Properties and Computational Issues of a Preconditioner for Interior Point Methods *

Venansius Baryamureeba† Trond Steihaug‡

Department of Informatics
University of Bergen
N-5020 Bergen
Norway
November 1, 1999

Abstract

This is a collection of four conference proceedings on scientific computation. In the proceedings, we discuss solving a sequence of linear systems arising from the application of an interior point method to a linear programming problem. The sequence of linear systems is solved by alternating between a direct and an iterative method. The preconditioner is based on low-rank modifications of the coefficient matrix where a direct solution technique has been used. We compare two different techniques of forming the low-rank modification matrix; namely one by Wang and O'Leary [11] and the other by Baryamureeba, Steihaug and Zhang [3]. The theory and numerical testing strongly support the latter. We derive a sparse algorithm for modifying the Cholesky factors by a low-rank matrix, discuss the computational issues of this preconditioner, and finally give numerical results that show the approach of alternating between a direct and an iterative method to be promising.

Key Words. Linear Programming, Preconditioner, Cholesky factorization, Sparse algorithm, modifications, Interior point methods.

1 Introduction

Consider the primal linear program

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to:} & Ax = b \\ & x \geq 0 \end{array}$$

and its dual program

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to:} & A^T y + z = c \\ & z \geq 0 \end{array}$$

where $A \in \Re^{m \times n}$ $(m \leq n)$ has full rank m, $b \in \Re^m$ and $c \in \Re^n$. The optimality conditions for the above linear program pair can be written as a system of equations plus non-negativity constraints,

$$F(x, y, z) = \begin{pmatrix} Ax - b \\ A^T y + z - c \\ XZe \end{pmatrix} = 0, \quad (x, z) \geq 0,$$

where $X = \text{diag}(x)$, $Z = \text{diag}(z)$ and $e = (1, \ldots, 1)^T$ is an n-vector of all ones.

*This is Technical Report No. 180, Department of Informatics, University of Bergen, N-5020 Bergen, Norway
†E-mail address: barya@ii.uib.no
‡E-mail address: trond@ii.uib.no

The basic framework of primal-dual interior point methods can be viewed as applying Newton's method to the so-called perturbed Karush-Kuhn-Tucker conditions

$$F_\mu(x,y,z) = \begin{pmatrix} Ax-b \\ A^Ty+z-c \\ XZe-\mu e \end{pmatrix} = 0, \quad (x,z) > 0, \tag{1}$$

for a sequence of positive μ values that decreases to zero, while keeping x and z positive. The parameter μ is called the centering (or barrier) parameter, and the set of triples (x,y,z) that satisfy (1) for all $\mu > 0$ is called the (primal-dual) central path.

We will mainly consider primal-dual interior point algorithms based on this approach for linear programming, although the results in this paper can be applied to most interior point algorithms. At a given iterate (x,y,z), where $x, z > 0$, and for a fixed value of μ, the linear equation that defines a Newton step for (1) is

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = -\begin{pmatrix} Ax-b \\ A^Ty+z-c \\ XZe-\mu e \end{pmatrix}. \tag{2}$$

The next iterate is obtained by the update

$$(x,y,z) \leftarrow (x,y,z) + \alpha(\Delta x, \Delta y, \Delta z)$$

where the step length α is chosen to ensure that the new x and z remain positive.

The most widely-used approach for solving (2) is to use block elimination to reduce it to a smaller system

$$(AGA^T)\Delta y = AG(c - A^Ty - \mu X^{-1}e) + (b - Ax) \tag{3}$$

followed by a Cholesky factorization, $G = Z^{-1}X$ is a positive definite diagonal matrix. After Δy is determined from solving the above linear system, Δx and Δz can then be easily computed. Hence, in this approach, the primary computation at each iteration is to solve (3).

It is worth noting that when $b = A\tilde{x}$ for some $\tilde{x}$, system (3) can be written as

$$(AGA^T)\Delta y = AGh \tag{4}$$

where $h = (c - A^Ty - \mu X^{-1}e) + G^{-1}(\tilde{x} - x)$.

Let k denote the interior point iteration (step). The for $k = 0, 1, 2 \ldots$, we solve the linear system

$$(AG^kA^T)\Delta y = AG^kh^k. \tag{5}$$

Thus, we can say that the primary computation for interior point algorithms based on the Newton's method approach is to solve a sequence of normal equations (5) where the weight G and the right-hand side h vary from iteration to iteration. Also in other interior point methods like the predictor-corrector method, the normal equation linear system can be written in the form (5).

The idea of using low-rank modifications (updates or downdates) is not new. In his seminal paper on the first polynomial-time interior point method for linear programming, Karmarkar [6] proposed to use low-rank modifications to decrease the theoretical complexity bounds. His idea has been pursued by many later papers for the similar reason. On the other hand, a combination of a direct method and an iterative method was reported by Karmarkar and Ramakrishnan [8].

Wang and O'Leary [11] propose a strategy of preconditioning the normal equation system based on low-rank modifications. They also propose a dynamic approach based on the performance of the preconditioner. However, they base the decision of the low-rank modifications on major changes between the current diagonal weight matrix G and the diagonal weight matrix for which there exists a Cholesky factorization. On the other hand, Baryamureeba, Steihaug and Zhang [3] give both theoretical and numerical justification of a different criteria for the low-rank modifications.

2 Discussion of the Proceedings

The main issue for this type of preconditioner is on how to choose a low-rank matrix that is used to modify the factorized matrix. For rapid convergence for conjugate gradient type methods, either the spectral condition number of the preconditioned matrix is close to one or the eigenvalues of the preconditioned matrix are clustered. Thus it is necessary to know the eigenvalues of the preconditioned matrix or a lower and an upper bound for these eigenvalues. Next we need to know how the preconditioner performs on the corresponding linear system. Let G be the current diagonal matrix and H be the one for which we have a Cholesky factorization $AHA^T = LDL^T$. For any n-by-n matrix X, let X_{jj} denote the j-th diagonal element of X. Wang and O'Leary [11, 10] base the low-rank modifications on the largest values of $|G_{jj} - H_{jj}|$. Baryamureeba, Steihaug and Zhang [3] suggest that the low-rank modifications be based on the magnitude of G_{ii}/H_{ii} instead of $|G_{ii} - H_{ii}|$. In Baryamureeba and Steihaug [1] a comparative study on these two techniques for choosing the low-rank modifications is done. In particular, we carry out two sets of numerical experiments. One set is on the eigenvalue distributions. For the second set, we implement a modified preconditioned conjugate gradient least squares algorithm and apply it on the positive definite linear system arising at a given interior point iteration. The theory and numerical experiments strongly support the technique by Baryamureeba, Steihaug and Zhang [3].

In Baryamureeba and Steihaug [2] we discuss two approaches of solving the linear system with the preconditioner as coefficient matrix. In particular, we discuss the Sherman Morrison-Woodbury formula approach and the approach based on modifying the Cholesky factors. When we alternate for every step between a direct and an iterative method while solving a sequence of linear systems of the form (5), then Sherman Morrison-Woodbury formula approach has been found to work well. But when we want to reuse the factorized matrix at more than one step by simply carrying out low-rank modifications, then it may be cheaper to modify the Cholesky factors. In this case, we derive a sparse algorithm [2] that can be used to modify the Cholesky factors by a low-rank matrix. Finally we give numerical results on the computational issues of this preconditioner.

In [3, 5] we apply the preconditioner to linear programming problems. A mixed primal-dual method where (5) is solved by a direct method for every even step and an iterative method (PCGLS) for every odd step is proposed. We implement a modified preconditioned conjugate gradient least squares (PCGLS) algorithm to solve the linear system at every odd step. In this implementation, we employ the Sherman Morrison-Woodbury formula approach. Numerical results with the mixed method are given. The results by the mixed method approach are promising and the approach merits further study.

In the proceedings, we only considered linear programming problems. Here, we note that apart from interior point methods for linear programming, the sequence of problems of the form (5) has applications in electrical networks, elliptic boundary value problems, structural analysis, and recursive robust linear regression.

References

[1] V. Baryamureeba and T. Steihaug, *On a class of preconditioners for interior point methods*, The Proceedings for the 6th Meeting of the Nordic Section of the Mathematical Programming Society (Nordic MPS'99), Västerås, Sweden, September 25-26, 1999, to appear.

[2] V. Baryamureeba and T. Steihaug, *Computational issues for a new class of preconditioners,* VIEWEG Series Notes on Numerical Fluid Mechanics (The Proceedings for the 2nd Workshop on Large-Scale Scientific Computations, Bulgaria, 1999), to appear.

[3] V. Baryamureeba, T. Steihaug and Y. Zhang, *Properties of a class of preconditioners for weighted least squares problems,* Technical Report No. 170, Department of Informatics, University of Bergen, 5020 Bergen, Norway, April 30, 1999 (Revised July 6, 1999).

[4] V. Baryamureeba, T. Steihaug and Y. Zhang, *Application of a class of preconditioners to large scale linear programming problems*, P. Amestoy, P. Berger, M. Dayd, I. Duff, V. Frayss, L. Giraud and D. Ruiz, (Eds), Euro-Par'99 Parallel Processing. Lecture Notes in Computer Science, No. 1685, Springer-Verlag, pp. 1044 -1048, 1999.

[5] V. Baryamureeba and T. Steihaug, *Application of a new class of preconditioners to large-scale linear programming problems*, Presented at "The 1999 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Industrial Applications", Minneapolis, Minnesota, USA, June 10-12, 1999.

[6] N. Karmarkar, *A new polynomial time algorithm for linear programming.* Combinatorica, Vol. 4, pp. 373-395, 1984.

[7] N.K. Karmarkar and K.G. Ramakrishnan, *Computational results of an interior point algorithm for large-scale linear programming,* Mathematical Programming, Vol. 52, pp. 555-586, 1991.

[8] W. Wang and D.P. O'Leary, *Adaptive use of iterative methods in interior point methods for linear programming,* Computer Science Department Report CS-TR-3560, Institute for Advanced Computer Studies Report UMIACS-95-111, University of Maryland, November 1995.

[9] W. Wang and D.P. O'Leary, *Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming*, Computer Science Department Report CS-TR-4011, Institute for Advanced Computer Studies Report UMIACS-TR-99-21, University of Maryland, April 1999.

On a Class of Preconditioners for Interior Point Methods

Venansius Baryamureeba   Trond Steihaug

Presented at the 6th Meeting of the Nordic Section of the
Mathematical Programming Society (Nordic MPS'99),
Västerås, Sweden,
September 25-26, 1999.

On a Class of Preconditioners for Interior Point Methods [1]

Venansius Baryamureeba Trond Steihaug

Department of Informatics
University of Bergen
5020 Bergen Norway

Abstract

In most primal-dual interior point methods, a sequence of weighted normal equations are solved, where the only change from one problem to the next are the weights and the right hand side. Solving the normal equations alternating between a direct method and an iterative method was introduced in Wang and O'Leary [10]. A class of preconditioners based on a low-rank correction of a Cholesky factorization of a weighted normal equation coefficient matrix was used in [10] and the low-rank correction was based on the difference of the weights. In Baryamureeba, Steihaug and Zhang [3] it was shown that the low-rank correction should be based on the ratios of the weights to reduce the condition number. The purpose of this paper is to compare the approach in forming the low-rank preconditioners in [3] with the approach by Wang and O'Leary [10, 11]. The theory and numerical testing strongly support the approach by Baryamureeba, Steihaug and Zhang [3].

Key Words. Weighted linear least squares, Preconditioners, Preconditioned conjugate gradient for least squares, Linear programming, Primal-dual infeasible interior point algorithms.

A Introduction

An interior point algorithm solves the linear programming problem by generating a sequence of interior points from an initial interior point. At every interior point iteration weighted linear least squares problems are solved. The class of preconditioners we consider is based on a low-rank correction of a factorization of a weighted normal equation coefficient matrix. The condition number of the preconditioned coefficient matrix can be kept below a prescribed constant for special choices of the low-rank corrections. The numerical results in Section 5 compare two approaches of forming the low-rank preconditioners. One approach is by Wang and O'Leary [10, 11]. In Baryamureeba, Steihaug and Zhang [3] it was shown that the low-rank correction should be based on the ratios of the weights.

In Section B we formulate the linear system arising from the application of an interior point method to a linear programming problem as a weighted least squares problem. In Section C we discuss the low-rank preconditioner and its computational aspects. Section D discusses the iterative solver, and in Section E we give numerical results.

Throughout this paper we use the following notation. $\min_i$ or $\max_i$ is for all i for which the argument is defined. For any matrix A, A_{ij} is the element in the i-th row and j-th column, and A_j is the j-th column. The vector $e = (1, \ldots, 1)^T$ is an n-vector of all ones. For any square m-by-m matrix X, $\lambda_i(X)$ are the eigenvalues of X arranged in non-decreasing order, $\lambda_1(X) \leq \lambda_2(X) \leq \cdots \leq \lambda_m(X)$. The letter L represents the lower Cholesky factor.

B Problem

Consider the primal linear program

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to:} & Ax = b \\ & x \geq 0 \end{array}$$

and its dual program

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to:} & A^T y + z = c \\ & z \geq 0 \end{array}$$

[1]Proceedings of the 6th Meeting of the Nordic Section of the Mathematical Programming Society (Nordic MPS'99), Västerås, Sweden, September 25-26, 1999, to appear.

where $A \in \Re^{m\times n}$ $(m \leq n)$ has full rank m, $b \in \Re^m$ and $c \in \Re^n$. The optimality conditions for the above linear program pair can be written as a system of equations plus non-negativity constraints,

$$F(x,y,z) = \begin{pmatrix} Ax-b \\ A^Ty+z-c \\ XZe \end{pmatrix} = 0, \quad (x,z) \geq 0,$$

where $X = \operatorname{diag}(x)$, and $Z = \operatorname{diag}(z)$.

The basic framework of primal-dual interior-point methods can be viewed as applying Newton's method to the so-called perturbed Karush-Kuhn-Tucker conditions

$$F_\mu(x,y,z) = \begin{pmatrix} Ax-b \\ A^Ty+z-c \\ XZe-\mu e \end{pmatrix} = 0, \quad (x,z) > 0, \tag{1}$$

for a sequence of positive μ values that decreases to zero, while keeping x and z positive. The parameter μ is called the centering (or barrier) parameter, and the set of triples (x,y,z) that satisfy (1) for all $\mu > 0$ is called the (primal-dual) central path. References on primal-dual interior-point methods include [5, 8, 12], for example.

We will mainly consider primal-dual interior-point algorithms based on this approach for linear programming, although the results in this paper can be applied to most interior-point algorithms. At a given iterate (x,y,z), where $x, z > 0$, and for a fixed value of μ, the linear equation that defines a Newton step for (1) is

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = - \begin{pmatrix} Ax-b \\ A^Ty+z-c \\ XZe-\mu e \end{pmatrix}. \tag{2}$$

The next iterate is obtained by the update

$$(x,y,z) \leftarrow (x,y,z) + \alpha(\Delta x, \Delta y, \Delta z)$$

where the step length α is chosen to ensure that the new x and z remain positive.

The most widely-used approach for solving (2) is to use block elimination to reduce it to a smaller system

$$(AGA^T)\Delta y = AG(c - A^Ty - \mu X^{-1}e) + (b - Ax) \tag{3}$$

followed by a Cholesky factorization, where $G = Z^{-1}X$ is a positive definite diagonal matrix. After Δy s determined from solving the above linear system, Δx and Δz can then be easily computed. Hence, in this approach, the primary computation at each iteration is to solve (3).

It is worth noting that when $b = A\tilde{x}$ for some $\tilde{x}$, system (3) can be written as

$$(AGA^T)\Delta y = AGh \tag{4}$$

where $h = (c - A^Ty - \mu X^{-1}e) + G^{-1}(\tilde{x} - x)$. The equation (4) is the normal equations for the weighted least squares problem

$$\min_{\Delta y} \|A^T\Delta y - h\|_G^2, \tag{5}$$

with the weighted norm $\|\cdot\|_G \equiv \|G^{1/2}(\cdot)\|_2$. For this reason, we can say that the primary computation for interior-point algorithms based on the Newton's method approach is to solve a sequence of weighted least squares problems (5) where the weights G and the right-hand side h vary from iteration to iteration. Also in other interior point methods like the predictor-corrector method, the normal equation linear system can be written in the form (5).

In this paper, we consider solving (4) by a preconditioned conjugate gradient method. The preconditioner is based on a low-rank correction of a matrix AHA^T where (i) H is a positive definite diagonal matrix from a previous iteration and is not "vastly different" from G; (ii) the Cholesky factorization of AHA^T is already available.

C The class of preconditioners

Let $G, H \in \Re^{n\times n}$ be positive definite diagonal matrices. For a given index set $\mathcal{Q} \subseteq \{1 \le j \le n | \; G_{jj} \neq H_{jj}\}$, let the $n \times n$ diagonal matrices D and K be given, respectively, by

$$D_{jj} = \begin{cases} G_{jj} - H_{jj}, & \text{if } j \in \mathcal{Q}, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$K = H + D. \tag{6}$$

Note that $K_{jj} = G_{jj}$ if $j \in \mathcal{Q}$, otherwise $K_{jj} = H_{jj}$. Clearly, AKA^T is a symmetric positive definite matrix for all possible $\mathcal{Q}$.

Let $\bar{A} \in \Re^{m\times q}$, where $q = |\mathcal{Q}|$, consists of all columns A_j such that $j \in \mathcal{Q}$ and let $\bar{D} \in \Re^{q\times q}$ be the diagonal matrix corresponding to the nonzero diagonal elements of D. In this notation,

$$AKA^T = A(H + D)A^T = AHA^T + \bar{A}\bar{D}\bar{A}^T, \tag{7}$$

so, AKA^T is a rank q-correction of AHA^T. In particular, if

$$\mathcal{Q} = \{1 \le j \le n | G_{jj} \neq H_{jj}\},$$

then $AKA^T = AGA^T$. The elements in the class of preconditioners are determined by the choice of the index set $\mathcal{Q}$. We use the following result:

Theorem C.1 *[3] Let $G, H \in \Re^{n\times n}$ be positive definite diagonal matrices. Let γ_j be the sorted elements of G_{jj}/H_{jj} in non-decreasing order, i.e.,*

$$\gamma_1 = \min_j \{G_{jj}/H_{jj}\} \le \gamma_2 \le \cdots \le \gamma_n = \max_j \{G_{jj}/H_{jj}\}.$$

Let $\mathcal{Q}$ consist of the indices corresponding to the q_1 largest and the q_2 smallest diagonal elements of $H^{-1}G$. Then

$$\min\{\gamma_{q_2+1}, 1\} \le \lambda_i((AKA^T)^{-1}AGA^T) \le \max\{\gamma_{n-q_1}, 1\}. \blacksquare$$

Theorem C.1 suggests choosing $\mathcal{Q}$ to consist of indices corresponding to the largest and smallest diagonal elements of $H^{-1}G$. It also gives a bound on the eigenvalues of $(AKA^T)^{-1}AGA^T$ for this particular choice of $\mathcal{Q}$. Further, the condition number $\kappa((AKA^T)^{-1}AGA^T)$ is bounded above by

$$\kappa((AKA^T)^{-1}AGA^T) \le \frac{\max\{\gamma_{n-q_1}, 1\}}{\min\{\gamma_{q_2+1}, 1\}}.$$

We can keep the condition number of $(AKA^T)^{-1}AGA^T$ bounded above by a predescribed constant if we carefully choose q_1 and q_2 sufficiently large.

C.1 The index set $\mathcal{Q}$

Let q be the number of elements in $\mathcal{Q}$. For a fixed q, the index set $\mathcal{Q}$ consists of indices corresponding to q_1 largest and q_2 smallest diagonal elements of $H^{-1}G$ (where $q = q_1 + q_2$) such that the condition number of $K^{-1}G$ is minimized.

Wang and O'Leary [11, 10] discuss a strategy of preconditioning the normal equation system based on formulation (7). They choose the index set $\mathcal{Q}$ to consist of the indices corresponding to the large $|G_{jj} - H_{jj}|$. This kind of choice does not guarantee a small condition number of the preconditioned matrix especially in the last stage of the interior point algorithm as we will see later. Further, this may slow down the convergence of the conjugate gradient iteration.

C.2 Computing the preconditioner

We want to solve the linear system

$$AKA^Ts = v \tag{8}$$

efficiently, where AKA^T is given by (7). In this paper we compute the solution of the linear system (8) by the Sherman-Morrison-Woodbury formula approach. However, it is possible to solve (8) based on modifying (updating or downdating) the triangular factors [2]. Applying the Sherman-Morrison-Woodbury formula ([8], pp. 50) to (7) gives

$$(AKA^T)^{-1} = (AHA^T)^{-1} - (AHA^T)^{-1}\bar{A}(\bar{D}^{-1} + \bar{A}^T(AHA^T)^{-1}\bar{A})^{-1}\bar{A}^T(AHA^T)^{-1}.$$

Let $LL^T = AHA^T$ and compute $V = L^{-1}\bar{A}$. Thus

$$(AKA^T)^{-1} = (LL^T)^{-1} - L^{-T}V(\bar{D}^{-1} + V^TV)^{-1}V^TL^{-1} = L^{-T}(I - VF^{-1}V^T)L^{-1}, \tag{9}$$

where $F = \bar{D}^{-1} + V^TV$. The solution of (8) is then

$$s = (AKA^T)^{-1}v = L^{-T}[d - (V(F^{-1}(V^Td)))], \tag{10}$$

where $d = L^{-1}v$. In computing $(AKA^T)^{-1}v$ for different values of v we need to store L, V and the factors of F. The main cost of computing the solution s is in the two solves $L^{-1}v$ and $L^{-T}(\cdot)$, and the matrix vector products V^Td and $V(\cdot)$. Note that F is symmetric but may be indefinite.

D The iterative solver–PCGLS

The preconditioned conjugate gradient least squares (PCGLS) algorithm ([5], pp. 294) is designed to solve linear systems of the form $A^TAy = Ah$. Here, we rewrite the PCGLS of [5] to handle weighted problems of the form $AGA^Ty = AGh$ with a preconditioner based on the Sherman-Morrison-Woodbury formula (9). We state the modified algorithm below:

Algorithm D.1 *PCGLS*
Let $M = (AKA^T)^{-1}$ and y_0 be the starting vector
$r_0 = h - A^Ty_0$, $v_0 = AGr_0$, $p_0 = s_0 = Mv_0$, $\gamma_0 = s_0^Tv_0$
$k \leftarrow 0$
while not converged **do**
$q_k = A^Tp_k$
$\alpha_k = \frac{\gamma_k}{q_k^TGq_k}$
$y_{k+1} = y_k + \alpha_k p_k$
$r_{k+1} = r_k - \alpha_k q_k$
$v_{k+1} = AGr_{k+1}$
$s_{k+1} = Mv_{k+1}$
$\gamma_{k+1} = s_{k+1}^Tv_{k+1}$
$\beta_k = \frac{\gamma_{k+1}}{\gamma_k}$
$p_{k+1} = s_{k+1} + \beta_k p_k$
$k \leftarrow k+1$

Note that the main cost per PCGLS iteration consists of computing s in (10) and the matrix vector products with A and A^T. In addition to the storage of the nonzero elements of A and G, we need storage for the nonzero elements of L and V, the factors of F and the vector d in (10), and the vectors q, p, v and r in Algorithm D.1.

E Numerical results

In this section we present some numerical results. We have implemented the mixed primal-dual Newton algorithm. Our implementation in this paper is done entirely in MATLAB[2]. We use a starting point (x_0, y_0, z_0) that is identical to the one given by Mehrotra [9]. The termination criteria in Algorithm D.1 is when number of

[2]MATLAB is a registered trademark of The Math Works, Inc

iterations has reached a predefined value t or when $\|v_{k+1}\|_2 \leq 10^{-5}$. We terminate the interior point iterations when the relative error ϵ in the interior point algorithm is less than or equal to a predefined relative error tolerance $\epsilon^* = 10^{-5}$. The relative error ϵ is given by

$$\epsilon = \max\left\{ \frac{\|Ax-b\|_2}{\max\{1, \|b\|_2\}}, \frac{\|A^Ty+z-c\|_2}{\max\{1, \|c\|_2\}}, \frac{|c^Tx - b^Ty|}{\max\{1, |c^Tx|\}} \right\}. \tag{11}$$

The step-length parameter is $\tau = 0.99995$, and the barrier parameter $\mu = \sigma(x^Tz)/n$, where $\sigma = 0.1$. The direct solver used is the standard solver (**chol** function) in MATLAB. In our implementation, Symmetric minimum degree permutation is used to reduce fill in.

E.1 How Good Are the Corrections and Bounds? How Should the Index Set be Selected?

Here we compare two techniques for selecting the index set for the low-rank corrections. The first technique is based on choosing the indices corresponding to the largest diagonal elements of $|G - H|$, as discussed in [11, 10]. The second technique is based on choosing indices corresponding to the smallest and the largest diagonal elements of $H^{-1}G$ as proposed in Baryamureeba, Steihaug and Zhang [3]. We present examples based on some Netlib test problems [6] and mention that the observed behavior seems to be typical out of many similar experiments we conducted on different test problems.

Figure 1 shows the distribution of the eigenvalues of the preconditioned matrix using the low-rank preconditioner based on $|G - H|$ and $H^{-1}G$. We see that the preconditioned matrix $(AKA^T)^{-1}AGA^T$ based on $H^{-1}G$ has a smaller condition number than that of $(AVA^T)^{-1}AGA^T$ which is based on $|G - H|$.

Figure 2 shows the distribution of eigenvalues of $(AHA^T)^{-1}AGA^T$ and $H^{-1}G$. We observe that the spectrum of $(AHA^T)^{-1}AGA^T$ is always contained in the spectrum of $H^{-1}G$.

From Figure 3 we see that the spectrum of $(AKA^T)^{-1}AGA^T$ is always contained in the spectrum of $K^{-1}G$. We also note that the tightness of the bounds of the eigenvalues of $(AKA^T)^{-1}AGA^T$ based on $K^{-1}G$ has improved compared to the bounds of the eigenvalues of $(AHA^T)^{-1}AGA^T$ based on $H^{-1}G$. Clearly, from Figure 3 we see that the upper and lower bounds on the eigenvalues of $(AKA^T)^{-1}AGA^T$ are very good. The smallest and largest eigenvalues of $K^{-1}G$ are very good bounds of the eigenvalues of $(AKA^T)^{-1}AGA^T$.

On the other hand, Figure 4 shows that $V^{-1}G$ and $H^{-1}G$ have almost the same distribution of eigenvalues less than one, and the spectrum of $(AVA^T)^{-1}AGA^T$ is always contained in the spectrum of $V^{-1}G$. We also observe from Figure 4 that choosing the index set based on the large $|G_{jj} - H_{jj}|$ does not usually include the indices corresponding to the small G_{jj}/H_{jj}. This implies that the preconditioned matrix $(AVA^T)^{-1}AGA^T$ has approximately the same distribution of eigenvalues as $(AHA^T)^{-1}AGA^T$ for eigenvalues less than one, see Figure 1. Further, the index set based on the large $|G_{jj} - H_{jj}|$ does not include indices corresponding to the case when G_{jj} and H_{jj} are small but G_{jj}/H_{jj} is large. When this case exists in a test problem, then the condition numbers of $(AHA^T)^{-1}AGA^T$ and $(AVA^T)^{-1}AGA^T$ may be approximately the same for a low-rank correction.

We have used the test problem **stocfor1** to show the distribution of the eigenvalues. We generate a test problem at an interior point iteration in the last stage of the interior point algorithm. In Figure 5 we give examples to demonstrate the effectiveness of the low-rank preconditioner based on $|G - H|$ and $H^{-1}G$. The gragh for **stocfor1** in Figure 5 corresponds to the test problem used in Figure 1, 2, 3 and 4. For the results in Figure 5, we apply PCGLS to the generated linear systems for different test problems and iterate until the termination criteria is satisfied in each case. From Figure 5 we that for the preconditioner [11, 10] based on large $|G_{jj} - H_{jj}|$, PCGLS requires considerably more iterations to achieve the prescribed accuracy than using the preconditioner [3] based on large and small G_{jj}/H_{jj}.

This is well explained by the results in Figure 1, where we observe that the preconditioned matrix corresponding to $M_1 = AVA^T$ has a larger condition number than that corresponding to $M_2 = AKA^T$. To attain rapid convergence for PCGLS we require that either the spectral condition number of the preconditioned matrix be close to one or the eigenvalues of the preconditioned matrix be clustered. Thus we conclude that we should choose the index set based on $H^{-1}G$, and not $|G - H|$.

For the numerical results on how the mixed strategy performs we refer the reader to [2, 3, 3].

Our preliminary numerical results indicate that the distribution of the eigenvalues of the preconditioned matrix approximated by the distribution of the ratios G_{ii}/H_{ii} is quite satisfactory. Further, our numerical results indicate that the theoretical bounds on the spectrum of the preconditioned matrix obtained in [3] are well suited to to be used as a guide to select the elements in $\mathcal{Q}$. The theory and the numerical testing strongly support the principle of selecting the index set $\mathcal{Q}$ based on the magnitude of $H^{-1}G$.

References

[1] V. Baryamureeba, and T. Steihaug, *Computational issues for a new class of preconditioners,* The Proceedings for the 2nd Workshop on Large-Scale Scientific Computations, Bulgaria, 1999, to appear.

[2] V. Baryamureeba, T. Steihaug and Y. Zhang, *Properties of a class of preconditioners for weighted least squares problems,* Technical Report No. 170, Department of Informatics, University of Bergen, 5020 Bergen, Norway, April 30, 1999 (Revised July 6, 1999).

[3] V. Baryamureeba, T. Steihaug and Y. Zhang, *Application of a Class of Preconditioners to Large Scale Linear Programming Problems*, P. Amestoy, P. Berger, M. Dayd, I. Duff, V. Frayss, L. Giraud and D. Ruiz, (eds), EuroPar'99 Parallel Processing. Lecture Notes in Computer Science, No. 1685, Springer-Verlag, pp. 1044 -1048, 1999.

[4] Å. Björck, *Numerical methods for least squares problems*, SIAM, 1996.

[5] A.S. El-Bakry, R.A. Tapia, T. Tsuchiya, and Y. Zhang, *On the formulation of the primal-dual Newton interior-point method for nonlinear programming,* J. Optimization Theory and Applications, Vol. 89:507-541, 1996.

[6] D.M. Gay, *Electronic mail distribution of linear programming test problems,* Mathematical Programming Society COAL Newsletter, No. 13, pp.10-12, 1985.

[7] G. H. Golub, and C.F. Van Loan, *Matrix computations,* Third Edition, 1996.

[8] M. Kojima, S. Mizuno, and A. Yoshise, *A primal-dual interior point method for linear programming.* In Nimrod Megiddo, editor, *Progress in Mathematical programming, interior-point and related methods*, pages 29–47. Springer-Verlag, New York, 1989.

[9] S. Mehrotra, *Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods,* ORSA J. Comput., 4(1992), pp. 103-118.

[10] W. Wang, and D.P. O'Leary, *Adaptive use of iterative methods in interior point methods for linear programming,* Computer Science Department Report CS-TR-3560, Institute for Advanced Computer Studies Report UMIACS-95-111, University of Maryland, November 1995.

[11] W. Wang, and D.P. O'Leary, *Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming*, Computer Science Department Report CS-TR-4011, Institute for Advanced Computer Studies Report UMIACS-TR-99-21, University of Maryland, April 1999, Preprint (1999).

[12] S. Wright, *Primal-dual interior-point methods.* SIAM, Philadelphia, 1997.

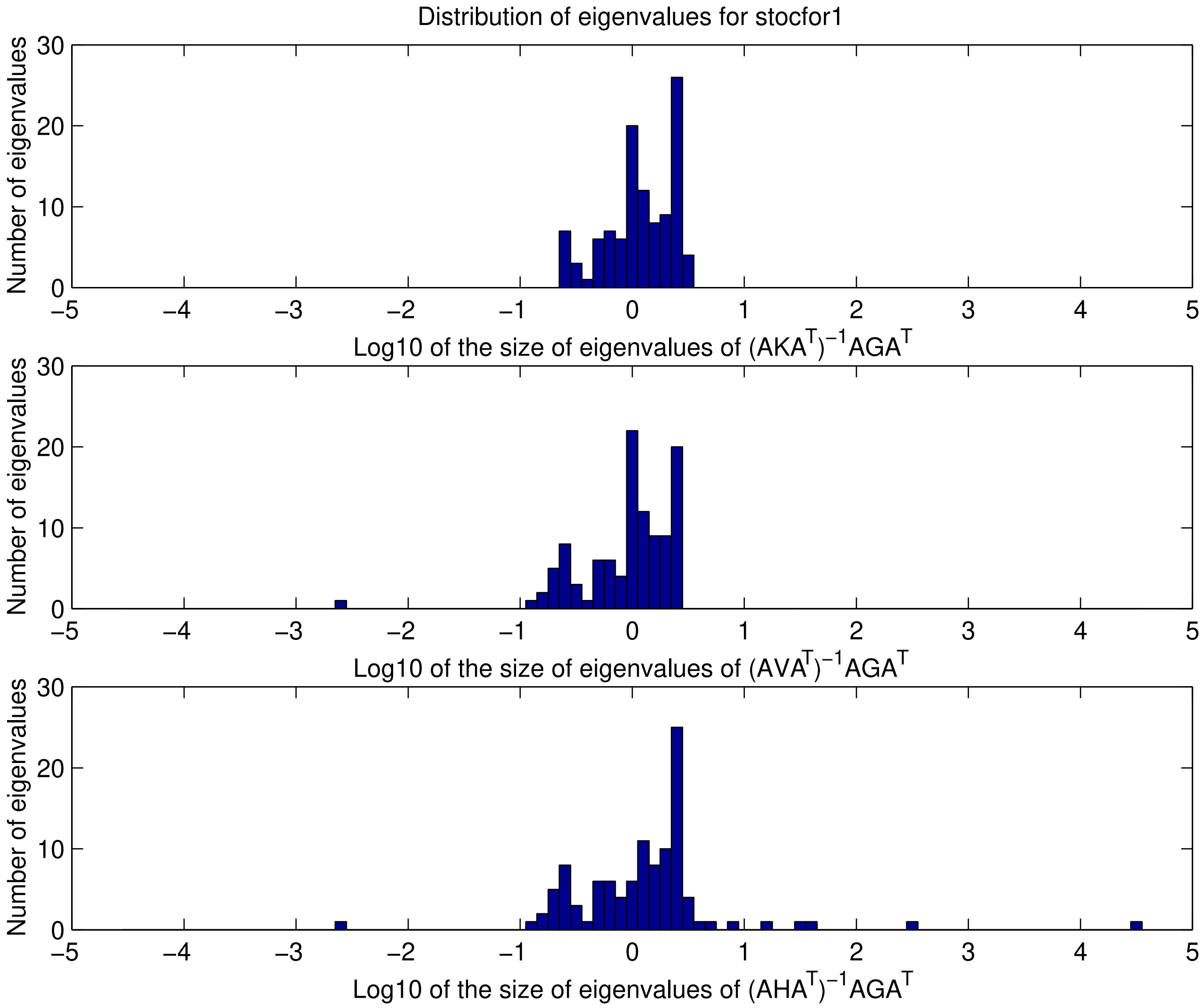

Figure 1: For preconditioner $M_1 = AVA^T$, we choose indices corresponding to 20 largest diagonal elements of $|G - H|$. For preconditioner $M_2 = AKA^T$, we choose indices corresponding to 10 largest and 10 smallest diagonal elements of $H^{-1}G$ at that same iteration. The test problem is obtained at an interior point iteration in the last stage for test problem **stocfor1** from the Netlib.

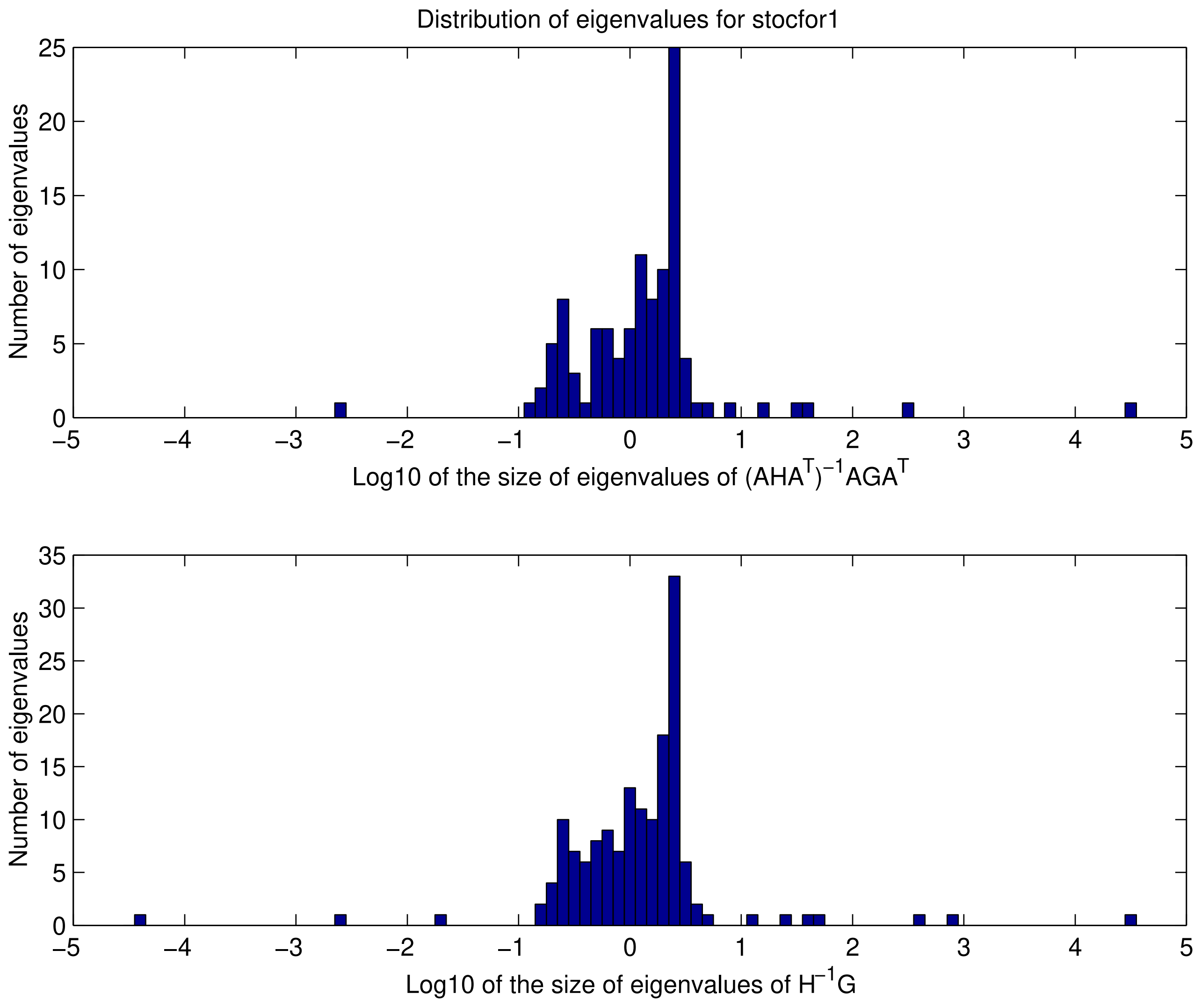

Figure 2: The test problem is obtained at an interior point iteration in the last stage for test problem **stocfor1** from the Netlib.

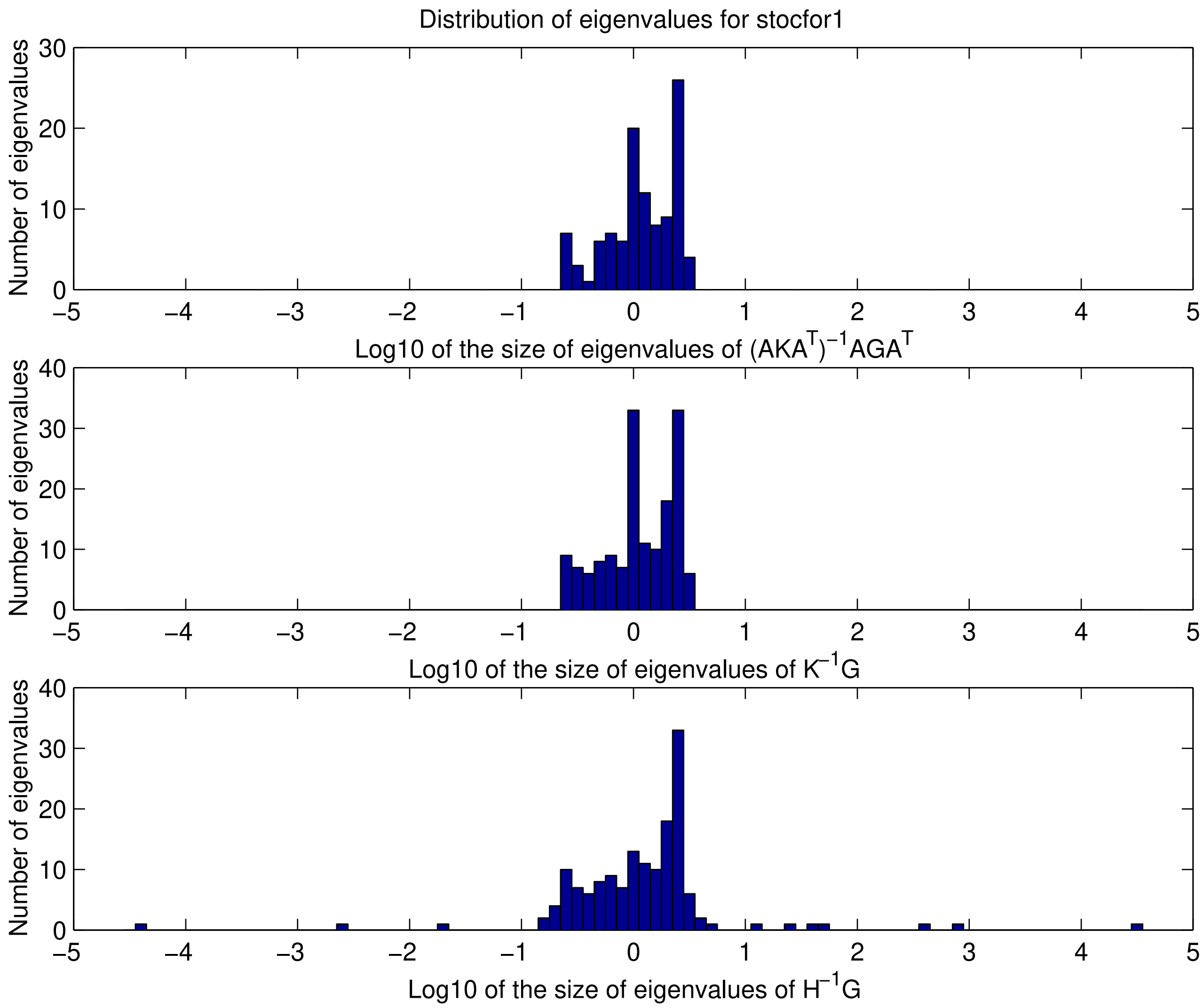

Figure 3: For preconditioner $M_1 = AVA^T$, we choose indices corresponding to 20 largest diagonal elements of $|G - H|$. For preconditioner $M_2 = AKA^T$, we choose indices corresponding to 10 largest and 10 smallest diagonal elements of $H^{-1}G$ at that same iteration. The test problem is obtained at an interior point iteration in the last stage for test problem **stocfor1** from the Netlib.

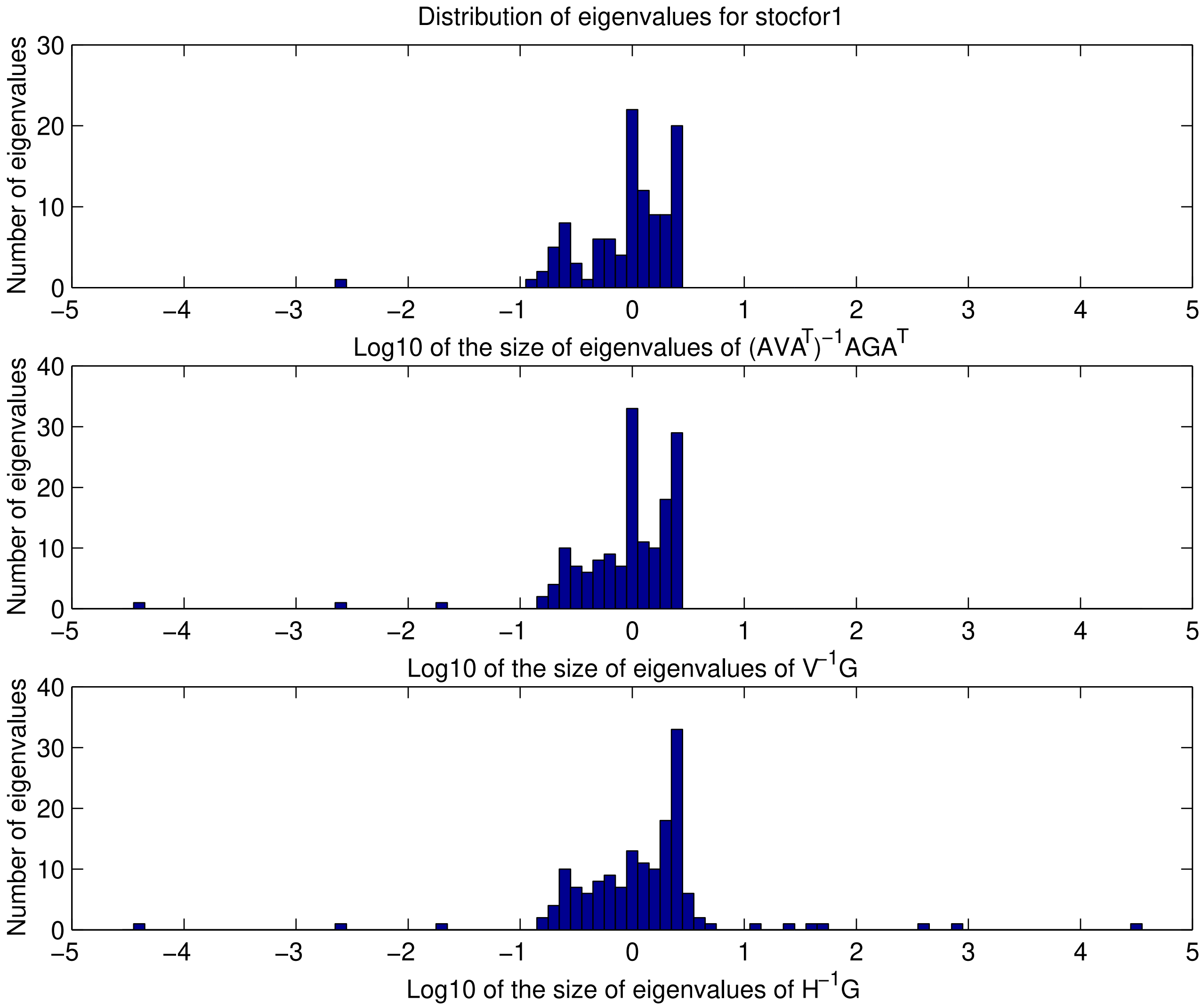

Figure 4: For preconditioner $M_1 = AVA^T$, we choose indices corresponding to 20 largest diagonal elements of $|G - H|$. For preconditioner $M_2 = AKA^T$, we choose indices corresponding to 10 largest and 10 smallest diagonal elements of $H^{-1}G$ at that same iteration. The test problem is obtained at an interior point iteration in the last stage for test problem **stocfor1** from the Netlib.

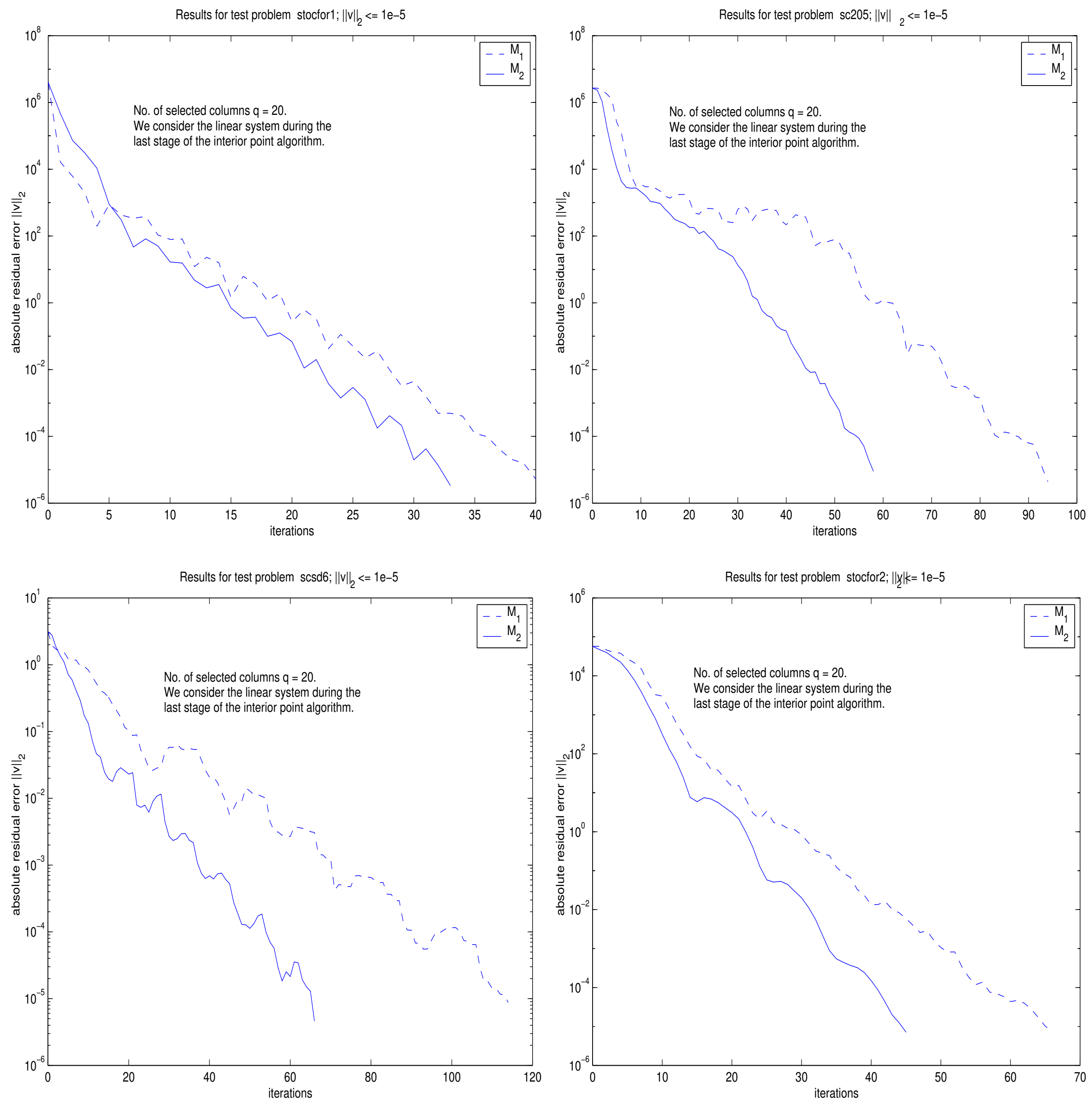

Figure 5: For preconditioner $M_1 = AVA^T$, we choose indices corresponding to 20 largest diagonal elements of $|G - H|$; and for preconditioner $M_2 = AKA^T$, we choose indices corresponding to 10 largest and 10 smallest diagonal elements of $H^{-1}G$ at that same iteration.

Computational Issues for a New Class of Preconditioners

Venansius Baryamureeba   Trond Steihaug

Presented at the 2nd Workshop on
Large-Scale Scientific Computations (WLSSC'99),
Sozopol, Bulgaria,
June 2-6, 1999.

Computational Issues for a New Class of Preconditioners [1]

Venansius Baryamureeba Trond Steihaug

Department of Informatics
University of Bergen
5020 Bergen Norway

Abstract

In this paper we consider solving a sequence of weighted linear least squares problems where the only changes from one problem to the next are the weights and the right hand side (or data). We alternate between iterative and direct methods to solve the normal equations for the least squares problems. The direct method is the Cholesky factorization. For the iterative method we discuss a class of preconditioners based on a low rank correction of a Cholesky factorization of a weighted normal equation coefficient matrix. Different ways to compute the preconditioner are given. Further, a sparse algorithm for modifying the Cholesky factors by a low rank matrix is derived.

Key Words. Weighted linear least squares, Linear programming, Infeasible interior point algorithms, Sherman Morrison-Woodbury formula, Cholesky factor

A Introduction

The weighted linear least squares problem is

$$\min_y \|A^T y - h\|_G, \tag{1}$$

where $y \in \Re^m$, $h \in \Re^n$. The weight G is an n-by-n symmetric positive definite matrix and $\|z\|_G^2 = z^T G z$.

The matrix $A \in \Re^{m\times n}$ $(m \leq n)$ is assumed to have full rank m. The solution of (1) is given by the normal equations

$$AGA^T y = AGh. \tag{2}$$

Our interest is in sequences of weighted linear least squares problems, where both the weight matrix G and the vector h change from system to system while A is constant. These applications include linear programming, electrical networks, elliptic boundary value problems, and structural analysis (see Strang [11] and the references therein.)

Consider the sequence of problems of the form

$$AG^k A^T y = h^k,\ k \geq 0 \tag{3}$$

where k denotes the computation step. These problems can be solved using a direct or an iterative method. We suggest a combination of these methods in solving (3). A natural choice of preconditioner for the iterative method at step k will be the Cholesky factorization of AG^jA^T for $j < k$. The combination involves solving the linear system (3) with a direct method at computation step j and use an iterative method to solve the system at step k $(k > j)$. In the class of preconditioners [3, 11], only important changes from AG^jA^T to AG^kA^T are incorporated. In this paper we discuss computational issues for this class of preconditioners.

Throughout this paper we use the following notation: $\min_i$ or $\max_i$ is for all i for which the argument is defined. For any matrix A, A_{ij} is the element in the i-th row and j-th column, $A_{j\bullet}$ is the j-th row, and A_j is the j-th column. For any square matrix X, $\lambda_i(X)$ are the eigenvalues of X arranged in non decreasing order. For any X, $X^{(i)}$ denotes X at the i-th computation stage, and $\mathrm{nnz}(X)$ is the number of nonzero elements in X.

[1] VIEWEG Series Notes on Numerical Fluid Mechanics (The Proceedings for the 2nd Workshop on Large-Scale Scientific Computations, Bulgaria, 1999), to appear.

B The Class of Preconditioners

Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal matrices. For a given index set $\mathcal{Q} \subseteq \{j | 1 \le j \le n \text{ and } G_{jj} \ne H_{jj}\}$, the $n \times n$ diagonal matrices $\bar{H}, \bar{G}$ are given by

$$\bar{H}_{jj} = \begin{cases} H_{jj} & \text{if } j \in \mathcal{Q} \\ 0 & \text{otherwise,} \end{cases}$$

$$\bar{G}_{jj} = \begin{cases} G_{jj} & \text{if } j \in \mathcal{Q} \\ 0 & \text{otherwise,} \end{cases}$$

and the n-by-n diagonal matrix K is

$$K = H + \bar{G} - \bar{H}. \tag{4}$$

Note that $K_{jj} = G_{jj}$ if $j \in \mathcal{Q}$, otherwise $K_{jj} = H_{jj}$. Let $q = |\mathcal{Q}|$ and $\bar{A} \in \Re^{m\times q}$ consist of all columns A_j such that $j \in \mathcal{Q}$. Further, let $\bar{D} \in \Re^{q\times q}$ be the nonzero diagonal submatrix of $\bar{G} - \bar{H}$ corresponding to the columns in $\bar{A}$. The class of preconditioners [3, 11] is defined by

$$AKA^T = AHA^T + \bar{A}\bar{D}\bar{A}^T. \tag{5}$$

The weight matrix H is the weight at the most recent computation step with a Cholesky factorization. If $\mathcal{Q} = \{j : | 1 \le j \le n \text{ and } G_{jj} \ne H_{jj}\}$ then $AKA^T = AGA^T$. However, computing solutions to linear systems of the form

$$AKA^T v = h \tag{6}$$

can be done much cheaper than solving

$$AGA^T v = h \tag{7}$$

provided $\mathcal{Q}$ has few elements. The next two results from [3] give guidelines for determining $\mathcal{Q}$.

Theorem B.1 *[3]: Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal. For a given $\mathcal{Q}$ let K be defined as in (4). The eigenvalues of $(AKA^T)^{-1}AGA^T$ satisfy*

$$\min\left\{1, \min_{j\notin\mathcal{Q}}\left\{\frac{G_{jj}}{H_{jj}}\right\}\right\} \le \lambda_i((AKA^T)^{-1}AGA^T) \le \max\left\{1, \max_{j\notin\mathcal{Q}}\left\{\frac{G_{jj}}{H_{jj}}\right\}\right\}.$$

This theorem suggests that we include in $\mathcal{Q}$ indices corresponding to the large $G_{jj}/H_{jj} > 1$ in order to reduce on the upper bound of the eigenvalues of $(AKA^T)^{-1}AGA^T$. Similarly, to increase the lower bound, indices corresponding to the small $G_{jj}/H_{jj} < 1$ should be included.

The next result gives the lower bound on $\lambda_m((AKA^T)^{-1}AGA^T)$ and an upper bound on $\lambda_1((AKA^T)^{-1}AGA^T)$ in terms of the eigenvalues of the preconditioned matrix without the low rank corrections $\lambda_j((AHA^T)^{-1}AGA^T)$.

Theorem B.2 *[3]: Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal. Let $|\mathcal{Q}| = q = q_1 + q_2 < m$, where q_1 is the number of indices in $\mathcal{Q}$ where $G_{jj}/H_{jj} > 1$. Then*

$$\lambda_1((AKA^T)^{-1}AGA^T) \le \lambda_{q_2+1}((AHA^T)^{-1}AGA^T)$$

and

$$\lambda_m((AKA^T)^{-1}AGA^T) \ge \lambda_{m-q_1}((AHA^T)^{-1}AGA^T).$$

B.1 Computing the Preconditioner

We consider two approaches of computing solutions of (6), namely the Sherman Morrison-Woodbury formula approach, and the approach based on modifying the triangular factors.

B.1.1 The Sherman Morrison-Woodbury Formula

To simplify the notation in the rest of this section, let $M = (AKA^T)^{-1}$. For $AKA^T = AHA^T + \bar{A}\bar{D}\bar{A}^T$ the Sherman Morrison-Woodbury formula ([9] pp. 50) gives

$$M = (AHA^T)^{-1} - (AHA^T)^{-1}\bar{A}(\bar{D}^{-1} + \bar{A}^T(AHA^T)^{-1}\bar{A})^{-1}\bar{A}^T(AHA^T)^{-1}. \tag{8}$$

Let $LL^T = AHA^T$ and compute $V = L^{-1}\bar{A}$. Thus

$$M = (LL^T)^{-1} - L^{-T}V(\bar{D}^{-1} + V^TV)^{-1}V^TL^{-1} = L^{-T}(I - VF^{-1}V^T)L^{-1}, \tag{9}$$

where $F = \bar{D}^{-1} + V^TV$. For an m-vector h the solution to (6) is

$$v = Mh = L^{-T}(r - (V(F^{-1}(V^Tr)))), \tag{10}$$

where $r = L^{-1}h$. Computing Mh is equivalent to solving for v in (6). In computing Mh we need to store L, V, r, and the factors of F. Note that F is a $q \times q$ symmetric matrix that may be indefinite. Forming F requires approximately $2q + 2q\ \mathrm{nnz(V)}$ floating point operations.

A solve is either a backward or forward substitution, and its floating point operations depend on the sparsity of the right hand side. Forming M consists of q solves with a sparse right hand side to compute $V = L^{-1}\bar{A}$, and forming and factorizing the q-by-q matrix F. The computation of Mh consists of 2 solves with a full right hand side, plus approximately $4\mathrm{nnz}(V) + 2q^2 + m$ floating point operations.

To exploit even more of the sparsity structure of $\bar{A}$ for very sparse problems, first determine the nonzero rows of $\bar{A}$. Let $p \ll q$ be the number of nonzero rows of $\bar{A}$ and let E_i be the i-th unit vector corresponding to row index i. The $m \times p$ matrix E consists of the columns E_i for all distinct row indices i. Then $L^{-1}\bar{A} = (L^{-1}E)(E^T\bar{A})$. Again note that the sparsity in the right hand side (E) should be exploited.

Let $\tilde{V} = L^{-T}V$ and

$$\tilde{M} = (LL^T)^{-1} - \tilde{V}\tilde{F}^{-1}\tilde{V}^T,$$

where $\tilde{F} = D^{-1} + \bar{A}^T\tilde{V}$. Forming $\tilde{F}$ requires approximately $2q + 2q\ \mathrm{nnz}(\bar{A})$ floating point operations. Hager [10] suggests computing

$$\tilde{M}h = L^{-T}(L^{-1}h) - (\tilde{V}(\tilde{F}^{-1}(\tilde{V}^Th))).$$

Forming $\tilde{M}$ requires $2q$ solves with a sparse right hand side. This is twice as many solves as for forming M. Forming $\tilde{M}$ also involves forming and factorizing the q-by-q matrix $\tilde{F}$. Computing $\tilde{M}h$ requires 2 solves with a full right hand side, and approximately $4\mathrm{nnz}(\tilde{V}) + 2q^2 + m$ floating point operations. Numerical testing on the large Netlib test problems [6] shows that V is a sparse matrix and $\tilde{V}$ is almost a dense $m \times q$ matrix. When using the Sherman Morrison-Woodbury formula approach, Mh is computed using (10).

B.1.2 Modification of the Triangular Factors

We consider the case where a factorization of a symmetric positive definite matrix is modified by a weighted symmetric matrix of rank q ($q \geq 1$). Let

$$RTR^T = LDL^T + \bar{A}\bar{D}\bar{A}^T = L(D + V\bar{D}V^T)L^T,$$

where $V = L^{-1}\bar{A}$. The special nature of the matrix $D + V\bar{D}V^T$ is used in computing its triangular factors. Let

$$\hat{L}\hat{D}\hat{L}^T = D + V\bar{D}V^T,$$

then the modified factors are of the form

$$RTR^T = (L\hat{L})\hat{D}(L\hat{L})^T,$$

which gives $R = L\hat{L}$ and $T = \hat{D}$, using that the product of two lower-triangular matrices is a lower-triangular matrix. By extending the derivation in [1, 8], it can be shown that

$$\hat{L}_{rs} = V_{r\bullet}u^{(s)},\ r = 2, \ldots, m;\ s = 1, 2, \ldots, r-1 \tag{11}$$

where $u^{(s)}$ is a q-vector. By induction it can be shown that the factors $\hat{L}$ and $\hat{D}$ are unique, and $\hat{L}$ is of the form (11). This implies that we need only to compute the values of $\hat{D}_{ii}$ and $u^{(i)}$, for $i = 1, 2, \ldots, m$ in order

$C^{(1)} = \bar{D}$
for $i = 1, \ldots, m$ **do**
  $p^{(i)} = C^{(i)} V_{i\bullet}^T$
  $\hat{D}_{ii} = D_{ii} + V_{i\bullet}\, p^{(i)}$
  $u^{(i)} = \frac{1}{\hat{D}_{ii}} p^{(i)}$
  $C^{(i+1)} = C^{(i)} - u^{(i)} {p^{(i)}}^T$

Table 1: Computing $\hat{D}$ and $u^{(i)}$

Set $W^{(1)} = LV$
for $i = 1, \ldots, m$ **do**
  for $j = i+1, \ldots, m$ **do**
    $W_{j\bullet}^{(i+1)} = W_{j\bullet}^{(i)} - L_{ji} V_{i\bullet}$
    $R_{ji} = L_{ji} + W_{j\bullet}^{(i+1)} u$

Table 2: Computing R

to obtain the factorization of $D + V\bar{D}V^T$. Given V, D and $\bar{D}$ we can show that the recurrence relations for computing $\hat{D}_{ii}$ and $u^{(i)}$ are as given in Table 1. We can compute $R = L\hat{L}$ in terms of the $u^{(i)}$, for $i = 1, \ldots, m$ by forward recurrence using Lemma V in [8]. Given V we compute R by the recurrence relation in Table 2.

The matrix $W^{(1)}$ needed to initialize the recurrence relation is known since $W^{(1)} = LV = \bar{A}$. The matrices $W^{(i)}$ for $i = 1, \ldots, m$ obtained during the i-th stage are

$$W_{r\bullet}^{(i)} = \bar{A}_{r\bullet} - \sum_{j=1}^{i-1} L_{rj} V_{j\bullet}, \quad r = i, i+1, \ldots, m.$$

At the i-th computation stage we have $W_{j\bullet}^{(i)} = L^{-1}\bar{A}_{j\bullet}$, $j \le i$. Thus $W_{i\bullet}^{(i)} = V_{i\bullet}$, and $W^{(m)} = L^{-1}\bar{A} = V$. The final recurrence relations for modifying L and D take on the form of Algorithm B.1 in Table 3.

The inner loop in Algorithm B.1 can be (using the notation in [9]) written as:

$$\begin{aligned} V(i+1:m,:) &= V(i+1:m,:) - L(i+1:m,i)V(i,:) \\ R(i+1:m,i) &= L(i+1:m,i) + (V(i+1:m,:)u). \end{aligned} \tag{12}$$

Bartels and Kaufman [1] derived Algorithm B.1 for the case $q = 2$. The symmetric version of the algorithm by Bennett [4] on $LDL^T + \bar{A}\bar{D}\bar{A}^T$ is identical to Algorithm B.1. This algorithm is stable [8] if $LDL^T + \bar{A}\bar{D}\bar{A}^T$ is positive definite.

B.1.3 Modification of Sparse Factors by a Sparse Matrix of Rank q

The Cholesky factors of AHA^T and AKA^T are unique (see Theorem 4.2.5, [9]), and when $\bar{A}$ consists of columns from A the positive definite matrices AHA^T and AKA^T have the same symbolic factorization and consequently, R and L have the same sparsity structure. This is reflected in the sparse algorithm by setting $R_{ji} = 0$ whenever $L_{ji} = 0$. For $V_{i\bullet} = 0$ we have from Table 1 that $u^{(i)} = 0$ and $\hat{L}_{ri} = 0$ for $r = i+1, \ldots, m$ in (11). Thus $R_{ri} = L_{ri}$ for $r = i+1, \ldots, m$. The sparse algorithm is given in Table 3.

Algorithm B.1 *Dense case*
Define $V = \bar{A}$; $C = \bar{D}$

for $i = 1, \ldots, m$ **do**
  $p = (V_{i\bullet} C)^T$
  $T_{ii} = D_{ii} + V_{i\bullet}\, p$
  $u = (1/T_{ii})\, p$
  $C \leftarrow C - u\, p^T$
  for $j = i+1, \ldots, m$ **do**
    $V_{j\bullet} \leftarrow V_{j\bullet} - L_{ji} V_{i\bullet}$
    $R_{ji} = L_{ji} + V_{j\bullet}\, u$

Algorithm B.2 *Sparse case*
Define $V = \bar{A}$; $C = \bar{D}$; $R = L$; $T = D$

for $i = 1, \ldots, m$ **do**
  if $V_{i\bullet} \ne 0$ **then**
    $p = (V_{i\bullet} C)^T$
    $T_{ii} = D_{ii} + V_{i\bullet}\, p$
    $u = (1/T_{ii})\, p$
    $C \leftarrow C - u\, p^T$
    for $j = i+1, \ldots, m$ **do**
      if $L_{ji} \ne 0$ **then**
        $V_{j\bullet} \leftarrow V_{j\bullet} - L_{ji} V_{i\bullet}$
        $R_{ji} = L_{ji} + V_{j\bullet}\, u$

Table 3: Modifying the triangular factors for matrices of the form $RTR^T = LDL^T + \bar{A}\bar{D}\bar{A}^T$

To extend the algorithm for the sparse case when $\bar{A}$ is a general sparse matrix (where columns in $\bar{A}$ are not necessarily from A) the only modification is to exclude the **if** test on $L_{ji} \neq 0$ in the inner loop. This algorithm is stable for all updates of the form $LDL^T + \bar{A}\bar{D}\bar{A}^T$, provided $LDL^T + \bar{A}\bar{D}\bar{A}^T$ is positive definite.

Rank-one modifications (updates and downdates) of the triangular factors are given in [4, 6, 8] and the references therein. Given a sparse symmetric positive definite matrix AA^T and an assoicated sparse triangular factorization LDL^T, Davis and Hager [6] develop sparse techniques for obtaining the new factorization when a column is added or deleted from A.

C Numerical Results

In this section, some preliminary numerical results are presented. We have implemented the Primal-Dual Newton (PDN). The starting point is identical to one given by Mehrotra [9]. The interior point iterations (in the following called steps) are terminated when an error ϵ is less than or equal to a predefined relative error tolerance $\epsilon^* = 10^{-5}$. The variable ϵ is a proximity measure of the interior point iterations to a solution of the linear programming problem. The sparse Cholesky factorization is applied after using the symmetric minimum degree ordering. The step length parameter $\tau = 0.99995$ and the barrier parameter $\mu = 0.1x^Tz/n$.

In the mixed PDN methods (3) is solved with a direct method for every even step and an iterative method at every odd step. For simplicity, we will call an interior point iteration (or step) a *direct step* when (2) is solved using the Cholesky factorization and an *iterative step* when an iterative method is used to find an approximate solution of (2). The iterative method is preconditioned conjugate gradient and least squares (PCGLS) method [5]. The PCGLS iterations are terminated when the maximum number of iterations is reached or when the residual error in the normal equation (2) is less than 10^{-5}. When ϵ is large the maximum number of iterations in the iterative solver is a small value, and when ϵ falls below a certain predefined constant the maximum number of iterations is increased in order to attain both primal and dual feasibility. Test problems from the Netlib set of linear programs [6] are used in the testing. The test code is implemented in MATLAB[2].

A direct step consist of computing AGA^T and its Cholesky factor, and two solves with a full right hand side. An iterative step consist of forming M (for a given q) and the PCGLS iterations. We define the break-even point for using M as the largest number of PCGLS iterations for which the floating point operations (flops) of an iterative step is less than or equal to that of a direct step. We will denote the M matrix defined in (9) by M_1,

Problem				Maximal q	
Name	m	n	$s\%$	M_1	M_2
czprob	737	3141	0.41	18	30
d2q06c	2171	5831	0.26	152	85
d6cube	404	6184	1.51	94	56
stocfor2	2157	3045	0.14	105	69
scsd8	397	2750	0.79	17	13
agg3	516	758	1.22	54	36
maros	835	1921	0.63	49	34

Table 4: Break-even point for using M in one PCGLS iteration compared to a direct step. It is expressed in terms of q. Interior-point step = 2.

and M_2 will be given by $M_2 = (RTR^T)^{-1}$. Obviously, $M_1 = M_2$; but we use two distinct symbols because the amounts of computation associated with the two formulas are different. In Table 4 the columns for M_1 and M_2 indicate the maximum q so that the number of flops for one PCGLS iteration per iterative step do not exceed the number of flops for a direct step. The quantity $s\%$ indicates the density of nonzero elements in percentage. Sparsity density of an m-by-n matrix A is $\text{nnz}(A)/mn$.

For a constant q, we define the break-even point for using M as the largest number of PCGLS iterations for which the cost of a PCGLS step is less than or equal to the cost of a direct step, where the cost is measured by floating point operations. From Table 4 it is seen that column $\bar{A}_j$ from A is likely to be sparse. For $\bar{A}_j$ sparse, $L^{-1}\bar{A}_j$ is a solve with a sparse right hand side. Computing Mh in PCGLS, the vector h is a full vector. As a result, the number of floating point operations to increase the preconditioner by an additional rank-one update is usually less than those for increasing the number of PCGLS iteration by one for small values of q. Table 5

[2]MATLAB is a registered trademark of The Math Works, Inc

shows the relationship between number of PCGLS iterations and number of elements q in $\mathcal{Q}$. The q columns from A are chosen according to Theorem B.1 and $q_1 = q/2$.

In (3) we can use the results at $k = 0$ and $k = 1$ to compute the ratio of flops for one PCGLS iteration to a solve with a sparse right hand side, and use this ratio as an indicator for choosing the size of q. When the ratio is high we choose a large q and do few PCGLS iterations in solving problem (3). The mixed PDN method is

Maximal number of PCGLS iterations										
q	stocfor2		czprob		scsd8		d6cube		d2q06c	
	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
1	34	35	1	1	2	2	34	35	66	67
2	33	34	1	1	2	2	34	35	65	66
3	33	34	1	1	2	2	34	34	65	66
4	32	33	1	1	2	2	34	34	65	65
5	32	33	1	1	2	2	33	33	64	64
6	32	32	1	1	2	2	33	32	64	63
10	31	30	1	1	1	1	32	30	62	60
20	28	26		1			29	24	58	53
30	26	22		1			25	19	53	45
40	23	17					22	12	50	40
50	20	12					18	5	46	32

Table 5: The break-even point for using M for a fixed value of q expressed in number of PCGLS iterations. Interior-point iteration = 2.

likely to use less flops than PDN method if q and maximum number of iterations per step are carefully chosen. For most problems we do not get close to these break-even numbers since we terminate after fewer iterations (except for czprob and scsd8 problems). The results in Table 6 show that Mixed PDN is a promising method for solving large sparse linear programming problems. The cpu-time for this MATLAB implementation are better for the mixed method than for the method only using a direct method for solving the linear system. The mixed method approach can be extended to the predictor-corrector method.

Problem	PDN		Mixed PDN^1		Mixed PDN^2		Mixed PDN^3	
name	iter	time	iter	time	iter	time	iter	time
czprob	56	89.01	59	59.58	58	56.52	58	56.83
d2q06c	54	544.85	57	421.33	61	442.24	55	405.17
d6cube	42	317.58	45	239.16	47	226.72	46	229.93
fit1p	22	167.89	24	130.01	23	116.83	22	110.32
stocfor2	41	113.26	44	98.05	43	85.32	42	90.74
scsd8	16	21.41	19	17.49	19	17.31	19	17.33

Table 6: Comparison of Newton methods: $q = 6$ for Mixed PDN^1; $q = 20$ for Mixed PDN^2; and $q = 40$ for Mixed PDN^3; The preconditioner M used here is given by (9).

Acknowledgment We appreciate the discussions with Fredrik Manne and Alex Pothen on solving systems with a sparse right hand side.

References

[1] R. Bartels, and L. Kaufman, *Cholesky factor updating techniques for rank 2 matrix modifications,* SIAM J. Matrix Anal. Appl, Vol. 10, pp. 557-592, 1989.

[2] V. Baryamureeba, T. Steihaug, and Y. Zhang, *A class of preconditioners for weighted least squares problems,* Technical Report No. 170, Department of Informatics, University of Bergen, 5020 Bergen, Norway, April 30, 1999.

[3] J.M. Bennett, *Triangular factors of Modified matrices*, Numerische Mathematik Vol. 7, pp. 217-221, 1965.

[4] C.H. Bischof, C.T. Pan, and P.T.P. Tang, *A Cholesky up- and downdating algorithm for systolic and SIMD architectures,* SIAM J. Sci. Comput., Vol. 14, pp. 670-676, 1993.

[5] Å. Björck, *Numerical methods for least squares problems*, SIAM, 1996.

[6] T.A. Davis, and W.W. Hager, *Modifying a sparse Cholesky factorization,* Technical Report No. TR-97-025, Department of Computer and Information Science and Engineering, University of Florida. Revised January 25, 1999. To appear, SIAM J. Matrix Analysis and Applications.

[7] D.M. Gay, *Electronic mail distribution of linear programming test problems,* Mathematical Programming Society COAL Newsletter, No. 13, pp.10-12, 1985.

[8] P.E. Gill, G.H. Golub, W. Murray, and M.A. Saunders, *Methods for modifying matrix factorizations,* Mathematics of Computation, Vol. 28, pp. 505-535, 1974.

[9] G.H. Golub, and C.H. Van Loan, *Matrix computations,* The Johns Hopkins University Press, Third Edition, 1996.

[10] W.W. Hager, *Updating the inverse of a matrix,* SIAM Review, Vol. 31, pp. 221-239, 1989.

[11] G. Strang, *A framework for equilibrium equations,* SIAM Review, Vol. 30, pp. 283-297, 1988.

[12] S. Mehrotra, *Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods,* ORSA J. Comput., 4(1992), pp. 103-118.

[13] W. Wang, and D.P. O'Leary, *Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming* Technical Report No. CS-TR-4011, Computer Science Department, University of Maryland, April 1999.

Application of a

Class of Preconditioners to

Large-Scale Linear Programming Problems

Venansius Baryamureeba   Trond Steihaug   Yin Zhang

Presented at the European Conference on
Parallel Computing (Euro-Par99),
Toulouse, France,
August 31–September 3, 1999.

Venansius Baryamureeba, Trond Steihaug and Yin Zhang. Application of a Class of Preconditioners to Large-Scale Linear Programming Problems. P. Amestoy, P. Berger, M. Dayd, I. Duff, V. Frayss, L. Giraud and D. Ruiz, (eds), EuroPar'99 Parallel Processing. Lecture Notes in Computer Science, No. 1685, Springer-Verlag, 1999, pp. 1044-1048

Application of a Class of Preconditioners to Large Scale Linear Programming Problems [1]

Venansius Baryamureeba [2] Trond Steihaug [3] Yin Zhang[4]

Abstract

In most interior point methods for linear programming, a sequence of weighted linear least squares problems are solved, where the only changes from one iteration to the next are the weights and the right hand side. The weighted least squares problems are usually solved as weighted normal equations by the direct method of Cholesky factorization. In this paper, we consider solving the weighted normal equations by a preconditioned conjugate gradient method at every other iteration. We use a class of preconditioners based on a low rank correction to a Cholesky factorization obtained from the previous iteration. Numerical results show that when properly implemented, the approach of combining direct and iterative methods is promising.

Key Words. Weighted linear least squares, Parallel processing, Preconditioners, Linear programming, Primal-dual infeasible interior point algorithms.

A Introduction

The class of preconditioners we will consider is a low rank correction of a Cholesky factorization of a weighted normal equation coefficient matrix. The combination of a direct method and an iterative method was reported by Karmarkar and Ramakrishnan [8]. A low rank correction to the matrix was used by Goldfarb and Liu [6] to prove a low complexity bound for convex quadratic programming. A frequently used iterative method [3, 8, 9] is the preconditioned conjugate gradient method on the normal equations.

Throughout this paper we use the following notation: $\min_i$ or $\max_i$ is for all i for which the argument is defined. For any matrix A, A_{ij} is the element in the i-th row and j-th column, A_j is the j-th column, and $A_{j\bullet}$ is the j-th row. The symbol I is used to denote the identity matrix; its size will always be apparent from the context. For any square matrix X, $\lambda_i(X)$ are the eigenvalues of X arranged in non decreasing order. The letters L and R represent lower triangular factors or lower Cholesky factors, and the type of factors will always be apparent from the context.

B Problem

Consider a primal linear programming problem in standard form:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to:} & Ax = b, \;\; x \geq 0 \\ & A \in \Re^{m\times n}, \;\; b \in \Re^m, \;\; c, x \in \Re^n. \end{array} \tag{1}$$

It can be shown (see [2]) that the normal equation form for a primal-dual interior point method corresponding to problem (1) is of the form

$$AGA^T y = AGh \tag{2}$$

where G is a diagonal and positive definite matrix. At every interior point iteration, (2) is solved.

In the following, the matrix A will be a full rank m-by-n matrix, $m \leq n$.

---

[1]P. Amestoy, P. Berger, M. Dayd, I. Duff, V. Frayss, L. Giraud and D. Ruiz, (eds), EuroPar'99 Parallel Processing. Lecture Notes in Computer Science, No. 1685, Springer-Verlag, 1999, pp. 1044-1048.

[2]Department of Informatics, Uinversity of Bergen, N-5020 Bergen, Norway, barya@ii.uib.no

[3]Department of Informatics, Uinversity of Bergen, N-5020 Bergen, Norway, trond@ii.uib.no

[4]Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005, USA, zhang@caam.rice.edu

C The Class of Preconditioners

Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal matrices. For a given index set $\mathcal{Q} \subseteq \{1 \le j \le n |\ G_{jj} \ne H_{jj}\}$, let the $n \times n$ diagonal matrices $\bar{H}$ and $\bar{G}$ be given by

$$\bar{H}_{jj} = \begin{cases} H_{jj} & \text{if } j \in \mathcal{Q} \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } \bar{G}_{jj} = \begin{cases} G_{jj} & \text{if } j \in \mathcal{Q} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the diagonal matrix K defined in (3) is positive definite.

$$K = H + \bar{G} - \bar{H}. \tag{3}$$

Note that $K_{jj} = G_{jj}$ if $j \in \mathcal{Q}$, otherwise $K_{jj} = H_{jj}$. In the mixed interior point method H is a weight matrix at a previous iteration where a direct method is used to solve (2), and G is the weight matrix at the current interior point iteration. We can thus assume that we have a factorization of $AHA^T = LL^T$.

We denote the number of elements in $\mathcal{Q}$ by $q = |\mathcal{Q}|$. Let $\bar{A} \in \Re^{m\times q}$ consist of all columns A_j such that $j \in \mathcal{Q}$. Further, let $\bar{D} \in \Re^{q\times q}$ be the nonzero diagonal submatrix of $\bar{G} - \bar{H}$ corresponding to the submatrix $\bar{A}$. Then

$$AKA^T = A(H + \bar{G} - \bar{H})A^T = AHA^T + \bar{A}\bar{D}\bar{A}^T. \tag{4}$$

An approximation to AGA^T is AKA^T. Observe that if $\mathcal{Q} = \{1 \le j \le n \mid G_{jj} \ne H_{jj}\}$ then $AKA^T = AGA^T$ so the approximation can be made arbitrary good.

Theorem C.1 *[2]: Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal. Let γ_j be the sorted elements of G_{jj}/H_{jj} in non-decreasing order*

$$\gamma_1 = \min_j \{G_{jj}/H_{jj}\} \le \gamma_2 \le \cdots \le \gamma_n = \max_j \{G_{jj}/H_{jj}\}.$$

Let $\mathcal{Q}$ consist of the indices corresponding to the q_1 largest and the q_2 smallest diagonal elements of $H^{-1}G$. Then

$$\min\{\gamma_{q_2+1}, 1\} \le \lambda_i((AKA^T)^{-1}AGA^T) \le \max\{\gamma_{n-q_1}, 1\}.$$

Theorem C.1 suggests choosing $\mathcal{Q}$ to consist of indices j corresponding to q_1 largest and q_2 smallest diagonal elements of $H^{-1}G$. Further, the condition number of the preconditioned matrix is bounded above

$$\kappa((AKA^T)^{-1}AGA^T) \le \frac{\max\{\gamma_{n-q_1}, 1\}}{\min\{\gamma_{q_2+1}, 1\}}.$$

We can keep the condition number of $(AKA^T)^{-1}AGA^T$ bounded by a prescribed constant if we carefully choose q_1 and q_2 sufficiently large. The next result from [2] gives a lower bound on the condition number.

Theorem C.2 *Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal. Let $|\mathcal{Q}| = q_1 + q_2 < m$, where q_1 is the number of indices in $\mathcal{Q}$ where $G_{jj}/H_{jj} > 1$. Let K be defined as in (3). Then*

$$\kappa((AKA^T)^{-1}AGA^T) \ge \frac{\lambda_{m-q_1}((AHA^T)^{-1}AGA^T)}{\lambda_{q_2+1}((AHA^T)^{-1}AGA^T)}.$$

Wang and O'Leary [11] discuss a strategy of preconditioning the normal equation system based on formulation (4). In their implementation, they choose the index set $\mathcal{Q}$ to consist of the indices corresponding to the largest values of $|G_{jj} - H_{jj}|$.

C.1 Computing the Preconditioner

We consider two approaches of solving the linear system $AKA^Tz = r$, namely the Sherman Morrison-Woodbury formula approach, and the approach based on updating the triangular factors.

C.1.1 Sherman Morrison-Woodbury Formula

Let $AHA^T = LL^T$. Applying the Sherman Morrison-Woodbury formula ([7] pp. 50) to $AKA^T = LL^T + \bar{A}\bar{D}\bar{A}^T$ followed with some manipulation yields

$$(AKA^T)^{-1} = L^{-T}(I - VF^{-1}V^T)L^{-1}, \tag{5}$$

where $V = L^{-1}\bar{A}$ and $F = \bar{D}^{-1} + V^TV$. Solving $AKA^Tz = r$ for z is equivalent to computing

$$z = (AKA^T)^{-1}r = L^{-T}[d - (V(F^{-1}(V^Td)))], \tag{6}$$

where $d = L^{-1}r$. To compute z in (6) we need to store L, V, d, and the factors of F. Note that F is a $q \times q$ symmetric matrix that may be indefinite.

C.1.2 Updating the Triangular Factors

Suppose we have the factorization $AHA^T = LDL^T$. Now, we want to compute R and T so that $AKA^T \equiv LDL^T + \bar{A}\bar{D}\bar{A}^T = RTR^T$. Since the sparsity structure of AKA^T is the same as for AHA^T, sparsity structure of R is the same as for L. An algorithm for updating factors of dense and sparse matrices is given in Baryamureeba and Steihaug [2]. For algorithms updating factors of dense matrices see [2] and references therein. We state the algorithms in Figure 1. The sparse-case algorithm is very effective when there are either many zero rows in $\bar{A}$ or few nonzero elements in the unit lower triangular factor L.

Algorithm C.1 *Dense case*
Define $V = \bar{A}$; $C = \bar{D}$

for $i = 1, \ldots, m$ **do**
  $p = (V_{i\bullet}C)^T$
  $T_{ii} = D_{ii} + V_{i\bullet}\, p$
  $u = (1/T_{ii})\, p$
  $C \leftarrow C - u\, p^T$
  for $j = i+1, \ldots, m$ **do**
    $V_{j\bullet} \leftarrow V_{j\bullet} - L_{ji}V_{i\bullet}$
    $R_{ji} = L_{ji} + V_{j\bullet}\, u$

Algorithm C.2 *Sparse case*
Define $V = \bar{A}$; $C = \bar{D}$; $R = L$; $T = D$

for $i = 1, \ldots, m$ **do**
  if $V_{i\bullet} \neq 0$ **then**
    $p = (V_{i\bullet}C)^T$
    $T_{ii} = D_{ii} + V_{i\bullet}\, p$
    $u = (1/T_{ii})\, p$
    $C \leftarrow C - u\, p^T$
    for $j = i+1, \ldots, m$ **do**
      if $L_{ji} \neq 0$ **then**
        $V_{j\bullet} \leftarrow V_{j\bullet} - L_{ji}V_{i\bullet}$
        $R_{ji} = L_{ji} + V_{j\bullet}\, u$

Figure 1: Updating the triangular factors

The major computation in an interior point algorithm is in solving linear systems. Demmel et al. [4] discuss parallel algorithms for conjugate gradients and computing Cholesky factors. The construction of the preconditioner (updating of the triangular factors) can be done in parallel.

D Numerical Results

Here, we give some numerical results using the same implementation details as in [2]. The test problems are from the Netlib set [6] of linear programs. The test code is implemented in MATLAB The variable ϵ is a proximity measure of the interior point iterations to a solution of the linear programming problem and τ is the number of iterations (or corrections) allowed for the preconditioned conjugate gradient method. The percentages in Table 1 are based on that approximately half of the direct solves are replaced by an iterative solution. The gain is therefor approximately the ratio of the difference between the two methods and half of the total time. The results in Table 1 show that the mixed primal-dual Newton (mixed PDN) interior-point method, which alternatively uses a direct (Cholesky factorization) method and a preconditioned (with the preconditioner described in Section 3) conjugate gradient method to solve (2), competes favourably with the primal-dual Newton (PDN) interior-point method on large-scale problems.

The numerical results show that the mixed PDN method is promising and merits further study. In this paper the mixed PDN method is based on the odd-even alternation technique. Other dynamic alternation schemes [11] which could be based on the easily computable bounds on condition numbers should be investigated.

Problem name	Rows (m)	Columns (n)	PDN		Mixed PDN		Gain %
			iter	cpu-time	iter	cpu-time	
czprob	737	3141	56	89.01	59	59.58	66
d2q06c	2171	5831	54	544.85	57	421.33	45
d6cube	404	6184	42	317.58	45	239.16	51
stocfor2	2157	3045	41	113.26	44	98.05	27
scsd8	397	2750	16	21.41	19	17.49	37

Table 1: Comparison of two implementations of primal-dual Newton interior point method. If $\epsilon \geq 0.1$ then $\tau = 7$, otherwise $\tau = 40$. Fixed value of $q = 6,\ q_1 = 3$.

References

[1] V. Baryamureeba, and T. Steihaug, *Computational issues for a new class of preconditioners,* To appear in the Proceedings for the 2nd Workshop on Large-Scale Scientific Computations, 1999.

[2] V. Baryamureeba, T. Steihaug and Y. Zhang, *A class of preconditioners for weighted least squares problems,* Technical Report No. 170, Department of Informatics, University of Bergen, 5020 Bergen, Norway, April 30, 1999.

[3] T.J. Carpenter and D.F. Shanno, *An interior point method for quadratic programs based on conjugate projected gradients,* Computational Optimization and Applications, Vol. 2, pp. 5-28, 1993.

[4] J. W. Demmel, M.T. Heath, and H.A. Van der Vorst, *Parallel numerical linear algebra,* Acta Numerica, pp. 111-197, 1993.

[5] D.M. Gay, *Electronic mail distribution of linear programming test problems,* Mathematical Programming Society COAL Newsletter, No. 13, pp.10-12, 1985.

[6] D. Goldfarb, and S. Liu, *An $O(n^3L)$ primal interior-point algorithm for convex quadratic programming,* Mathematical Programming, Vol. 49, pp. 325–340, 1991.

[7] G. H. Golub, and Charles H. Van Loan, *Matrix computations,* Third Edition, 1996.

[8] N.K. Karmarkar and K.G. Ramakrishnan, *Computational results of an interior point algorithm for large-scale linear programming,* Mathematical Programming, Vol. 52, pp. 555-586, 1991.

[9] S. Mehrotra, *Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods,* ORSA J. Comput., 4(1992), pp. 103-118.

[10] W. Wang, and D.P. O'Leary, *Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming* Technical Report No. CS-TR-4011, Computer Science Department, University of Maryland, April 1999. (Revised version of TR-3560, November 1995)

Application of a

New Class of Preconditioners to

Large-Scale Linear Programming Problems

Venansius Baryamureeba Trond Steihaug

Presented at the 1999 International Conference on
Preconditioning Techniques for Large Sparse Matrix Problems in
Industrial Applications (Sparse'99),
Minneapolis, Minnesota, USA,
June 10-12, 1999.

Application of a new class of preconditioners to large-scale linear programming problems [1]

Venansius Baryamureeba Trond Steihaug

Department of Informatics
University of Bergen
5020 Bergen Norway

Abstract

In every primal-dual interior point method, a sequence of weighted linear least squares problems are solved, where the only change from one problem to the next are the weights and the right hand side (or data). We propose a mixed primal-dual method where we solve the weighted least squares problems with a direct method for every even interior point iteration and an iterative method for every odd iteration. A class of preconditioners based on a low rank correction of a Cholesky factorization of a weighted normal equation coefficient matrix is introduced.

Key Words. Weighted linear least squares, Preconditioners, Preconditioned conjugate gradient for least squares, Linear programming, Primal-dual infeasible interior point algorithms.

A Introduction

An interior point algorithm solves the linear programming problem by generating a sequence of interior points from an initial interior point. At every interior point iteration weighted linear least squares problems are solved. The new class of preconditioners is based on a low rank correction of a Cholesky factorization of a weighted normal equation coefficient matrix. The condition number of the preconditioned coefficient matrix can be kept below a prescribed constant for special choices of the low rank corrections. The numerical results in Section 5 indicate that the proposed mixed primal-dual method performs at least as good as a primal-dual Newton method on large-scale problems.

Throughout this paper we use the following notation. $\min_i$ or $\max_i$ is for all i for which the argument is defined. For any matrix X, $nnz(X)$ is the number of nonzero elements in X. For any matrix A, A_{ij} is the element in the i-th row and j-th column, A_j is the j-th column, and $A_{j\bullet}$ is the j-th row. The vector $e = (1, \dots, 1)^T$ is an n-vector of all ones. The symbol I is used to denote the identity matrix; its dimensions will always be apparent from the context. For any square matrix X, $\lambda_i(X)$ are the eigenvalues of X arranged in non-decreasing order. If X has real eigenvalues, then $\lambda_{\min}(X)$ and $\lambda_{\max}(X)$ denote the smallest and largest eigenvalues of X respectively. Thus, $\lambda_{\min}(X) = \lambda_1(X) \leq \lambda_2(X) \leq \cdots \leq \lambda_m(X) = \lambda_{\max}(X)$. The letters L and R represent lower triangular factors or lower Cholesky factors, and the type of factors will always be apparent from the context. Finally, A is an $m \times n$ $(m \leq n)$ matrix with full rank m.

B Problem

Consider a primal linear programming problem in standard form:

$$\begin{array}{lll} & \text{minimize} & c^T x \\ (1) & \text{subject to:} & Ax = b, \;\; x \geq 0 \\ & & A \in \Re^{m\times n}, \;\; b \in \Re^m, \;\; x \in \Re^n, \, m \leq n. \end{array}$$

It can be shown (see [3]) that the weighted least squares problem corresponding to (1) for a primal-dual interior point method is of the form

$$\min_{\Delta y} \|A^T \Delta y - h\|_G, \tag{2}$$

[1]Presented at "The 1999 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Industrial Applications", Minneapolis, Minnesota, June 10-12, 1999.

where , $\Delta y \in \Re^m$, $h \in \Re^n$, and for an n-vector z, the G-norm is given by $\|z\|_G^2 = z^T G z$. The weight matrix G is positive definite and diagonal. This weight matrix and the n-vector h change from iteration to iteration. The solution of (2) is given by the normal equations

$$AGA^T \Delta y = AGh. \tag{3}$$

In an interior point algorithm, linear systems of the form (3) are solved at every iteration.

C The class of preconditioners

Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal matrices. For a given index set $\mathcal{Q} \subseteq \{j | 1 \le j \le n \text{ and } G_{jj} \ne H_{jj}\}$, let the $n \times n$ diagonal matrices $\bar{H}$ and $\bar{G}$ be given by

$$\bar{H}_{jj} = \begin{cases} H_{jj} & \text{if } j \in \mathcal{Q} \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \bar{G}_{jj} = \begin{cases} G_{jj} & \text{if } j \in \mathcal{Q} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the diagonal matrix K defined in (4) is positive definite.

$$K = H + \bar{G} - \bar{H}. \tag{4}$$

Note that $K_{jj} = G_{jj}$ if $j \in \mathcal{Q}$, otherwise $K_{jj} = H_{jj}$. In the mixed interior point method, H is the weight matrix at the previous iteration and G is the weight matrix at the current interior point iteration. We have a factorization of AHA^T and will use $(AKA^T)^{-1}$ as an approximation to $(AGA^T)^{-1}$. Let $\bar{A} \in \Re^{m\times q}$ consist of all columns A_j such that $j \in \mathcal{Q}$. Let $\bar{D} \in \Re^{q\times q}$ be the nonzero diagonal submatrix of $\bar{G} - \bar{H}$ corresponding to the submatrix $\bar{A}$. Then

$$AKA^T = A(H + \bar{G} - \bar{H})A^T = AHA^T + \bar{A}\bar{D}\bar{A}^T. \tag{5}$$

If $\mathcal{Q} = \{j : | 1 \le j \le n \text{ and } G_{jj} \ne H_{jj}\}$ then $AKA^T = AGA^T$. However, it will become evident later that computing solutions to linear systems of the form

$$AKA^T v = d \tag{6}$$

can be done much cheaper than solving $AGA^T v = d$ provided $\mathcal{Q}$ has few elements.

Wang and O'Leary [10, 11] discuss a strategy of preconditioning the normal equation system based on formulation (5). They choose the index set $\mathcal{Q}$ to consist of the indices corresponding to the large $|G_{jj} - H_{jj}|$. This kind of choice does not guarantee a small condition number of the preconditioned matrix especially in the last stage of the interior point algorithm as we will see later. Further, this may slow down the convergence of the conjugate gradient iteration.

We use the following result:

Theorem C.1 *[3] Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal. Let γ_j be the sorted elements of G_{jj}/H_{jj} in non-decreasing order, i.e.,*

$$\gamma_1 = \min_j \{G_{jj}/H_{jj}\} \le \gamma_2 \le \cdots \le \gamma_n = \max_j \{G_{jj}/H_{jj}\}.$$

Let $\mathcal{Q}$ consist of the indices corresponding to the q_1 largest and the q_2 smallest diagonal elements of $H^{-1}G$. Then

$$\min\{\gamma_{q_2+1}, 1\} \le \lambda_i((AKA^T)^{-1}AGA^T) \le \max\{\gamma_{n-q_1}, 1\}. \blacksquare$$

Theorem C.1 suggests choosing $\mathcal{Q}$ to consist of indices corresponding to the largest and smallest nonzero elements of $H^{-1}G$. It also gives a bound on the eigenvalues of $(AKA^T)^{-1}AGA^T$ for this particular choice of $\mathcal{Q}$. Further, the condition number
$\kappa((AKA^T)^{-1}AGA^T)$ is bounded above by

$$\kappa((AKA^T)^{-1}AGA^T) \le \frac{\max\{\gamma_{n-q_1}, 1\}}{\min\{\gamma_{q_2+1}, 1\}}.$$

We can keep the condition number of $(AKA^T)^{-1}AGA^T$ bounded above by a predescribed constant if we carefully choose q_1 and q_2 sufficiently large. The next result gives a lower bound on the condition number.

Theorem C.2 *[3] Let $G, H \in \Re^{n\times n}$ be positive definite and diagonal. Let $|\mathcal{Q}| = q_1 + q_2 < m$, where q_1 is the number of indices in $\mathcal{Q}$ where $G_{jj}/H_{jj} > 1$. Let K be defined as in (4). Then*

$$\kappa((AKA^T)^{-1}AGA^T) \geq \frac{\lambda_{m-q_1}((AHA^T)^{-1}AGA^T)}{\lambda_{q_2+1}((AHA^T)^{-1}AGA^T)}. \blacksquare$$

C.1 The index set $\mathcal{Q}$

The choice of index set $\mathcal{Q}$ defines the class of preconditioners $(AKA^T)^{-1}$. We suggest to choose $\mathcal{Q}$ as follows: If $\lambda_{\min}(H^{-1}G) < 1$ and $\lambda_{\max}(H^{-1}G) > 1$ then choose $\mathcal{Q}$ to comprise of the indices j corresponding to the q_1 largest and the q_2 smallest diagonal elements of $H^{-1}G$. If $\lambda_{\min}(H^{-1}G) > 1$ then choose $\mathcal{Q}$ to comprise of indices j corresponding to the q largest diagonal elements of $H^{-1}G$. Finally, if $\lambda_{\max}(H^{-1}G) < 1$ then choose $\mathcal{Q}$ to comprise of the indices j corresponding to the q smallest diagonal elements of $H^{-1}G$.

C.2 Computing the preconditioner

We shall consider two approaches of computing solutions to linear systems of the form (6), namely the Sherman-Morrison-Woodbury formula approach, and the approach based on updating the triangular factors.

C.2.1 Sherman-Morrison-Woodbury formula

To simplify the notation in the rest of this section, let $M = (AKA^T)^{-1}$. For $AKA^T = (AHA^T + \bar{A}\bar{D}\bar{A}^T)$ the Sherman-Morrison-Woodbury formula ([8] pp. 50) gives

$$M = (AHA^T)^{-1} - (AHA^T)^{-1}\bar{A}(\bar{D}^{-1} + \bar{A}^T(AHA^T)^{-1}\bar{A})^{-1}\bar{A}^T(AHA^T)^{-1}.$$

Let $LL^T = AHA^T$ and compute $V = L^{-1}\bar{A}$. Thus

$$M = (LL^T)^{-1} - L^{-T}V(\bar{D}^{-1} + V^TV)^{-1}V^TL^{-1} = L^{-T}(I - VF^{-1}V^T)L^{-1}, \tag{7}$$

where $F = \bar{D}^{-1} + V^TV$. For an m-vector d we have

$$Md = L^{-T}[r - (V(F^{-1}(V^Tr)))],$$

where $r = L^{-1}d$. In computing Md we need to store L, V, and the factors of F. Note that F is symmetric but may be indefinite.

An alternative for computing $V = L^{-1}\bar{A}$ that further exploits the sparsity of $\bar{A}$ is the following: First, compute the number of distinct row indices p in $\bar{A}$. If $p \ll q$, let E_i be the i-th unit vector corresponding to row index i. Let E be the $m \times p$ matrix comprising of the columns E_i for all distinct row indices i. Then $L^{-1}\bar{A} = (L^{-1}E)(E^T\bar{A})$.

C.2.2 Updating the triangular factors

In this approach $AHA^T = LDL^T$. We compute R and T so that $LDL^T + \bar{A}\bar{D}\bar{A}^T = RTR^T$. The algorithm by Bennett [4] for updating triangular factors is known to be stable [7] when applied to the positive definite matrix $LDL^T + \bar{A}\bar{D}\bar{A}^T$, and it can be shown to be identical to Algorithm C.1. In computing R and T, L and D can be used to store the elements of R and T respectively. Algorithm C.1 is an extension of the algorithm by Gill et al. [7] to rank q corrections for $q \geq 1$.

Baryamureeba and Steihaug [2] extend Algorithm C.1 to sparse matrices. The sparse algorithm (Algorithm C.2) is developed for a special class of problems where $\bar{A}$ consists of columns from A. In this algorithm we set $R_{ji} = 0$ whenever $L_{ji} = 0$ to avoid fill in.

D The iterative solver–PCGLS

The unpreconditioned conjugate gradient least squares (CGLS) algorithm in Björck ([5] pp. 288) is designed to solve linear systems of the form $A^TAy = Ah$. Here, we rewrite the preconditioned form to handle weighted problems of the form $AGA^Ty = AGh$. The preconditioner M used in this preconditioned algorithm (PCGLS) is an approximation to $(AGA^T)^{-1}$.

Algorithm C.1 *Dense algorithm*
Define $V = \bar{A}$; $C = \bar{D}$

for $i = 1, \ldots, m$ **do**
$p = (V_{i\bullet} C)^T$
$T_{ii} = D_{ii} + V_{i\bullet}\, p$
$u = (1/T_{ii})\, p$
$C \leftarrow C - u\, p^T$
for $j = i+1, \ldots, m$ **do**
$V_{j\bullet} \leftarrow V_{j\bullet} - L_{ji} V_{i\bullet}$
$R_{ji} = L_{ji} + V_{j\bullet}\, u$

Algorithm C.2 *Sparse algorithm*
Define $V = \bar{A}$; $C = \bar{D}$; $R = L$; $T = D$

for $i = 1, \ldots, m$ **do**
if $V_{i\bullet} \neq 0$ **then**
$p = (V_{i\bullet} C)^T$
$T_{ii} = D_{ii} + V_{i\bullet}\, p$
$u = (1/T_{ii})\, p$
$C \leftarrow C - u\, p^T$
for $j = i+1, \ldots, m$ **do**
if $L_{ji} \neq 0$ **then**
$V_{j\bullet} \leftarrow V_{j\bullet} - L_{ji} V_{i\bullet}$
$R_{ji} = L_{ji} + V_{j\bullet}\, u$

Table 1: Updating the triangular factors for matrices of the form: $RTR^T = LDL^T + \bar{A}\bar{D}\bar{A}^T$

Algorithm D.1 *PCGLS*
y_0 is the initial starting vector
$r_0 = h - A^T y_0,\ v_0 = AGr_0,\ p_0 = s_0 = Mv_0,\ \gamma_0 = s_0^T v_0$
$k \leftarrow 0$
While not converged **do**
$q_k = A^T p_k$
$\alpha_k = \frac{\gamma_k}{q_k^T G q_k}$
$y_{k+1} = y_k + \alpha_k p_k$
$r_{k+1} = r_k - \alpha_k q_k$ (1.c)
$v_{k+1} = AGr_{k+1}$
$s_{k+1} = Mv_{k+1}$
$\gamma_{k+1} = s_{k+1}^T v_{k+1}$
$\beta_k = \frac{\gamma_{k+1}}{\gamma_k}$
$p_{k+1} = s_{k+1} + \beta_k p_k$
$k \leftarrow k+1$
end {**while**}

E Numerical results

In this section we present some numerical results. Here, m and n are the number of rows and columns respectively, after preprocessing the problems [6]. We have implemented the Primal-Dual Newton (PDN), and mixed primal-dual Newton (mixed PDN) algorithms. Our implementation in this paper is done entirely in MATLAB[2]. We use a starting point (x_0, y_0, z_0) that is identical to the one given by Mehrotra [9]. The termination criteria in Algorithm D.1 is when number of iterations has reached a predefined value t or when $\|v_{k+1}\|_2 \leq 10^{-5}$. We terminate the interior point iterations when the relative error ϵ in the interior point algorithm is less than or equal to a predefined relative error tolerance $\epsilon^* = 10^{-5}$ or when we exceed the maximum number of interior point iterations allowed, which is 300. The relative error ϵ is given by

$$\epsilon = \max\left\{ \frac{\|Ax-b\|_2}{\max\{1, \|b\|_2\}}, \frac{\|A^T y + z - c\|_2}{\max\{1, \|c\|_2\}}, \frac{|c^T x - b^T y|}{\max\{1, |c^T x|\}} \right\} \leq \epsilon^*. \tag{8}$$

The step-length parameter is $\tau = 0.99995$, and the barrier parameter $\mu = \sigma(x^T z)/n$, where $\sigma = 0.1$. The direct solver used is the standard solver (**chol** function) in MATLAB. In our implementation, Symmetric minimum degree permutation is used to reduce fill in.

In the rest of this paper we will denote the M matrix defined in (7) by M_1, and M_2 will be given by $M_2 = (RTR^T)^{-1}$. Obviously, $M_1 = M_2$; but we use two distinct symbols because the amounts of computation associated with the two formulas are different.

[2] MATLAB is a registered trademark of The Math Works, Inc

The cost of computing M_1 consists of the cost of computing V and the factors of F in (7). The cost of computing M_2 consists of the cost of computing the triangular factors of AKA^T. We define the break-even point for computing M as the smallest q required for the cost of computing M to exceed the cost of computing AGA^T and its Cholesky factor. We measure cost in floating point operations. Let the cost of an iterative step

Problem information				Computing M		Using M										
							Problem name									
						q	stocfor2		czprob		scsd8		d6cube		d2q06c	
Name	m	n	$\frac{nnz(A)}{m\times n}$	M_1	M_2		M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2	M_1	M_2
czprob	737	3141	0.0041	37	51	1	35	36	2	2	3	3	35	36	67	68
d2q06c	2171	5831	0.0026	156	87	2	34	35	2	2	2	3	35	36	66	67
d6cube	404	6184	0.0151	98	63	3	34	35	2	2	2	3	35	35	66	67
stocfor2	2157	3045	0.0014	110	71	5	33	34	2	2	2	3	34	34	65	65
scsd8	397	2750	0.0079	26	21	6	33	33	2	2	2	3	34	33	65	64
agg3	516	758	0.0122	60	40											
maros	835	1921	0.0063	54	38											

Table 2: Break-even point for computing M is in terms of q. The break-even point for using M is expressed in terms of PCGLS iterations; Interior point iteration = 2.

consist of the cost of computing M for a given q, plus the cost of the PCGLS iterations per step. By a *dense solve*, we mean the computation of $L^{-1}h$ for a dense vector h, and by a *sparse solve*, we mean the computation of $L^{-1}A_j$ for some column of A.

Let the cost of a direct step consist of the cost of computing AGA^T, plus the cost of computing its Cholesky factor and two dense solves. The break-even point for using M is defined as the smallest number of PCGLS iterations required for the cost of an iterative step to exceed the cost of a direct step. For most problems we do not get close to the break-even numbers for using M in Table 2 since we terminate after fewer iterations (except for czprob and scsd8 problems). For problems with small break-even values for using M, Table 2 suggests that we may choose a large q and do few PCGLS iterations per step. Further, the break-even numbers for computing M in Table 2 show that computing the Cholesky factor is much more expensive than carrying out a small number of sparse solves. Thus we may also choose a large q and do few PCGLS iterations. In practice, we compare the cost of two sparse solves ($L^{-T}[L^{-1}A_j]$) with the cost of one PCGLS iteration at the beginning of the interior point algorithm and use this to decide whether to choose a large q and do few PCGLS iterations or choose a small q and do many PCGLS iterations in the rest of the interior point algorithm.

Problem	Rows	Columns	PDN		Mixed PDN	
name	(m)	(n)	iter	cpu-time	iter	cpu-time
czprob	737	3141	56	89.01	59	59.58
d2q06c	2171	5831	54	544.85	57	421.33
d6cube	404	6184	42	317.58	45	239.16
stocfor2	2157	3045	41	113.26	44	98.05
scsd8	397	2750	16	21.41	19	17.49

Table 3: Comparison of Newton methods; If $\epsilon \geq 0.1$ then $t = 7$, otherwise $t = 40$, where t is the maximum number of iterations allowed in PCGLS. We use M_1 and set $q = 6$.

The numerical experiments in Tables 2 provide guidelines for selecting q based on floating point operations. Similar experiments can be carried out based on CPU time and a similar analysis given. The main cost of computing an inexact Newton step (PCGLS step) is mainly due to the sparse solves and dense solves in computing and using M respectively. The idea is to use a small t when the error in the interior point algorithm is still big, and a large t during the last stage of the algorithm when high accuracy is required. Table 3 shows that mixed PDN method costs less than PDN method on large sparse problems.

F Concluding remarks

The numerical results in this paper show that the mixed PDN approach is very promising and merits further study. There are many possibilities for improving the efficiency of the mixed PDN approach. Obviously, the even-odd alternation discussed in this paper is not generally optimal. Dynamic schemes of alternation based on easily computable bounds on condition numbers should be investigated. In addition, a mixture of the PCGLS steps with the predictor-corrector steps may also be promising [10].

References

[1] E. D. Andersen, J. Gondzio, C. Meszaros, X. Xu, *Implementation of interior point methods for large-scale linear programs,* T.Terlaky(ed), Interior Point Methods of Mathematical Programming pp. 189-252. Kluwer Academic Publishers, 1996. Printed in Netherlands.

[2] V. Baryamureeba, and T. Steihaug, *Computational issues for a new class of preconditioners,* To appear in the Proceedings for the 2nd Workshop on Large-Scale Scientific Computations, 1999.

[3] V. Baryamureeba, T. Steihaug and Y. Zhang, *A class of preconditioners for weighted least squares problems,* Technical Report No. 170, Department of Informatics, University of Bergen, 5020 Bergen, Norway, April 30, 1999.

[4] J.M. Bennett, *Triangular factors of Modified matrices*, Numerische Mathematik Vol. 7, pp. 217-221, 1965.

[5] Å. Björck, *Numerical methods for least squares problems*, SIAM, 1996.

[6] D.M. Gay, *Electronic mail distribution of linear programming test problems,* Mathematical Programming Society COAL Newsletter, No. 13, pp.10-12, 1985.

[7] P.E. Gill, G.H. Golub, W. Murray and M.A. Saunders, *Methods for modifying matrix factorizations,* Mathematics of Computation, Vol. 28, pp. 505-535, 1974.

[8] G. H. Golub, and C.H. Van Loan, *Matrix computations,* Third Edition, 1996.

[9] S. Mehrotra, *Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods,* ORSA J. Comput., 4(1992), pp. 103-118.

[10] W. Wang, and D.P. O'Leary, *Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming* Preprint (1999)

[11] W. Wang, and D.P. O'Leary, *Adaptive use of iterative methods in interior point methods for linear programming*

[12] J. H. Wilkinson, *The Eigenvalue Problem*, 1965.