REVUE FRANÇAISE D'AUTOMATIQUE, D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

J. GONDZIO

D. TACHAT

The design and application of IPMLO. A FORTRAN library for linear optimization with interior point methods

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 28, n° 1 (1994), p. 37-56.

<http://www.numdam.org/item?id=RO_1994__28_1_37_0>

Recherche opérationnelle/Operations Research
(vol. 28, n° 1, 1994, p. 37 à 56)

THE DESIGN AND APPLICATION OF IPMLO A FORTRAN LIBRARY FOR LINEAR OPTIMIZATION WITH INTERIOR POINT METHODS (*) (1)

by J. GONDZIO (2) and D. TACHAT (3)

Communicated by Pierre TOLLA

Abstract. – *The design principles of the IPMLO, a modularly structured library of FORTRAN subroutines for large scale Linear Optimization with Interior Point Methods are addressed. The objective of the library is to provide the base for the development and experiments with the new attractive approaches that apply interior point methods for solving linear programming problems. An example application of it for the implementation of the primal-dual logarithmic barrier interior point method of McShane* et al. *(1989) is described. The preliminary computational results of the code's application to the solution of medium scale LP test problems from* Netlib *collection are given and the comparison with the implementation of the simplex method is made.*

Keywords: Linear programming, interior point methods, program library.

Résumé. – *Dans cet article, on présente les principes de conception de IPMLO, une bibliothèque structurée de façon modulaire, de procédures FORTRAN pour l'Optimisation Linéaire de problèmes de grande taille à l'aide de Méthodes de Point Intérieur.*

L'objectif de cette bibliothèque est de fournir une base afin de développer et expérimenter de nouvelles approches intéressantes qui utilisent des méthodes intérieures pour résoudre des programmes linéaires.

On décrit un exemple d'application de cette bibliothèque pour implémenter la méthode primale-duale avec barrière logarithmique de McShane et al. *(1989).*

Enfin, on présente les premiers résultats expérimentaux de l'application de ce code à la résolution des problèmes-test de programmes linéaires de taille moyenne de la collection Netlib *et on le compare avec l'état de l'art en matière d'implémentation de la méthode du simplexe.*

Mots clés : Programmation linéaire, méthode intérieure, bibliothèque de programmes.

(*) Received April 1992.

(1) A preliminary version of the paper has been presented at the 14th International Symposium on Mathematical Programming ISMP'91 in Amsterdam, August 5-9, 1991 and at the 15th I.F.I.P. Conference on Systems Modelling and Optimization in Zurich, September 2-6, 1991.

(2) Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland.

The results discussed in the paper have been obtained when this author was staying at LAMSADE, University of Paris-Dauphine, place du Maréchal-de-Lattre-de-Tassigny, 75775 Paris Cedex 16, France.

(3) LAMSADE, University of Paris-Dauphine, place du Maréchal-de-Lattre-de-Tassigny, 75775 Paris Cedex 16, France.

Recherche opérationnelle/Operations Research, 0399-0559/94/01/$ 4.00

1. INTRODUCTION

Karmarkar's (1984) publication of the polynomial-time linear programming algorithm initiated a flood of research papers in which the application of different interior point methods to the solution of linear optimization problems was addressed. In parallel, much effort has been made so as to develop implementations of the interior point methods that could outperform the state-of-the-art simplex codes. It seems that, when large scale linear problems of over 1,000 constraints are solved, the interior point methods are faster than the simplex-type ones (*see e. g.*: Adler *et al.*, 1989 *b*, Cheng *et al.*, 1989, Marsten *et al.*, 1990 and Monma and Morton, 1987). Consequently, further search for its new attractive and more efficient variants seems well justified.

A need then arises for an experimental modularly structured library of subroutines that could facilitate this research. In this paper the design principles and example applications of such a library are addressed.

The IPMLO is a set of FORTRAN subroutines that can be applied to solve large scale *L*inear *O*ptimization problems with *I*nterior *P*oint *M*ethods. It has been designed to use as much as possible the ideas of structured programming. The interior point algorithm has been modularized and the closely related or even identical steps of its different variants (*e. g.*: input/output management, preprocessing, handling the projections or problem-oriented BLAS-basic linear algebra system) have been identified. They have later been implemented resulting in a software that accurately reflects the overall structure of the basic algorithm. In particular, the routines that incorporate the logic of the interior point method can almost never access directly two fundamental data structures: one for the original problem data and one for computing Karmarkar projections. The later are in our code handled by a direct approach *i. e.* the Cholesky factorization of Gondzio (1991).

Although in the library design a good structure rather than an efficiency of code has been emphasized, the program is competitive. Additionally, due to the careful exploiting of the sparsity of the linear program both in the management of original problem data and in the computations of orthogonal projections, it was possible to solve with it even medium scale problems (of up to 500 constraints and 1,000 variables) on a microcomputer with operational memory limited to 640 kB.

The most computationally attractive variant of the interior point algorithm, *i. e.* the primal-dual logarithmic barrier one of McShane *et al.* (1989) has already been developed on the basis of the IPMLO. Issues of its implementation are discussed in detail to illustrate possible applications of the library.

The experience gained so far indicates that any new variant of the interior point method can be quickly incorporated into IPMLO library and tested when applied to solve medium scale problems, which should show whether it is attractive for further study or not. The library seems thus to be a useful tool that may facilitate algorithmic research when an application of interior point methods to linear optimization is concerned.

The paper is organized as follows. In Section 2 general issues of the IPMLO design are addressed. It contains: description of the library structure, management of the linear programming (LP) problem data, computing Karmarkar projections, implementation of the problem-oriented basic linear algebra routines and the implicit treatment of rows and columns that are added to the constraint matrix in the preprocessing phase of different variants of the method. In Section 3 a theoretical background of the primal-dual method implemented on the basis of the library is discussed. Issues of its implementation are addressed in Section 4. In Section 5 the efficiency of the code is empirically evaluated and, on the basis of its application to the solution of medium scale problems from Gay's (1985) *Netlib* collection, compared with the one of the experimental simplex code of Gondzio (1990). The purpose of this comparison is to show reasonable efficiency of IPMLO library. Unfortunately, we do not know how our benchmark simplex code compares with the state-of-the-art simplex implementations. We only know that it is in the average 5-10 % faster than XMP of Marsten (1981). Finally, Section 6 brings our conclusions.

2. IPMLO DESCRIPTION

We start this section with some general remarks concerning the construction of a FORTRAN program library for linear optimization. Such a library should satisfy several widely accepted requirements (*see e. g.*, Gill *et al.*, 1979) that, unfortunately, often conflict with each other. Let us now briefly discuss the most important of them. For their analysis the reader is referred to the paper of Marsten (1981).

Subservience. The library should consist entirely of the set of subroutines that can be called (and linked) in different configurations. Argument lists should be the only mean for communication between different routines.

Readability. The source code should be well documented and thus readable to its intended users.

Extendibility. It should be possible to replace library routines by the alternative ones and add new capabilities.

Modularity and hierarchical structure. User programs should be able to call library routines at any level in their hierarchical structure.

Hidden data structures. No routine should access directly problem data structures (neither the routines that incorporate the logic of interior point algorithm nor the ones that handle Karmarkar projections).

Ability to solve large problems. It should be able to solve at least medium scale problems (of up to 2,000 constraints and 10,000 variables) such as for example those from *Netlib* collection.

Reliability. It should respond quickly to user's errors or numerical difficulties.

Portability. It should be easy to install and compile on any kind of computer.

In the following subsections the process of finding the compromise among these requirements is addressed.

2.1. Structure of the library

The structure of the library reflects as much as possible the natural phases of solving an LP problem. In particular, such functions as: reading MPS-formatted data, preprocessing, solving the problem and writing the results can be distinguished in it. To satisfy the modularity requirement, much effort has been made to minimize the cross references among these phases and within them.

MPS data input has practically been completely isolated from the rest of the library. It was possible since a special internal standard formulation of the LP problem, independent of the variant of interior point algorithm is kept (*see* section 2.2).

Analogously, the whole preprocessing phase is also independent of the logic of the interior point method chosen. This phase is in turn (*see* section 2.3) strongly dependent on the method of handling Karmarkar projections

for which we applied the Cholesky decomposition and should be replaced if the user wants to apply another approach for computing projections.

The most implementationally involved part of the code is the *driver routine* (solver) for the chosen variant of the interior point method. We describe it in section 4.

IPMLO output, *i. e.* writing the MPS formatted results, was also made general in wide extent.

2.2. Problem data management

IPMLO is intended to solve the linear programming problems

$$\begin{aligned} &\text{minimize} \quad c^T x, && (1\ a)\\ &\text{subject to} \quad A\,x = b, && (1\ b)\\ &0 \le x \le u, && (1\ c) \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$, c, x, $u \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. This internal standard form used in the library differs from the general one in which inequality constraints, ranges for the right hand sides and nonzero lower bounds of the variables are allowed. Any linear program of more general form

$$\begin{aligned} &\text{minimize} \quad c_y^T y, && (2\ a)\\ &\text{subject to} \quad b_y - r \le A_y y \le b_y, && (2\ b)\\ &l_y \le y \le u_y, && (2\ c) \end{aligned}$$

where $A_y \in \mathbb{R}^{p \times q}$, c_y, y, l_y, $u_y \in \mathbb{R}^q$ and $b_y, r \in \mathbb{R}^p$, can easily be transformed to its equivalent form (1) (*see e. g.*, Murtagh, 1981). It is achieved by applying to (2) some straightforward techniques such as: adding slack variables to constraints (2 *b*) (upper bounded, if ranges r are present), removing fixed variables and moving variables bounded from below to zero lower bound (these operations require simple modifications of b_y vector), splitting unbounded variables, etc.

We have found useful to stop problem transformations with form (1) although we are aware that many variants of the interior point algorithm (*see e. g.*, Vial, 1987 and Goldfarb and Todd, 1989) need further transformations (leading to zero right hand side, removing upper bounds of variables, etc.). As those usually depend on the chosen approach, it is advantageous to handle them implicitly within the logic of the algorithm instead of explicit bordering LP constraint matrix with additional (presumably dense) rows and columns. Since this problem seems particularly important for the modularity of the library, we shall address it in more detail in section 2.4.

Differently to the simplex method, in which it suffices to handle LP constraint matrix A by columns only, interior point algorithms need much more computations that involve A and, consequently, require comfortable access to both rows and columns of it (*see e. g.*, Alder *et al.*, 1989 *a*). Following the techniques discussed in chapter 2 of the book of Duff *et a/.* (1989), we thus store it as a collection of sparse column vectors (CLPNTS, RWNMBS and ELMNTS arrays are pointers to columns, row numbers and nonzero elements, respectively), and additionally remember the sparsity pattern of A by rows in the form of row linked lists (RWHEAD, RWLINK and CLNMBS arrays are hearders to the lists, row linked lists and column numbers where nonzero entries are present, respectively).

2.3. Preprocessing

The reading of the MPS input file ends up with the construction of data structures for the IPMLO internal standard problem formulation (1). Depending however on the method applied to Karmarkar projections these data structures may require further modifications.

Preprocessing phase depends on the choice of the method used for computing Karmarkar projections. As has already been stated, the Cholesky decomposition of the matrix of form $A\theta A^T$ is used for this purpose. The matrices that have to be inverted in successive iterations of any interior point method differ only with the diagonal weighting matrix θ. They thus share the same sparsity pattern although their numerical values change. This means that an expensive sparsity structure analysis, *i. e.* reordering that minimizes the fill-in of Cholesky matrix and the symbolic factorization, can be performed only once in the whole solution process (*see e. g.*, Gondzio, 1991). The building up of the matrix $A\theta A^T$ and the numerical phase of Cholesky decomposition (followed with the solves with the triangular factor) have to be repeated at every iteration of the method.

The whole sparsity structure analysis is then done in the following five steps:

(i) removing empty rows from A;

(ii) splitting dense columns of A;

(iii) finding the minimum degree ordering of AA^T;

(iv) permuting rows of A according to the reordering resulting from (iii);

(v) building static data structures for the Cholesky matrix (symbolic factorization).

The reader interested in more detail in the preprocessing phase is referred to Alder *et al.* (1989 *a*) and Gondzio (1991, 1992). Let us only mention that the idea of splitting dense columns of A applied in our library was first suggested by Karmarkar at the Asilomar Conference on Interior Point Methods in 1985 and later analysed by Vanderbei (1991).

It is a particularly important feature of the IPMLO library that all the preprocessing operates on the pure LP constraint matrix. Consequently, the Cholesky factorization is computed for matrix $A\,\theta A^T$ without bordered rows or columns. The routines computing the decomposition are thus well isolated from the logic of an interior point algorithm, which ensures their generality. Any modifications of A required by different variants of the interior point method, that cause usually bordering A with (probably dense) rows and/or columns, are always handled implicitly.

2.4. Bordered rows and columns

There exist several reasons which justify the choice of direct approach to the solution of equations with $A\,\theta A^T$ (*see e. g.*, Gondzio, 1991). (Direct approach is the only one available in a current version of IPMLO library.) As was already mentioned, IPMLO library applies Cholesky factorization to handle these equations

$$A\,\theta\,A^T = LL^T, \tag{3}$$

where L is a lower triangular matrix of dimension m. It is easy to observe that having computed (3), equations with $A\,\theta A^T$ can be replaced with two triangular solves with matrices L and L^T, respectively.

The computational practice of application of the Cholesky factorization indicates that the process of finding decomposition (3) involves usually much more *flops* (floating point operations) than applying this factorization to solve equations with $A\,\theta A^T$. The costs of computing the decomposition and one triangular solve with factor L (or L^T) $1/2 \sum_{i=1}^{m} n_i^2$ and $\sum_{i=1}^{m} n_i$, respectively (n_i denotes the number of entries of the i-th column of matrix L). Consequently, it is often advantageous to simplify the decomposition alone even if more triangular solves have to be done later. Such approach is particularly useful when dense columns are added to the LP constraint matrix, which is the case in many variants of the interior point method that transform the problem to be solved into a new equivalent form with zero right hand side or add an artificial variable when an initial feasible solution is looked for (*see e. g.*: Vial, 1987 and Lustig, 1991).

Consequently, instead of solving the equation

$$A\,\theta\,A^T\,\eta = d, \tag{4}$$

the computation of the projection requires solving the more complicated equation

$$A_c\,\theta_c\,A_c^T\,\xi = f, \tag{5}$$

where

$$A_c = \left[\, A \;\middle|\; C \,\right] \text{ with } C \in \mathbb{R}^{m\times k}, \tag{6}$$

and

$$\theta_c = \left[\, \operatorname{diag}\theta \;\middle|\; \operatorname{diag} D_C \,\right] \text{ with } \operatorname{diag} D_c \in \mathbb{R}^{k\times k}, \tag{7}$$

Observe that this technique could be applied not only to handle added columns but also to deal with dense columns removed from A, if such are present in the linear program. We do not however suggest to use it for such purpose since removing dense columns from A may lead to the rank-deficiency of $A_s\,\theta_s\,A_s^T$ (A_s and θ_s denote sparse parts of A and θ, respectively). We rather suggest a more robust approach that splits dense columns into shorter ones (*see e. g.*: Karmarkar, 1985, Vanderbei, 1991 and Gondzio, 1992) and can never affect the full row rank property of A.

Substituting (6) and (7) into (5) gives

$$(A\,\theta\,A^T + CD_cC^T)\,\xi = f \tag{8}$$

Applying (3) and the Sherman-Morrison-Woodbury formula (*see e. g.*, Golub and Van Loan, 1983, p. 3) to (8), we obtain

$$\begin{aligned}\xi &= (A\theta A^T + CD_c\,C^T)^{-1} f \\ &= (LL^T + CD_c\,C^T)^{-1} f \\ &= \left(LL^{-1}\right)(I - VS^{-1}\,W^T L^{-1})\,f\end{aligned} \tag{9}$$

where

$$V = CD_c^{1/2}, \tag{10}$$

$$W = L^{-1}V, \tag{11}$$

and

$$S = I + W^T\,W \tag{12}$$

is a $k\times k$ Schur complement (*see e. g.*: Cottle, 1974 or Hager, 1989).

Summing up, given the factorization (3), the solution ξ of (5) can be obtained by the following sequence of calculations (*see* (9)-(12)):

(i) solve $L g = f$,

(ii) solve $LW = V = CD_c^{1/2}$,

(iii) compute $S = I + W^T W$,

(iv) solve $Sh = W^T g$,

(v) solve $(L L^T)\xi = f - Vh$.

We thus avoid the decomposition of $A_c \theta_c A_c^T$ for which we pay with more triangular solves in steps (i) and (ii) and an additional factorization of the small, in general, Schur complement S.

Although the above presentation assumes adding a general $m \times k$ matrix C to A, in practice, C is usually built with only one or two columns. These two special cases have already been implemented in IPMLO.

One has to be aware that applying Schur complement approach to handling dense columns of A may lead to serious stability problems (even for well formulated programs the part of A remaining after removing dense columns may be rank deficient). Lustig *et al.* (1991) address this problem and discuss methods to overcome numerical difficulties in such case. IPMLO library uses splitting to prevent degrading influence of dense columns and applies the Schur complement mechanism only for handling artificial column [equation (35)] bordered to the LP problem. Usual difficulties associated with applying Schur complements cannot thus arise in it as long as the original linear program is well formulated, *i. e.* rank $(A) = m$.

Another problem constitutes bordering matrix A with new rows, *i. e.* the need of solving

$$A_R \theta A_R^T \xi = f, \tag{13}$$

where

$$A_R = \left[\frac{A}{R}\right] \quad \text{with } R \in \mathbb{R}^{k \times n}, \tag{14}$$

and $\xi,\ f \in \mathbb{R}^{m \times k}$.

In this case, formulas for solving (13) trace back to the block elimination technique (*see e. g.*, Duff *et al.*, 1989, p. 97):

$$A_R \theta A_R^T = \left[\frac{A}{R}\right] \theta \left[A^T | R^T\right]$$

$$= \left[\begin{array}{c|c} A\,\theta\,A^T & P^T \\ \hline P & K \end{array}\right]$$

$$= \left[\begin{array}{c|c} A\,\theta\,A^T & 0 \\ \hline P & S \end{array}\right] \left[\begin{array}{c|c} I & Q \\ \hline 0 & I \end{array}\right] \tag{15}$$

where

$$\left.\begin{array}{l} P = R\,\theta\,A^T, \qquad K = R\,\theta\,R^T, \\ Q = (A\,\theta\,A^T)^{-1}\,A\,\theta\,R^T, \end{array}\right\} \tag{16}$$

and

$$S = R\,\theta\,R^T - R\,\theta\,A^T\,(A\,\theta\,A^T)^{-1}\,A\,\theta\,R^T \tag{17}$$

Consequently, given the factorization (3) and partition

$$\xi = (\xi_A,\ \xi_R) \qquad \text{and} \qquad f = (f_A, f_R), \tag{18}$$

equation (13) may be replaced with the following sequence of calculations:

(i) solve $\left(LL^T\right) Q = A\,\theta\,R^T$,

(ii) solve $\left(LL^T\right) g = f_A$,

(iii) compute $S = R\,\theta\,R^T - R\,\theta\,A^T\,Q$,

(iv) solve $S\,\xi_R = f_R - P^T\,g$,

(v) compute $\xi_A = g - Q\xi_R$.

In other words we avoid the decomposition of $A_R\,\theta\,A_R^T$ but more equations with the Cholesky factor have to be solved in steps (i) and (ii) and an additional factorization of the small Schur complement has to be computed.

In the above presentation, a general $k \times n$ matrix R bordered to A was considered. In practice however only one row is usually added to A, which has already been implemented in IPMLO.

Summing up, rows and columns bordered to A by different reformulations of the original problem (1) can be handled implicity without affecting the Cholesky decomposition (3). The maintained generality of Cholesky factorization (it is independent of the variant of the interior point method used) and the preventing of the sparsity of L (added columns might degrade it substantially) are obvious advantages of such approach.

3. THE PRIMAL-DUAL LOGARITHMIC BARRIER METHOD

In this section we shall briefly remind a logarithmic barrier interior point algorithm that has already been implemented on the basis of IPMLO. We

address theoretical issues rather, leaving implementational details to be discussed in the next section.

We have chosen the primal-dual logarithmic barrier interior point method (*see e. g.*: Megiddo, 1986; Kojima *et al.*, 1986; Monteiro and Adler, 1989; McShane *et al.*, 1989 and Choi *et al.*, 1990) mostly due to its high efficiency. It applies the logarithmic barrier approach (*see e. g.*, Fiacco and McCormick, 1968) simultaneously to primal and dual problems. Consequently, it iterates at the same time on the interior estimates of both primal and dual variables and performs Newton steps that maintain feasibility and reduce the violation of the complementarity constraint. Although its single iteration is slightly more expensive than that of a pure primal or a pure dual barrier method, the primal-dual method has several advantages. It gives both primal and dual optimal solutions and it can earlier be terminated since it works with the exact duality gap as soon as the primal feasibility is obtained.

Gill *et al.* (1986) showed that the projective method of Karmarkar (1984) is under some assumptions equivalent to a logarithmic barrier one. Meddigo (1986) was to our knowledge the first to propose applying logarithmic barrier approach to primal and dual problems at the same time. This led Kojima *et al.* (1986), Monteiro and Alder (1989) and McShane *et al.* (1989) to practical (implementable) methods.

Let us consider the dual pair of LP problems

$$\text{(P)} \qquad \text{minimize} \quad c^T x, \tag{19a}$$

$$\text{subject to} \quad Ax = b, \tag{19b}$$

$$x \geq 0, \tag{19c}$$

and

$$\text{(D)} \qquad \text{minimize} \quad b^T y, \tag{20a}$$

$$\text{subject to} \quad A^T y + z = c, \tag{20b}$$

$$z \geq 0, \tag{20c}$$

where $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$.

Simple bounds (19*c*) and (20*c*) can be replaced in these problems by logarithmic barrier function (*see e. g.*, Fiacco and McCormick, 1968) leading to the following primal and dual objective functions

$$f_P(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \ln x_i, \tag{21}$$

$$f_D(y,\ z,\ \mu) = b^T x - \mu \sum_{i=1}^{n} \ln z_i, \tag{22}$$

where μ denotes the barrier coefficient.

Problems (19) and (20) can thus be replaced by the barrier equivalents

(PB) $$\begin{aligned} &\text{minimize} \quad f_P(x,\ \mu) && (23\,a)\\ &\text{subject to} \quad A\,x = b, && (23\,b) \end{aligned}$$

and

(DB) $$\begin{aligned} &\text{maximize} \quad f_D(y,\ z,\ \mu) && (24\,a)\\ &\text{subject to} \quad A^T y + z = c. && (24\,b) \end{aligned}$$

Having formulated lagrangians for (23) and (24):

$$L_{PB}\,(x,\ y,\ \mu) = c^T x - \mu \sum_{i=1}^{n} \ln x_i - y^T\,(A\,x - b), \tag{25}$$

$$L_{DB}\,(x,\ y,\ z,\ \mu) = b^T y - \mu \sum_{i=1}^{n} \ln z_i - x^T\,(A^T y + z - c), \tag{26}$$

we can easily derive the first order conditions for (23) and (24)

$$A\,x = b, \tag{27\,a}$$

$$A^T\,y + z = c, \tag{27\,b}$$

$$X\,Ze = \mu\,e, \tag{27\,c}$$

where X and Z denote diagonal matrices built with x_i and z_i, respectively and e is a vector of ones in $\mathbb{R}^n$.

(27 *a*, *b*) are primal and dual feasibility conditions while (27 *c*) leads to the complementarity condition as μ tends to 0. The algorithm assumes that feasible interior primal, dual and dual slack solutions $x \geq 0$, y and $z \geq 0$ are known and applies Newton's method to determine their corrections Δx, Δy and Δz, respectively. The corrections maintain the feasibility and reduce

the violation of the complementarity constraint (27 *c*). This leads to the following equations that define the corrections

$$A\,\Delta x = 0, \tag{28a}$$

$$A^T \Delta y + \Delta z = 0 \tag{28b}$$

$$Z\Delta x + X\Delta z = v(\mu), \tag{28c}$$

where

$$v\,(\mu) = X\,Z\,e - \mu\,e. \tag{29}$$

Their solution gives

$$\Delta y = -B^{-1}AZ^{-1}v(\mu), \tag{30a}$$

$$\Delta z = -A^T \Delta y, \tag{30b}$$

$$\Delta x = Z^{-1}v(\mu) - Z^{-1}X\Delta z, \tag{30c}$$

where

$$B = AZ^{-1}XA^T. \tag{30d}$$

Correction of the current solutions *x, y* and *z* with the direction (30) completes the iteration.

The algorithm proceeds until the duality gap becomes small. It is advantageous that it operates on feasible solutions, which allows its earlier termination. Additionally, if continued until the end, it gives both primal and dual optimal solutions never mind whether a degeneracy is present or not.

4. IMPLEMENTATION OF THE PRIMAL-DUAL METHOD

In this section we shall address the problem of method's implementation on the basis of IPMLO. We shall in particular focus our attention on exploiting modularity features of the library.

As was shown in section 3, the primal-dual method operates on feasible estimates *x*, *y* and *z* of primal, dual and dual slack variables. As those are not known in advance, we shall apply the approach of McShane *et al.* (1989) and transform problems (19) and (20) to some more complicated forms with *a priori* known feasible solutions. We assume that some initial values $x^0 \geq 0$, y^0 and $z^0 \geq 0$ that may violate constraints (19 *b*) and (20 *b*) are

given. Consequently, (19 b) may be replaced by a new constraint with the artificial variable x_a

$$A x + \left(b - A x^0\right) x_a = b, \tag{31}$$

that has a strictly positive feasible solution (x^0, 1). Similarly, given y^0, a sufficiently large β can be found such that for $y_a = -1$

$$z^0 = c - A^T y^0 - \beta e y_a, \tag{32}$$

is strictly positive.

Summing up, a new dual pair with *a priori* known feasible solutions can be formulated

(P1)

$$\text{minimize} \quad c^T x + c_a x_a, \tag{33a}$$

$$\text{subject to} \quad A x + \left(b - A x^0\right) x_a = b, \tag{33b}$$

$$\beta e^T x + x_b = b_a, \tag{33c}$$

$$x, x_a, x_b \geq 0, \tag{33d}$$

and

(D1)

$$\text{maximize} \quad b^T y + b_a y_a, \tag{34a}$$

$$\text{subject to} \quad A^T y + \beta e y_a + z = c, \tag{34b}$$

$$\left(b - A x^0\right)^T y + z_a = c_a, \tag{34c}$$

$$y_a + z_b = 0, \tag{34d}$$

$$z, z_a, z_b \geq 0. \tag{34e}$$

Observe that artificial variables x_a, y_a and slack variables x_b, z_a and z_b were added to the original problem leading to its bordering with two columns and one row

$$\underline{A} = \left[\begin{array}{c|c|c} A & d & 0 \\ \hline \beta e^T & 0 & 1 \end{array}\right], \tag{35}$$

where column $d = b - A x^0$ is presumably dense.

As shown in section 2.4 such bordered row and columns can and should be handled implicitly. This is then the approach chosen. We use the Cholesky factorization to decompose $AZ^{-1}XA^T$ and later, when the direction (30) has to be computed, solve equations with $\underline{AZ}^{-1}\underline{XA}^T$ applying techniques of section 2.4.

Observe that once primal (or dual) feasibility is achieved, the artificial column (or row) may be removed from (35), which simplifies the computation of Newton's direction.

We omit further discussion of implementational details and refer to Choi *et al.* (1990), Lustig (1991), Lustig *et al.* (1991) and McShane *et al.* (1989) for their studies. The numerical results reported in this paper were obtained for the parameters set up as below.

Initial solutions:

$$x^0 = 1, \qquad y^0 = 0,$$

primal and dual step lengths:

$$\alpha_P = \alpha_D = 0.999,$$

artificial cost coefficients:

$$c_a = 10\, n^2 \max |c_i|,$$

and $b_a = 10\, n^2 \max |b_i|$,

the barrier parameter determined at every iteration as:

$$\mu = \frac{c^T x - b^T y}{n^2},$$

and the stopping criteria

$$\frac{c^T x - b^T y}{c^T x} < \varepsilon,$$

with $\epsilon = 10^{-8}$.

5. NUMERICAL RESULTS

We shall now present some preliminary computational results. They show in particular that when the efficiency of code is concerned, IPMLO routines compare favorably with other LP codes. Consequently, below, the results of applying the primal-dual interior point method and the experimental simplex code of Gondzio (1990) to the solution of several medium scale problems from *Netlib* collection (those which have no upper bounded variables) are presented. The simplex method used in this comparison was based on Marsten's XMP library. LA05 routines of Reid (1982) that handle Bartels-Golub updates of the basis were however replaced in it with slightly faster and remarkably more storage efficient basis inverse representation that apply Schur complement updates.

The simplex implementation of Gonzio (1990) is an experimental one and is probably less efficient than state-of-the-art simplex codes (*see e.g.*: CPLEX of Bixby (1992), OSL of Forrest and Tomlin (1991) or MINOS 5.3 of Murtagh and Saunders (1987)). Comparison of IPMLO with this benchmark LP code does not aim to draw general conclusions on the performance of two practicable LP approaches: interior point and simplex ones. It is made only to show reasonable efficiency of the IPMLO library on some of *Netlib* tests.

Table I contains the description of the test problems (M and N denote problem dimensions, τ_A indicates the number of nonzero elements of the LP constraint matrix (1 *b*)) and the numbers of nonzero entries in appropriate inverse representations. In case of the interior point method this contain the number of subdiagonal entries of $A\,\theta\,A^T$, the number of subdiagonal entries of its Cholesky factor L, and the fill-in. For the simplex method, the number of nonzero entries of the largest Schur complement encountered during the whole run is reported.

TABLE I
Nonzeros of the inverse representations.

Problem	M	N	τ_A	Interior point method			Simplex
				$\tau(AA^T)$	τ_L	Fill-in	nz of Schur
AFIRO	27	51	102	63	80	17	36
ADLITTLE . . .	56	138	424	328	355	27	900
SHARE2B . . .	96	162	777	775	941	166	961
SHARE1B . . .	117	253	1,179	884	1,266	382	1,521
SCAGR7	129	185	465	500	637	137	961
SCSD6	147	1,350	4,316	1,952	2,398	446	2,025
BEACONFD . .	173	295	3,408	2,669	2,728	59	1,849
ISRAEL	174	318	2,443	11,053	11,259	206	961
ISRAEL (split)	182	326	2,459	5,819	7,640	1,821	–
BRANDY	182 (*)	292	2,191	2,541	3,231	690	1,849
SC205	204 (*)	316	664	451	1,000	549	729
E226	223	472	2,768	2,600	3,443	843	1,936
SCTAP1	300	660	1,872	1,386	2,360	974	2,304
BANDM	305	472	2,494	3,419	4,358	939	1,849
SCFXM1	330	600	2,732	2,903	4,452	1,549	2,401
SCAGR25 . . .	471	671	1,725	1,922	2,509	587	1,843
SCRS8	490	1,275	3,288	1,708	5,804	4,096	1,936

(*) Empty rows have been removed from BRANDY (38) and SC205 (1).

The analysis of table I results substantiates the well known conclusion on the storage efficiency of simplex code when compared with interior point method, especially that the memory needs of the interior point algorithm are at least two times larger than that indicated in column τ_L (results collected in table I indicate numbers of nonzero entries of different inverse representations and not their storage needs).

Table II collects results on the efficiency of the two methods considered: primal-dual logarithmic barrier one and the benchmark simplex one. For every method, both the number of iterations and the solution times on a 20 MHz IBM 80386 computer with the arithmetic coprocessor 80387, the memory limited to 640 kB and the relative precision $\epsilon = 2.2 \times 10^{-16}$ are given.

TABLE II

Comparison of the methods' efficiency.

Problem	Primal-dual method		Simplex method	
	iters	time	iters	time
AFIRO	14	2 sec.	9	1 sec.
ADLITTLE	20	8 sec.	171	9 sec.
SHARE2B	18	12 sec.	146	12 sec.
SHARE1B	51	42 sec.	330	34 sec.
SCAGR7	22	10 sec.	112	10 sec.
SCSD6	15	33 sec.	543	1 min. 04 sec.
BEACONFD	21	1 min. 2 sec.	118	16 sec.
ISRAEL	–	–	464	1 min. 04 sec.
ISRAEL (split)	33	3 min. 22 sec.	–	–
BRANDY (*)	31	1 min. 20 sec.	329	56 sec.
SC205 (*)	21	15 sec.	50	7 sec.
E226	36	1 min. 32 sec.	806	2 min. 02 sec.
SCTAP1	27	40 sec.	349	51 sec.
BANDM	32	1 min. 30 sec.	578	2 min. 01 sec.
SCFXM1	32	1 min. 36 sec.	475	1 min. 25 sec.
SCAGR25	33	55 sec.	642	2 min. 05 sec.
SCRS8	53	3 min. 18 sec.	683	2 min. 33 sec.

(*) Empty rows have been removed from BRANDY (38) and SC205 (1).

As the results of table II show, primal-dual implementation based on IPMLO compares favorably with the simplex code. They also indicate that even medium scale LP problems may be solved on a microcomputer with only 640 kB of operational memory. IPMLO library seems thus to be a useful and easy to install tool when a practical development of new interior point methods is concerned.

6. CONCLUSIONS

IPMLO library can be used for different research purposes:

– it can be the basis for implementing new attractive variants of interior point method (*see e. g.*, Tolla, 1987);

– it can be modified to deal with specially structured linear programs such as dynamic, stochastic, network (*see e. g.*, Lisser and Tolla, 1989);

– it can be used for experiments with different projection techniques (iterative as *e. g.*, Karmarkar and Ramakrishnan, 1991 and others as *e. g.*, Tachat, 1991).

The code itself is small and uses storage efficient data structures. Consequently, even on a microcomputer with 640 kB of operational memory, problems of remarkable size (up to 500 constraints and 1,000 variables) can be solved with it. Additionally, it compares favorably with other LP solvers.

It is well-documented and easily extendible due to its modularity.

We end up this paper with indicating near future development of the IPMLO library. There are at least three attractive directions of these enhancements. The first one is the incorporation of upper bounds on variables in it (current version does not accept them). The second one is the improvement of the efficiency of the already implemented primal-dual logarithmic barrier interior point method and the coding of other computationally attractive methods. The third one is the implementation of different methods for computing projections that could deal with badly conditioned problems.

ACKNOWLEDGMENTS

The first author is grateful to Professor Pierre Tolla for the possibility of spending 15 months on sabbatical leave with his research group at LAMSADE, University of Paris Dauphine. Financial support of the French Government during the first 9 months of this period is kindly acknowledged.

The authors are grateful to anonymous referee for careful reading of the manuscript and pointing out a number of improvements.

AVAILABILITY OF CODE

It is our intention to make the IPMLO code available for any research purposes. More information regarding this can be obtained by contacting Jacek Gondzio.

REFERENCES

I. ADLER, N. KARMARKAR, M. G. C. RESENDE and G. VEIGA, Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm, *O.R.S.A. Journal on Computing*, 1989 *a*, *1*, No. 2, pp. 84-106.

I. ADLER, N. KARMARKAR, M. G. C. RESENDE and G. VEIGA, An Implementation of Karmarkar's Algorithm for Linear Programming, *Mathematical Programming*, 1989 *b*, *44*, pp. 297-335.

R. BIXBY, Implementing the Simplex Method: the Initial Basis, *O.R.S.A. Journal on Computing*, 1992, *4*, No. 3, pp. 267-284.

Y.-C. CHENG, D. J. HOUCK Jr., J.-M. LIU, M. S. MEKETON, L. SLUTSMAN, R. J. VANDERBEI and P. WANG, The AT&T KORBX System, *AT&T Technical Journal*, 1989, May/June, pp. 7-19.

I. C. CHOI, C. L. MONMA and D. F. SHANNO, Further Development of a Primal-Dual Interior Point Method, *O.R.S.A. Journal on Computing*, 1990, *2*, pp. 304-311.

R. W. COTTLE, Manifestations of the Schur Complement, *Linear Algebra and its Applications*, 1974, *8*, pp. 189-211.

I. S. DUFF, A. M. ERISMAN and J. K. REID, *Direct methods for sparse matrices*, Oxford University Press, New York, 1989.

A. V. FIACCO and G. P. MCCORMICK, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York, 1968.

J. J. H. FORREST and J. A. TOMLIN, Implementing the Simplex Method for the Optimization Subroutine Library, *I.B.M. Systems Journal*, 1991, *31*, No. 2, pp. 11-25.

D. M. GAY, Electronic Mail Distribution of Linear Programming Test Problems, *Mathematical Programming Society COAL Newsletter*, 1985.

A. GEORGE and J. W. H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Inc., Englewood Cliffs, 1981.

P. E. GILL, W. MURRAY, S. M. PICKEN and M. H. WRIGHT, The Design and Structure of a FORTRAN Program Library for Optimization. *A.C.M. Transactions on Mathematical Software*, 1979, *5*, No. 3, pp. 259-283.

P. E. GILL, W. MURRAY, M. A. SAUNDERS, J. A. TOMLIN and M. H. WRIGHT, On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method, *Mathematical Programming*, 1986, *36*, pp. 183-209.

D. GOLDFARB and M. J. TODD, Linear Programming, *in*: G. L. NEMHAUSER, A. H. G. RINNOOY KAN and M. J. TODD Eds., *Optimization*, North Holland, Amsterdam, 1989.

G. H. GOLUB and C. F. VAN LOAN, *Matrix Computations*, John Hopkins University Press, Baltimore, 1983.

J. GONDZIO, On Exploiting Original Problem Data in the Inverse Representation of the Linear Programming bases, *O.R.S.A. Journal on Computing*, 1990 (to appear).

J. GONDZIO, Implementing Cholesky Factorization for Interior Point Methods of Linear Programming, *Optimization*, 1991 (to appear).

J. GONDZIO, Splitting Dense Columns of Constraint Matrix in Interior Point Methods for Large Scale Linear Programming, *Optimization*, 1992, *24*, pp. 285-297.

W. HAGER, Updating the Inverse of a Matrix, *S.I.A.M. Review*, 1989, *31*, No. 2, pp. 221-239.

N. K. KARMARKAR, A New Polynomial Time Algorithm for Linear Programming, *Combinatorica*, 1984, *4*, pp. 373-395.

N. K. KARMARKAR, A Talk at the Asilomar Conference on Interior Point Methods, Asilomar, California, U.S.A., 1985.

N. K. KARMARKAR and K. G. RAMAKRISHNAN, Computational Results of an Interior Point Algorithm for Large Scale Linear Programming, *Mathematical Programming*, 1991, *52*, pp. 555-586.

M. KOJIMA, S. MIZUNO and A. YOSHISE, A Primal-Dual Interior Point Algorithm for Linear Programming, *in*: N. MEGIDDO Ed., *Progress in Mathematical Programming*, Springer-Verlag, New York, 1986.

A. LISSER and P. TOLLA, *Variants of Karmarkar's Algorithm*, Technical Report 90, LAMSADE, University of Paris Dauphine, March, 1989.

I. LUSTIG, Feasibility Issues in a Primal-Dual Interior Point Method for Linear Programming, *Mathematical Programming*, 1991, *49*, pp. 145-162.

I. LUSTIG, R. E. MARSTEN and D. F. SHANNO, Computational Experience with a Primal-Dual Interior Point Method for Linear Programming, *Linear Algebra and its Applications*, 1991, *152*, pp. 191-222.

R. E. MARSTEN, The Design of XMP Linear Programming Library, *A.C.M. Transactions on Mathematical Software*, 1981, *7*, pp. 481-497.

R. E. MARSTEN, R. SUBMARANIAN, M. SALTZMAN, I. LUSTIG and D. SHANNO, Interior Point Methods for Linear Programming: Just Call Newton, Lagrange and Fiacco and McCormick, *Interfaces*, 1990, *20*, pp. 105-116.

K. A. MCSHANE, C. L. MONMA and D. F. SHANNO, An Implementation of a Primal-Dual Interior Point Method for Linear Programming, *O.R.S.A. Journal on Computing*, 1989, *1*, pp. 70-83.

N. MEGIDDO, Pathways to the Optimal Set in Linear Programming, *in*: N. MEGIDDO Ed., *Progress in Mathematical Programming*, Springer-Verlag, New York, 1986.

C. L. MONMA and A. J. MORTON, Computational Experience with a Dual Affine Variant of Karmarkar's Method for Linear Programming, *Operations Research Letters*, 1987, *6*, pp. 261-267.

R. C. MONTEIRO and I. ADLER, Interior Path Following Primal-Dual Algorithms – part I: Linear Programming, *Mathematical Programming*, 1989, *44*, pp. 27-41.

B. MURTAGH, *Advanced Linear Programming, Computation and Practice,* McGraw-Hill, New York, 1981

B. MURTAGH and M. A. SAUNDERS, *MINOS 5.1 User's guide*, Technical Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, California, 1983 (revised 1987).

J. K. REID, A Sparsity Exploiting Variant of the Bartels-Golub Decomposition for linear programming Bases, *Mathematical Programming*, 1982, *24*, pp. 55-69.

D. TACHAT, Interior Point Methods for Linear Programming: Computing Exact and Approximate Projections, *Ph. D. Thesis*, LAMSADE, University of Paris-Dauphine, Paris, 1991 (in French).

P. TOLLA, *Improvement of the Efficiency of Karmarkar's Algorithm for Linear Programs with Upper Bounded Variables*, Technical Report 82, LAMSADE, University of Paris-Dauphine, Paris, November, 1987.

R. J. VANDERBEI, Splitting Dense Columns in Sparse Linear Systems, *Linear Algebra and its Applications*, 1991, *152*, pp. 107-117.

J.-P. VIAL, *A Unified Approach to Projective Algorithms for Linear Programming*, Technical Report, Département d'Economie commerciale et industrielle, University of Geneva, Geneva, September 1987.