_j\right) \right\| \;\leq\; \sqrt{(j+1)}\tau_j$$

for the norm of the residual residual vector $d' - Bz'_j$ associated with the jth iterate z'_j. This upper bound is usually very close to the true residual norm, and we use it as part of the convergence check in Step 5 of Algorithm 5.1.

Finally, we remark that the work and storage requirements of Algorithm 5.1 are essentially the same as those for other existing Krylov-subspace algorithms for symmetric linear systems, such as SYMMLQ and MINRES [28]. However, all these other algorithms can only be used with symmetric positive definite preconditioners M, whereas there are no restrictions on the choice of the symmetric preconditioner M for the symmetric QMR algorithm. If M is symmetric positive definite, then, without loss of generality, one can set $D = I$ in (33) and (35). It follows that $B = U^{-T}KU^{-1}$ is congruent to K, and hence, the preconditioned matrix B and the original matrix K have the same number of positive (negative) eigenvalues. In other words, a symmetric positive definite preconditioner M cannot change the indefiniteness of K, and—when limited to positive definite M—the best one can do is to choose M such that the positive and negative eigenvalues of B are clustered around 1 and -1, respectively; this is the approach considered in [17]. On the other hand, for the symmetric QMR algorithm, we can use arbitrary symmetric indefinite preconditioners M that allow to cluster all eigenvalues of B around 1. This is a definite advantage for symmetric linear systems with highly indefinite matrices, such as the KKT matrix (21).

6. Indefinite preconditioning

We now return to the solution of the special symmetric indefinite systems (25), and from now on, we assume that K is the KKT matrix defined in (21). In this section, we discuss the construction of suitable symmetric indefinite preconditioners for the KKT matrix K.

The first question when deciding on a certain preconditioner for K is whether it might not be advisable after all to apply an iterative solver to the original 3×3-block system (10), rather than the reduced 2×2-block system (25). We also looked at possible preconditioners for the 3×3-block systems, and the only promising strategies we could think of implicitly used a reduction to the 2×2-block system. Therefore, we decided to carry out this reduction beforehand just once rather than at each iteration of the linear systems solver.

6.1 Preconditioning strategies

Next, we discuss four types of indefinite preconditioners for the matrix K.

- The first and simplest strategy for preconditioning K that one might think of is to factor K for some given values of x and s (i.e., for some given diagonal $-SX^{-1}$), and to reuse this factorization for different values of x and s later on. However, our numerical experiments have shown that, typically, after 3 interior-point iterations the old factorization is a bad preconditioner, and it results in a high number of QMR iterations. We did not pursue this approach further.

- A second idea for finding a simple preconditioner is to form the Schur complement $A^T X S^{-1} A$, and to use SSOR [2] as a preconditioner, say M_0, for this positive definite matrix. It is then easy to construct from M_0 a symmetric indefinite preconditioner M for the KKT matrix K. However, since this Schur complement is inherently unstable, it was not surprising to observe that this preconditioner worked quite well only in the first interior-point iterations (when x and s are large), but deteriorated in the later stages of the interior-point algorithm. Another disadvantage of this approach is that in the Schur complement, considerable fill-in may occur. We did not pursue this approach either.

- A more sophisticated strategy for constructing a preconditioner is to compute an approximate incomplete version of the Bunch-Parlett factorization [3] of K. This strategy seems best suited to approximate K. However, the construction of an efficient incomplete Bunch-Parlett factorization is fairly complicated, and it is beyond the scope of this paper.

- The last strategy, which we list here, might form a reasonable compromise between simplicity and effectiveness. It is based on extending the SSOR preconditioner to indefinite matrices, as described next.

6.2 A modified SSOR preconditioner

We now describe our modification of the SSOR preconditioner to the indefinite matrix K in detail. If K were positive definite, the SSOR preconditioner would simply consist of decomposing K as $K = L + D + L^T$, where L is a strictly lower triangular matrix and D is a diagonal matrix, and then using the matrix

$$M = (L + D)D^{-1}(L^T + D)$$

as a preconditioner for K; see [2]. Since

$$M = K + LD^{-1}L^T,$$

the approximation $M \approx K$ is better the smaller $LD^{-1}L^T$ is. For the indefinite matrix K of (21), however, the last m diagonal entries of D are all zero, and therefore, a direct application of the above approximation is impossible.

Instead, we first permute K symmetrically to obtain $\hat{K} = \Pi^T K \Pi$, and then approximate the matrix $\hat{K}$ in a similar fashion as the SSOR preconditioner for positive definite K, namely by identifying a block-diagonal matrix D from $\hat{K}$ with either 1×1 or 2×2 blocks and using the decomposition

$$\hat{K} = \Pi^T K \Pi = L + D + L^T, \tag{49}$$

where L is strictly lower triangular. The resulting SSOR-type preconditioner for K is then given by

$$M = U^T D^{-1} U, \quad \text{where} \quad U = \left(L^T + D\right) \Pi^T. \tag{50}$$

The main difficulty lies in determining the 2×2 blocks in the initial matrix K. Once these are known, it is straightforward to construct some permutation Π that orders the 2×2 blocks about the diagonal of $\Pi^T K \Pi$.

From the structure of the matrix K, it follows immediately that the only possibility for constructing an invertible block-diagonal matrix D with either 1×1 or 2×2 blocks from the diagonal of $\Pi^T K \Pi$ is to choose exactly m blocks of size 2×2 and $n-m$ blocks of size 1×1. All 1×1 blocks of course are elements of the diagonal matrix $-SX^{-1}$. The 2×2 blocks are uniquely determined by choosing for each of the remaining m elements of $-SX^{-1}$ one (large) pivot in the corresponding column of A such that each row of A contains exactly one pivot. Thus, each jth diagonal element of $-XS^{-1}$ is either a 1×1-pivot or part of a 2×2-pivot. In the latter case, there is some row index $i = i(j)$, with $1 \leq i \leq m$, such that $a_{i,j} \neq 0$ and

$$\begin{bmatrix} -s_j x_j^{-1} & a_{i,j} \\ a_{i,j} & 0 \end{bmatrix}$$

is a 2×2-pivot. Here and in the sequel, we use the notation $a_{i,j}$ for the entries of the matrix $A = [\, a_{i,j} \,]$. Since each pivot row is unique, it follows that $i(j) \neq i(j')$ for $j \neq j'$.

Ideally, the choice of the 2×2 blocks is to be carried out such that $\|LD^{-1}L^T\|$ is as small as possible. We approximate this goal by minimizing the product of the largest eigenvalues of the (1×1 or 2×2) blocks of D^{-1} over all possible choices of 2×2 blocks. The problem of finding the best-possible choice of 2×2 pivots can be written as a weighted bipartite matching problem.

Let G be an undirected graph with vertex set $\mathcal{V}_1 \cup \mathcal{V}_2$ and edge set $\mathcal{E} \subset \mathcal{V}_1 \times \mathcal{V}_2$ given as follows. The set $\mathcal{V}_1 = \{\, 1, 2, \ldots, n \,\}$ corresponds to the diagonal entries of $-XS^{-1}$. The set $\mathcal{V}_2$ is defined as $\mathcal{V}_2 = \{\, \phi_1, \phi_2, \ldots, \phi_n \,\}$, where $\phi_1, \phi_2, \ldots, \phi_m$ correspond to rows $1, 2, \ldots, m$ of A, and $\phi_{m+1}, \phi_{m+2}, \ldots, \phi_n$ are "dummys" corresponding to 1×1 blocks in D. Each vertex k in $\mathcal{V}_1$ is connected to each vertex in $\{\, \phi_{m+1}, \phi_{m+2}, \ldots, \phi_n \,\}$ by an edge with weight $\log(s_k/x_k)$, which is just the logarithm of the norm of the inverse of $-x_k s_k^{-1}$. These edges represent 1×1-pivots. Further, each vertex k in $\mathcal{V}_1$ is connected with some $\phi_l \in \mathcal{V}_2$ for $l \leq m$ if $a_{l,k} \neq 0$. In this case, the weight of the edge is

$$\log \left(\frac{x_k/s_k + \sqrt{4a_{l,k}^2 + x_k^2/s_k^2}}{2a_{l,k}^2} \right),$$

which is just the logarithm of the inverse of the smaller eigenvalue of the corresponding 2×2 block in D. The solution of the weighted bipartite matching problem minimizes the sum of the weights, or, equivalently, the product of the norms of the inverses of the diagonal blocks. We note that it is not essential to know the exact minimum of the weighted bipartite matching problem, and instead, a heuristic that finds a nearly optimal solution is sufficient of course. Furthermore, slight changes in the data x and s in the course of the interior-point algorithm are likely to result in a matching problem for which the (nearly) optimal solution of the previous problem is a good starting point. The plots of our numerical experiments in the next section result from a simple greedy heuristic for approximately solving the weighted bipartite matching problem.

This greedy heuristic is as follows. Starting from the smallest diagonal element (in absolute value) of $-XS^{-1}$, we assign the largest element in the corresponding column of A to form a 2×2-pivot and then cancel the corresponding row of A. This is repeated until all rows of A are canceled. This strategy, however, is too simple in general, and it is easy to construct examples in which it might break down. A more elaborate heuristic, which is guaranteed not to break down, is described in our forthcoming paper [11].

6.3 Eisenstat's trick for modified SSOR

In this section, we show that for the modified SSOR preconditioner M defined by (50) and (49), the matrix-vector products (48) in the symmetric QMR Algorithm 5.1 can be computed very efficiently, using the so-called Eisenstat trick [4].

First, by (49) and (50), we can rewrite the matrix $B = U^{-T}KU^{-1}D$ of the preconditioned system (34) as follows:

$$\begin{aligned} B &= U^{-T}\Pi(L + D + L^T)\Pi^T U^{-1}D \\ &= (L + D)^{-1}(L + D + L^T)(L^T + D)^{-1}D \\ &= \left(I + (L + D)^{-1}(L^T + D - D)\right)(L^T + D)^{-1}D \\ &= (L^T + D)^{-1}D + (L + D)^{-1}\left(D - D(L^T + D)^{-1}D\right). \end{aligned} \tag{51}$$

Setting $w_{j-1} = Dv_{j-1}$ and using (51), it follows that the matrix-vector product $t_{j-1} = Bv_{j-1}$ in Step 1 of Algorithm 5.1 is given by

$$\begin{aligned} \hat{t} &= (L^T + D)^{-1}w_{j-1}, \\ t_{j-1} &= \hat{t} + (L + D)^{-1}\left(w_{j-1} - D\hat{t}\right). \end{aligned} \tag{52}$$

Recall that the vector w_{j-1} is available in Algorithm 5.1 anyway, and therefore, by using (52), the computation of the matrix-vector product Bv_{j-1} requires two triangular solves and one multiplication with D. Thus, the multiplication by $\Pi^T K\Pi$ that would occur in the direct computation

$$Bv_{j-1} = (L + D)^{-1}\Pi^T K\Pi(L^T + D)^{-1}Dv_{j-1}$$

has been eliminated. Since one triangular solve is about half the cost of a multiplication by K, the overall work per QMR step is reduced by nearly 50%.

7. Numerical experiments

To illustrate some of the features of the approach proposed in this paper, we report numerical results for two examples (SC105 and ISRAEL) from the NETLIB test set [16]. In both cases, we applied the symmetric QMR Algorithm 5.1 (without look-ahead) using the SSOR preconditioner with the simple greedy heuristic described at the end of Section 6.2.

The first three figures refer to Example SC105. Our interior-point method took a total of 18 iterations for this example. In Figure 7.1, we show the convergence behavior of the QMR method applied to one of the KKT systems that arose within our interior-point method. In Figures 7.2 and 7.3, we show the corresponding eigenvalue distributions for the coefficient matrix K of the original 2×2-block system and for the preconditioned matrix B defined in (35), respectively. Note that the preconditioned matrix has almost all eigenvalues with positive real part, and the highly indefinite structure is nearly eliminated. Furthermore, except for a few outliers that QMR easily eliminates in the first few iterations, the eigenvalues are nicely clustered around 1. As mentioned in Section 5.3, this clustering explains the fast convergence of the QMR method for this example.

The next three figures refer to Example ISRAEL. Our interior-point method took a total of 48 iterations for this example. In Figure 7.4, we show the convergence behavior of the QMR method for the solution of a typical KKT system that arose within our interior-point method. We can observe a rapid convergence in the first QMR iterations, thereafter, convergence slows down. While the

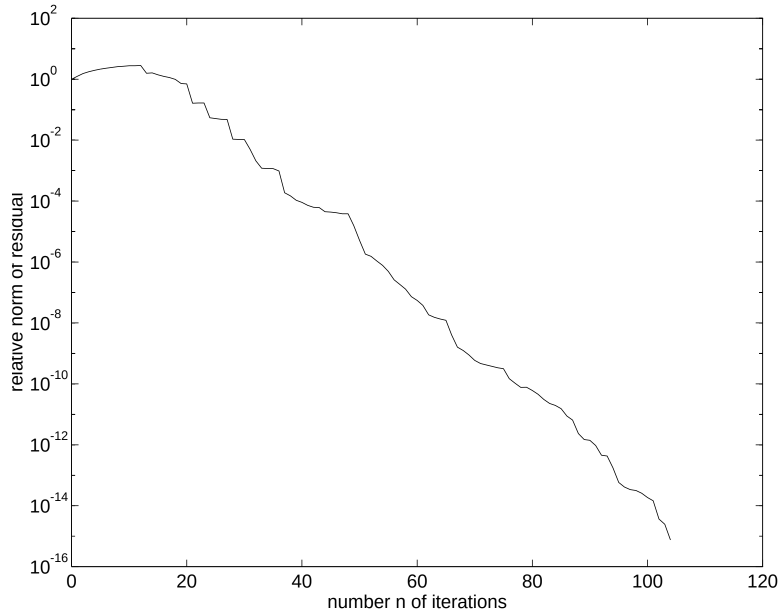

Fig. 7.1. Relative norm of residuals for Example SC105.

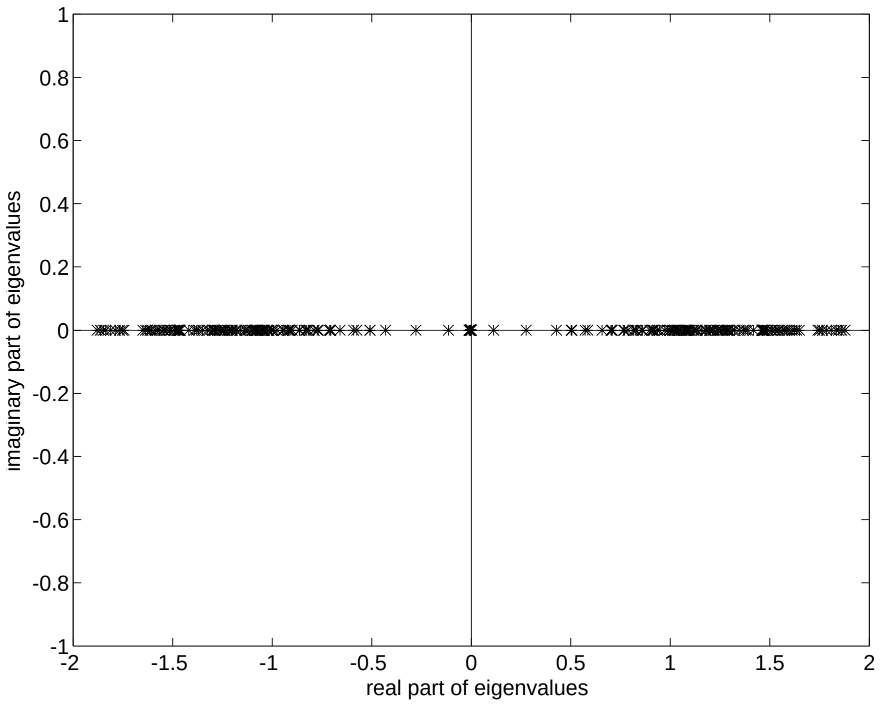

Fig. 7.2. Eigenvalues of the original KKT matrix for Example SC105.

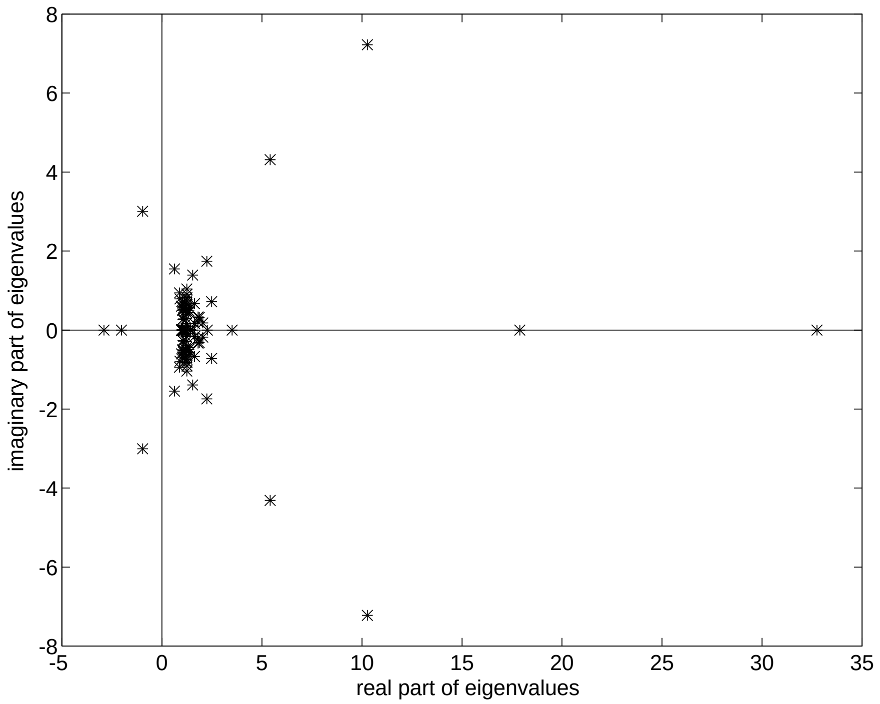

Fig. 7.3. Eigenvalues of the preconditioned KKT matrix for Example SC105.

use of a more sophisticated heuristic for solving the weighted bipartite matching problem and the incorporation of look-ahead into the QMR method might improve the convergence behavior also for this example, we point out that a reduction of the norm of the residual by a factor 10^{-3} after just a few QMR iterations for this example might already satisfy the moderate accuracy condition of Section 3.3.

Finally, in Figures 7.5 and 7.6, we display the sparsity structure of the original and the symmetrically permuted KKT matrix for Example ISRAEL. The 1×1- or 2×2-diagonal blocks of the reordered KKT matrix are all nonsingular. The lower part of the permuted KKT matrix forms our SSOR preconditioner, and, of course, there is no fill-in for this preconditioner. However, in a numerical implementation, it may be most efficient to store both A and A^T in sparse format, and to construct D and L in sparse format from these two matrices. In this case, the additional storage needed is twice the storage for A, and in this sense, one may think of the SSOR preconditioner as creating fill-in by a factor two.

8. Concluding remarks

We presented an interior-point algorithm that uses a preconditioned QMR method to approximately solve the linear KKT systems at each interior-point iteration. Based on a simple reduction introduced in Section 4.2, we derived a modified SSOR preconditioner for these symmetric indefinite KKT systems. One of the features of our approach is a controllable amount of fill-in by a factor two, independent of the particular problem. A further feature when applying an iterative scheme

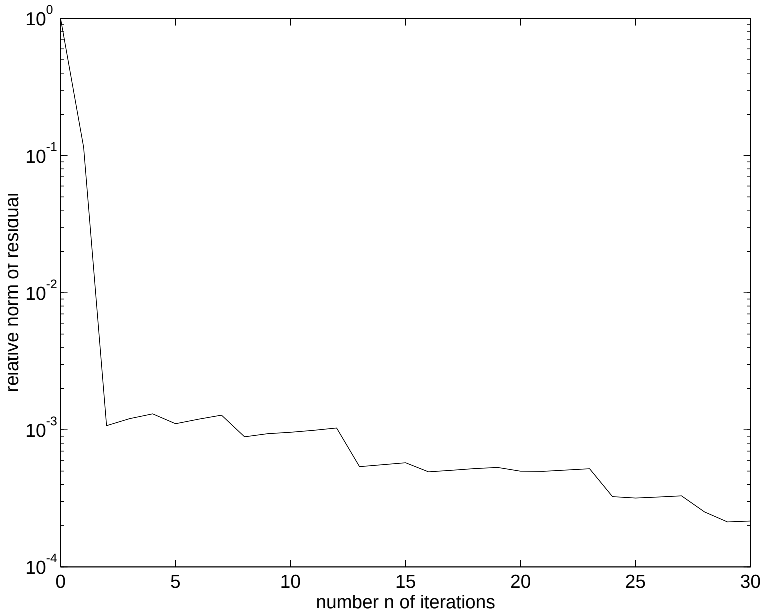

Fig. 7.4. Relative norm of residuals for Example ISRAEL.

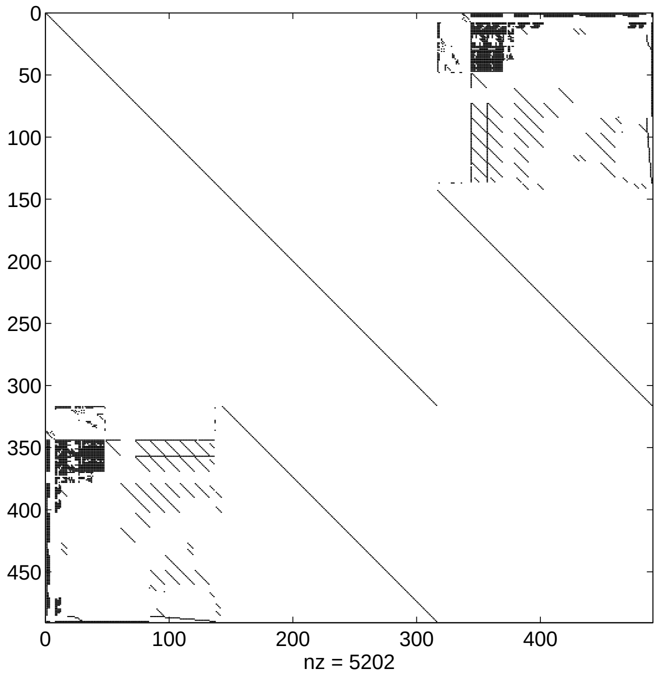

Fig. 7.5. Sparsity structure of the original KKT matrix for Example ISRAEL.

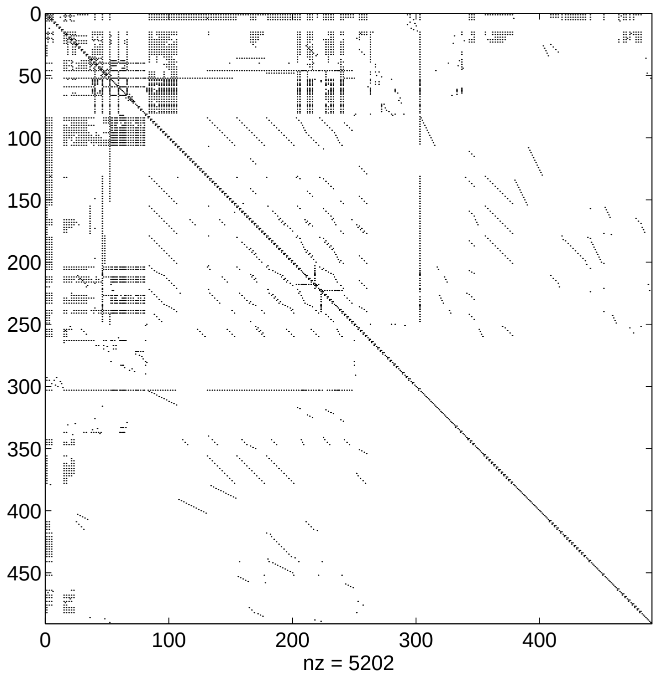

Fig. 7.6. Sparsity structure of the reordered KKT matrix for Example ISRAEL.

to the indefinite system—rather than to the standard positive definite Schur complement—is a low relative accuracy that is sufficient when solving the linear systems at each iteration, as well as a natural treatment of free variables.

These features contrast the standard approach of solving systems with the positive definite Schur complement $AXS^{-1}A^T$ as coefficient matrix. Especially in the final stages of the algorithm, the Schur-complement systems must be solved to very high accuracy to obtain primal and dual feasibility. Fill-in may occur during the computation of both $AXS^{-1}A^T$ and its factorization. The actual amount of fill-in is dependent on the linear program to be solved, and it is hard to predict. Ordering schemes for the factorization are necessary, and modifications like splitting of dense columns may also be needed to reduce fill-in. Also, free variables require a special treatment, such as splitting into two nonnegative variables. Finally, the Schur complement is often worse conditioned than the full KKT system. Nevertheless, in many numerical examples (in fact in almost all NETLIB problems) the factorization of the Schur complement is very cheap, and only for those problems that create substantial fill-in in the Schur complement do we expect a QMR-based interior-point method to be superior.

A final comparison of the two approaches will certainly depend on the running times of our algorithm. Such a comparison will be presented in our forthcoming paper [11].

Acknowledgements

We would like to express our gratitude to Jörg Menniken for making his interior-point code [26] available to us; we have used parts of his program continuously in our numerical implementations. Part of this work was done while the second author was visiting the Institute of Statistical Mathematics in Tokyo, Japan, and he would like to thank Shinji Mizuno for his warm hospitality as well as for helpful discussions about this paper. The second author is also grateful to Masakazu Kojima for his help and for the office and computing facilities at the Tokyo Institute of Technology. The authors are grateful to the two anonymous referees for their helpful and constructive comments that greatly improved the presentation of the paper.

References

[1] I. Adler, N.K. Karmarkar, M.G.C. Resende and G. Veiga, "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming* 44 (1989) 297–335.

[2] O. Axelsson, "A survey of preconditioned iterative methods for linear systems of algebraic equations," *BIT* 25 (1985) 166–187.

[3] J.R. Bunch and B.N. Parlett, "Direct methods for solving symmetric indefinite systems of linear equations," *SIAM Journal on Numerical Analysis* 8 (1971) 639–655.

[4] S.C. Eisenstat, "Efficient implementation of a class of preconditioned conjugate gradient methods," *SIAM Journal on Scientific and Statistical Computing* 2 (1981) 1–4.

[5] R. Fourer and S. Mehrotra, "Solving symmetric indefinite systems in an interior-point method for linear programming," *Mathematical Programming* 62 (1993) 15–39.

[6] R.W. Freund, "Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations," in: D. Braess and L. Schumaker, eds., *Numerical Methods of Approximation Theory* (Birkhaüser, Basel, 1992) pp. 77–95.

[7] R.W. Freund, "Transpose-free quasi-minimal residual methods for non-Hermitian linear systems," in: G. Golub, A. Greenbaum, and M. Luskin, eds., *Recent Advances in Iterative Methods*, The IMA Volumes in Mathematics and its Applications, Vol. 60 (Springer-Verlag, New York, 1994) pp. 69–94.

[8] R.W. Freund, G.H. Golub and N.M. Nachtigal, "Iterative solution of linear systems," *Acta Numerica* 1 (1992) 57–100.

[9] R.W. Freund, M.H. Gutknecht and N.M. Nachtigal, "An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices," *SIAM Journal on Scientific Computing* 14 (1993) 137–158.

[10] R.W. Freund, F. Jarre and S. Mizuno, "Convergence of a class of inexact interior-point algorithms for linear programs," Numerical Analysis Manuscript 96–16, Bell Laboratories (Murray Hill, NJ, 1996).

[11] R.W. Freund and F. Jarre, "Numerical experiences with a QMR-based interior-point algorithm for linear programs," AT&T Numerical Analysis Manuscript, Bell Laboratories (Murray Hill, NJ, 1996), in preparation.

[12] R.W. Freund and N.M. Nachtigal, "QMR: a quasi-minimal residual method for non-Hermitian linear systems," *Numerische Mathematik* 60 (1991) 315–339.

[13] R.W. Freund and N.M. Nachtigal, "An implementation of the QMR method based on coupled two-term recurrences," *SIAM Journal on Scientific Computing* 15 (1994) 313–337.

[14] R.W. Freund and T. Szeto, "A transpose-free quasi-minimal residual squared algorithm for non-Hermitian linear systems," in: R. Vichnevetsky, D. Knight, and G. Richter, eds., *Advances in Computer Methods for Partial Differential Equations – VII* (IMACS, New Brunswick, 1992) pp. 258–264.

[15] R.W. Freund and H. Zha, "Simplifications of the nonsymmetric Lanczos process and a new algorithm for Hermitian indefinite linear systems," AT&T Numerical Analysis Manuscript, Bell Laboratories (Murray Hill, NJ, 1995).

[16] D.M. Gay, "Electronic mail distribution of linear programming test problems," *Mathematical Programming Society COAL Newsletter* 13 (1985) 10–12.

[17] P.E. Gill, W. Murray, D.B. Ponceleón and M.A. Saunders, "Preconditioners for indefinite systems arising in optimization," *SIAM Journal on Matrix Analysis and Applications* 13 (1992) 292–311.

[18] M.R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards* 49 (1952) 409–436.

[19] N.K. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica* 4 (1984) 373–395.

[20] N.K. Karmarkar and K.G. Ramakrishnan, "Computational results of an interior point algorithm for large scale linear programming," *Mathematical Programming* 52 (1991) 555–586.

[21] M. Kojima, N. Megiddo and S. Mizuno, "A primal-dual infeasible-interior-point algorithm for linear programming," *Mathematical Programming* 61 (1993) 263–280.

[22] M. Kojima, S. Mizuno and A. Yoshise, "A primal-dual interior point algorithm for linear programming," in: N. Megiddo, ed., *Progress in Mathematical Programming, Interior-Point and Related Methods* (Springer-Verlag, New York, 1989) pp. 29–47.

[23] I.J. Lustig, R.E. Marsten and D.F. Shanno, "Computational experience with a primal-dual interior point method for linear programming," *Linear Algebra and Its Applications* 152 (1991) 191–222.

[24] I.J. Lustig, R.E. Marsten and D.F. Shanno, "On implementing Mehrotra's predictor-corrector interior-point method for linear programming," *SIAM Journal on Optimization* 2 (1992) 435–449.

[25] N. Megiddo, "Pathways to the optimal set in linear programming," in: N. Megiddo, ed., *Progress in Mathematical Programming, Interior-Point and Related Methods* (Springer-Verlag, New York, 1989) pp. 131–158.

[26] J. Menniken, "Implementation of a primal-dual path following method for solving large linear programs," Dissertation, Fakultät für Mathematik und Informatik, Universität Würzburg (Würzburg, Germany, 1991).

[27] R.D.C. Monteiro and I. Adler, "Interior path following primal-dual algorithms, Part I: Linear programming," *Mathematical Programming* 44 (1989) 27–41.

[28] C.C. Paige and M.A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM Journal on Numerical Analysis* 12 (1975) 617–629.

[29] K. Tanabe, "Centered Newton method for mathematical programming," in: M. Iri and K. Yajima, eds., *System Modelling and Optimization* (Springer, New York, 1988) pp. 197–206.

[30] R.J. Vanderbei and T.J. Carpenter, "Symmetric indefinite systems for interior point methods," *Mathematical Programming* 58 (1993) 1–32.

[31] X. Xu, P.F. Hung and Y. Ye, "A simplified homogeneous and self-dual linear programming algorithm and its implementation," Working Paper, College of Business Administration, The University of Iowa (Iowa City, IA, 1993).