The Role of the Augmented System in Interior Point Methods*

István Maros[†] Csaba Mészáros[‡]

Brunel, The University of West London, U.K.,
and Computer and Automation Institute, Budapest, Hungary

Revised
September 1996

Abstract

We present a way to use augmented system approach in interior point methods. We elaborate on the increased freedom in determining the pivot order which makes this approach computationally very competitive. This means that with the pivot search heuristics presented here, in most of the cases we can achieve a performance not worse than the $AD^{-1}A^T$ method, and in several cases much better. In reality, the augmented system seems to be the only safe method in case of dense columns or "bad" nonzero pattern. We found that both methods are important for their usefulness. Our implementation includes both. It is also equipped with an analyzer that is able to determine which of them to use. It is based on the evaluation of the nonzero pattern of constraint matrix. We also point out that the treatment of free variables is also more efficient in the framework of the augmented system. We report on some very favorable computational experiences achieved with our implementation of the augmented system based on these ideas.

Keywords: Linear programming, interior point methods, augmented system

1 Introduction

The augmented system variant of interior point methods has long been known, but until recently it has not been considered as a serious alternative to the well established $AD^{-1}A^T$ version.

*This work was supported in part by Hungarian Research Fund OTKA 2587.

[†]Now at Imperial College, London, Email: `i.maros@doc.ic.ac.uk`

[‡]This author was partly supported by Hungarian Research Fund OTKA W-015800.

A recent feature article [13] gives a survey of the development achieved with the different algorithmic variants of interior point methods. However, computationally it restricts the discussion to solving the normal system that leads to $AD^{-1}A^T$ with some positive diagonal matrix D. Though it is numerically very convenient, it may suffer from heavy fill-in. If some tricks (e.g. Schur complement to restrict the growth of nonzeros during factorization) are implemented then the favorable numerical properties may disappear, as pointed out by Lustig, Marsten and Shanno [12] and, indeed, problems may become unsolvable. This can be well observe with the most recent version of the state-of-the-art commercial solver CPLEX (V4.0), as we demonstrate it in section 6. Additionally, the efficient handling of the free variables is not easy with this approach.

These problems have been recognized by several researchers and alternative ways have been looked for. Some authors refer to it as search for a "spare engine" for the case when (tuned) Cholesky factorization of $AD^{-1}A^T$ fails.

The first approach to keep fill-in under control was introduced by Todd [21] and elaborated by Hurd and Murphy [10]. The authors consider the case where the constraint matrix has a special primal block angular or doubly bordered block diagonal structure. The presented method works well in such cases, but cannot easily be generalized for handling general sparse matrices. The authors demonstrated the efficiency of this approach only a special set of the applications, such as network and stochastic problems or for handling generalized and variable upper bounds.

Another technique due to Birge, Freund and Vanderbei [2] uses the choice between the AA^T and A^TA forms. The systems of equations to be solved are $m \times m$ or $n \times n$ depending on the choice of the algorithm. Experiments and comparisons show that the approach works well only in the case if $m \gg n$ and A does not contain dense row(s).

One more possibility is the well know "column splitting" technique. An implementation and some computational results can be found in [7]. The comparison of the results reported in [23], [25], and [7] with ours can be found in section 4.

Several attempts have been made to use the augmented system as an alternative to the normal equations. Fourer and Mehrotra [5] reported that, using their pivot order, the augmented system is inferior in most of the cases. Similar can be said about the findings of Vanderbei and Carpenter in [25]. They experienced significantly better efficiency only on four problems of the NETLIB set [6] (`fit1p, fit2p, israel` and `seba`) with their pivot selection heuristics.

The methods for handling free variables obtained much less attention than it would have deserved. One reason for it is that from the 100 problems of the commonly used LP test set NETLIB only 7 problems have more than 4 free variables. However, in practice much more problems can occur with many free variables. One source of them is the dualization of problems, where $m \gg n$. The simplest method for handling the free variables is to split them into positive and negative parts [13], also known as *variable replacement*. This approach may result in serious numerical troubles on problems with several free variables. 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 fill-in.

Vanderbei introduced a method for both the primal and the dual affine-scaling algorithm [27] with algebraic modification of the $AD^{-1}A^T$ factorization and with limit investigation of the variable replacement. Mehrotra in [16] derived the same approach for the primal-dual method by considering a modification of the logarithmic barrier function and using the augmented system. Numerically both methods are superior to the variable replacement method and theoretically equivalent to explicit variable elimination. The only disadvantage is the possibility of big fill-in on problems with several free variables. We show another way to derive a rule for handling free variables, which allows more freedom for the pivot selection and can result in reduced fill-in.

The advantage of the augmented system was particularly brought to our attention in respect of a few industrial models which were under investigation by the Mathematical Programming Research Group at Brunel University. The research system FortMP [14] developed by the group has sparse simplex and predictor-corrector IPM capabilities. This latter uses the normal equation approach, incorporates supernode elimination strategy to take advantage of vector processing [18] and is also adapted to run on SIMD parallel computers [9]. Yet, in case of these industrial models FortMP/IPM on parallel machines was unable to match computational results achieved by the augmented system approach of solver BPM using a powerful PC.

Our aim in this paper is to exploit the possibilities of the augmented system as much as possible but preserve the benefits of normal equations when appropriate. This results in an implementation that includes both versions. One of the most important steps of a solver based on the above idea is the choice between the two systems at the analyze phase.

An important step in verifying some computational ideas is testing. To compare algorithms, testing must be done on the same set of test problems. For linear programming, there is a well established set of test problems in NETLIB. Nowadays, it contains some 100 reasonably sized real life models, including some hard ones. Any claim on the performance of an implemented algorithm can easily be checked on them.

Unfortunately, this is not true in the case of some recent reports and claims ([3],[13]) on computational performance. The authors successfully solved some proprietary large scale problems clearly displaying the extended capabilities of their solvers. We attempted to obtain copies of the problems to try to solve them with our new solvers that we believe to be competitive with them. Since the requests have been turned down due to their proprietary nature of the problems, we were obliged to do a comparative testing in the following way.

We have access to CPLEX 4.0 with the barrier optimizer feature. We ported our solvers to the same Pentium 90 MHz based computer where CPLEX is available and carried out the comparison there. In the first part of the testing, we solved the problems of NETLIB/LP/DATA. To get some more insight into performance, we took problems from other sources and solved them by CPLEX and by our solvers. The details of the testing are described in section 6.

The rest of the paper is organized in the following way. In section 2 we present a common form of the interior point methods with the help of the augmented system and show the relationship between the two approaches (normal equations and augmented system). This enables us to discuss our new pivot selection heuristics in section 3. In section 4 we compare the performance of these heuristics with other reported results in case of some critical problems. Efficient handling of free variables is discussed in section 5. Our ideas have been implemented in our solver BPM. Section 6 is devoted to presenting results of detailed computational testing. The test involves problems of NETLIB and some other large real-life problems that appeared to have been interesting for the solvers. We summarize our findings in section 7.

2 The algorithm paradigm

To present our ideas, we need the general framework of the augmented system. In practice, the logarithmic barrier methods from the huge family of interior point algorithms seem to be the most powerful tools for solving large scale LP problems. characteristics to the competitive efficient implementation techniques logarithmic barrier method have appeared in the literature; readers interested in the full development of these algorithms are referred to the survey by Gonzaga [8], or Lustig, Marsten and Shanno [13].

We consider the following primal linear programming problem:

$$\begin{array}{lrcl} \text{minimize} & c^T x, & & \\ \text{subject to} & Ax & = & b, \\ & x & \geq & 0, \end{array} \tag{1}$$

where A is an $m \times n$ matrix, c and x are n vectors, and b is an m vector. We assume that A is of full row rank.

Let y denote the m vector of dual variables. The dual problem associated with (1) can be written as follows:

$$\begin{array}{lrcl} \text{maximize} & b^T y, & & \\ \text{subject to} & A^T y + z & = & c, \\ & z & \geq & 0, \end{array} \tag{2}$$

where z is the n vector of dual slack variables.

Let X denote the diagonal matrix whose diagonal elements are the components of x, i.e., $X = \mathrm{diag}(x)$, and let $Z = \mathrm{diag}(z)$.

The barrier algorithms generate a sequence of primal and/or dual solutions, which converge to the optimum. The fundamental task of the logarithmic barrier methods is to compute the search direction. In the general case the search direction can be obtained from the following set of equations:

$$\begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{3}$$

In the context of interior point methods this set of equations is usually referred to as the *augmented system*. It is to be noted that the matrix of coefficients on the left hand side of (3) is a symmetric indefinite matrix. Practically any interior point method can be discussed within this framework. In every iteration the diagonal matrix D and the right hand side vector of (3) are changed, corresponding to the rule of the algorithm in question.

In the paper we discuss the primal-dual log barrier algorithm, where:

$$\begin{array}{llll} D = -Z^2 & \alpha = -\mu Z^{-1} e & \beta = b & \text{in case of the dual method} \\ D = -X^{-1} Z & \alpha = Z - \mu X^{-1} e & \beta = 0 & \text{in case of the primal-dual method.} \end{array}$$

The so-called one-phase variant of these algorithm can be derived with minor modifications. For further details on this the reader is referred to the excellent summaries in [25] and [13].

To solve (3) an LU decomposition can be used. Depending on the pivot agenda implemented one can solve the system with varying efficiency (and reliability). If Δx is eliminated from (3) (which is equivalent to pivoting down the diagonal elements of D) then the remaining system will be just the system of normal equations $AD^{-1}A^T \Delta y = AD^{-1}\alpha - \beta$. This simply means that the normal equation approach is a special case of the augmented system (for more details, c.f. [26]).

The big advantage of the normal equation approach is that it deals with a system of linear equations with a positive definite coefficient matrix. As such, the Cholesky factorization can be used with great success. In its symbolic phase full attention can be paid to sparsity since numerically it is theoretically always stable. Once the symbolic phase is done it need not be repeated during iterations, that is, the pivot agenda is the same all the way. This, of course, results in an increased efficiency and in the capability of handling large scale problems.

However, difficulty arises when A is such that it contains dense columns. In such a case $AD^{-1}A^T$ is dense and the Cholesky factors are also dense. This immediately prevents the solution of large problems and results in a serious deterioration of efficiency on still tractable smaller problems.

To get around density, several ideas have been tried in the framework of the normal equations. One straightforward thought is *column splitting* (c.f. [7]). This requires the introduction of additional variables and a constraint per columns split. Though this in itself is usually not a problem since large scale problems do not have too many dense columns, but the method is not efficient in practice. The other way that was accepted by several authors is using the *Schur complement* mechanism. Unfortunately, this method has its own limitations. If the "sparse" part of A is not of full row rank it does not work, as pointed out in [11] and [12]. Our computational investigations (see section 6) show that solvers based on this technique can fail.

As an alternative, instead of trying to find further tricks for the normal equations, we advocate using the augmented system. If it is implemented properly the difficulty of coping with dense columns can be eliminated. Though in this case we have to deal with a considerably larger system it does not necessarily mean more work. The other issue that has to be made clear is that in using (3) we are faced with an indefinite system. This means that a pivot agenda determined at the beginning may not be valid for all the iterations. At the same time, we have a much greater freedom in choosing the pivot elements and this can be exploited for sparsity to a large extent.

3 Pivot selection for the augmented system

The pivot order has a very important influence on fill-in and therefore on the efficiency of interior point algorithms. Clearly, factoring the augmented system is the most general direct method for the computation of the search directions. It has much more freedom in determining the pivot order than the normal equation system approach, and its better numerical behavior is known [1]. The only one disadvantageous property is the indefiniteness of the matrix in (3). Similarly to the normal equation approach, we set up the pivot order for the augmented system at the beginning of the algorithm, and we use it as long as possible.

Actually, as a first step, we make a numerical factorization with $D = I$. The result of this factorization is used to determine the starting point.

In general, indefiniteness may cause problem, if some variables go very close to their boundaries at early stages of the iterations, still far away from optimality. This happens very rarely if we apply a variant of the primal dual method that efficiently follows the central path. If we reach a very small pivot value during the factorization process, and there are still some unprocessed columns of the original constraint matrix in the pivot order that can modify this diagonal value then there are two alternatives: (i) The very small values can simply be changed (see [20]) and then correcting later during the back-solve operation. Unfortunately, this carries some undesirable computational overhead per back-solve. (ii) The pivot order can be modified by moving this pivot to the end of the pivot sequence. In this way, this pivot will be positioned in the AA^T part of the order and, therefore, will be positive. This process is very simple and fast, but may create some more fill-in than previously. In our experience it has hardly any influence on the total solution efficiency.

Our basic idea in determining the pivot order for the augmented system is that we try to partition the augmented matrix into two parts. The first one is considered sparse and the second part is considered dense. (This does not necessarily mean that different storage schemes are used for the two parts.) In the first stage of the factorization we make pivot steps in the sparse part as long as possible. This stage may terminate because all columns have been pivoted, or because it was not possible to select a proper pivot element in the remaining columns. In the second stage we determine the pivot elements for the rest of the problem (part two and the remaining columns of part one). This selection is primarily based on the column count and a pivot tolerance. During this stage we may select pivots from the second but also from the first part if it was not fully pivoted.

We experienced that the information supplied by column counts may not be sufficient and deeper knowledge about the structure of the matrix would be needed. Therefore, we feel justified a partitioning procedure that can serve this purpose at little extra computational cost.

More formally, we look for a permutation that results in partitioning of A in such a way that

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}, \tag{4}$$

where A_1 is sparse with no dense columns or rows, A_2 and A_3 are considered dense borders. We consider this partitioning successful if the borders are relatively small, that is, if the number of columns in A_2 and, similarly, the number of rows in A_3 are much smaller than the number of columns/rows in A_1.

Let us note that our pivoting strategy is equivalent to the Schur-complement method up to the end of the first pass. Thereafter, however, the Schur-complement approach builds the decomposition of $\begin{bmatrix} A_1 \\ A_3 \end{bmatrix} D_1^{-1} \begin{bmatrix} A_1 \\ A_3 \end{bmatrix}^T$ immediately which is not necessarily done by our method.

Regarding system (3), first we pivot along the diagonal elements in D falling outside A_2. This will result in a very little fill-in. We go in this way as long as possible. When this part is exhausted we use a minimum degree rule for the rest of the augmented matrix.

It is clear that partitioning (4) plays a very important role in the pivot selection. If an algorithm based on a fixed threshold value is used to group columns into dense and sparse parts then we can observe better behavior only in very few cases [25], namely in those for which the threshold value is favorable. For different problems it can be different and manual tuning would be necessary to exploit the scheme. But a further question arises: On what information can such a manual tuning be based? We believe that no general rule can be determined for a decision to use the augmented system without examining the nonzero structure of the constraint matrix. At the same time, this examination itself and the algorithm based on it must be fast. This calls for some kind of an adaptive heuristic that is efficient and cheap.

In an attempt to fulfill these requirements as much as possible, we developed an algorithm that is described in the sequel.

The purpose of our heuristic algorithm is to find a partitioning where the density of A_1 and that of A_2 differ considerably. Further purposes are that A_1 should be as sparse as possible but should not be too small in size.

The main steps of the algorithm can be demonstrated by the following verbal description.

First, the procedure creates groups of columns having the same number of nonzero counts by group (groups of "equidense" columns). Each group will be characterized by the common nonzero count referred to as size.

At the beginning all columns are marked to belong to A_1. The groups are processed in decreasing order of size. The current next group becomes a candidate to join A_2. At each step we examine whether extending A_2 by this new group is acceptable or not. We use two criteria to decide.

Dimension condition (DIMCON): Will the dimension of A_1 remain greater than a predefined **relative measure**?

Density condition (DENCON): Is the size of the last group in A_2 significantly different from that of the current next group (which is the largest group by size in A_1)?

If DIMCON is true we check DENCON and proceed with the next group.

If DIMCON is false and DENCON was not true yet then we try to enlarge A_3 (if it is still not too large) by the row with largest row count and restart the procedure with the reduced matrix (and recomputed counts). If A_3 cannot be enlarged or DENCON was true at least once previously the procedure terminates.

If DENCON is true A_2 is enlarged by all the candidate groups identified so far.

There are many subtleties of the procedure that can be identified in the more formal description that follows. The procedure assigns label 'sparse' or 'dense' to each column of A.

Let c_j denote the number of nonzeros in column j (column c and r_i the number of nonzeros in row i (row count). We introduce the active set of column counts denoted by K, that is $k \in K \iff \exists j, c_j = k$. Let f_k denote the number of columns with column count value k.

1 let $M = m$, set all rows 'enabled', mark all columns 'sparse'
2 let $P = M + 1$
3 $N = n$
4 $p = \max\{k \in K$ and $k < P\}$
5 if $p < minlength$ GOTO 11
6 $N = N - f_p$
7 if $(M + N) < (m + n)(1 - \beta)$ GOTO 11
8 $P = \max\{k \in K$ and $k < p\}$
9 if $p^2/P^2 > dgap$ then
  mark all columns with $c_j > P$ as 'dense' and
  all others with $c_j < p$ as 'sparse'
10 GOTO 5
11 if $(M + n) < (m + n)(1 - \beta)$ or Step 9 was at least once true GOTO 17
12 select row j from the 'enabled' with maximal nonzero count
13 set row j disabled
14 $M = M - 1$
15 update counters c_* and f_* and the index set K
16 GOTO 2
17 terminate procedure

We can summarize the algorithm as follows:

- We accept the partitioning only if A_1 is relatively big (test in the Step 7).
- We do not search the partitioning when the maximum column count value is small.
- The difference between the sparsity of A_1 and A_2 is tested in the Step 9, where p^2 is the upper bound of the fill-in when pivoting on a column with column count p. The partitioning is accepted only if the difference between the 'sparse' and 'dense' parts is significant.
- We use the 'last' accepted partitioning (if there are more possibilities) where A_1 is the sparsest.

The parameter values used in our experiments are: $minlength = 10$, $\beta = 0.17$, $dgap = 2.5$.

4 Comparison with other results

The main role of the augmented system is to be a viable alternative for the case when the normal equation approach suffers heavy fill-in. We believe that the method described in this paper is a good candidate for this purpose.

Since there have been other attempts to handle such situations, we thought it instructive to compare their performance with that of ours. Unfortunately, the published results are rather scarce. Therefore, the usefulness of any of them cannot be judged in a reliable way. We think that the detailed performance results of our technique presented in Table 7 can serve as a basis for further comparisons.

Column splitting is an attempt to alleviate the problems caused by the presence of dense columns in A. Gondzio [7] presents the effect of this idea on two NETLIB problems, `israel` and `seba`. Vanderbei [23] used dualization for the same purpose. He analyzed three problems, `agg`, `fit1p`, and `fit2p`. We combine their findings with ours in Table 1, which shows the number of nonzeros and arithmetic operations during the factorization of the augmented system. We make a few modifications to make the results comparable. Namely, we have added the number of nonzeros to the off-diagonal nonzeros of the factorization in case of HOPDM [7] and ALPO [23]. Additionally, the number of arithmetic operations for ALPO has been divided by 2 to bring it on a common platform with our solver BPM. (In BPM an operation is defined as a pair of operations: a multiplication and an addition.) In Table 1 the number of operations is given in thousands. Cases when results are not available are denoted by N.A. This table clearly displays the favorable consequences of our pivot scheme.

Here we would like to emphasize again that heavy fill-in during factorization of $AD^{-1}A^T$ can be caused not only by a dense column but also by other nonzero patterns in A. The techniques quoted so far do not offer a remedy for these cases. Apparently, our partitioning scheme (4) can be used with more success. To demonstrate this view, we show

Table 1: Number of nonzeros and operations (the latter in thousands) with different sparsity preserving techniques

	Nonzeroes/Operations				
	agg	fit1p	fit2p	israel	seba
HOPDM	N.A.	N.A.	N.A.	9909/N.A.	24156/N.A.
ALPO	6937/96	33184/201	240684/1668	N.A.	N.A.
BPM	3440/30	10142/63	50583/266	4231/52	5486/27

the nonzero pattern of problem `route` in Figure 1. The dimensions of the problem are $m = 20894, \quad n = 23923$ and the number of nonzeroes is $nz = 187686$. The longest column has 44 nonzeros, which reflects a very low density (0.21%). It is noteworthy that our partitioning heuristic classifies this problem to be solved by the augmented system. Next, in Figure 2 we show the nonzero pattern of the factorization of the augmented system with the normal equations approach and with our heuristic of the previous section. On the left hand side of Figure 2 the result with the normal equations (using the minimum degree ordering) can be seen. The procedure generated 3281093 nonzeros in the lower triangular part of the factored form of (3). The right hand side of Figure 2 shows the nonzero pattern of the factorization of the augmented system using the heuristic of section 3. In this case the number of nonzeros is 444206.

Figure 1: Nonzero pattern of problem `route`

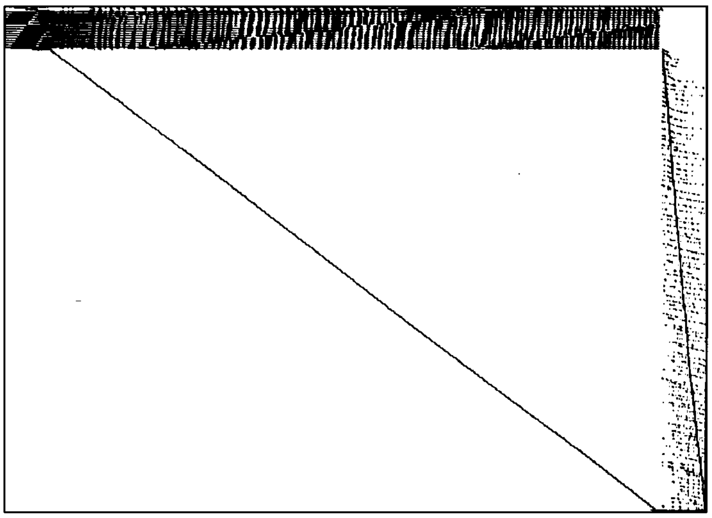
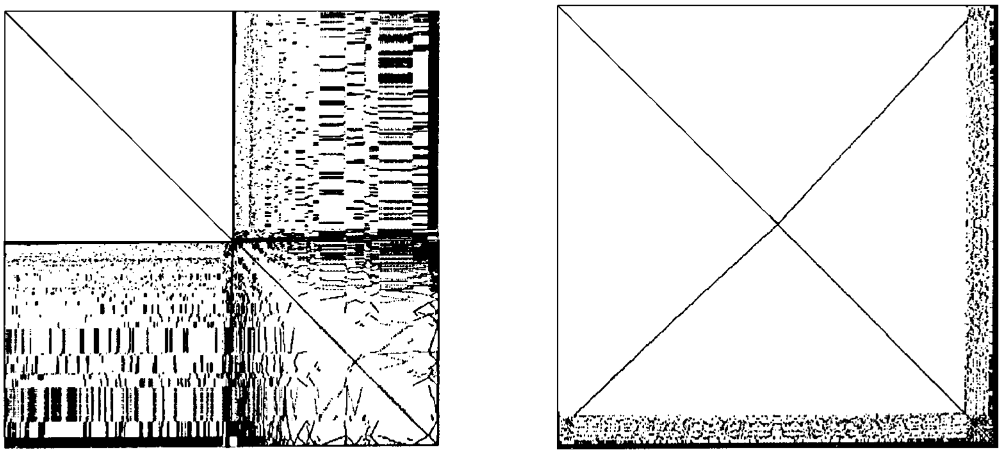


Figure 2: Nonzero pattern of the factorization via NE and AS pivot agenda



Vanderbei and Carpenter [25] used a version of the augmented system to test its general usefulness in their solver LOQO. To show the power of our pivot search heuristic, for comparison we selected those problems from the NETLIB set on which the factorization of the augmented system either in BPM or in LOQO was significantly better than the normal equation approach with respect to fill-in. In Table 2 we display the number of operations (in thousands) needed to compute the factorization of the augmented system. Again, the original figures of LOQO have been divided by two for reasons explained above. Note that in case of problems `agg2` and `agg3` the favorable fill-in figures were prevented by the unfavorable numerical properties. Namely, our default relative pivot tolerance was too large. When we reduced it we obtained much less fill-in. These figures are given in parenthesis in the table. It is to be noted that on no other problem have we experienced such a heavy dependence of fill-in on the pivot tolerance. However, the example of these two problems suggests that this should not be taken for granted.

5 Handling free variables

In this section we develop a treatment of free variables for the primal algorithm. This approach has some notational conveniences. At the same time, arguments used here can similarly be applied for all the log-barrier algorithms.

The simplest way of handling the free variables is the variable replacement method. Unfortunately, it has some disadvantages. One of them is that the optimal (and also the feasible) region is unbounded which may cause numerical problems. Although we also start with splitting a free variable into positive and negative parts we do not put the centrality term in the right hand side of (3) because it makes little if any sense to try to centralize both the positive and negative part of a free variable.

In the following description we assume that a partition of free variables F has been added to the original A matrix. For further notations, let x_F be the partition of the free

Table 2: Number of nonzeros (in thousands) generated in a factorization of the augmented system

Problem	LOQO	BPM	
agg2	435	709	(285)
agg3	428	709	(285)
bore3d	45	39	
fit1p	74	63	
fit2p	324	266	
ganges	877	307	
israel	52	52	
pilot4	539	266	
seba	34	27	
stair	538	188	
vtp_base	34	20	

variables, and the splitting $x_F = x_F^+ - x_F^-$ with $x_F^+, x_F^- \geq 0$. Then, the primal LP problem has the following form:

$$\text{minimize} \left[c^T, c_F^T, -c_F^T\right] \begin{bmatrix} x \\ x_F^+ \\ x_F^- \end{bmatrix},$$

$$[A \quad F \quad -F] \begin{bmatrix} x \\ x_F^+ \\ x_F^- \end{bmatrix} = b,$$

$$\begin{bmatrix} x \\ x_F^+ \\ x_F^- \end{bmatrix} \geq 0.$$

From these assumptions, (3) will look as follows:

$$\begin{bmatrix} D_x & 0 & 0 & A^T \\ 0 & D_{x_F^+} & 0 & F^T \\ 0 & 0 & D_{x_F^-} & -F^T \\ A & F & -F & 0 \end{bmatrix} \begin{bmatrix} dx \\ dx_F^+ \\ dx_F^- \\ dy \end{bmatrix} = \begin{bmatrix} \alpha_x \\ \alpha_F \\ -\alpha_F \\ \beta \end{bmatrix}.$$

Let M_1 denote the row transformations that add the rows of block F^T to the rows of block $-F^T$ in (5), and M_2 the column transformations, that eliminate block $-F$ in similar way. Then we have the equivalent system:

$$M_1 \begin{bmatrix} D_x & 0 & 0 & A^T \\ 0 & D_{x_F^+} & 0 & F^T \\ 0 & 0 & D_{x_F^-} & -F^T \\ A & F & -F & 0 \end{bmatrix} M_2 M_2^{-1} \begin{bmatrix} dx \\ dx_F^+ \\ dx_F^- \\ y \end{bmatrix} = M_1 \begin{bmatrix} \alpha_x \\ \alpha_F \\ -\alpha_F \\ \beta \end{bmatrix},$$

and we obtain after the transformations:

$$\begin{bmatrix} D_x & 0 & 0 & A^T \\ 0 & D_{x_F^+} & D_{x_F^+} & F^T \\ 0 & D_{x_F^+} & D_{x_F^-} + D_{x_F^+} & 0 \\ A & F & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ w_{x_F^+} \\ w_{x_F^-} \\ dy \end{bmatrix} = \begin{bmatrix} \alpha_x \\ \alpha_F \\ 0 \\ \beta \end{bmatrix}$$

where

$$\begin{bmatrix} w \\ w_{x_F^+} \\ w_{x_F^-} \\ dy \end{bmatrix} = M_2^{-1} \begin{bmatrix} dx \\ dx_F^+ \\ dx_F^- \\ dy \end{bmatrix}. \tag{5}$$

The solution of (5) is:

$$\begin{aligned} dx &= w\,, \\ dx_F^+ &= w_{x_F^+} + w_{x_F^-} \\ dx_F^- &= w_{x_F^-}. \end{aligned}$$

It is easy to see that the primal search directions (dx, dx_F^+, dx_F^-) and $(w, w_{x_F^+}, 0)$ are equivalent because $dx_F = dx_F^+ - dx_F^- = w_{x_F^+}$. We can pivot out the variables $w_{x_F^-}$, and obtain:

$$\begin{bmatrix} D_x & 0 & A^T \\ 0 & D_{x_F} & F^T \\ A & F & 0 \end{bmatrix} \begin{bmatrix} dx \\ dx_F \\ dy \end{bmatrix} = \begin{bmatrix} \alpha_x \\ \alpha_F \\ \beta \end{bmatrix}$$

thereby

$$D_{x_F} = \frac{D_{x_F^+} D_{x_F^-}}{D_{x_F^+} + D_{x_F^-}}.$$

It follows from the definition of D that if we tend with the splitting values to infinity, that is $x_{F_i}^+ \to \infty$ and $x_{F_i}^- \to \infty$, then $D_{x_{F_i}} \to 0$. This leads to the following system:

$$\begin{bmatrix} D_x & 0 & A^T \\ 0 & 0 & F^T \\ A & F & 0 \end{bmatrix} \begin{bmatrix} dx \\ dx_F \\ dy \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha_F \\ \beta \end{bmatrix},$$

which is the same as the result of Mehrotra and the projection matrices of Vanderbei. It is easy to see that the search directions obtained with this way are the same as the search direction after eliminating the free variables. Mehrotra reports that the numerical behavior of this technique is superior to that of the variable replacement method. Unfortunately, the zero diagonal values in D_{x_F} may cause more fill-in. To avoid this while allowing the same freedom for choosing pivots as with the variable replacement method, we use a constant $\epsilon > 0$ instead of 0 for the diagonal values of D_{x_F}. We experimented on some LP problems with several free variables. Table 3 shows these problems (number of columns, rows, nonzeros, and the number of the free variables for each problem).

Table 3: Problem characteristics

Problem	Number of			
name	rows	columns	nonzeroes	free vars
perold	625	1376	6018	88
pilot_ja	940	1988	14698	88
pilot_we	722	2789	9126	80
complex	1023	1408	46463	385
df2177	900	9728	31434	450
l9	244	1401	4577	186
l30	2701	15380	51169	1880

Consider the degree of fill-in. Table 4 compares alternatives $\epsilon = 0$ and $\epsilon > 0$ in the diagonal of D_{x_F} with respect to the number of nonzeroes in the factorization of (3), the number of the operations for one factorization of (3), and the "set up time", that is the time to build the nonzero structure of the factorization, including one numerical factorization.

Table 4: Fill-in with different handling of free variables

Problem	Nonzeroes		Operations		Set up time (sec)	
name	$\epsilon > 0$	$\epsilon = 0$	$\epsilon > 0$	$\epsilon = 0$	$\epsilon > 0$	$\epsilon = 0$
perold	30425	31450	874658	952213	0.66	2.04
pilot_ja	67041	71352	3878928	4276072	2.09	7.47
pilot_we	24903	25722	392623	405083	0.52	1.20
complex	120418	120418	11570403	11570403	15.26	15.26
df2177	172361	309854	25713237	63061428	8.43	108.89
l9	11178	9341	125531	65421	0.34	0.33
l30	152974	261276	3020826	12475021	5.22	12.81

Table 5 shows the number of iterations needed to reach an optimal solution.

Table 5: Influence of ϵ to the iteration numbers

Problem name	Number of iterations $\epsilon = 10^{-1}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-7}$	$\epsilon = 0$
perold	LIMIT	28	28	26
pilot_ja	LIMIT	26	24	27
pilot_we	LIMIT	LIMIT	33	32
complex	10	9	9	9
df2177	6	4	4	4
l9	14	13	13	12
l30	LIMIT	29	21	22

Clearly, increasing ϵ may result in less efficient search directions. We compare the numbers of necessary IPM iterations for the values $\epsilon = 10^{-1}$, $\epsilon = 10^{-4}$, $\epsilon = 10^{-7}$ and $\epsilon = 0$. The results in Table 5 show the influence of ϵ to the efficiency on the computed search direction. In Table 5 the case when the iteration limit of 99 was exceeded is denoted by "LIMIT". With our default choice of $\epsilon = 10^{-7}$ we could reduce the number of fill-ins significantly in all cases without noticeable loss of efficiency in the general case.

The differences in "set up time" can be caused not only by the different number of operations (and fill-in), but also by the different ways the set up is performed. Namely, for $\epsilon > 0$ we can use the symbolic factorization, but for $\epsilon = 0$ a numerical factorization of the augmented system is necessary.

In case of problem `complex` the factorization of the augmented system was used for both approaches because of the structure of the matrix.

The most striking difference can be observed on problem `df2177`, which is a dualized version of a primal problem, where the primal had 450 equality constraints. In the case of this problem we want to point out a common difficulty with using the dual of an originally primal problem with a larger number of equalities. In the primal model of `df2177` some equality constraints are linearly dependent, which causes linear dependency among the free variables of the dual. The "result" of the presence of linearly dependent free variables is dual infeasibility (primal unboundedness) unless the dependency is true for the vectors augmented by the objective coefficients. When considering the variable replacement method, this dependency in the objective function may become false due to numerical errors in scaling the problem. Using $\epsilon = 0$ the dependencies among the free variables can be detected during the factorization of the augmented system and we can either detect dual infeasibility or reduce the set of free variables to its maximal linearly independent set. As shown in Table 4, $\epsilon = 0$ requires a lot of computational effort and causes much fill-in. When using $\epsilon > 0$ the linearly dependency is not detectable, but numerical errors do not appear due to the fixed scaling of the free variables.

As a conclusion, we can say that $\epsilon = 10^{-7}$ is a good choice for the default scaling of the free variables. In this case the efficiency of the computed steps is very near to the theoretically best $\epsilon = 0$ case and, additionally, the performance in the number of fill-in and in the set up time is much better in the majority of cases. At the same time, it seems to be a good idea to maintain the possibility of working with $\epsilon = 0$.

6 Computational testing

For computational testing, we have selected the NETLIB/LP problems and some additional nontrivial real life problems. Characteristics of the NETLIB/LP problems are widely known (c.f., [6]), therefore, in Table 6 we include only the characteristics of the additionally investigated problems. Note that we give the number of nonzeros of the longest column in each problem since this number can strongly effect the normal equation method.

The purpose of our computational testing is twofold. First, we compare the performance of the two algorithms (normal equations and augmented system) of our solver BPM. Second, we compare the performance of BPM with that of CPLEX 4.0. The purpose of the second testing is to see the difference in coping with fill-in between an advanced implementation of the Schur complement (CPLEX) and the augmented system with our heuristic. The test runs were performed in June 1996 on a 90 MHz Pentium based PC with 32 MB of RAM.

The solution times and also times per iteration are influenced by many factors. Perhaps the most important factor is the actual size of the problem that remains after the initial problem reduction (presolve). First we tested the performance of the presolve procedures of CPLEX and BPM and we found that they often result in very different problem sizes. To enable proper comparison, we decided to disable presolve for both solvers. Similarly, to perform comparison on equal basis, we used CPLEX with the minimum local fill-in ordering, which is the ordering scheme of BPM.

In some cases the available memory was not enough for one or the other solver. This situation is indicated in the tables by ROOM (Run Out Of Memory).

Table 6: Problem characteristics

Problem	Number of			Longest
name	rows	columns	nonzeroes	column
aircraft	3754	7517	20267	751
car2	400	1200	38890	109
car4	16384	33052	63724	109
cep1	1521	3248	6712	219
complex	1023	1408	46463	512
cr42	905	1513	6614	302
cre_a	3516	4067	14987	14
				ctd. →

Problem characteristics, ctd.				
Problem	Number of			Longest
name	rows	columns	nonzeroes	column
cre_c	3068	3678	13244	13
df2177	900	9728	21706	8
pgp2	4034	9220	18440	578
pf2177	9728	900	21706	37
route	20894	23923	187686	44
sto8	4409	10193	27424	21
sto27	14441	34114	90903	40
tad12064	9564	14514	53799	132
tad12096	14300	21714	80359	196
zed	117	43	594	49

In Table 7 we give comparison of some important characteristics of the performance of the normal equation and the augmented system method of BPM. They are the number of nonzeros (NZ), the number of operations (OP) in the factorization, and time per factorization (TM) with the given factored form. To assist viewing, we use the following typographic notations. If two corresponding values in the NZ or TM column differ less than 10% (15% for OP) then none of them is marked. If the difference is between 10% and 20% (15% and 25%) then the better is put into an oval box. If the difference is bigger than 20% (25%) then the better value is put into a rectangular box. The number of nonzeros and the number of operations refer to the number of nonzeros/operations during the factorization of the matrix of the augmented system in (3). As such, in case of the normal equations (i) the number of nonzeros includes the nonzeros of the lower triangular part of the factorized $AD^{-1}A^T$ plus the nonzeros of A, (ii) the number of operations includes the number of operations used to compute $AD^{-1}A^T$ and its factorization. By operation we mean a pair consisting of a multiplication and addition.

Table 7: Performance of BPM's normal equation and augmented system

Problem	Nonzeros in		Operations in		Time(sec)/fac	
name	norm	augm	norm	augm	norm	augm
25fv47	44005	44806	1309567	1376145	$\boxed{0.2550}$	0.2817
80bau3b	61308	60305	1260719	1148595	0.3370	0.3573
adlittle	738	738	2752	2752	0.0000	0.0000
afiro	163	163	338	338	0.0000	0.0000
agg2	25269	$\boxed{20118}$	$\boxed{586699}$	708856	$\boxed{0.1095}$	0.1358
agg3	25285	$\boxed{20128}$	$\boxed{587233}$	709121	$\boxed{0.1095}$	0.1279
bandm	6859	6852	57050	56861	$\boxed{0.0000}$	0.0147
beaconfd	6103	6102	69952	69921	0.0300	$\boxed{0.0200}$
						ctd. →

Performance of the normal and augmented system, ctd.						
Problem	Nonzeros in		Operations in		Time(sec)/fac	
name	norm	augm	norm	augm	norm	augm
blend	1423	1431	9962	10154	0.0109	$\boxed{0.0060}$
bnl1	16937	17417	252621	276781	0.0758	0.0754
bnl2	96184	100198	$\boxed{6376912}$	7263861	$\boxed{1.1029}$	1.3291
boeing1	10498	10671	111699	116621	$\boxed{0.0319}$	0.0420
boeing2	3788	3758	36869	36267	$\boxed{0.0079}$	0.0113
bore3d	4312	4170	42253	38770	0.0169	$\boxed{0.0138}$
brandy	5350	5399	65060	66749	$\boxed{0.0113}$	0.0193
capri	7103	7391	$\boxed{96272}$	111925	$\boxed{0.0194}$	0.0265
cycle	95131	89796	2977821	$\boxed{2420611}$	0.5819	$\boxed{0.5287}$
czprob	16987	16987	62271	62271	$\boxed{0.0431}$	0.0573
d2q06c	198093	$\boxed{180065}$	17655900	$\boxed{13622255}$	3.2537	$\boxed{2.5789}$
degen2	19853	19771	482892	486022	$\boxed{0.0931}$	0.1100
degen3	144049	143489	7958190	7867373	1.6912	1.7557
dfl001	1669889	1632660	703846942	668867236	177.9323	169.6777
e226	5976	5967	56250	56361	$\boxed{0.0138}$	0.0547
etamacro	13163	13926	$\boxed{306348}$	360907	$\boxed{0.0595}$	0.0690
fffff800	24414	24215	513690	504749	0.1210	0.1250
finnis	7352	7371	56475	56847	$\boxed{0.0210}$	0.0238
fit1d	13676	13676	98018	98018	$\boxed{0.0314}$	0.0412
fit1p	206119	$\boxed{10142}$	42919782	$\boxed{62748}$	7.2781	$\boxed{0.0494}$
fit2d	129317	129317	880960	880960	0.4450	0.4676
fit2p	-	$\boxed{50583}$	-	$\boxed{266015}$	-	$\boxed{0.2865}$
ganges	36368	$\boxed{24058}$	799817	$\boxed{307111}$	0.1353	$\boxed{0.0913}$
gfrd_pnc	3925	3913	7003	6934	$\boxed{0.0086}$	0.0223
greenbea	112629	114437	2922943	3140816	$\boxed{0.6559}$	0.7647
greenbeb	111179	111179	2751441	2751441	0.6090	0.5971
grow15	11410	11410	105750	105750	0.0413	$\boxed{0.0267}$
grow22	16870	16842	157102	156458	0.0550	0.0556
grow7	5202	5202	47798	47798	0.0121	$\boxed{0.0043}$
israel	13583	$\boxed{4231}$	552820	$\boxed{52279}$	0.0889	$\boxed{0.0111}$
lotfi	2828	2799	18168	17732	0.0121	$\boxed{0.0071}$
maros	34415	35593	$\boxed{706316}$	784677	$\boxed{0.1445}$	0.1809
maros_r7	1339955	1319196	252576266	244861308	46.8293	45.5120
modszk1	15425	$\boxed{6900}$	170435	$\boxed{42071}$	0.0570	$\boxed{0.0211}$
nesm	34212	33502	625994	577319	$\boxed{0.1385}$	0.1581
perold	30425	32314	$\boxed{874658}$	1042317	$\boxed{0.1646}$	0.1956
pilot	234229	249281	$\boxed{24461562}$	28508040	$\boxed{4.5944}$	5.6105
						ctd. →

Performance of the normal and augmented system, ctd.						
Problem name	Nonzeros in norm	augm	Operations in norm	augm	Time(sec)/fac norm	augm
pilot4	19198	14426	495091	265835	0.1003	0.0645
pilot87	491876	537254	88420073	102162584	18.4223	22.3425
pilotnov	64592	63953	3214098	3029208	0.5359	0.5753
pilot_ja	67041	63213	3678928	3040508	0.6142	0.5742
pilot_we	24903	24958	392623	402563	0.0988	0.1112
recipe	1292	1290	8421	8418	0.0000	0.0000
sc205	1504	1471	4025	3815	0.0056	0.0189
scagr25	4064	4062	13655	13641	0.0000	0.0279
scagr7	1057	1056	3387	3381	0.0000	0.0042
scfxm1	7002	6989	59857	59507	0.0211	0.0211
scfxm2	14145	14174	121570	122493	0.0350	0.0517
scfxm3	21297	21396	183514	186523	0.0644	0.0595
scorpion	3609	3112	14594	11361	0.0142	0.0083
scrs8	8833	9362	76318	98942	0.0230	0.0233
scsd1	3703	3703	19029	19029	0.0000	0.0275
scsd6	6714	6714	32177	32177	0.0000	0.0170
scsd8	14066	14066	61379	61379	0.0525	0.0325
sctap1	3949	4009	17926	19101	0.0077	0.0050
sctap2	20575	20550	303771	298576	0.0573	0.0825
sctap3	26006	26152	306679	313738	0.0900	0.1015
seba	58100	5486	5588880	27070	0.9164	0.0214
share1b	2488	2488	14725	14725	0.0031	0.0129
share2b	1639	1615	9221	9067	0.0050	0.0000
shell	6601	6811	32839	39483	0.0195	0.0245
ship04l	10760	10760	44410	44410	0.0300	0.0331
ship04s	7604	7604	31798	31798	0.0175	0.0269
ship08l	21730	21742	90303	90463	0.0587	0.0629
ship08s	12522	12582	51583	52575	0.0369	0.0275
ship12l	27307	27347	111452	111852	0.0847	0.0943
ship12s	14643	14635	60124	60108	0.0514	0.0479
sierra	18922	17544	159897	124863	0.0519	0.0588
stair	17992	11698	461479	188040	0.0931	0.0508
standata	5696	5719	26136	26830	0.0129	0.0094
stocfor1	1282	1269	5429	5304	0.0056	0.0000
stocfor2	35199	35434	282987	296943	0.1100	0.1274
stocfor3	271606	230733	2201407	1922324	0.9746	1.1032
truss	80345	91883	1729700	2748653	0.3787	0.6394
						ctd. →

Performance of the normal and augmented system, ctd.						
Problem	Nonzeros in		Operations in		Time(sec)/fac	
name	norm	augm	norm	augm	norm	augm
tuff	12807	12800	202213	202121	$\boxed{0.0365}$	0.0483
vtp_base	2922	3012	18062	19842	0.0142	$\boxed{0.0092}$
wood1p	88297	88296	1779322	1779320	0.4988	0.5285
woodw	85131	84966	1716500	1706564	$\boxed{0.4110}$	0.4657
aircraft	-	$\boxed{20317}$	-	$\boxed{36904}$	-	$\boxed{0.0792}$
car2	105052	99901	8711130	$\boxed{7473173}$	1.7208	1.8246
car4	218638	$\boxed{100424}$	22560003	$\boxed{4185620}$	4.4500	$\boxed{1.1618}$
cep1	268408	$\boxed{11902}$	47675528	$\boxed{23576}$	10.8035	$\boxed{0.0287}$
complex	562025	$\boxed{120418}$	176825613	$\boxed{11570403}$	34.5120	$\boxed{2.1800}$
cr42	294667	$\boxed{11817}$	56115569	$\boxed{56682}$	14.3567	$\boxed{0.0367}$
cre_a	47835	46817	375667	351307	0.1764	0.1892
cre_c	44423	43858	359097	342803	0.1630	0.1739
df2177	172361	172780	25713237	25766698	4.5900	4.8480
pgp2	1854441	$\boxed{33998}$	896801677	$\boxed{72606}$	266.0609	$\boxed{0.1178}$
pf2177	-	$\boxed{305486}$	-	$\boxed{60692994}$	-	$\boxed{15.4375}$
route	-	$\boxed{444206}$	-	$\boxed{7480407}$	-	$\boxed{2.8900}$
zed	4477	$\boxed{790}$	102259	$\boxed{3302}$	0.0178	$\boxed{0.0060}$

To get a first message from Table 7 we prepared a much smaller summary table (Table 8). It contains counts of how many times one or the other method was better. More precisely, we set up the following categories: not significantly different (corresponds to no boxing), little better (oval boxes), much better (part of the rectangular boxes), and qualitatively better. This latter contains the other part of rectangular boxes and refers to the situations in which a problem could be solved with one method and not with the other or one method was better by at least an order of magnitude.

Table 8: Number of cases one system is better than the other

	Nonzeros in		Operations in		Time(sec)/fac	
	norm	augm	norm	augm	norm	augm
Nearly equal	76		64		24	
Little better	1	3	7	3	16	1
Much better	0	10	5	11	26	23
Qualitatively better	0	9	0	9	0	9

Table 9 shows the comparison of the performance of CPLEX and BPM. Figures given include nonzeros in the factor, number of iterations to optimality, solution time, and the choice of the algorithm selection heuristics of BPM. Therefore, the figures given for BPM are those obtained with the selected algorithm. Since the number of nonzeros in CPLEX is $|L|$ (where L is defined by $LL^T = AD^{-1}A^T$), we give the number of nonzeros for BPM in such a way that the number of nonzeros in Table 7 is adjusted by subtracting $|A|$. In this way we compare the number of nonzeros generated in addition to A.

Table 9: Comparison of CPLEX and BPM

Problem	Nonzeros in L		Iterations		Sol. time(sec)		BPM
Name	CPLEX	BPM	CPLEX	BPM	CPLEX	BPM	choice
25fv47	42017	38648	22	18	8.78	$\boxed{7.85}$	NE
80bau3b	62734	57450	29	39	$\boxed{24.55}$	36.03	NE
agg	19179	$\boxed{6134}$	27	17	3.62	$\boxed{1.31}$	NE
agg2	25780	$\boxed{20118}$	18	17	4.23	4.67	AS
agg3	25796	$\boxed{20128}$	18	18	$\boxed{4.07}$	4.95	AS
bandm	7636	$\boxed{6803}$	16	15	0.99	1.04	NE
beaconfd	6631	6102	12	10	0.77	0.77	NE
bnl1	18016	$\boxed{15895}$	55	24	7.47	$\boxed{3.95}$	NE
bnl2	93330	87019	29	26	48.88	$\boxed{37.41}$	NE
boeing1	11695	$\boxed{10176}$	24	18	2.31	$\boxed{1.87}$	NE
boeing2	4498	$\boxed{3759}$	17	14	0.66	0.71	NE
bore3d	4721	$\boxed{4194}$	17	14	0.77	0.77	AS
brandy	5879	5352	18	15	0.94	0.94	NE
capri	8096	$\boxed{6886}$	19	16	1.49	$\boxed{1.26}$	NE
cycle	79780	73376	28	20	19.94	$\boxed{14.00}$	NE
czprob	17663	16986	33	23	5.98	5.44	NE
d2q06c	130039	128816	30	22	47.40	44.22	NE
d6cube	91296	89646	17	16	37.89	$\boxed{27.84}$	NE
degen2	21665	$\boxed{19172}$	13	9	3.30	$\boxed{2.52}$	NE
degen3	156941	$\boxed{140496}$	18	13	67.83	$\boxed{35.60}$	NE
dfl001	1504318	1405086	44	37	7532.93	$\boxed{4239.92}$	NE
e226	6629	$\boxed{5972}$	18	16	$\boxed{0.93}$	1.04	NE
etamacro	13937	$\boxed{12494}$	24	19	3.18	2.91	NE
fffff800	27190	$\boxed{23917}$	29	24	6.81	6.64	NE
finnis	8955	$\boxed{7429}$	22	20	1.92	1.82	NE
fit1d	13662	13676	16	18	$\boxed{1.98}$	2.37	NE
fit1p	9868	10142	15	14	2.58	2.59	AS
fit2d	128864	129317	18	20	$\boxed{21.75}$	26.91	NE
fit2p	50284	50583	19	18	21.92	23.24	AS
ganges	27308	$\boxed{23870}$	16	16	$\boxed{3.95}$	4.67	AS
gfrd-pnc	4609	$\boxed{3910}$	20	12	1.43	$\boxed{1.09}$	NE
greenbea	112249	$\boxed{99947}$	51	28	53.88	$\boxed{28.28}$	NE
greenbeb	112158	$\boxed{99472}$	35	38	36.69	36.97	NE
							ctd. →

Comparison of CPLEX and BPM, ctd.							
Problem Name	Nonzeros in L CPLEX	BPM	Iterations CPLEX	BPM	Sol. time(sec) CPLEX	BPM	BPM choice
grow15	15370	11410	12	12	1.42	1.59	NE
grow22	22932	16842	12	13	2.20	2.37	NE
grow7	6942	5202	10	11	0.61	0.83	NE
israel	13950	4231	20	17	3.52	0.82	AS
kb2	790	746	18	10	0.22	0.27	NE
lotfi	3137	2783	16	13	0.55	0.61	NE
maros-r7	2008872	1298272	10	12	3008.93	577.82	NE
maros	33420	30143	29	19	7.75	5.32	NE
modszk1	17289	6900	$*^{(3)}$ (23)	21	$*^{(3)}$ (3.68)	2.97	AS
nesm	36269	32822	37	26	15.11	10.43	NE
perold	31309	29137	35	24	10.38	9.22	NE
pilot	226987	211750	29	29	167.36	140.28	AS
pilot4	18375	14426	35	32	6.20	5.05	NE
pilot87	496945	457743	34	31	841.18	546.94	NE
pilotnov	62779	63330	21	12	20.43	12.20	NE
pilot-ja	64354	58972	29	20	28.56	18.95	NE
pilot-we	25480	23436	37	27	8.46	7.25	NE
sc105	836	710	10	8	0.22	0.28	NE
sc205	1727	1435	11	9	0.33	0.44	NE
sc50a	361	312	9	7	0.16	0.28	NE
sc50b	362	297	8	6	0.11	0.22	NE
scagr25	4987	4027	16	14	0.83	0.87	NE
scagr7	1279	1043	13	14	0.27	0.44	NE
scfxm1	7362	6528	17	15	1.04	1.04	NE
scfxm2	14766	13138	20	17	2.26	2.20	NE
scfxm3	22170	19734	20	17	3.79	3.24	NE
scorpion	3819	3471	12	10	0.50	0.66	NE
scrs8	9899	8694	20	17	1.82	1.92	NE
scsd6	7007	6722	11	9	0.71	0.93	NE
scsd8	14230	14066	9	9	1.27	1.70	NE
sctap1	4532	3960	14	12	0.60	0.72	NE
sctap2	21220	18774	13	13	2.97	2.91	NE
sctap3	28470	25096	13	12	4.23	3.68	NE
seba	4813	5486	18	14	1.54	1.59	AS
share1b	2414	2241	20	16	0.44	0.54	NE
shell	7308	6407	20	18	1.98	2.09	NE
ship04l	11468	10720	14	11	1.60	1.76	NE
							ctd. →

Comparison of CPLEX and BPM, ctd.							
Problem Name	Nonzeros in L CPLEX	BPM	Iterations CPLEX	BPM	Sol. time(sec) CPLEX	BPM	BPM choice
ship04s	8224	7428	14	11	1.10	1.31	NE
ship08l	23158	21690	16	12	3.84	3.74	NE
ship08s	13790	12226	14	11	1.98	2.03	NE
ship12l	29523	27307	18	17	5.27	6.31	NE
ship12s	15267	13987	15	13	2.47	2.75	NE
sierra	20989	18887	15	16	3.85	4.01	NE
stair	16645	11695	15	11	2.31	1.42	AS
standata	5946	5694	12	17	0.94	1.54	NE
standgub	6002	5802	12	18	0.94	1.65	NE
standmps	8804	7853	17	22	1.59	2.19	NE
stocfor1	1437	1227	13	10	0.22	0.33	NE
stocfor2	35507	30244	22	16	6.48	5.17	NE
stocfor3	281717	229540	32	29	89.64	91.84	AS
truss	90423	79429	17	13	17.20	15.55	NE
tuff	13530	12367	21	16	2.37	1.81	NE
vtp-base	3283	2897	36	15	1.05	0.65	AS
wood1p	90853	88297	18	18	18.90	16.97	NE
woodw	87669	81994	23	23	20.98	22.41	NE
aircraft	20267	20317	$*^{(4)}$ (22)	9	$*^{(4)}$ (18.07)	5.60	AS
car2	118690	99901	14	10	58.11	31.03	NE
car4	63724	100424	$*^{(4)}$ (17)	9	$*^{(4)}$ (385.47)	46.80	AS
cep1	11248	11902	$*^{(4)}$ (17)	15	$*^{(4)}$ (4.17)	4.12	AS
complex	525956	120418	$*^{(4)}$ (8)	8	$*^{(4)}$ (579.08)	29.94	AS
cr42	7214	11817	27	12	4.67	1.98	AS
cre-a	52996	46761	30	21	17.63	14.17	NE
cre-c	50402	43300	30	24	17.47	14.22	NE
df2177	176332	169506	$*^{(3)}$ (13)	2	$*^{(3)}$ (159.72)	23.40	NE
pgp2	30536	33998	$*^{(4)}$ 23	25	$*^{(4)}$ (15.49)	18.79	AS
pf2177	11977	306470	$*^{(1)}$	2	$*^{(1)}$	98.43	AS
route	$*^{(2)}$	444346	$*^{(2)}$	19	$*^{(2)}$	190.43	AS
sto8	147174	114897	28	33	54.21	63.54	AS
sto27	242805	322273	38	52	507.40	254.03	AS
tad12064	138799	166612	$*^{(4)}$ (49)	45	$*^{(4)}$ (252.21)	110.12	AS
tad12096	207567	248212	$*^{(4)}$ (36)	26	$*^{(4)}$ (276.50)	93.43	AS
zed	4911	790	16	8	0.83	0.28	AS
Total	8671018	7992515	2180	1847	14667.21	7165.34	

Remarks to Table 9:

$*^{(1)}$ CPLEX was able to make the symbolic factorization but ran out of memory when it started iterations

$*^{(2)}$ No data because CPLEX ran out of memory during the symbolic phase

$*^{(3)}$ CPLEX declared the problem infeasible while it has optimal solution

$*^{(4)}$ CPLEX was unable to solve due to numerical troubles

7 Conclusion

In the paper we have pointed out some favorable features of the augmented system approach to the interior point methods. We demonstrated how free variables can be handled efficiently. We introduced a pivot selection method that is generally competitive with the normal equation method (iteration efficiency is not significantly different) and is qualitatively better in some critical cases (many free variables, serious fill-in in $AD^{-1}A^T$).

We concluded that a good implementation should include both approaches. A method to make a choice between the normal equation and the augmented system has also been shown. This is a heuristic method and is based on the structural analysis of the problem to be solved.

The results of the detailed computational testing demonstrate that the augmented system can indeed be the requested spare engine of interior point methods for cases when the normal equation approach is not satisfactory or cannot work at all.

8 Acknowledgement

The authors gratefully acknowledge the attention and support of Gautam Mitra at Brunel University during the preparation of this paper. The authors also express their indebtedness to the two anonymous referee for their helpful comments and remarks.

References

[1] Arioli, M., Duff, I.S. and Rijk, P.P.M., "On the Augmented System Approach to Sparse Least-Squares Problems" *Numerische Mathematik* 55, 1989, p. 667–684.

[2] Birge, J.R., Freund, R.M. and Vanderbei, R.J., "Prior reduced fill-in solving equations in interior point algorithms" *Operations Research Letters*, Vol. 11, 1992, p. 195–198.

[3] Bixby, R.E., "Progress in Linear Programming", *ORSA Journal on Computing*, Vol. 6, No. 1, Winter 1994, p. 15–22.

[4] Duff, I., Gould, J., Reid, J., Scott, Turner, "The factorization of sparse symmetric indefinite matrices", *IMA Journal of Numerical Analysis*, 11, 1991, p. 181–204.

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

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

[7] Gondzio, J., "Splitting dense columns of constraint matrix in interior point methods for large scale linear programming", *Optimization*, Vol. 24, 1992, p. 285–297.

[8] Gonzaga, C.G., "Path following methods for linear programming" *SIAM Review* , Vol. 34, (2) 1992, p. 167–227

[9] Hafsteinsson, H., Levkovitz R., Mitra, G., "Solving Large Scale Linear Programming Problems Using an Interior Point Method on a Massively Parallel SIMD Computer", *Journal of Parallel Algorithms and Applications*, Vol. 4, No. 3&4, p. 301–316.

[10] Hurd, J.K. and Murphy, F.H., "Exploiting Special Structure in Primal Dual Interior Point Methods", *ORSA Journal on Computing*, Vol. 4, No. 1, 1992, p. 38–44.

[11] Lustig, I.J., Marsten, R.E. and Shanno, D.F. (1991) "Computational Experience with a Primal–Dual Interior Point Method" for Linear Programming, *Linear Algebra and its Applications* 152, 191-222.

[12] Lustig I.J., Marsten, R.E. and Shanno, D.F. (1992) "On Implementing Mehrotra's Predictor–Corrector Interior Point Method for Linear Programming", *SIAM Journal on Optimization* 2, 435-449.

[13] Lustig, I.J., Marsten, R.E. and Shanno, D.F., "Interior Point Methods for Linear Programming: Computational State of the Art", *ORSA Journal on Computing*, Vol. 6, No. 1, Winter 1994, p. 1–14.

[14] Maros, I., Mitra, G., "Simplex algorithms", in Beasley, J., (ed.) Recent Advances in *Linear and Integer Programming*, Oxford University Press, 1996, p. 1–46.

[15] Marsten, R.E., Saltzman, M.J., Shanno, D.F., Ballantijn, J.F. and Pierce, G.S., "Implementation of a Dual Affine Interior Point Algorithm for Linear Programming", *ORSA Journal on Computing*, Vol. 1(1989), p. 287–297.

[16] Mehrotra, S., "Handling Free Variables in Interior Methods", *Technical Report 91-06R1*, March 1991.

[17] Mészáros, Cs., "Fast Cholesky factorization for interior point methods of linear programming", *LORDS WP 94-5*, MTA SZTAKI, Budapest, 1994.

[18] Mitra, G., Levkovitz, R., "Solution of Large Linear Programs: A Review of Hardware, Software and Algorithmic Issues", p. 139–171 in Ciriani, T., Leachman, R., (eds.), *Optimization in Industrial Environments*, John Wiley, 1993.

[19] Saunders, M.A., "Major Cholesky Would Feel Proud", *ORSA Journal on Computing*, Vol. 6, No. 1, Winter 1994, p. 23–27.

[20] Stewart, G.W., "Modifying pivot elements in Gaussian elimination", *Math. Comp.*, 28, 1974, p. 537–542.

[21] Todd, M.J., "Exploiting special structure in Karmarkar's linear programming algorithm", *Mathematical Programming*, 41, 1988, p. 81–103.

[22] Turner, "Computing projections for the Karmarkar algorithm", *Linear Algebra and its Applications*, 152, 1991, p. 141–154.

[23] Vanderbei, R.J., "ALPO: Another Linear Program Optimizer", *ORSA Journal on Computing*, Vol. 3, No. 2, 1993, p. 134–146.

[24] Vanderbei, R.J., Meketon, M.S., Freedman, B.F., "A modification of Karmarkar's linear programming algorithm", *Algorithmica* 1 (1986), p. 395–407.

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

[26] Vanderbei, R.J., "Interior-Point Methods: Algorithms and Formulations", *ORSA Journal on Computing*, Vol. 6, No. 1, Winter 1994, p. 32–34.

[27] Vanderbei, R.J., "Affine Scaling for Linear Programs with Free Variables", *Mathematical Programming*, 43, 1989, p. 31–44.