ON FREE VARIABLES IN INTERIOR POINT METHODS *

CSABA MÉSZÁROS

Imperial College, London SW7 2BZ, UK
and
MTA SZTAKI, H-1518 Budapest, P.O. BOX 63, Hungary

3. April 1997

Interior point methods, especially the algorithms for linear programming problems are sensitive if there are unconstrained (free) variables in the problem. While replacing a free variable by two nonnegative ones may cause numerical instabilities, the implicit handling results in a semidefinite scaling matrix at each interior point iteration. In the paper we investigate the effects if the scaling matrix is regularized. Our analysis will prove that the effect of the regularization can be easily monitored and corrected if necessary. We describe the regularization scheme mainly for the efficient handling of free variables, but a similar analysis can be made for the case, when the small scaling factors are raised to larger values to improve the numerical stability of the systems that define the search direction. We will show the superiority of our approach over the variable replacement method on a set of test problems arising from water management application.

KEY WORDS: Constrained Optimization, Interior Point Methods

1 INTRODUCTION

Interior point methods for constrained optimization are usually formulated for nonnegative variables. Methods for handling free variables have received little attention. One of the reasons for this is that in nonlinear optimization the computational difficulties caused by free variables are absorbed by the nonlinearity of the problem. Free variables can cause trouble when linear programming (LP) problems are solved by traditional interior point methods. However, LP problems with large numbers of free variables are rare in the current set of test libraries. For example, from the more than 100 problems of the commonly used LP test set NETLIB [7] only 7 problems have more than 4 free variables. However, in practice many more problems can occur with many free variables. The water management decision support system Aquarius [8], which has been developed at the Delft University of Technology especially draws our attention to the necessity for efficient handling of free variables.

* This work was supported in part by EPSRC grant No. GR/J52655 and Hungarian Research Fund OTKA T-016413.

The simplest way to handle free variables is to split them into positive and negative parts, also known as *variable replacement*. This approach, however, may result in serious numerical troubles. Another straightforward idea is the elimination of free variables during presolve. While this can be very favorable in some cases, it can also be a major source of instability or fill-in. Numerical results indicate [10] that an efficient way to handle free variables can be derived by a modification of the logarithmic barrier function. It is easy to see that this will result in the same iteration trajectory as that for the problem after elimination of all free variables. Therefore the approach is also referred to as *implicit elimination*. The disadvantages of this technique are the possibility of big fill-in on problems with several free variables and the loss of the numerically advantageous positive definiteness of the scaling matrix at each interior point iteration [9].

Our aim in the paper is to combine the benefits of the variable replacement and implicit elimination techniques and to derive an efficient way to handle free variables. In Section 2 we give a brief summary of the primal-dual logarithmic barrier algorithm which is the framework of our investigations. In Section 3 we suggest a regularization approach which makes the use of the normal equation system possible, but does not encounter the numerical problems inherent to the variable replacement. In Section 4 we investigate the proposed regularization and show its properties. In Section 5 we describe a presolve technique which detects hidden free variables in the model and eliminates free variables unless they introduce heavy fill–in or numerical instability. In Section 6 we show computational results and compare our method, implemented in BPMPD [13], with the primal–dual interior point implementation PCx [5] which uses variable replacement. We summarize our findings in Section 7.

2 THE PRIMAL-DUAL LOG BARRIER ALGORITHM

In this paper we have selected the primal-dual logarithmic barrier algorithm to present our ideas, because it and its modified versions are considered, in general, to be the most efficient in practice. The computational results presented in this paper were obtained using implementations of this algorithm. It is to be noted, however, that this choice has notational consequences only. Practically, any interior point method, even nonlinear ones can be discussed in a similar linear algebra framework.

Let us consider the linear programming problem

$$\begin{aligned} \min \quad & c^T x \\ & Ax = b, \\ & x \geq 0, \end{aligned} \tag{1}$$

where $c, x \in R^n$, $b \in R^m$ and $A \in R^{m \times n}$ and its dual

$$\begin{aligned} \max \quad & b^T y, \\ & A^T y + z = c, \end{aligned} \tag{2}$$

$$z \geq 0,$$

where $y \in R^m$ and $z \in R^n$. In this paper we will assume that A is of full row rank.

The primal-dual logarithmic barrier algorithm constructs a sequence of nonlinear optimization problems, where the goal is to minimize the logarithmic barrier function

$$L(x,\mu) = c^T x - \mu \sum_{i=1}^{n} \ln\,(x_j) \tag{3}$$

for a barrier parameter $\mu \geq 0$ under the condition $Ax = b$. It can be seen that the application of the Newton method to solve the first order optimality conditions of each barrier problem results in the so called augmented system:

$$\begin{bmatrix} -D & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{4}$$

where $X = diag(x_1, ..., x_n)$, $Z = diag(z_1, ..., z_n)$, $D = X^{-1}Z$, $\alpha = X^{-1}(\mu e - XZe)$, $\beta = 0_m$ and e is the n-vector of all ones. In one iteration of the primal-dual interior point method the current iterate (x, y, z) is shifted along the search direction $(\Delta x, \Delta y, \Delta z)$, where solution of (4) results in Δx and Δy, and Δz can be obtained as $\Delta z = X^{-1}(\mu e - XZe - Z\Delta x)$.

An efficient way to solve (4) is the normal *equations approach*, where Δx is eliminated first from (4). The matrix of the resulting equation system, $AD^{-1}A^T$ is positive definite, therefore a Cholesky decomposition can be performed on it. The big advantage of this approach is the positive definite property, which has favorable numerical benefits and makes efficient symbolic pivot determination possible. For further details on the interior point algorithms the reader is referred to the recent paper [2].

3 FREE VARIABLES IN THE PRIMAL-DUAL LOG BARRIER METHOD

In what follows, we investigate the case when some of the variables are not constrained by nonnegativity. Let us modify problem (1) as

$$\begin{aligned} \min \quad & c^T x, \\ & Ax = b, \\ & x_i \geq 0, \qquad i \in \{1, ..., n\} \backslash F \end{aligned} \tag{5}$$

where $F \subseteq \{1, ..., n\}$ denotes the index set of free variables. In our investigations we will assume that the columns of the free variables are linearly independent. Otherwise, linearly dependent free variables will cause dual infeasibility (primal unboundedness) unless the dependency is true for the vectors augmented by the objective coefficients. In this case, the set of free variables can be reduced to a maximal independent subset.

Most of the interior point implementations replace each free variable by two nonnegative ones: $x_i = x_i^+ - x_i^-$ for $i \in F$. After this transformation the standard interior point technology can be applied. The variable replacement increases the number of variables in the linear programming problem, but has no further effect on the sparsity of $AD^{-1}A^T$ because the sparsity pattern of the newly introduced columns is the same than that one of the original free variable. The main drawback of this approach is that function (3) becomes unbounded. Furthermore the dual of the reformulated problem will have an empty interior. This results in numerical instabilities, which make the algorithm less effective.

A natural idea is to modify (3) in such a way that we do not include the free variables in the logarithmic barrier term, in other words we define the logarithmic barrier function for problem (5) as

$$L(x,\mu) = c^T x - \mu \sum_{\substack{i=1,\ldots,n \\ i \notin F}} \ln\,(x_i). \tag{6}$$

It is easy to see that application of the log barrier methodology to the above problem yields the system

$$\begin{bmatrix} -\tilde{D} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix} = \begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix}, \tag{7}$$

where

$$\tilde{D}_{ii} = \begin{cases} x_i^{-1} z_i & i \notin F \\ 0 & i \in F \end{cases}, \tag{8}$$

$$\tilde{\alpha}_i = \begin{cases} x_i^{-1}(\mu - x_i z_i) & i \notin F \\ 0 & i \in F \end{cases}, \quad \tilde{\beta} = 0_m.$$

If A is of full row rank and the columns of the free varibales are linearly independent, then the matrix of system (7) is of full rank. It is to be noted that the diagonal scaling matrix $\tilde{D}$ is no longer positive definite, therefore $A\tilde{D}^{-1}A^T$ does not exist. This is why Cholesky factorization with the symbolic reordering scheme can not be used for this system.

However, as it was shown in [12], the decomposition of the augmented system with 1×1 pivoting is possible, but it must be based on both numerical and sparsity investigations. Such an approach can not be superior to the Cholesky factorization, unless the linear programming problem has special structure, which results in disadvantageous fill–in in AA^T [9]. The other advantage of the normal equations approach is that the pivot order determined at the very beginning of the algorithm will be valid during all interior point iterations, which is not the case when the augmented system is factored [6, 9].

In [6] the Bunch-Parlett factorization was proposed to solve the indefinite system (7). Computational results indicated the usefulness of the Bunch-Parlett factorization, but showed that it tends to be computationally more burdensome.

Vanderbei derived the implicit elimination for the primal affine scaling algorithm [15]. The approach presented by the author uses the Schur complement mechanism to overcome the semidefiniteness of $\tilde{D}$. Further investigations of the Schur complement method in context of interior point methods indicated that it works efficiently if only few columns of A have to be handled with it (in our case, if the LP problem contains few free variables only) [4]. Otherwise the approach and its supplementary stabilization techniques are excessively expensive.

We will combine the split variable representation method (which actually regularizes $\tilde{D}$, see in [9]), and the implicit variable elimination method to exploit their benefits. We will use a regularized scaling matrix:

$$D_{ii} = \begin{cases} x_i^{-1} z_i & i \notin F \\ \epsilon & i \in F \end{cases}, \tag{9}$$

where $\epsilon > 0$, but keep the right hand side of (7).

4 PROPERTIES OF THE REGULARIZED SYSTEM

In our approach we regularize matrix (8) by replacing its diagonal zero elements by ϵ. Obviously, changing a zero value in a matrix to a nonzero one can heavily influence the behavior of the system, and it does not stand to reason that such a modification in our case does not change the solution of the system substantially. In this section we will derive error bounds on the difference of the solution by the original and regularized systems.

Let $\tilde{M}$ denote the matrix at the left hand side of equation (7), and M its regularized version, where (9) is used in place of the $\tilde{D}$. Let $\|\,.\,\|$ denote the Euclidean norm. Furthermore, let

$$\begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix} = \tilde{M}^{-1} \begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = M^{-1} \begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix} .$$

In the first step we examine the case if only one position (i,i) is regularized in (7). In this case, $\tilde{M} = M + \epsilon e_i e_i^T$. Let $u^i = M^{-1} e_i$. We can compute $\tilde{M}^{-1}$ via the formula of the modified matrix inverse as:

$$\tilde{M}^{-1} = M^{-1} - \frac{\epsilon M^{-1} e_i e_i^T M^{-1}}{1 + \epsilon e_i M^{-1} e_i} \tag{10}$$

It follows from this that

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} - \begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix} = \frac{\epsilon \Delta x_i M^{-1} e_i}{1 + \epsilon (M^{-1})_{ii}}. \tag{11}$$

Unfortunately, in (11) the denominator also depends on ϵ. In the following, we will show that $1 + \epsilon (M^{-1})_{ii}$ is bounded.

Lemma 4.1.

$$0 < 1 + \epsilon (M^{-1})_{ii} \le 1.$$

Proof. To compute $(M^{-1})_{ii}$ one can use the rule for computing the inverse of a block matrix. Let

$$M^{-1} = \begin{bmatrix} \Delta & B^T \\ B & C \end{bmatrix}$$

the block decomposition suitable to the blocks of M, then

$$\begin{aligned} \Delta &= -D^{-1} + D^{-1}A^T(AD^{-1}A^T)^{-1}AD^{-1}, \\ (\Delta)_{ii} &= -\frac{1}{\epsilon} + \frac{1}{\epsilon}(A^T(AD^{-1}A^T)^{-1}AD^{-1})_{ii}. \end{aligned}$$

In this way,

$$1 + \epsilon(M^{-1})_{ii} = (A^T(AD^{-1}A^T)^{-1}AD^{-1})_{ii}.$$

Let us substitute D^{-1} by $D^{-\frac{1}{2}}D^{-\frac{1}{2}}$ (note, D is a positive definite diagonal matrix), and we gain

$$\begin{aligned} (A^T(AD^{-1}A^T)^{-1}AD^{-1})_{ii} &= \frac{1}{\sqrt{\epsilon}}\left(a_i^T\left(AD^{-\frac{1}{2}}D^{-\frac{1}{2}}A^T\right)^{-1}AD^{-\frac{1}{2}}\right)_i & (12) \\ &= \frac{1}{\sqrt{\epsilon}}\left(a_i^T\left(AD^{-\frac{1}{2}}\right)^{+}\right)_i & (13) \end{aligned}$$

where $\left(AD^{-\frac{1}{2}}\right)^{+}$ denotes the Moore-Penrose pseudoinverse of $AD^{-\frac{1}{2}}$ and a_i the i-th column of A. As well known, $v = a_i^T\left(AD^{-\frac{1}{2}}\right)^{+}$ is that minimizer of

$$\| AD^{-\frac{1}{2}}\hat{v} - a_i \|^2 \qquad (14)$$

which has the smallest L_2 norm. Since for $\bar{v} = \sqrt{\epsilon}e_i$

$$\| AD^{-\frac{1}{2}}\bar{v} - a_i \|^2 = 0,$$

we can obtain v by the following optimization problem:

$$\begin{aligned} \min \; & \| v \|^2, \\ AD^{-\frac{1}{2}}v &= a_i. \end{aligned} \qquad (15)$$

Now we write v as the convex combination of two orthogonal components:

$$v = p\bar{v} + (1-p)\tilde{v}$$

where $0 \le p \le 1$, $\bar{v} = \sqrt{\epsilon}e_i$ and $\tilde{v} \in \{R^n,\ \tilde{v}_i = 0\}$. Because of the orthogonality, $\tilde{v}$ can be obtained by the following problem:

$$\begin{aligned} \min \; & \| \hat{v} \|^2, \\ AD^{-\frac{1}{2}}\hat{v} &= a_i \\ \hat{v}_i &= 0. \end{aligned} \qquad (16)$$

Let s denote the infimum of the objective function of problem (16). Let $s = +\infty$ if the problem has no feasible solution. Let us observe that s is independent of ϵ. To obtain v, we have to minimize the expression

$$\| v \|^2 = p^2 \epsilon + (1-p)^2 s. \tag{17}$$

It is easy to derive that the minimum of (17) takes place at

$$p = \frac{s}{s+\epsilon}$$

and so the denominator of the expression on the right hand side of (11) is

$$\begin{aligned} 1 + \epsilon (M^{-1})_{ii} &= \frac{1}{\sqrt{\epsilon}} v_i \\ &= \frac{1}{\sqrt{\epsilon}} (p \bar{v}_i + (1-p) \tilde{v}_i) \\ &= \frac{s}{s+\epsilon}. \end{aligned}$$

It is to be observed that

$$0 < \frac{s}{s+\epsilon} \leq 1 \tag{18}$$

which proves our lemma. □

Note that $\frac{s}{s+\epsilon} = 1$ if and only if (16) has no feasible solution, that is if a_i is linearly independent from the other columns of A.

Theorem 4.2. *The regularization error is of order ϵ that is*

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} - \begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix} = \epsilon \Delta(\epsilon)$$

where $\Delta(\epsilon)$ is bounded.

Proof. Substituting the result of the lemma as

$$\epsilon (M^{-1})_{ii} = \frac{s}{s+\epsilon} - 1 \tag{19}$$

into (11) results in

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} - \begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix} = \epsilon \Delta x_i u^i + \frac{\epsilon^2 \Delta x_i u^i}{s}. \tag{20}$$

Hence $\tilde{M}$ furthermore M for any ϵ are nonsingular and s is independent of ϵ, expression (20) shows that the perturbation introduced by the regularization is of order ϵ. □

Clearly, during interior point iterations, once we have computed a decomposition of M, with only one more backsolve operation, namely by computing $u^i = M^{-1} e_i$ we can compute the solution of the unregularized equations system with an arbitrary right hand side.

In the following we will give a computationally cheap explicite method to reduce the regularization error.

Theorem 4.3. $\begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix}$ *can be obtained by the iterative refinement scheme*

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^0 = M^{-1} \begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix}, \tag{21}$$

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^{k+1} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^k + M^{-1} \left(\begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix} - \tilde{M} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^k \right). \tag{22}$$

Proof. To prove the convergence of (21-22) one should observe that

$$\tilde{M} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^k = \begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix} + \epsilon \left(\Delta x^k\right)_i e_i$$

hence

$$M^{-1} \left(\begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix} - \tilde{M} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^k \right) = -\epsilon \left(\Delta x^k\right)_i M^{-1} e_i.$$

By induction on k it is easy to see that

$$\epsilon \left(\Delta x^k\right)_i = \epsilon^{k+1} \left(\Delta x^0\right)_i (u_i^i)^k$$

therefore

$$-\epsilon \left(\Delta x^k\right)_i M^{-1} e_i = -\epsilon^{k+1} \left(\Delta x^0\right)_i (u_i^i)^k u^i.$$

It follows from the foregoing that

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^k = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^0 + \left(\sum_{j=0}^{k-1} -\epsilon^{j+1} \left(\Delta x^0\right)_i (u_i^i)^j \right) u^i \quad \text{for} \quad k > 0. \tag{23}$$

Let us observe that the multiplicator of u^i in (23) is a geometric series with $-\epsilon(\Delta x^0)_i$ starting value and $-\epsilon u_i^i$ quotient. Since $(M^{-1})_{ii} = u_i^i$, it follows from (19) and (18) that $0 \leq -\epsilon u_i^i < 1$, therefore the geometric series has a finite limit value, and

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^\infty = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}^0 - \frac{\epsilon \left(\Delta x^0\right)_i u^i}{1 + \epsilon u_i^i} = \begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix}.$$

□

In other words, the iterative refinement which is considered as a standard technique to improve numerical stability in interior point implementations, for any regularization $\epsilon > 0$ can be performed with matrix M to compute the solution by the unregularized system.

Now let us consider the case when the set of the free variables is $F = \{i_1, i_2, ..., i_k\}$. We can apply (20) recursively, at all times taking one more regularization into account. As a result, we gain

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} - \begin{bmatrix} \tilde{\Delta x} \\ \tilde{\Delta y} \end{bmatrix} = \epsilon \sum_{i \in F} \Delta x_i u^i + o(\epsilon^2). \tag{24}$$

In this case, one can obtain the first order error term by computing

$$M^{-1}(\sum_{i \in F} \Delta x_i e_i).$$

As in what has gone once before, by investigating each component in the residual, the convergence of the iterative refinement (21-22) can be derived.

The important parameter of our approach is the amount of regularization, ϵ, which balances between stability and efficiency. Increasing ϵ can result in more stable factorizations, but also alters the search directions stronger which may lead to less efficient steps or more necessary corrections during iterations. This effect was demonstrated in [9]. Rather than using a fixed value, we describe here an adaptive procedure to determine the regularization ϵ in each iteration. To derive a practically efficient rule for obtaining ϵ we suppose that the columns of A are of the same norm (this can be done by trivial scaling of the LP problem prior to applying the interior point algorithm) and that the positive components of x at the optimum are of the same order of magnitude. This latter requirement is less trivial, but also fulfilled by most real life applications.

First, we define at iteration k the tentative partition P^k of those nonnegative primal variables which are positive at the optimum, as

$$P^k = \{j : j \notin F, \ |\Delta x_j^a|/x_j \le |\Delta z_j^a|/z_j\},$$

where $(\Delta x^a, \Delta z^a)$ is the primal-dual affine scaling direction at the previous iteration [11]. We selected this indicator because it is independent of problem scaling and has been justified by both the theory and practice [2]. Since the effect of the regularization depends on the relation between the columns of free variables and the remaining part of $AD^{-\frac{1}{2}}$, our assumption is that variables $\{x_j : j \in \{1, ..., n\} \setminus (P^k \cup F)\}$ play less important roles and we have to concentrate on the behavior of the variables defined by P^k. We compute the "average" scaling factor of the "important" variables and multiply it by the $rtol$ parameter:

$$\delta = rtol \left(\prod_{j \in P^k} D_{jj} \right)^{\frac{1}{|P^k|}}. \tag{25}$$

To avoid an increase in the condition number of D, we define the regularization for the forthcoming iteration as

$$\epsilon_{k+1} = \max \left\{ \min_{i \in P^k} D_{ii} \ , \ \delta \right\}. \tag{26}$$

Our experiments indicated that the above technique with $rtol = 10^{-6}$ is reliable both numerically and algorithmically.

5 SPECIAL PRESOLVE TECHNIQUES

Investigations of interior point implementations indicated that presolve techniques which reduce the original problem size are particularly important in interior point implementations. One such reduction possibility, which is not used frequently in practice is the elimination of free variables. Since linear programming problems often have no or only few free variables, the space for this kind of reduction is very limited in most of the cases. First, we will describe how some of those variables can be identified as "free" which have finite bound(s). Next, we will describe an elimination process which reduces the problem by elimination of (not necessarily all) free variables.

In the LP problem the variables have explicit, possibly infinite bounds. An often used presolve technique is to define upper and lower limits for constraints $i = 1, ..., m$ as

$$\underline{b}_i = \sum_{\substack{j=1,..n \\ a_{ij}>0}} a_{ij} l_j + \sum_{\substack{j=1,..n \\ a_{ij}<0}} a_{ij} u_j \qquad \text{and} \qquad \bar{b}_i = \sum_{\substack{j=1,..n \\ a_{ij}>0}} a_{ij} u_j + \sum_{\substack{j=1,..n \\ a_{ij}<0}} a_{ij} l_j$$

where u_j and l_j are individual upper and lower bounds of x_j. Based on the property that

$$\underline{b}_i \leq \sum_{j=1,..,n} a_{ij} x_j \leq \bar{b}_i$$

for any solution x which satisfies the individual bounds, infeasibility or redundancy in the constraint set can be detected [3]. From investigation of the upper and lower constraint limits and their relation with individual variables one can derive "tighter" bounds on the variables (that is either larger lower or smaller upper bounds). This technique is referred as *tightening variable bounds* [3]. Recent interior point presolve techniques (see e.g. [1]) restore the original bounds or keep the tightened bounds and solve the modified LP problem. In our approach we concentrate on making as many variables free as possible, and eliminating as many of those possible. Therefore instead of restoring the original or keeping the modified bounds, we relax those for which the bound tightening procedure generated a tighter bound than the original one.

New implied bounds for variables can be computed as

$$u_j^{'} = \min_i \begin{cases} \left(b_i - \sum\limits_{\substack{k \in \{1,..n\} \setminus \{j\} \\ a_{ik}>0}} a_{ik} l_k + \sum\limits_{\substack{k \in \{1,..n\} \setminus \{j\} \\ a_{ik}<0}} a_{kj} u_j \right) / a_{ij}, & a_{ij} > 0 \\ \left(b_i - \sum\limits_{\substack{k \in \{1,..n\} \setminus \{j\} \\ a_{ik}<0}} a_{ik} l_k + \sum\limits_{\substack{k \in \{1,..n\} \setminus \{j\} \\ a_{ik}>0}} a_{kj} u_j \right) / a_{ij}, & a_{ij} < 0 \end{cases},$$

$$l_j^{'} = \max_i \begin{cases} \left(b_i - \sum\limits_{\substack{k\in\{1,..n\}\setminus\{j\}\\ a_{ik}<0}} a_{ik} l_k + \sum\limits_{\substack{k\in\{1,..n\}\setminus\{j\}\\ a_{ik}>0}} a_{kj} u_j \right) / a_{ij}, & a_{ij} > 0 \\ \left(b_i - \sum\limits_{\substack{k\in\{1,..n\}\setminus\{j\}\\ a_{ik}>0}} a_{ik} l_k + \sum\limits_{\substack{k\in\{1,..n\}\setminus\{j\}\\ a_{ik}<0}} a_{kj} u_j \right) / a_{ij}, & a_{ij} < 0 \end{cases}.$$

If $u_j^{'} < u_j$ (or $l_j^{'} > l_j$) then we tighten the bound as $u_j := u_j^{'}$ (or $l_j := l_j^{'}$) and mark the bound which was tightened. Otherwise if $u_j^{'} = u_j$ (or $l_j^{'} = l_j$) we relax the bound by setting $u_j := +\infty$ (or $l_j := -\infty$). This investigation may be performed successively until no modification is available. At the end of the procedure we relax all bounds which were marked during the bound tightening. It is to be noted that relaxation of bounds in this case is possible because the bound tightening "proved" that the variable is more constrained by other constraints than by its own bound, therefore the individual bound can be relaxed.

This approach has two benefits. It opens more space for the elimination which in turn reduces the size of the problem; and removal of bounds simplifies the logarithmic barrier function (6) and makes the first order optimality conditions of the barrier problem less complex by decreasing its nonlinearity.

The successive computation of implied bounds can be implemented efficiently by using counters and update techniques. We observed that the order of the rows and columns in the bound tightening investigation has strong influence on the success of the procedure. Based on our numerical experiments, we suggest that the rows and columns are processed in increasing order of their nonzero count.

It is widely known that free variables can be eliminated from the linear programming problems using the standard Gauss elimination procedure. But, apart from special cases (singleton columns, doubleton rows) this possibility was not exploited up to recently. In this section we describe the elimination process, which is implemented as a feature of the presolve in BPMPD.

Let C denote the set of column indices and R the set of row indices. Let c_j the number of nonzeros in column $j \in C$ and r_i the number of nonzeros in row $i \in R$. Furthermore, to avoid numerical instability as well as fill–in, we define a pivot tolerance *ptol* and a sparsity tolerance *stol*. We search for a pivot a_{ij}, $j \in F$, $i \in R$ for which

$$|a_{ij}| \geq ptol\,(\max_{k\in R} |a_{kj}|), \quad \text{and} \tag{27}$$

$$(c_j - 1)(r_i - 1) \leq stol\,(c_j + r_i). \tag{28}$$

If more than one pivot candidate satisfies the stability (27) and sparsity (28) requirements, we select one for which the Markowitz count $(c_j - 1)(r_i - 1)$ is minimal. Once a pivot is selected, the constraint matrix is to be transformed as

$$a_{kl} := a_{kl} - \frac{a_{kj} a_{il}}{a_{ij}} \;\; \text{for} \;\; k \in R\setminus\{i\},\; l \in C\setminus\{j\}, \tag{29}$$

the right hand side as

$$b_k := b_k - \frac{a_{kj} b_j}{a_{ij}} \quad \text{for } k \in R \backslash \{i\} \tag{30}$$

and the objective function as

$$c_l := c_l - \frac{a_{il} c_i}{a_{ij}} \quad \text{for } l \in C \backslash \{j\}. \tag{31}$$

Furthermore we remove i from C and F, j from R and update the column and row counts. The process terminates if no other pivot can be selected, either because F is empty, or no pivot satisfies the tolerances. In our experiment $ptol = 10^{-3}$ and $stol = 4.0$ were used.

TABLE 1: Bound relaxation and elimination of free variables

Problem name	Orig.free variables	Bounds relax.	Final free variables	Elimin.free variables	Fill–in by elimination
80bau3b	0	549	434	0	0
bore3d	0	60	53	53	225
capri	0	134	115	104	1150
dfl001	0	2179	2179	2179	9576
ganges	0	479	433	433	966
greenbeb	4	1100	1017	704	5910
ken-18	0	86526	24893	24893	28770
nesm	0	110	72	72	618
psd-10	0	20412	6702	6682	43855
pilot	0	660	506	106	1928
pilotnov	0	259	232	133	1524
scfxm256	0	7215	6413	6413	63867
stocfor3	0	7515	7515	7242	60155
world	0	9116	4812	35	1996

Tab. 1 shows on few examples, how many "hidden" free variables can be detected and how many of them can be eliminated by the afore mentioned technique. Figures given include the number of free variables in the original problem, the number of relaxed individual bounds, the number of free variables after bound relaxation, the number of free variables eliminated and the fill–in during the elimination.

As a result of the elimination, the problems contain some new "fill-in" nonzeroes but less rows and columns. An important issue is how this influences the efficiency of the computations in the interior point algorithm. Tab. 2 compares some of the important characteristics of the symmetric decomposition of the augmented system with ("elim") and without ("noelim") the elimination of free variables. As performance indicators, the number of nonzeros in the factorization of the augmented systems (FNZ), the number of floating point operations (in thousands) needed to compute one decomposition (FLOPS) and the time in seconds on a Sparc-2000 workstation to compute one factorization (TIME) were used.

TABLE 2: Fill–in and and speed of factorizations

Problem	FNZ		FLOPS		TIME	
name	noelim	elim	noelim	elim	noelim	elim
80bau3b	53476	53476	893	893	0.74	0.74
bore3d	1377	613	10	4	0.01	0.01
capri	4983	5020	52	89	0.03	0.04
dfl001	14313801	1090680	506614	350559	149.83	116.23
ganges	25000	15021	291	126	0.13	0.05
greenbeb	67414	58184	801	748	0.51	0.45
ken-18	2194501	2072844	101168	100600	62.43	58.11
nesm	32646	30805	524	463	0.21	0.18
psd-10	1675223	1006234	460186	190676	141.94	68.13
pilot	208150	206365	18336	18408	5.15	5.12
pilotnov	51451	54551	2001	2124	0.54	0.55
scfxm256	551647	591776	6362	8353	5.31	6.73
stocfor3	215413	159583	1477	1396	1.31	0.97
world	1104334	1113884	52000	52226	19.60	19.60

Results presented in Tab. (2) indicate that elimination of free variables applied prior to the interior point method does not increase remarkably the computational work per iteration. In some of the cases, however, the computation speed is improved significantly. An additional effect of the smaller problem dimensions is that better numerical and algorithmical behavior can be expected.

6 COMPUTATIONAL RESULTS

In our computational results we compare the interior point codes PCx and BPMPD. Both of the codes are implementations of the infeasible primal–dual logarithmic barrier algorithm, but PCx uses variable replacement while BPMPD implements the afore mentioned techniques for handling free variables. Since the goal of our experiments is to demonstrate the differences between the variable replacement and our approach, we have selected test problems from the water management area [8]. These models, whereas they are of small to medium size, contain a significant number (up to 15%) of free variables. The problems presented in Tab. 3 have about 3100 constraints, 3250 variables of which 15 % are free and 9000 nonzeros.

Tab. 3 shows the comparison of the performance of the two implementations. The figures given include the number of iterations to optimality, the total solution time and the accuracy of the solution. All timing results are given in seconds on a Sparc 2000 workstation, and reflect on the pure algorithmic time, without the input of the MPS file. Both of the solvers were configured to stop when the relative

TABLE 3: Comparison of PCx and BPMPD

Problem	Iterations		Solution time		Significant digits	
name	PCx	BPMPD	PCx	BPMPD	PCx	BPMPD
pldd000b	39	25	14.08	11.60	6	9
pldd001b	39	24	13.71	12.56	6	10
pldd002b	39	27	13.78	13.02	6	8
pldd003b	37	26	12.76	13.01	5	9
pldd004b	39	25	13.96	11.11	5	9
pldd005b	36	25	12.71	11.25	5	9
pldd006b	37	23	13.39	10.82	7	8
pldd007b	37	24	12.80	10.82	7	9
pldd008b	38	25	14.07	12.86	6	9
pldd009b	37	24	14.23	11.84	5	9
pldd010b	37	24	14.17	11.60	6	9
pldd011b	37	25	13.90	11.61	6	9
pldd012b	35	24	13.45	11.00	5	8

duality gap, computed by formula

$$\frac{c^T x - b^T y}{|c^T x| + 1.0}$$

was less than 10^{-8} and both the relative primal infeasibility $\frac{\|Ax-b\|_2}{\|b\|_2+1.0}$ and relative dual infeasibility $\frac{\|c-A^T y-z\|_2}{\|c\|_2+1.0}$ were below 10^{-8}.

Results show that the desired accuracy was not achieved by PCx, because of numerical instabilities at the last stage of the iterations, but BPMPD always resulted in a solution within the tolerance limits. In the case of PCx during the last few (usually 5–7) iterations the quality of the current iterate was not improved, until the solver detected the bad numerical behavior, stopped the iterations and restored the best solution found so far.

7 CONCLUDING REMARKS

As pointed out in [10], efficient handling of free variables may result in a semidefinite scaling matrix in interior point methods. Since this prevents the use of the powerful normal equations approach, many researchers implement the variable replacement for handling free variables [5, 14]. Variable replacement, however, results in serious numerical difficulties on linear programming problems with several free variables. In the paper we presented an approach which regularizes the semidefinite scaling matrix during interior point iterations. The regularization makes the scaling matrix definite, therefore the use of the normal equations approach is possible. But, our approach avoids variable replacement and numerical difficulties. In the paper we discussed the effect of the regularization. We showed how the solution of the unregularized system depends on the regularized one. A cheap and reliable method was proposed to control the influence of the regularization. It was shown that independently of the amount of regularization, an iterative refinement scheme can always serve to approach closer to the unregularized solution. We demonstrated how "hidden" free variables can be detected. We described an algorithm which is able to reduce the problem by elimination of some (not necessary all) free variables without considerable fill–in or loss in numerical behavior.

In our analysis we considered the case where each singularity was regularized by the same $\epsilon > 0$. The extension of the approach, in which different singularities are regularized by different values is also possible. This needs only small changes in our proofs.

At last, we would like to mention that our observations are valid if "near" semidefinite scaling matrices are regularized. This means that "too small" diagonal values in (8), which may cause numerical instability, can be improved during iterations.

The regularization techniques, mentioned above are recently implemented in BPMPD. For computing the scaling matrix instead of (9) the formula

$$D_{ii} = \begin{cases} x_i^{-1} z_i & \text{if} \quad i \notin F \quad \text{and} \quad x_i^{-1} z_i \geq \delta \\ \sqrt{x_i^{-1} z_i \delta} & \text{if} \quad i \notin F \quad \text{and} \quad x_i^{-1} z_i < \delta \\ \epsilon & \text{if} \quad i \in F \end{cases}$$

is used in the solver, where δ is defined by (25) and ϵ by (26). According to our experiences, this regularization scheme works efficiently and helps to avoid numerical difficulties in the practice.

REFERENCES

1. E.D. Andersen and K.D. Andersen. Presolving in linear programming. *Math. Programming*, 71(1):221–245, 1995.
2. E.D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programs. In T. Terlaky, editor, *Interior point methods of mathematical programming*, pages 189–252. Kluwer Academic Publisher, 1996.
3. A.L. Brearley, G. Mitra, and H.P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Math. Programming*, 15:54–83, 1975.
4. T.J. Carpenter, I.J. Lustig, J.M. Mulvey, and D.F. Shanno. Higher order predictor-corrector interior point methods with application to quadratic objectives. *SIAM Journal on Optim.*, 3:696–725, 1993.
5. J. Czyzyk, S. Mehrotra, and S. J. Wright. PCx user guide. Technical Report OTC 96/01, Optimization Technology Center, Aragone National Laboratory and Northwestern University, 1996.
6. R. Fourer and S. Mehrotra. Solving symmetric indefinite systems in an interior point method for linear programming. *Math. Programming*, 62:15–40, 1993.
7. D. M. Gay. Electronic mail distribution of linear programming test problems. *COAL Newsletter*, 13:10–12, 1985.
8. A.H. Lobbrecht. *Dymanic Water-System Control, Design and Operation of Regional Water-Resources Systems*. PhD thesis, TU Delft, Department of Civil Engineering, 1997.
9. I. Maros and Cs. Mészáros. The role of the augmented system in interior point methods. Technical Report TR/06/95, Brunel University, Department of Mathematics and Statistics, London, 1995. to appear in European Journal of Operations Research.

10. S. Mehrotra. Handling free variables in interior methods. Technical Report 91-06, Department of Industrial Engineering and Managment Sciences, Northwestern University, Evanston, USA., March 1991.
11. S. Mehrotra and Y. Ye. Finding an interior point in the optimal face of linear programs. *Math. Programming*, 62(3):497–515, 1993.
12. Cs. Mészáros. The augmented system variant of IPMs in two–stage stochastic linear programming computation. Working paper WP 95–1, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1995. to appear in European Journal of Operations Research.
13. Cs. Mészáros. *The Efficient Implementation of Interior Point Methods for Linear Programming and their Applications.* PhD thesis, Eötvös Loránd University of Sciences, 1996.
14. R. Vanderbei. LOQO user's manual. Technical Report SOR-96-07, Princeton University, School of Engineering and Applied Science, Dept. of Civil Engineering and Op.Res., 1996.
15. R. J. Vanderbei. Affine-scaling for linear programs with free variables. *Math. Programming*, 43:31–44, 1989.