CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
(Université de Paris-Dauphine)
Unité de Recherche Associée au CNRS n^0 825

THE DESIGN AND APPLICATION OF THE IPMLO - A FORTRAN LIBRARY FOR LINEAR OPTIMIZATION WITH INTERIOR POINT METHODS [1]

CAHIER N° 108
janvier 1992

Jacek GONDZIO [2]
Dominique TACHAT [3]

Received : November 1991.

---

[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, 191 and at the 15th IFIP 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.*

[3] *LAMSADE, Université de Paris-Dauphine.*

CONTENTS

THE DESIGN AND APPLICATION OF THE IPMLO - A FORTRAN LIBRARY FOR LINEAR OPTIMIZATION WITH INTERIOR POINT METHODS

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 mediate scale LP test problems from *Netlib* collection are given and the comparison with the state-of-the-art implementation of the simplex method is made.

Key words. Linear Programming, Interior Point Methods, Program Library.

CONCEPTION ET APPLICATION DE IPMLO - UNE BIBLIOTHEQUE FORTRAN POUR L'OPTIMISATION LINEAIRE AVEC DES METHODES INTERIEURES

Résumé

Dans ce cahier, 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

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 1000 constraints are solved, the interior point methods are faster than the simplex-type ones (see e.g.: Cheng et al., 1989 and Marsten et al.,1990). 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 Linear Optimization problems with Interior Point Methods. 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 (1991b).

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 mediate scale problems (of up to 500 constraints and 1000 variables) on a microcomputer with operational memory limited to 640kB.

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 (even on a 640kB microcomputer) when applied to solve mediate 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 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 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 analysed and, on the basis of its application to the solution of mediate scale problems from Gay's (1985) *Netlib* collection, compared with the one of the simplex code

of Gondzio (1990). 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 excellent detailed 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. None 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 large scale problems 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 on the variant of interior point algorithm is kept (see section 2.2).

Analogously, the whole preprocessing phase is also independent on 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

$$\text{minimize} \quad c^T x \,, \tag{1a}$$

$$\text{subject to} \quad Ax = b \,, \tag{1b}$$

$$0 \le x \le u \,, \tag{1c}$$

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

$$\text{minimize} \quad c_y^T y \,, \tag{2a}$$

$$\text{subject to} \quad b_y - r \le A_y y \le b_y \,, \tag{2b}$$

$$l_y \le y \le u_y \,, \tag{2c}$$

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 (2b) (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 on 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. Following the techniques discussed in chapter 2 of the book of Duff et al. (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 headers to the lists, row linked lists and column numbers where nonzero entries are present, respectively).

Figures 1 and 2 below, present an example matrix A and data structures that handle it, respectively.

$$A = \begin{bmatrix} 3.0 & & & & & & & 5.0 & \\ & 1.0 & 2.0 & & & & & & 3.3 \\ -2.0 & & & 3.0 & & 2.1 & & & \\ & & & 3.5 & 0.2 & & & & \\ & & -7.0 & & & 4.5 & & & \\ 7.0 & & & & 1.0 & & -2.0 & & \end{bmatrix}$$

Fig. 1. Example LP constraint matrix $A \in \mathbb{R}^{6\times 8}$.

subscripts	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CLPNTS	1	4	5	7	9	11	13	15	16							
RWNMBS	1	3	6	2	2	5	3	4	4	6	3	5	1	6	2	
ELMNTS	3.	-2.	7.	1.	2.	-7.	3.	3.5	.2	1.	2.1	4.5	5.	-2.	3.3	
RWHEAD	1	4	2	8	6	3										
RWLINK	13	7	10	5	15	12	11	9	0	14	0	0	0	0	0	
CLNMBS	1	1	1	2	3	3	4	4	5	5	6	6	7	7	8	

Fig. 2. FORTRAN arrays storing matrix A of Fig. 1.

Observe that, to access sparse column of A, say column 3, one has to determine where it begins in RWNMBS and ELMNTS arrays (in the example of Figs 1 and 2, CLPNTS(3) = 5) and to compute its length (LENCOL = CLPNTS(4) - CLPNTS(3) = 2). Starting from position 5, two subsequent elements of RWNMBS and ELMNTS arrays contain row numbers where nonzeros of column 3 are present (2 and 5) and its nonzero entries (2.0 and -7.0), respectively. To access sparse row of A, say row 2, row linked list has to be scanned. It starts from RWHEAD(2) = 4 and contains three elements indicated by RWLINK array: 4 is the first one, 5 = RWLINK(4) and 15 = RWLINK(5) are second and third, respectively. Zero value of RWLINK(15) indicates the end of the list. Consequently, row 2 of A is stored in positions 4, 5 and 15 of two arrays CLNMBS and ELMNTS handling column numbers where nonzeros are present (2, 3 and 8) and the nonzero entries (1.0, 2.0 and 3.3), respectively.

Let us observe that such data structures ensure high efficiency of arithmetic operations with A and A^T and allow easy construction of AA^T matrix. The sparsity of A may easily be exploited in these operations. Additionally, data structures presented have very small memory requirements. Since RWNMBS and CLNMBS arrays can be half-length integer (RWLINK array must be full-length integer), the storage needed for LP constraint matrix management is

$$r_A = \tau_A \tag{3}$$

and $$i_A = n + m + 2\tau_A + 1 \tag{4}$$

of real and integer numbers, respectively (τ_A denotes the number of nonzero entries of A).

An important feature of IPMLO library is that data structures of Fig. 2 are never accessed directly by the high level routines. Those can only call two specialized routines GETROW and GETCOL to build up a single row or column of A. Our approach to this is similar in spirit to the one of

Marsten (1981) in which problem data was hidden in two work arrays and could be accessed by the limited number of routines specialized for this purpose. We are aware that such approach must lead to some loss of code's efficiency. Consequently two routines of those handling Cholesky factorization of AA^T that extensively operate on matrix A have exceptionally been given a direct access to data structures of Fig. 2.

2.3. Preprocessing

The reading of the MPS input file ends up with the construction of data structures of Fig. 2 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, 1991b). 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 papers of Gondzio (1991a,b).

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, 1991b). As was already mentioned, IPMLO library applies Cholesky factorization to handle these equations

$$A\Theta A^T = LL^T, \tag{5}$$

where L is a lower triangular matrix of dimension m. It is easy to observe that having computed (5), 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 (5) 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) are $\frac{1}{2}\sum_{i=1}^{m} n_i^2$ 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

$$A\Theta A^T\eta = d, \tag{6}$$

the computation of the projection requires solving the more complicated equation

$$A_c\Theta_c A_c^T\xi = f, \tag{7}$$

where

$$A_c = \left[\begin{array}{c|c} A & C \end{array}\right] \quad \text{with} \quad C \in \mathbb{R}^{m\times k}, \tag{8}$$

and

$$\Theta_c = \left[\begin{array}{c|c} \operatorname{diag}\Theta & \operatorname{diag} D_c \end{array}\right] \quad \text{with} \quad \operatorname{diag} D_c \in \mathbb{R}^{k\times k}, \tag{9}$$

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 split dense columns into shorter ones (see e.g., Gondzio, 1991a) and can never affect the full row rank property of A.

Substituting (8) and (9) into (7) gives

$$(A\Theta A^T + CD_C C^T)\xi = f. \tag{10}$$

Applying (5) and the Sherman-Morrison-Woodbury formula (see e.g., Golub and Van Loan, 1983, page 3) to (10), we obtain

$$\begin{aligned}\xi &= (A\Theta A^T + CD_C C^T)^{-1} f \\ &= (LL^T + CD_C C^T)^{-1} f \\ &= (LL^{-1})(I - VS^{-1}W^T L^{-1}) f \,.\end{aligned} \tag{11}$$

where

$$V = CD_C^{1/2}, \tag{12}$$

$$W = L^{-1}V \,, \tag{13}$$

and $$S = I + W^T W \,, \tag{14}$$

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

Summing up, given the factorization (5), the solution ξ of (7) can be obtained by the following sequence of calculations (see (11)-(14)):

(i) solve $Lg = f$,

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

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

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

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

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 assumed 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.

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

$$A_R \Theta A_R^T \xi = f, \tag{15}$$

where

$$A_R = \left[\begin{array}{c} A \\ \hline R \end{array}\right] \quad \text{with} \quad R \in \mathbb{R}^{k \times n}, \tag{16}$$

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

In this case, formulas for solving (15) trace back to the block elimination technique (see e.g., Duff et al., 1989, page 97):

$$A_R \Theta A_R^T = \left[\begin{array}{c} A \\ \hline R \end{array}\right] \Theta \left[\begin{array}{c|c} A^T & R^T \end{array}\right]$$

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

$$= \left[\begin{array}{c|c} A \Theta A^T & 0 \\ \hline P & S \end{array}\right] \left[\begin{array}{c|c} I & Q \\ \hline 0 & I \end{array}\right] \tag{17}$$

where

$$P = R \Theta A^T, \qquad K = R \Theta R^T,$$

$$Q = (A \Theta A^T)^{-1} A \Theta R^T, \tag{18}$$

and $$S = R \Theta R^T - R \Theta A^T (A \Theta A^T)^{-1} A \Theta R^T, \tag{19}$$

Consequently, given the factorization (5) and the partition

$$\xi = (\xi_A, \xi_R) \quad \text{and} \quad f = (f_A, f_R) , \tag{20}$$

the equation (15) may be replaced with the following sequence of calculations:

(i) solve $(LL^T)Q = A\Theta R^T$,

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

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

(iv) solve $S\xi_R = f_R$,

(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 implicitly without affecting the Cholesky decomposition (5). The maintained generality of Cholesky factorization (it is independent on 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. Megiddo (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 Adler (1989) and McShane et al. (1989) to practical (implementable) methods.

Let us consider the dual pair of LP problems

$$\begin{aligned} &\text{minimize} && c^T x \,, && (21a)\\ (P)\quad &\text{subject to} && Ax = b \,, && (21b)\\ & && x \geq 0 \,, && (21c) \end{aligned}$$

and

$$\begin{aligned} &\text{maximize} && b^T y \,, && (22a)\\ (D)\quad &\text{subject to} && A^T y + z = c \,, && (22b)\\ & && z \geq 0 \,, && (22c) \end{aligned}$$

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

Simple bounds (21c) and (22c) 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

$$f_P(x,\mu) = c^T x - \mu \sum_{i=1}^{n} \ln x_i, \tag{23}$$

$$f_D(y,z,\mu) = b^T x - \mu \sum_{i=1}^{n} \ln z_i, \tag{24}$$

where μ denotes the barrier coefficient.

Problems (21) and (22) can thus be replaced by the barrier equivalents

(PB) minimize $f_P(x,\mu)$ (25a)

subject to $Ax = b$, (25b)

and

(DB) maximize $f_D(y,z,\mu)$ (26a)

subject to $A^T y + z = c$. (26b)

Having formulated lagrangians for (25) and (26):

$$L_{PB}(x,y,\mu) = c^T x - \mu \sum_{i=1}^{n} \ln x_i - y^T(Ax-b) , \tag{27}$$

$$L_{DB}(x,y,z,\mu) = b^T y - \mu \sum_{i=1}^{n} \ln z_i - x^T(A^T y+z-c) , \tag{28}$$

we can easily derive the first order conditions for (25) and (26)

$$Ax = b , \tag{29a}$$

$$A^T y + z = c , \tag{29b}$$

$$XZe = \mu e , \tag{29c}$$

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

(29a,b) are primal and dual feasibility conditions while (29c) 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 (29c). This leads to the following equations that define the corrections

$$A\Delta x = 0 , \tag{30a}$$

$$A^T\Delta y + \Delta z = 0 , \tag{30b}$$

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

where $v(\mu) = XZe - \mu e$. (31)

Their solution gives

$$\Delta y = - B^{-1}AZ^{-1}v(\mu) , \tag{32a}$$

$$\Delta z = - A^T\Delta y , \tag{32b}$$

$$\Delta x = Z^{-1}v(\mu) - Z^{-1}X\Delta z , \tag{32c}$$

where $B = AZ^{-1}XA^T$. (32d)

Corrections of the current solutions x, y and z with the direction (32) 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 (21) and (22) to some more complicated forms with a priori known feasible solutions. We assume that some initial values $x^o \geq 0$, y^o and $z^o \geq 0$ that may violate constraints (21b) and (22b) are given. Consequently, (21b) may be replaced by a new constraint with the artificial variable x_a

$$Ax + (b - Ax^o)x_a = b \ , \tag{33}$$

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

$$z^o = c - A^T y^o - \beta e y_a \ , \tag{34}$$

is strictly positive.

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

$$\text{minimize} \quad c^T x + c_a x_a \ , \tag{35a}$$

$$\text{(P1) subject to} \quad Ax + (b - Ax^o)x_a = b \ , \tag{35b}$$

$$\beta e^T x + x_b = b_a, \tag{35c}$$

$$x, x_a, x_b \geq 0 \ , \tag{35d}$$

and

$$\text{maximize} \quad b^T y + b_a y_a \,, \tag{36a}$$

$$\text{(D1) subject to} \quad A^T y + \beta e y_a + z = c \,, \tag{36b}$$

$$(b - Ax^o)^T y + z_a = c_a \,, \tag{36c}$$

$$y_a + z_b = 0 \,, \tag{36d}$$

$$z,\ z_a,\ z_b \geq 0 \,. \tag{36e}$$

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

$$\underline{A} = \left[\begin{array}{c|c|c} A & d & 0 \\ \hline \beta e^T & 0 & 1 \end{array}\right] \,, \tag{37}$$

where column $d = b - Ax^o$ 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 (32) has to be computed, solve equations with $\underline{AZ^{-1}XA^T}$ applying techniques of section 2.4.

Observe that once a primal (or dual) feasibility is achieved, the artificial column (or row) may be removed from (37), which simplifies the computation of Newton's direction.

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

Initial solutions:

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

primal and dual step lengths:

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

artificial cost coefficients:

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

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

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 $\varepsilon = 10^{-6}$.

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 state-of-the-art LP codes. Consequently, below, the results of applying the primal-dual interior point method and the simplex code of Gondzio (1990) to the solution of several mediate 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.

Table 1 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 (1b)) 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 1. Nonzeros of the inverse representations.

				Interior Point Method			SIMPLEX
PROBLEM	M	N	τ_A	$\tau(AA^T)$	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	1179	884	1266	382	1521
SCAGR7	129	185	465	500	637	137	961
SCSD6	147	1350	4316	1952	2398	446	2025
BEACONFD	173	295	3408	2669	2728	59	1849
ISRAEL	174	318	2443	11053	11259	206	961
ISRAEL (split)	182	326	2459	5819	7640	1821	-
BRANDY	182*	292	2191	2541	3231	690	1849
SC205	204*	316	664	451	1000	549	729
E226	223	472	2768	2600	3443	843	1936
SCTAP1	300	660	1872	1386	2360	974	2304
BANDM	305	472	2494	3419	4358	939	1849
SCFXM1	330	600	2732	2903	4452	1549	2401
SCAGR25	471	671	1725	1922	2509	587	1843
SCRS8	490	1275	3288	1708	5804	4096	1936

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

The analysis of Table 1 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 col-

lected in Table 1 indicate numbers of nonzero entries of different inverse representations and not their storage needs).

Table 2 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 20MHz IBM 80386 computer with the arithmetic coprocessor 80387, the memory limited to 640kB and the relative precision $\varepsilon = 2.2\ 10^{-16}$ are given.

Table 2. Comparison of the methods' efficiency.

PROBLEM	Primal-Dual Method		Simplex Method	
	iters	time	iters	time
AFIRO	13	2s	9	1s
ADLITTLE	19	7s	171	9s
SHARE2B	17	11s	146	12s
SHARE1B	54	45s	330	34s
SCAGR7	22	10s	112	10s
SCSD6	15	33s	543	1m04s
BEACONFD	20	1m10s	118	16s
ISRAEL	-	-	464	1m04s
ISRAEL (split)	34	3m28s	-	-
BRANDY *	29	1m16s	329	56s
SC205 *	21	15s	50	7s
E226	35	1m30s	806	2m02s
SCTAP1	27	40s	349	51s
BANDM	29	1m24s	578	2m01s
SCFXM1	31	1m34s	475	1m25s
SCAGR25	33	55s	642	2m05s
SCRS8	52	3m18s	683	2m33s

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

As the results of Table 2 show, implementations based on IPMLO compare favorably with the efficient simplex code. They also indicate that even mediate scale LP problems may be solved on a microcomputer with only 640kB 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, 1989 and others as e.g., Tachat, 1991).

The code itself is small and uses storage efficient data structures. Consequently, even on a microcomputer with 640kB of operational memory, problems of remarkable size (up to 500 constraints and 1000 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.

Availabity 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 the authors.

References

Adler I., Karmarkar N., Resende M.G.C., Veiga G. (1989). An implementation of Karmarkar's algorithm for linear programming, *Mathematical Programming* **44**, pp. 297-335.

Cheng Y.-C., Houck D.J.Jr., Liu J.-M., Meketon M.S., Slutsman L., Vanderbei R.J., Wang P. (1989). The AT&T KORBX System, AT&T Technical Journal, May/June, pp. 7-19.

Choi I.C., Monma C.L., Shanno D.F. (1990). Further development of a primal-dual interior point method, *ORSA Journal on Computing* **2**, pp. 304-311.

Cottle R.W. (1974). Manifestations of the Schur complement, *Linear Algebra and its Applications* **8**, pp. 189-211.

Duff I.S., Erisman A.M., Reid J.K.(1989) Direct methods for sparse matrices, Oxford University Press, New York.

Fiacco A.V., McCormick G.P. (1968). Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, New York.

Forrest J.J.H., Tomlin J.A. (1990). Vector processing in simplex and interior point methods for linear programming, *Annals of Operations Research* **22**, pp. 71-100.

Gay D.M. (1985). Electronic mail distribution of linear programming test problems, Mathematical Programming Society COAL Newsletter.

George A., Liu J.W.H. (1981). Computer Solution of Large Sparse Positive Definite Systems, Prentice Hall, Inc., Englewood Cliffs.

Gill P.E., Murray W., Picken S.M., Wright M.H. (1979). The design and structure of a FORTRAN program library for optimization. *ACM Transactions on Mathematical Software* **5**, No 3, pp. 259-283.

Gill P.E., Murray W., Saunders M.A., Tomlin J.A., Wright M.H. (1986). On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method, *Mathematical Programming* **36**, pp. 183-209.

Goldfarb D., Todd M.J. (1989). Linear programming, in: G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd eds., Optimization, North Holland, Amsterdam, pp. 73-170.

Golub G.H., Van Loan C.F. (1983). Matrix Computations, John Hopkins University Press, Baltimore.

Gondzio J. (1990). On exploiting original problem data in the inverse representation of the linear programming bases, Technical Report ZTSW-1-A1214/90, Systems Research Institute, Warsaw, July.

Gondzio J. (1991a). Splitting dense columns of constraint matrix in interior point methods for large scale linear programming, Technical Report 105, LAMSADE, University of Paris Dauphine, Paris, April, revised in July 1991, to appear in *Optimization*.

Gondzio J., (1991b). An advanced implementation of Cholesky factorization for computing projections in interior point methods of large scale linear programming, Technical Report 107, LAMSADE, University of Paris Dauphine, Paris, December.

Hager W. (1989). Updating the inverse of a matrix, *SIAM Review* **31**, No 2, pp. 221-239.

Karmarkar N.K. (1984). A new polynomial time algorithm for linear programming, *Combinatorica* **4**, pp. 373-395.

Karmarkar N.K., Ramakrishnan K.G. (1989). Implementation and computational results of the Karmarkar algorithm for linear programming using an iterative method for computing projections, Technical Memorandum, AT&T

Bell Laboratories, Murray Hills, September.

Kojima M., Mizuno S., Yoshise A. (1986). A primal-dual interior point algorithm for linear programming, in: N. Megiddo ed., *Progress in Mathematical Programming*, Springer-Verlag, New York, pp. 29-48.

Lisser A. and Tolla P. (1989). Variants of Karmarkar's algorithm, Technical Report 90, LAMSADE, University of Paris Dauphine, Paris, March.

Lustig I. (1991). Feasibility issues in a primal-dual interior point method for linear programming, *Mathematical Programming* **49**, pp. 145-162.

Marsten R.E. (1981). The design of XMP linear programming library, *ACM Transactions on Mathematical Software* **7**, pp. 481-497.

Marsten R., Submaranian R., Saltzman M., Lustig I., Shanno D. (1990). Interior point methods for linear programming: just call Newton, Lagrange and Fiacco and McCormick, *Interfaces* **20**, pp. 105-116.

McShane K.A., Monma C.L., Shanno D.F. (1989). An implementation of a primal-dual interior point method for linear programming, *ORSA Journal on Computing* **1**, pp. 70-83.

Megiddo N. (1986). Pathways to the optimal set in linear programming, in: N. Megiddo ed., *Progress in Mathematical Programming*, Springer-Verlag, New York, pp. 131-158.

Monteiro R.C., Adler I. (1989). Interior path following primal-dual algorithms - part I: linear programming, *Mathematical Programming* **44**, pp. 27-41.

Murtagh B. (1981). Advanced Linear Programming, Computation and Practice, McGrew-Hill, New York.

Reid J.K. (1982). A sparsity exploiting variant of the Bartels-Golub decomposition for linear programming bases, *Mathematical Programming* **24**, pp. 55-69.

Tachat D. (1991). Interior point methods for linear programming: computing exact and approximate projections, Ph.D. Thesis, LAMSADE, University of Paris Dauphine, Paris 1991 (in French).

Tolla P. (1987). 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.

Vial J.-P. (1987). A unified approach to projective algorithms for linear programming, Technical Report, Departement d'Economie Commerciale et Industrielle, University of Geneva, Geneva, September 1987.

THE DESIGN AND APPLICATION OF THE IPMLO - A FORTRAN LIBRARY FOR LINEAR OPTIMIZATION WITH INTERIOR POINT METHODS[1]

Jacek Gondzio[2]

Systems Research Institute, Polish Academy of Sciences,

Newelska 6, 01-447 Warsaw, Poland

Dominique Tachat

LAMSADE, University of Paris - Dauphine,

Place du Marechal de Lattre de Tassigny, 75775 Paris Cedex 16, France

Received : 4/9/91

December 19, 1991

---

[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 IFIP Conference on Systems Modelling and Optimization in Zurich, September 2-6, 1991.

[2] the results discussed in the paper have been obtained when this author was staying at LAMSADE, University of Paris Dauphine, Place du Marechal De Lattre de Tassigny, 75775 Paris Cedex 16, FRANCE.