

# A New Class of Preconditioners for Large-Scale Linear Systems from Interior Point Methods for Linear Programming

A. R. L. Oliveira\*      D. C. Sorensen†

October 27, 2004

## Abstract

A new class of preconditioners for the iterative solution of the linear systems arising from interior point methods is proposed. For many of these methods, the linear systems are symmetric and indefinite. The system can be reduced to a system of normal equations which is positive definite. We show that all preconditioners for the normal equations system have an equivalent for the augmented system while the opposite is not true. The new class of preconditioners works better near a solution of the linear programming problem when the matrices are highly ill-conditioned. The preconditioned system can be reduced to a positive definite one. The techniques developed for a competitive implementation are rather sophisticated since the subset of columns is not known a priori. The new preconditioner applied to the conjugate gradient method compares favorably with the Cholesky factorization approach on large scale problems whose Cholesky factorization contains too many nonzero entries.

**Keywords:** linear programming, interior point methods, preconditioning, augmented system.

---

\*Applied Mathematics Department (DMA), State University of Campinas (UNICAMP), C.P. 6065, 13083-970 Campinas, SP, Brazil; aurelio@ime.unicamp.br.

†Department of Computational and Applied Mathematics, Rice University, 77005-1892, Houston TX, (sorensen@caam.rice.edu).

# 1 Introduction

In this work, a new class of preconditioners for the iterative solution of the linear systems arising from interior point methods is proposed. Each iteration of an interior point method involves the solution of one or more linear systems. The most common approach for solving these systems is to reduce an indefinite system, which is known as the augmented system, to a smaller positive definite one, called normal equations (Schur complement).

We will show that every preconditioner for the reduced system yields an equivalent preconditioner for the augmented system but the converse is not true. Therefore, we choose to work with the augmented system because of the greater opportunity to find an effective preconditioner.

The new class of preconditioners avoids computing the normal equations. These preconditioners rely on an  $LU$  factorization of an a priori unknown subset of the constraint matrix columns instead. We also develop some theoretical properties of the preconditioned matrix and reduce it to a positive definite one. Several techniques for an efficient implementation of these preconditioners are presented. Among the techniques used is the study of the nonzero structure of the constraint matrix to speed up the numerical factorization. We also investigate the performance of some particular cases with this preconditioner.

We use the following notation throughout this work. Lower case Greek letters denote scalars, lower case Latin letters denote vectors and upper case Latin letters denote matrices. The symbol  $0$  will denote the scalar zero, the zero column vector and the zero matrix. Its dimension will be clear from context. The dimension of identity matrix  $I$  will be given by a subscript when it is not clear from context. The Euclidean norm is represented by  $\|\cdot\|$  which will also represent the 2-norm for matrices. The relation  $X = \text{diag}(x)$  means that  $X$  is a diagonal matrix whose diagonal entries are the components of  $x$ . The range of the matrix  $A$  will be denoted by  $\mathcal{R}(A)$  and its null space by  $\mathcal{N}(A)$ .

## 2 Primal-Dual Interior Point Methods

Consider the linear programming problem in the *standard form*:

$$\begin{aligned} & \text{minimize} && c^t x \\ & \text{subject to} && Ax = b, \quad x \geq 0, \end{aligned} \tag{2.1}$$

where  $A$  is a full row rank  $m \times n$  matrix and  $c$ ,  $b$  and  $x$  are column vectors of appropriate dimension. Associated with problem (2.1) is the dual linear programming problem

$$\begin{aligned} & \text{maximize} && b^t y \\ & \text{subject to} && A^t y + z = c, \quad z \geq 0, \end{aligned} \tag{2.2}$$

where  $y$  is an  $m$ -vector of free variables and  $z$  is the  $n$ -vector of dual slack variables. The duality gap is defined as  $c^t x - b^t y$ . It reduces to  $x^t z$  for feasible points.

Since Karmarkar [17] presented the first polynomial time interior point method for linear programming, many methods have appeared. Among them, one of the most successful is the predictor–corrector method [22, 20]. In the predictor–corrector approach, the search directions are obtained by solving two linear systems. First we compute the *affine directions*

$$\begin{pmatrix} 0 & I & A^t \\ Z & X & 0 \\ A & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta \tilde{x} \\ \Delta \tilde{z} \\ \Delta \tilde{y} \end{pmatrix} = \begin{pmatrix} r_d \\ r_a \\ r_p \end{pmatrix} \tag{2.3}$$

where  $X = \text{diag}(x)$ ,  $Z = \text{diag}(z)$  and the residuals are given by  $r_p = b - Ax$ ,  $r_d = c - A^t y - z$  and  $r_a = -XZe$ . Then, the search directions  $(\Delta x, \Delta y, \Delta z)$  are computed solving (2.3) with  $r_a$  replaced by  $r_c = \mu e - XZe - \Delta \tilde{X} \Delta \tilde{Z} e$  where  $\mu$  is the centering parameter and  $e$  is the vector of all ones.

## 2.1 Computing the Search Directions

In terms of computational cost, the key step of an iteration is the solution of a linear system like (2.3). Since both systems share the same matrix, we will restrict the discussion to one linear system. By eliminating the variables  $\Delta z$  the system reduces to:

$$\begin{pmatrix} -D & A^t \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r_d - X^{-1} r_a \\ r_p \end{pmatrix} \tag{2.4}$$

where,  $D = X^{-1}Z$  is an  $n \times n$  diagonal matrix and the lower block diagonal matrix 0 has dimension  $m \times m$  (recall that  $A \in \mathfrak{R}^{m \times n}$ ). We refer to (2.4) as the *augmented system*. Eliminating  $\Delta x$  from (2.4) we get

$$AD^{-1}A^t\Delta y = r_p + A(D^{-1}r_d - Z^{-1}r_a) \tag{2.5}$$

which is called the *normal equations*. The diagonal matrix  $D$  changes at each iteration but the other matrices in (2.4) and (2.5) remain fixed.

We remark that the augmented and normal equations systems for problems with bounded variables have the same structure of the systems for the standard form. Therefore, the ideas developed here can be readily applied to these problems.

## 2.2 Approaches for Solving the Linear Systems

Using the Cholesky factorization of the normal equations system for computing the search directions in interior point methods is by far the most widely used approach (see for example [1, 9, 18]). However, the factored matrix can have much less sparsity and is often more ill-conditioned than the matrix of system (2.4).

One way around this problem is to use iterative methods. In most applications, however, it is essential to modify a hard to solve linear system into an equivalent system easier to solve by the iterative method. This technique is known as preconditioning.

Consider the following situation: given  $Kx = r$ , we solve the equivalent linear system  $M^{-1}KN^{-1}\tilde{x} = \tilde{r}$ , where  $\tilde{x} = Nx$  and  $\tilde{r} = M^{-1}r$ . The system is said to be preconditioned and  $M^{-1}KN^{-1}$  is called the preconditioned matrix.

Since the iterative methods require the matrix only for computing matrix-vector products there is no need to compute the normal equations unless the preconditioner depends on it.

Attempts to solve (2.5) using the preconditioned conjugate gradient method have achieved mixed results [18, 19], mainly because the linear systems become highly ill-conditioned as the interior point method approaches an optimal solution.

Therefore, the augmented system began to be considered even though it is indefinite. Implementations using the Bunch-Parlett factorization [6] proved to be more stable but they are slower than solving (2.5) (see [13, 15]). Better results were recently obtained for the symmetric indefinite system [12]. A multifrontal approach applied to the augmented system has been investigated in [11]. Approaches that avoid the 2 dimensional diagonal blocks are given in [21, 28].

In [3], several iterative methods applied to the preconditioned augmented system are compared with the direct approach obtaining good results for some instances.

### 3 The Augmented System

A slightly more general form for the augmented system (2.4) arises naturally in several areas of applied mathematics such as optimization, fluid dynamics, electrical networks, structural analysis and heat equilibrium. In some of these applications, the matrix  $D$  is not necessarily diagonal although it is symmetric positive definite. The techniques for solving the linear systems vary widely within these areas since the characteristics of the system are problem dependent. See for example [5, 26, 29, 16].

In this section we shall work with a more general form for the augmented system:

$$\begin{pmatrix} -D & A^t \\ A & E \end{pmatrix} \quad (3.6)$$

where  $E$  is a positive definite matrix. Vanderbei [27] shows that this form can be always obtained for linear programming problems.

The following lemma supports the choice of considering the augmented system instead of the normal equations when designing preconditioners for iterative methods. It will be shown that for every preconditioner of the normal equations, an equivalent preconditioner can be derived for the augmented system. However, the converse statement is not true.

**Lemma 3.1** *Let (3.6) be nonsingular and  $D$  be symmetric positive definite. Given any nonsingular matrices  $J$  and  $H$  of appropriate dimension, it is possible to construct a preconditioner  $(M, N)$ , such that*

$$M^{-1} \begin{pmatrix} -D & A^t \\ A & E \end{pmatrix} N^{-1} = \begin{pmatrix} I & 0 \\ 0 & J(AD^{-1}A^t + E)H^t \end{pmatrix}.$$

**Proof.** Let  $D = LL^t$  and consider the preconditioner

$$M^{-1} = \begin{pmatrix} -L^{-1} & 0 \\ JAD^{-1} & J \end{pmatrix} \text{ and } N^{-1} = \begin{pmatrix} L^{-t} & D^{-1}A^tH^t \\ 0 & H^t \end{pmatrix}$$

then,

$$M^{-1} \begin{pmatrix} -D & A^t \\ A & E \end{pmatrix} N^{-1} = \begin{pmatrix} I & 0 \\ 0 & J(AD^{-1}A^t + E)H^t \end{pmatrix} \bullet$$

The next lemma shows that for the augmented system (2.4) the converse statement is not true.

**Lemma 3.2** Consider the augmented system given by

$$\begin{pmatrix} -D & A^t \\ A & E \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r_d - X^{-1}r_a \\ r_p \end{pmatrix}$$

and its normal equations  $AD^{-1}A^t + E$  where  $D$  is nonsingular and  $A$  has full row rank. Then any symmetric block triangular preconditioner

$$\begin{pmatrix} F & 0 \\ H & J \end{pmatrix}$$

will result in a preconditioned augmented system whose normal equations are independent of  $F$  and  $H$ .

**Proof.** Consider again the class of symmetric preconditioners given by:

$$\begin{pmatrix} F & 0 \\ H & J \end{pmatrix} \begin{pmatrix} -D & A^t \\ A & E \end{pmatrix} \begin{pmatrix} F^t & H^t \\ 0 & J^t \end{pmatrix} = \begin{pmatrix} -FDF^t & FA^tJ^t - FDH^t \\ JAF^t - HDF^t & C \end{pmatrix}$$

where,  $C = -HDH^t + HA^tJ^t + JAH^t + JEJ^t$ .

Therefore, the preconditioned system will be as follows:

$$\begin{pmatrix} -FDF^t & FA^tJ^t - FDH^t \\ JAF^t - HDF^t & C \end{pmatrix} \begin{pmatrix} \Delta \tilde{x} \\ \Delta \tilde{y} \end{pmatrix} = \begin{pmatrix} F(r_d - X^{-1}r_a) \\ Jr_p + H(r_d - X^{-1}r_a) \end{pmatrix}$$

now, by eliminating  $\Delta \tilde{x}$  we get:

$$J(AD^{-1}A^t + E)J^t\Delta \tilde{y} = J(r_p + AD^{-1}(r_d - X^{-1}r_a)) \quad (3.7)$$

this system does not depend on either  $F$  or  $H$ , thus any choice for these matrices that preserves nonsingularity are valid preconditioners which lead to (3.7). •

These results can be applied in a more general context. For instance  $D$  can be any symmetric positive definite matrix not necessarily diagonal. Moreover, there is no restriction to  $E$  whatsoever. However, if  $E$  is positive semi-definite, the normal equations will be positive definite.

## 4 The Splitting Preconditioner

In this section we will develop and study a class of symmetric preconditioners for the augmented system that exploits its structure. In view of the discussion in previous sections, the preconditioners are designed to avoid forming the normal equations.

## 4.1 Building a Preconditioner

Since the augmented system is naturally partitioned into block form, let us start with the most generic possible block symmetric preconditioner for the augmented system

$$M^{-1} = N^{-t} = \begin{pmatrix} F & G \\ H & J \end{pmatrix}$$

and choose the matrix blocks step by step according to our goals. The preconditioned augmented matrix (2.4) will be the following

$$\begin{pmatrix} -FDF^t + FA^tG^t + GAF^t & -FDH^t + FA^tJ^t + GAH^t \\ -HDF^t + HA^tG^t + JAF^t & -HDH^t + HA^tJ^t + JAH^t \end{pmatrix}.$$

At this point we begin making decisions about the blocks. We start by observing that the lower-right block is critical in the sense that the normal equations matrix  $AD^{-1}A^t$  appears in the expression for many reasonable choices of  $J$  and  $H$ . Setting  $J = 0$  helps to avoid the normal equations. This choice seems to be rather drastic at first glance leaving few options for the selection of the other blocks. But, as we will soon see, the careful selection of the remaining blocks will lead to promising situations. With  $J = 0$  the preconditioned augmented system reduces to

$$\begin{pmatrix} -FDF^t + FA^tG^t + GAF^t & -FDH^t + GAH^t \\ -HDF^t + HA^tG^t & -HDH^t \end{pmatrix}.$$

Let us turn our attention to the off diagonal blocks. If we can make them zero blocks, the problem decouples into two smaller linear systems. Thus, one idea is to write  $F^t = D^{-1}A^tG^t$ . However, this choice leads to  $\mathcal{N}(F^t) \supset \mathcal{N}(G^t)$  and is not acceptable. A more reasonable choice is  $G^t = (HA^t)^{-1}HDF^t$  giving

$$\begin{pmatrix} -FDF^t + FA^tG^t + GAF^t & 0 \\ 0 & -HDH^t \end{pmatrix}.$$

Now, let us decide how to chose  $H$ . The choices  $A$ ,  $AD^{-1}$  or variations of it will not be considered since matrices with the nonzero pattern of the normal equations will appear in the lower-right block and also as part of  $G$ . On the other hand, setting  $H = [I \ 0]P$  where  $P$  is a permutation matrix such

that  $HA^t$  is nonsingular does not introduce a normal equations type matrix. The lower-right block reduces to a diagonal matrix  $-D_B \equiv -HDH^t$  where,

$$PDP^t = \begin{pmatrix} D_N & 0 \\ 0 & D_B \end{pmatrix} \quad (4.8)$$

and we achieve one of the main goals, namely avoiding the normal equations.

Finally, we can concentrate on  $F$  at the upper left block. The choice  $F = D^{-\frac{1}{2}}$  seems to be natural and as we shall see later leads to some interesting theoretical properties for the preconditioned matrix. Summarizing, the final preconditioned matrix takes the form

$$M^{-1} \begin{pmatrix} -D & A^t \\ A & 0 \end{pmatrix} M^{-t} = \begin{pmatrix} -I + D^{-\frac{1}{2}}A^tG^t + GAD^{-\frac{1}{2}} & 0 \\ 0 & -D_B \end{pmatrix} \quad (4.9)$$

where,

$$M^{-1} = \begin{pmatrix} D^{-\frac{1}{2}} & G \\ H & 0 \end{pmatrix}$$

with  $G = H^t D_B^{\frac{1}{2}} B^{-1}$ ,  $HP^t = [I \ 0]$  and  $AP^t = [B \ N]$ .

The price paid for avoiding the normal equations system is to find  $B$  and solve linear systems using it. However, the factorization  $QB = LU$  is typically easier to compute than the Cholesky factorization. In fact, it is known [14] that the sparsity pattern of  $L^t$  and  $U$  is contained in the sparsity pattern of  $R$ , where  $AA^t = R^tR$ , for any valid permutation  $Q$ . In practice, the number of nonzero entries of  $R$  is much larger than the number of nonzero entries of  $L$  and  $U$  added.

Let us make a few observations about the preconditioned matrix (4.9). It has the same inertia as the augmented system (2.4). Also, since the lower-right block matrix has  $m$  negative eigenvalues, the preconditioned matrix

$$S = -I + D^{-\frac{1}{2}}A^tG^t + GAD^{-\frac{1}{2}} \quad (4.10)$$

has  $m$  negative and  $n - m$  positive eigenvalues. Therefore the preconditioned matrix is indefinite except for the odd case where  $m = n$ .

We close this section by showing a property of the matrices in the preconditioned block (4.10) which leads to the results of the next section.

**Lemma 4.1** *Let  $A = [B \ N]$  with  $B$  nonsingular and  $G = H^t D_B^{\frac{1}{2}} B^{-1}$  where  $H = [I \ 0]$ . Then  $G^t D^{-\frac{1}{2}} A^t = AD^{-\frac{1}{2}} G = I$ .*

**Proof.** It is sufficient to show that  $G^t D^{-\frac{1}{2}} A^t = I$ .

$$\begin{aligned} B^{-t} D_B^{\frac{1}{2}} [I \ 0] D^{-\frac{1}{2}} [B \ N]^t &= \\ B^{-t} D_B^{\frac{1}{2}} [D_B^{-\frac{1}{2}} \ 0] [B \ N]^t &= \\ B^{-t} [I \ 0] [B \ N]^t &= I \quad \bullet \end{aligned}$$

This result can be easily extended for any permutation  $A = [B \ N]P$  with  $B$  nonsingular.

## 4.2 Theoretical Properties

In this section we will study some properties of matrices of the type

$$\begin{aligned} K &= -I_n + U^t V^t + VU \\ \text{where } UV &= V^t U^t = I_m \end{aligned} \quad (4.11)$$

and  $U, V^t$  are  $m \times n$  matrices. The preconditioned matrix (4.10) given in the previous section belongs to this class of matrices. First we show that the eigenvalues of  $K$  are bounded away from zero.

**Theorem 4.1** *Let  $\lambda$  be an eigenvalue of  $K$  given by (4.11) where  $U$  and  $V^t \in R^{m \times n}$ , then  $|\lambda| \geq 1$ .*

**Proof.** Let  $v$  be a normalized eigenvector of  $K$  associated with  $\lambda$ , then

$$\begin{aligned} Kv &= \lambda v \\ K^2 v &= \lambda^2 v \\ (I - U^t V^t - VU + U^t V^t VU + VU U^t V^t)v &= \lambda^2 v \\ v + (U^t V^t - VU)(VU - U^t V^t)v &= \lambda^2 v. \end{aligned}$$

Multiplication on the left by  $v^t$  gives  $1 + w^t w = \lambda^2$  where,  $w = (VU - U^t V^t)v$ . Thus, we obtain  $\lambda^2 \geq 1$ .  $\bullet$

**Corollary 4.1** *The preconditioned matrix (4.9) is nonsingular.*

Thus,  $K$  is not only nonsingular but it has no eigenvalues in the neighborhood of zero. The following remarks are useful for showing other important results.

**Remark 4.1** Since  $U$  and  $V^t \in R^{m \times n}$  are such that  $UV = I_m$ ,  $VU$  is an oblique projection onto  $\mathcal{R}(V)$ . Thus, if  $x \in \mathcal{R}(V)$ , then  $VUx = x$ .

**Theorem 4.2** The matrix  $K$  in (4.11) where  $U$  and  $V^t \in R^{m \times n}$  has at least one eigenvalue  $\lambda$  such that  $|\lambda| = 1$ .

**Proof.** Observe that if  $K + I$  is singular,  $K$  has at least an eigenvalue  $\lambda = -1$ . Let us consider three cases:

- (i)  $n > 2m$ ,  $K + I$  is singular, since for any square matrices  $A$  and  $B$   $\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$ .
- (ii)  $n < 2m$ , observe that  $\dim(\text{span}\{\mathcal{R}(U^t) \cup \mathcal{R}(V)\}) \leq n$ . Also,  $U^t$  and  $V$  have rank  $m$  since  $UV = I_m$ . Thus,  $\dim(\mathcal{R}(U^t) \cap \mathcal{R}(V)) > 0$  since  $n < 2m$ . Hence, there is at least one eigenvector  $v \neq 0$  such that  $v \in \mathcal{R}(U^t) \cap \mathcal{R}(V)$  therefore, by Remark 4.1,  $Kv = (-I + U^tV^t + VU)v = -v + v + v = v$ .
- (iii)  $n = 2m$ , from (ii) if  $\mathcal{R}(U^t) \cap \mathcal{R}(V) \neq \{0\}$  there is a eigenvalue  $\lambda = 1$ . Otherwise there exists an eigenvector  $v$  such that  $v_{\mathcal{R}(V)} = 0$  or  $v_{\mathcal{R}(U^t)} = 0$ . Without loss of generality consider that  $v_{\mathcal{R}(V)} = 0$ , where  $v_{\mathcal{R}(V)} \in \mathcal{R}(V)$  and  $v_{\mathcal{N}(V^t)} \in \mathcal{N}(V^t)$ . Thus,  $v \in \mathcal{N}(V^t)$  and there is an eigenpair  $(\theta = \lambda + 1, v)$  where  $VUv = \theta v$ . But then  $\lambda + 1 = (0 \text{ or } 1)$  since  $UV = I$  and from Theorem 4.1 it must be  $\lambda = -1$ .  $\bullet$

**Corollary 4.2** The condition number  $\kappa_2(K)$  in (4.11) is given by  $\max |\lambda(K)|$ .

The proof is immediate from Theorems 4.1, 4.2, the definition of  $\kappa_2(K) = \frac{\sigma_{max}}{\sigma_{min}}$  and recalling that  $K$  is symmetric.

#### 4.2.1 Reduction to Positive Definite Systems

Consider again the indefinite linear system at the left upper block (4.10). It is possible to reduce it to a smaller positive definite system. Expanding the above equation we obtain the following matrix

$$S = P^t \begin{pmatrix} I & D_B^{\frac{1}{2}} B^{-1} N D_N^{-\frac{1}{2}} \\ D_N^{-\frac{1}{2}} N^t B^{-t} D_B^{\frac{1}{2}} & -I \end{pmatrix} P. \quad (4.12)$$

Therefore the problem can be reduced to solve a positive definite linear system involving either matrix

$$I_m + D_B^{\frac{1}{2}} B^{-1} N D_N^{-1} N^t B^{-t} D_B^{\frac{1}{2}} \quad (4.13)$$

or

$$I_{n-m} + D_N^{-\frac{1}{2}} N^t B^{-t} D_B B^{-1} N D_N^{-\frac{1}{2}}. \quad (4.14)$$

These matrices have some interesting theoretical properties related to the indefinite matrix  $K$ .

Let us first define  $W = D_N^{-\frac{1}{2}} N^t B^{-t} D_B^{\frac{1}{2}}$ . Then, the positive definite matrices (4.13) and (4.14) can be written as  $I + W^t W$  and  $I + W W^t$  respectively.

**Lemma 4.2** *The matrices in (4.13) and (4.14) are positive definite and their eigenvalues are greater or equal to one.*

**Proof.** Let  $v$  be a normalized eigenvector of  $I + W W^t$  and  $\theta$  its associated eigenvalue, then

$$\begin{aligned} (I + W W^t)v &= \theta v \\ v^t(v + W W^t v) &= \theta v^t v \\ 1 + u^t u &= \theta \end{aligned}$$

where  $u = W^t v$ . Thus  $\theta \geq 1$ . The proof for  $I + W^t W$  is similar. •

**Remark 4.2** *The matrices in (4.13) and (4.14) have the same set of eigenvalues with the exception of the extra eigenvalue equal to one for the matrix of higher dimension.*

The next result is also important since it relates the eigenpairs of the indefinite matrix with the eigenpairs of the positive definite matrices. Notice that

$$\begin{pmatrix} I & W^t \\ W & -I \end{pmatrix} \begin{pmatrix} I & W^t \\ W & -I \end{pmatrix} = \begin{pmatrix} I + W^t W & 0 \\ 0 & I + W W^t \end{pmatrix}$$

thus if,

$$\begin{pmatrix} I + W^t W & 0 \\ 0 & I + W W^t \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \lambda^2 \begin{pmatrix} u \\ v \end{pmatrix}$$

then  $(\theta, u)$  is an eigenpair of  $I + W^tW$  and  $(\theta, v)$  is an eigenpair of  $I + WW^t$ , where  $\theta = \lambda^2$ .

Therefore the indefinite system can still be an option for solving the linear system since it has a better eigenvalue spectrum distribution. However, experiments solving (4.13) with the conjugate gradient method obtained better results than using the SYMMLQ method for solving (4.10).

From (4.12) we can get

$$(PSP^t)^{-1} = \begin{pmatrix} I - W^tTW & W^tT \\ TW & -T \end{pmatrix}$$

where,  $T = (I + WW^t)^{-1}$ . The smallest eigenvalue of  $S^{-1}$  is given by

$$|\lambda|^2 = \left\| \begin{array}{c} u - W^tTWu + W^tTv \\ TWu - W^tTv \end{array} \right\|$$

where,  $\|u^t v^t\| = 1$ . Let  $\tilde{N} = B^{-1}N$ , which can be seen as a scaling of the linear programming problem. Close to a solution, at least  $n - m$  entries of  $D$  are large. Thus, with a suitable choice of the columns of  $B$ , the diagonal entries of  $D_B$  and  $D_N^{-1}$  are very small close to a solution. In this situation,  $W = D_N^{-\frac{1}{2}}\tilde{N}^tD_B^{\frac{1}{2}}$  approaches the zero matrix,  $T$  approaches the identity matrix and both the largest eigenvalue of  $S$  and  $\kappa_2(S)$  approach one.

### 4.3 Choosing the Set of Columns

Given a good choice of columns of  $A$  to form  $B$  this preconditioner should work better close to a solution, where the linear systems are highly ill-conditioned.

A strategy to form  $B$  is to minimize  $\|D_B^{\frac{1}{2}}B^{-1}ND_N^{-\frac{1}{2}}\|$ . This problem is hard to solve but it can be approximately solved with an inexpensive heuristic. Select the first  $m$  linearly independent columns of  $AD^{-1}$  with smallest 1-norm. This choice of columns tends to produce better conditioned matrices as the interior point method approaches a solution. Due to the splitting nature of the preconditioner, we shall call it the *splitting* preconditioner.

### 4.4 Equivalence to the Normal Equations

It is useful to note that the matrix (4.13) can be obtained via the normal equations. Recall that  $A = [B \ N]P$  thus,  $AD^{-1}A^t = BD_B^{-1}B^t + ND_N^{-1}N^t$ .

Now, multiplying it by  $D_B^{\frac{1}{2}}B^{-1}$  and post-multiplying by its transpose leads to

$$D_B^{\frac{1}{2}}B^{-1}(AD^{-1}A^t)B^{-t}D_B^{\frac{1}{2}} = I + D_B^{\frac{1}{2}}B^{-1}ND_N^{-1}N^tB^{-t}D_B^{\frac{1}{2}}. \quad (4.15)$$

It can be shown that the right hand side vector for both preconditioned systems is the same.

A partition of matrix  $A$  has been used before as a preconditioner for network interior point methods [25]. In this situation  $B$  is a tree and is easy to find. Therefore, the preconditioner (4.13) can be viewed as a generalization. We also remark that the rules for choosing the set of columns are not the same.

## 5 Practical Aspects

Iterative methods only need to access the matrix to compute matrix-vector products. In this section we present a more stable way to compute this product versus using the matrix directly as in  $w = Kx$ . Let us consider for simplicity our matrix to be of the form (4.11). We can write any  $n$ -dimensional vector  $x$  as  $x = x_{\mathcal{R}(V)} + x_{\mathcal{N}(V^t)}$ . Thus,

$$\begin{aligned} Kx &= -x + U^tV^tx_{\mathcal{R}(V)} + VUx \\ &= -x_{\mathcal{N}(V^t)} + U^tV^tx_{\mathcal{R}(V)} + VUx_{\mathcal{N}(V^t)} \end{aligned}$$

by remark 4.1. Now, since  $V^t = [I \ 0]P$  its null and range spaces can be easily represented in a code and all the calculations for it consist of managing certain indices properly. Observe that the first two terms do not have nonzero entries in common for any of the positions. Hence, no floating point operations are needed to add them. If we compute the product  $Kx$  without these considerations, some round-off error will be introduced for the zero sum  $-x_{\mathcal{R}(V)} + UVx_{\mathcal{R}(V)}$  and often this error is large enough compared with the other entries of  $x$ . A welcomed side effect is that  $n$  floating point operations are saved with this procedure.

Another practical aspect concerns the recovery of the solution. The approximate solution for the original system can be easily recovered from the solution for the preconditioned system  $(\hat{x}, \hat{y})$  by computing

$$\begin{pmatrix} x \\ y \end{pmatrix} = M^{-t} \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}$$

where the residual is given by

$$\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} b_1 + Dx - A^t y \\ b_2 - Ax - Ey \end{pmatrix}.$$

Sometimes the norm of the error  $\|r\|^2 = \|r_1\|^2 + \|r_2\|^2$  is too large due to the round-off error introduced on computing the preconditioned system and recovering the solution. It can get particularly large at the final iterations of the interior point method. One way to reduce this error is to compute  $\tilde{x} = D^{-1}(A^t y - b_1)$  and form a new approximate solution  $(\tilde{x}, y)$ . The error for the new solution  $\tilde{r}$  will be given by  $\tilde{r}_2 = r_2 + AD^{-1}r_1$  if we assume that  $\tilde{r}_1 = b_1 + D\tilde{x} - A^t y$  is zero. Thus, we update the solution whenever  $\|r\|$  is above a given tolerance and  $\|\tilde{r}\| < \|r\|$ . This approach is related to iterative refinement for the augmented system [4] keeping  $y$  unchanged.

## 5.1 Inexact Solutions

An idea that immediately comes to mind when using iterative procedures for solving the linear systems is to relax the required tolerance. Thus, we start the interior point method with a relaxed tolerance ( $10^{-4}$ ) and, whenever an iteration does not (at least) halve the gap ( $x^t z$ ), the tolerance is changed to the square root of machine epsilon.

In the context of the predictor-corrector variant there is another place for applying this idea. Recall that for computing the search directions, two linear systems are solved. The first one gives the perturbation parameter and the nonlinear correction for the Newton's method. The second one is written in such a way that it gives already the final search directions. Thus, the first linear system may be solved with a more relaxed tolerance than the second one.

## 5.2 Keeping the Set of Columns

A nice property of the splitting preconditioner is that we can work with the selected set of columns for some iterations. As a consequence, the preconditioner is very cheap to compute for these iterations.

It is important to notice that keeping the matrix  $B$  from previous iterations does not mean to keep the same preconditioner since  $D$  will change from iteration to iteration and the preconditioner depends on it too.

In the experiments given later we change the set whenever the iterative method takes more iterations than a certain threshold value ( $\sqrt{m}$ ) or when the solution given by the iterative method is not accurate.

### 5.3 The $LU$ Factorization

This class of preconditioners is not competitive against the direct method approach by computing the Cholesky factorization without a careful implementation. The computation of an  $LU$  factorization where the set of independent columns is unknown at the beginning of the factorization may be too expensive. In this section we discuss the most important techniques for the implementation of a competitive code. This is a research topic in its own and a complete description of all techniques used can be found in [23].

For this application, the most economical way to compute the  $LU$  factorization is to work with the delayed update form. It fits very well with our problem because when a linearly dependent column appears, it is eliminated from the factorization and the method proceeds with the next column in the ordering.

One of the main drawbacks of a straightforward implementation of the splitting preconditioner is the excessive fill-in in the  $LU$  factorization. The reason is that the criterion for reordering the columns does not take the sparsity pattern of  $A$  into account. A good technique consists of interrupting the factorization when excessive fill-in occurs and reordering the independent columns found thus far by the number of nonzero entries. The factorization is then started from scratch and the process is repeated until  $m$  independent columns are found. In our implementation we consider excessive fill-in a factorization that produces more nonzero entries than the number of nonzero entries from the normal equations system.

A second factorization is applied on the chosen set of independent columns using standard techniques for computing an efficient sparse  $LU$  factorization. This approach improves the results significantly for some problems. Therefore, the second factorization is always done. As a welcome side effect, it is not necessary to store  $U$  in the factorizations that determine  $B$ .

One difficulty in determining the subset of independent columns is the number of dependent columns visited in the process. An idea consists in verify whether a column is dependent or not during the delayed update form of the  $LU$  factorization. If we find that a candidate column is already dependent on the first say,  $k$  columns, it is useless to continue updating the

candidate column for the remaining columns of  $L$ .

## 5.4 Symbolically Dependent Columns

Given an ordering of columns, we want to find the unique set of  $m$  independent columns that preserves the ordering. The brute force approach for this problem consists in computing the factorization column by column and discarding the (nearby) dependent columns along the way. The strategies developed here will indicate when a column can be ignored in the factorization. The set of independent columns found by these techniques is the same set obtained by the brute force approach.

*Symbolically dependent* columns are columns that are linearly dependent in structure for all numerical values of their nonzero entries. The idea is to find a set of say  $k$  columns with nonzero entries in at most  $k - 1$  rows. This set of columns is symbolically dependent.

Let us first consider a square matrix for simplicity. In this situation, the problem is equivalent to permuting nonzero entries onto the diagonal. This problem is equivalent to finding a matching of a bipartite graph where the rows and columns form the set of vertices and the edges are represented by the nonzero entries. This idea was first used by Duff [10] and it is applied as a first step for permuting a matrix to block triangular form. If a nonzero entry cannot be assigned to the diagonal in the matching process for a given column that column is symbolically dependent.

In [8] this idea is extended to rectangular matrices. The authors are concerned with finding a set of independent columns of the matrix which gives a sparse  $LU$  factorization. Thus, the columns are reordered by degree and the matching algorithm applied giving a set of candidate columns, denoted here as *key columns*, which are not symbolically dependent.

Our idea for using the key columns comes from the fact that the number of independent columns before the  $k$ th key column on the matrix is at most  $k - 1$ . Therefore, it is possible to speed up the  $LU$  factorization whenever we find  $k - 1$  numerically independent columns located before the  $k$ th key column. The speed up is achieved by skipping all the columns from the current one to the  $k$ th key column.

## 5.5 Symbolically Independent Columns

We now define the *symbolically independent* columns i.e., columns that are linearly independent in structure for all numerical values of their nonzero entries. A powerful strategy consists in moving the symbolically independent columns to the beginning of the ordered list since those columns are necessarily going to be in the factorization. Then these columns can be reordered further in order to reduce the number of fill-ins in the  $LU$  factorization. Notice that the symbolically dependent columns can be ignored in this step. Thus, we are concerned only with the key columns given by the matching algorithm.

We are not aware of any efficient algorithm for finding all the symbolically independent columns from a given ordered set. Therefore, we use heuristics approaches to identify some of the symbolically dependent columns.

### 5.5.1 First Nonzero Entries

On the description of the heuristic below, we say that column  $j$  is the first entry column of row  $i$  if  $j$  contains the first nonzero entry in row  $i$  on the ordered set. We consider a column  $j$  symbolically independent given an ordered set if at least one of the following rules applies:

1. Column  $j$  is the first entry column of at least one row;
2. Column  $j$  is the second entry column of a row  $i$  and the first entry column of row  $i$  is also first entry column for at least another row not present on column  $j$ .

This set of rules guarantees that the columns selected are symbolically independent but it does not guarantee that all symbolically independent columns are found. For instance, consider the following sparse matrix

$$\begin{pmatrix} \times & \times & \times \\ \times & \times & \\ \times & & \end{pmatrix}.$$

### 5.5.2 Strongly Connected Components

This strategy is also applied to the key columns. Since the key columns are determined by a matching procedure, a permutation for computing the strongly connected components is already at hand. Given the strongly connected components, their columns are reordered by the splitting criteria. Now, the following result holds:

**Theorem 5.1** *Let  $B$  be an  $m \times m$  matrix and  $BP$  a given ordering of  $B$ 's columns with nonzero diagonal entries. Consider the block triangular matrix  $QBPQ^t$  where the columns inside each strongly connected component are ordered according with  $P$ . Let  $k$  be the smallest index in  $P$  among the first symbolically dependent columns of each component considering only the rows from the respective component. Then every column whose index in ordering  $P$  is smaller than  $k$  is symbolically independent in  $BP$ .*

**Proof.** The first  $k - 1$  columns of  $BP$  are symbolically independent on their respective component. Moreover, the columns of each component are symbolically independent (considering only these  $k - 1$  columns). Since  $QBPQ^t$  is block triangular, any column is symbolically independent from the previous blocks. Therefore, these  $k - 1$  columns are symbolically independent among them. •

Thus, we look for the first symbolically dependent column in its own component considering only the rows from the respective component. All columns with smaller index in the ordering are symbolically independent.

Another advantage of this strategy is that we can apply the heuristics for finding symbolically independent columns inside each diagonal block.

## 5.6 Discarding Dependent Rows

In order for the splitting preconditioner to work, the constraint matrix  $A$  cannot have dependent rows. The following procedure finds the dependent rows and discards them before the interior point method starts.

The techniques used for finding  $B$  can be applied to the columns ordered by degree. Moreover, rows containing entries that are part of singleton columns can be ignored in this factorization since these rows are necessarily independent. This idea can be applied in the resulting matrix until there are no longer any singleton columns. Thus, finding dependent rows is inexpensive most of the time. Actually, there are problems like those with only inequality constraints where no factorization is performed at all. This factorization can be computed even more efficiently [2].

## 6 Numerical Experiments

In this section we present several numerical experiments with the new preconditioner. The experiments are meant to expose the type of problems

where the new approach performs better. Therefore, these experiments are not to be seen as a way to determine the best approach for the interior point methods in a general context. For instance, we observe that the new preconditioner fails to achieve convergence for several problems from the netlib collection, such as GREENBE and PILOT families, under the strict conditions the experiments were made. Nevertheless, the results do indicate that the new approach is an important option for some classes of problems.

The procedures for solving the linear systems with the splitting preconditioner are coded in C and are applied within the PCx code [9], a state of the art interior point methods implementation. PCx's default parameters are used except that multiple corrections are not allowed and all tolerances for the interior point and conjugate gradient methods are set to the square root of machine epsilon.

All the experiments are carried out on a Sun Ultra-60 station. The floating point arithmetic is IEEE standard double precision.

In order to illustrate the expected behavior of the iterative methods solution for solving the normal equations system Table 1 shows the number of iterations using the conjugate gradient method with the incomplete Cholesky factorization and the splitting preconditioner as the interior point method progresses. The Euclidian residual norm is used to measure convergence. No fill-in is allowed in the incomplete Cholesky approach and whenever a small pivot is found it is set to one and remaining entries of the corresponding column to zero. The starting vector used is the right hand side.

The chosen problem is KEN13 from netlib. The dimension of the linear system is 14627 after preprocessing. Iteration zero corresponds to computing the starting point. Only the number of iterations of the conjugate gradient method for solving the first linear system is shown. The number of iterations for solving the second linear system is very close to it. An interesting observation is that the incomplete Cholesky preconditioners generally take very few inner iterations to converge at the early stage of the interior point outer iterations, but this deteriorates in the later outer iterations as the interior point method nears convergence. With the splitting preconditioner the exact opposite occurs. The last few outer iterations are the ones where it performs better. This property of the splitting preconditioner is highly desirable since the linear systems in the last outer iterations are the most ill-conditioned.

We now briefly describe the problems used on the remaining numerical experiments. Problems FIT1P and FIT2P belong to the netlib collection of linear programming problems.

IP Iteration	Inner iterations	
	Incomplete Cholesky	Splitting Preconditioner
0	49	195
1	49	203
2	45	258
3	39	190
4	24	171
5	24	185
6	20	128
7	22	130
8	22	133
9	32	126
10	44	108
11	71	91
12	104	92
13	171	76
14	323	63
15	480	52
16	834	43
17	1433	34
18	2146	30
19	4070	22
20	7274	18
21	11739	17
22	15658	15
23	24102	12
24	13463	10
25	5126	6
Average	3360	84

Table 1: KEN13 Conjugate Gradient Method Iterations

The PDS model is a multi-commodity problem with 11 commodities and whose size is a function of the number of days being modeled. Thus, PDS-2 models two days, PDS-20 models twenty days and so on. A generator for this model is available and experiments for problems of a variety of sizes are presented.

DIFFICULT is the linear programming relaxation of a large network design problem. This problem was supplied by Eva Lee and formulated by Daniel Bienstock.

The QAP problems are models for the linearized quadratic assignment problem [24]. The problems tested here for the QAP model are from the QAPLIB [7] collection with the modification described in [24].

The following computational results compare the behavior of the direct method approach against the splitting preconditioner. The preconditioned positive definite matrix (4.15) is used for the experiments. Since the splitting preconditioner is designed for the last interior point iterations the diagonal of the normal equations matrix is first adopted as preconditioner for the conjugate gradient method until the initial gap  $(x_0^t z_0)$  for the linear programming problem is reduced by at least  $10^6$  or until the number of inner iterations for solving the linear system is above its own dimension divided by four when the splitting preconditioner is taken.

Table 2 contains the basic statistics about the test problems. The dimension and number of nonzero entries shown for the matrix of constraints refer to the preprocessed problems. The number of nonzero entries for the normal equations includes only the lower half of the matrix. The number of nonzero entries for matrix  $L$  of the Cholesky factorization is obtained after reordering the rows of  $A$  by the minimum degree criteria.

Table 3 reports the number of iterations for the interior point method where the linear systems are solved by either the Cholesky factorization of the normal equations, or with the conjugate gradient method with the splitting preconditioner. Column Fact. contains the number of  $LU$  factorizations needed for the interior point methods including the factorization for computing the starting point. Notice that the number of outer iterations for the interior point methods on both approaches is about the same for most problems. Only problem DIFFICULT presented a large difference. No results for problems PDS-25 to PDS-60 are reported for the Cholesky approach because it would take a large amount of time and memory to solve these problems.

It is interesting to notice that the direct approach does not obtain a clear advantage over the iterative approach for these problems. We remark that

Problem	Dimension	Number of Nonzero Entries		
		Matrix $A$	$AD^{-1}A^t$	Matrix $L$
FIT1P	$627 \times 1677$	9868	196878	196878
FIT2P	$3000 \times 13525$	50284	4501500	4501500
DIFFICULT	$31514 \times 274372$	806284	689260	5842076
PDS-01	$1291 \times 3623$	7726	5955	11230
PDS-02	$2609 \times 7339$	15754	12256	39613
PDS-06	$9156 \times 28472$	61120	46578	563278
PDS-10	$15648 \times 48780$	104550	79866	1647767
PDS-15	$24031 \times 77366$	165993	125528	4232804
PDS-20	$32287 \times 106180$	227541	170973	7123636
PDS-25	$40264 \times 131526$	281873	212083	10674326
PDS-50	$80339 \times 272513$	582206	432371	42074817
PDS-60	$96514 \times 332862$	710234	524563	-
CHR12A	$947 \times 1662$	5820	16217	81675
CHR12B	$947 \times 1662$	5820	16325	81549
CHR12C	$947 \times 1662$	5820	16169	81964
CHR15A	$1814 \times 3270$	11460	36794	221035
CHR15B	$1814 \times 3270$	11460	36899	221922
CHR15C	$1814 \times 3270$	11460	36749	226357
CHR18A	$3095 \times 5679$	19908	72458	557491
CHR18B	$3095 \times 5679$	19908	72368	562356
CHR20A	$4219 \times 7810$	27380	107509	942275
CHR20B	$4219 \times 7810$	27380	107349	924810
CHR20C	$4219 \times 7810$	27380	107809	1003786
CHR22A	$5587 \times 10417$	36520	153856	1424837
CHR22B	$5587 \times 10417$	36520	153680	1453225
CHR25A	$8149 \times 15325$	53725	249324	2653126
ELS19	$4350 \times 13186$	50882	137825	3763686
HIL12	$1355 \times 3114$	15612	34661	611836
NUG08	$383 \times 792$	3096	6363	44956
NUG15	$2729 \times 9675$	38910	88904	2604504
ROU10	$839 \times 1765$	8940	19274	242015
SCR10	$689 \times 1540$	5940	13094	129297
SCR12	$1151 \times 2784$	10716	24965	334090
SCR15	$2234 \times 6210$	24060	59009	1254242
SCR20	$5079 \times 15980$	61780	166709	6350444

Table 2: Problems Statistics

the solutions obtained in these experiments agree in at least significant eight digits for all the problems whose objective value are known to us.

Table 4 shows a comparison between both approaches for the total running time. Time to preprocess the linear programming problem and input data was not accounted, since those do not depend on the adopted method. All the remaining time procedures were measured. The splitting approach takes less total time for solving the problems. It is no surprise since these models were chosen for this reason. The purpose of these experiments is to show the type of problems where the new approach is expected to perform better. For example, on problems like FIT1P and FIT2P that have dense columns, the normal equations matrix is already very dense as it can be seen in Table 2. Thus, this approach takes much more computational effort.

The process of discarding dependent rows is very efficient. In particular, for problems FIT1P, FIT2P and DIFFICULT no factorization was performed in order to verify the independence among all rows. Finding singleton rows and columns also helps to obtain good results. On all PDS problems that strategy resulted in null block diagonal matrix in at least one iteration. That is, the refactorization for these matrices spent no floating point operations and generated no fill-in entries. The same happened on problem DIFFICULT. For these problems and also for many others block diagonal matrices with very small dimension often occurred.

The QAP model problems also lead to normal equations matrices that are not much sparse although in a lesser degree than the FIT problems since these problems do not have dense columns. This feature, together with the fact that the factorization generates a large number of fill-in entries makes the Cholesky approach less effective. That can be more easily observed as the size of the problems grows.

The PDS model problems do not generate dense normal equations matrices. On the other hand, the Cholesky factorization can generate many fill-in entries. As the dimension of the problem increases, the splitting approach performs better compared to the direct approach. Figure 1 is a good illustration of it.

Another factor that helps the splitting preconditioner obtain good results for the PDS model is that the number of  $LU$  factorizations is very small compared to the number of iterations. Therefore, computing a solution for the linear systems in a large number of iterations for these problems is inexpensive since no factorization is computed. This fact also applies for problem DIFFICULT explaining the good performance of the new approach

Problem	Cholesky Iterations	Splitting	
		Iterations	Fact.
FIT1P	22	23	3
FIT2P	25	29	5
DIFFICULT	32	41	4
PDS-01	22	24	4
PDS-02	26	28	4
PDS-06	39	40	4
PDS-10	51	46	4
PDS-15	63	59	4
PDS-20	69	72	5
PDS-25	-	73	6
PDS-50	-	84	5
PDS-60	-	85	5
CHR12A	16	17	6
CHR12B	12	12	2
CHR12C	13	14	5
CHR15A	12	15	3
CHR15B	16	17	4
CHR15C	16	18	4
CHR18A	24	25	6
CHR18B	14	13	2
CHR20A	23	26	8
CHR20B	26	29	8
CHR20C	20	20	5
CHR22A	23	25	5
CHR22B	32	33	8
CHR25A	31	32	8
ELS19	31	31	17
HIL12	18	16	13
NUG08	11	11	7
NUG15	22	25	18
ROU10	18	20	7
SCR10	18	15	5
SCR12	14	14	4
SCR15	23	27	18
SCR20	22	27	18

Table 3: Cholesky versus Splitting Preconditioner – Iterations

Problem	Cholesky		Splitting	
	Time	MFlops	Time	MFlops
FIT1P	153	87.6	5.3	4.2
FIT2P	44122	9110.3	67.1	33.0
DIFFICULT	26776	7335.2	8214	2264.7
PDS-01	2.24	0.3	8.8	4.4
PDS-02	10.99	1.9	34.6	15.7
PDS-06	760	201.4	497	161.0
PDS-10	5145	1014.9	1316	327.8
PDS-15	29004	4436.6	4285	772.3
PDS-20	71237	9441.6	8371	1277.6
PDS-25	-	17663.9	13940	1811.9
PDS-50	-	156416.2	60215	6625.9
PDS-60	-	-	89651	9073.1
CHR12A	12.5	8.5	5.9	4.6
CHR12B	9.9	8.5	6.2	8.2
CHR12C	10.3	8.5	5.2	5.0
CHR15A	37.7	31.7	12.7	11.6
CHR15B	49.0	32.4	13.1	10.0
CHR15C	48.9	33.2	14.1	10.3
CHR18A	252	116.7	49.1	24.8
CHR18B	160	118.0	28.6	32.1
CHR20A	567	244.1	97.6	45.9
CHR20B	598	232.6	113	48.9
CHR20C	586	285.8	70.0	45.0
CHR22A	1022	414.8	150	79.8
CHR22B	1395	433.0	189	61.2
CHR25A	3156	995.4	442	144.0
ELS19	15175	5212.0	2444	169.1
HIL12	547	386.6	89.4	69.1
NUG08	5.48	7.5	2.7	3.3
NUG15	6706	3473.8	2962	1064.5
ROU10	117	98.1	21.4	16.1
SCR10	40.5	34.3	6.4	6.0
SCR12	141	136.0	14.6	14.1
SCR15	1946	1008.3	257.7	72.5
SCR20	26171	12331.3	1929	455.9

Table 4: Cholesky versus Splitting Preconditioner – Time and Flops

even considering that the normal equations matrix is sparse.

## 7 Conclusions

We have shown that from the point of view of designing preconditioners, it is better to work with the augmented system instead of working with the normal equations. Two important results support this statement. First, all preconditioners developed for the normal equations system lead to an equivalent preconditioner for the augmented system. However, the opposite statement is not true. Whole classes of preconditioners for the augmented system can result in the same preconditioner for the normal equations.

Based upon this result we designed a preconditioner for the augmented system. This preconditioner reduces the system to positive definite matrices, and therefore the conjugate gradient method can be applied, and the resulting iteration is quite competitive with the normal equations approach.

An important advantage of the splitting preconditioner is that it becomes better in some sense as the interior point method advances towards an optimal solution. That is a very welcome characteristic since the linear system is known to be very ill-conditioned close to a solution, these systems are difficult to solve by iterative methods with most of the previously known preconditioners. Moreover, this new method seems to be well suited for classes of problems where the Cholesky factorization has a large amount of nonzero entries, even when the original normal equations matrix is fairly sparse. However, an efficient implementation of the splitting preconditioner is not trivial.

## Acknowledgments

Oliveira's research was partially sponsored by the Brazilian Council for the Development of Science and Technology (CNPq) and Foundation for the Support of Research of the State of São Paulo, (FAPESP). Sorensen's work was supported in part by NSF Grant CCR-9988393 and by NSF Grant ACI-0082645.

## References

- [1] I. ADLER, M. G. C. RESENDE, G. VEIGA, AND N. KARMARKAR, *An implementation of Karmarkar's algorithm for linear programming*, Mathematical Programming, 44 (1989), pp. 297–335.
- [2] E. D. ANDERSEN, *Finding all linearly dependent rows in large-scale linear programming*, Optimization Methods and Software, 6 (1995), pp. 219–227.
- [3] L. BERGAMASCHI, J. GONDZIO, AND G. ZILLI, *Preconditioning indefinite systems in interior point methods for optimization*, Computational Optimization and Applications, 28 (2004), pp. 149–171.
- [4] A. BJÖRCK, *Numerical Methods for Least Squares Problems*, SIAM Publications, SIAM, Philadelphia, PA, USA, 1996.
- [5] D. BRAESS AND P. PEISKER, *On the numerical solution of the biharmonic equation and the role of squaring matrices*, IMA J. Numer. Anal., 6 (1986), pp. 393–404.
- [6] J. R. BUNCH AND B. N. PARLETT, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655.
- [7] R. S. BURKARD, S. KARISCH, AND F. RENDL, *QAPLIB - A quadratic assignment problem library*, European Journal of Operations Research, 55 (1991), pp. 115–119.
- [8] T. F. COLEMAN AND A. POTHEN, *The null space problem II. Algorithms*, SIAM J. Alg. Disc. Meth., 8 (1987), pp. 544–563.
- [9] J. CZYZYK, S. MEHROTRA, M. WAGNER, AND S. J. WRIGHT, *PCx an interior point code for linear programming*, Optimization Methods & Software, 11-2 (1999), pp. 397–430.
- [10] I. S. DUFF, *On algorithms for obtaining a maximum transversal*, ACM Trans. Math. Software, 7 (1981), pp. 315–330.
- [11] ———, *The solution of large-scale least-square problems on supercomputers*, Annals Oper. Res., 22 (1990), pp. 241–252.

- [12] —, *MA57 a new code for the solution of sparse symmetric definite and indefinite systems*, tech. rep., RAL 2002-024, Rutherford Appleton Laboratory, Oxfordshire, England, 2002.
- [13] R. FOURER AND S. MEHROTRA, *Performance on an augmented system approach for solving least-squares problems in an interior-point method for linear programming*, Mathematical Programming Society COAL Newsletter, 19 (1992), pp. 26–31.
- [14] A. GEORGE AND E. NG, *An implementation of Gaussian elimination with partial pivoting for sparse systems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 390–409.
- [15] P. E. GILL, W. MURRAY, D. B. PONCELEÓN, AND M. A. SAUNDERS, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. and Applications, 13 (1992), pp. 292–311.
- [16] G. H. GOLUB AND A. J. WATHEN, *An iteration for indefinite systems and its application to the Navier-Stokes equations*, SIAM Journal on Scientific Computing, 19 (1998), pp. 530–539.
- [17] N. KARMARKAR, *A new polynomial-time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.
- [18] 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), pp. 435–449.
- [19] S. MEHROTRA, *Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods*, ORSA Journal on Computing, 4 (1992), pp. 103–118.
- [20] —, *On the implementation of a primal-dual interior point method*, SIAM Journal on Optimization, 2 (1992), pp. 575–601.
- [21] C. MÉSZÁROS, *The augmented system variant of IPMs in two-stage stochastic linear programming computation*, European Journal of Operations Research, 101 (1997), pp. 317–327.

- [22] R. D. C. MONTEIRO, I. ADLER, AND M. G. C. RESENDE, *A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension*, Mathematics of Operations Research, 15 (1990), pp. 191–214.
- [23] A. R. L. OLIVEIRA, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, tech. rep., PhD Thesis, TR97-11, Department of Computational and Applied Mathematics, Rice University, Houston TX, 1997.
- [24] M. PADBERG AND M. P. RIJAL, *Location, Scheduling, Design and Integer Programming*, Kluwer Academic, Boston, 1996.
- [25] M. G. C. RESENDE AND G. VEIGA, *An efficient implementation of a network interior point method*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 12 (1993), pp. 299–348.
- [26] T. RUSTEN AND A. J. WINTHER, *A preconditioned iterative method for saddlepoint problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.
- [27] R. J. VANDERBEI, *Symmetric quasi-definite matrices*, SIAM J. Optimization, 5 (1995), pp. 100–113.
- [28] R. J. VANDERBEI AND T. J. CARPENTER, *Indefinite systems for interior point methods*, Mathematical Programming, 58 (1993), pp. 1–32.
- [29] S. A. VAVASIS, *Stable numerical algorithms for equilibrium systems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1108–1131.

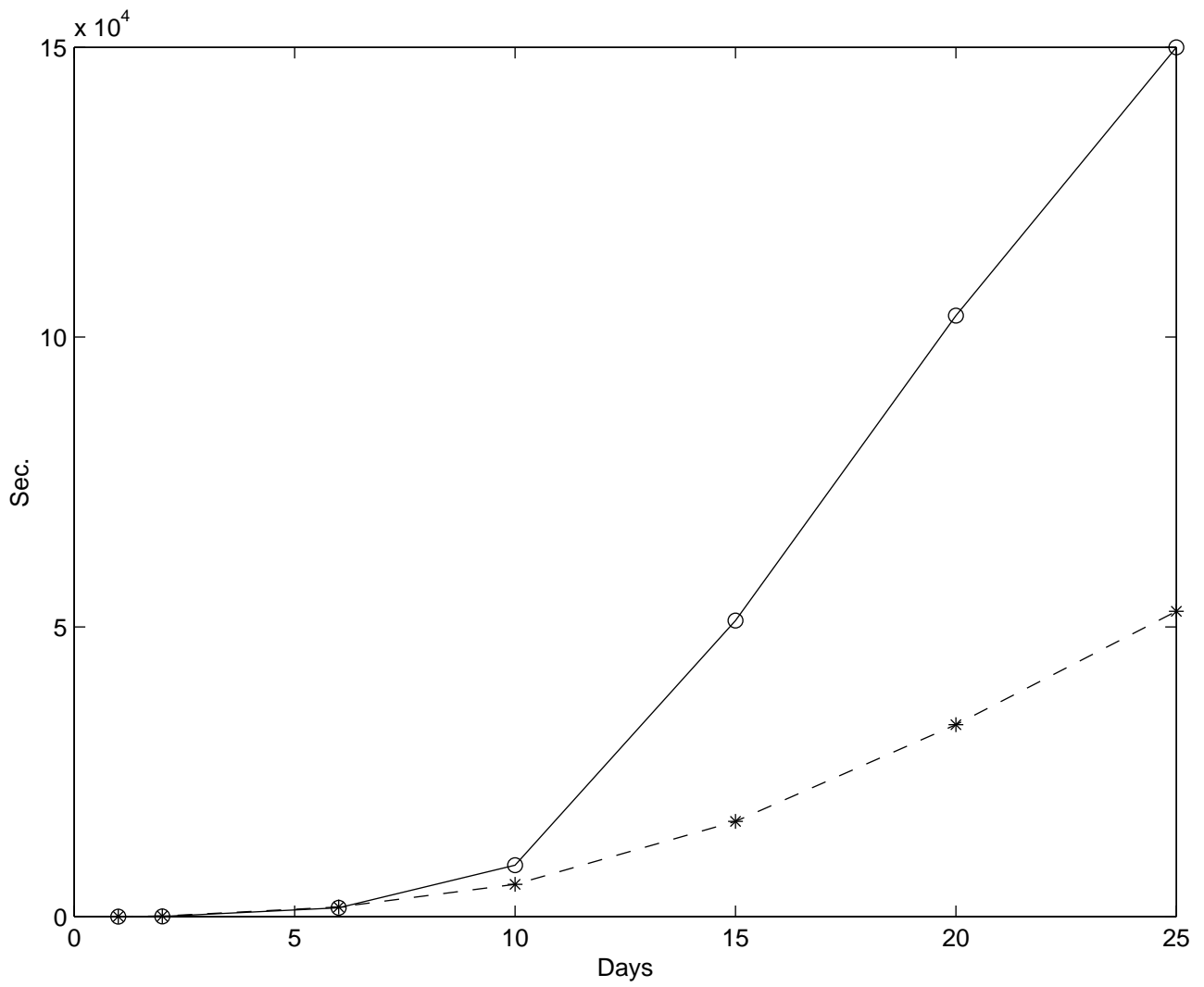


Figure 1: Splitting Approach (\*) versus Direct Approach (o) – PDS Model