Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (18): 3402-3406, 2009
ISSN 1812-5654

Efficient Hybrid Algorithm for Solving Large Scale Constrained Linear Programming Problems

[1]H. Navidi, [2]A. Malek and [3]P. Khosravi
[1]Department of Mathematics, Shahed University, P.O. Box 18151-159, Tehran, Iran
[2]Department of Mathematics, Tarbiat Modares University, P.O. Box 14115-175, Tehran, Iran
[3]Department of Mathematics, Shahed University, Tehran, Iran

Abstract: The aim of this study is to find an exact least 2-norm solution to a primal constrained linear programming problem in the standard form. Moreover, we can generate an exact solution to the dual programming problem using the exact least 2-norm solution to the primal problem. The proposed algorithm is suitable for solving linear programming problems with a very large number of variables (10^5) and a very large number of constraints (10^6). The exact least 2-norm solution to the primal problem is based on the minimization of exterior penalty functions dual of the dual programming problem. In practice for minimization, we use generalized Newton method and strong wolf conditions in order to find a corresponding suitable step size. This hybrid algorithm converges to the correct optimal solution independent of the values of the given starting point. It is shown that there are some problems that without using strong wolf conditions cannot be solved. The new algorithm can obtain least solution in comparison with MATLAB. Numerical results for a subset of problems from the Netlib collection and a subset of generated large scale linear programs are given. The hybrid algorithm is easy to implement and computationally very efficient. Comparisons are made with available literature.

Key words: Large scale linear programming, newton method, strong wolf conditions, penalty function, least 2-norm solution

INTRODUCTION

Newton method proposed for the unconstrained minimization of strongly convex, piecewise quadratic function arising from quadratic programs (Mangasarian, 1995). Attention to this approach, Golikov and Evtushenko (2000) investigate the least 2-norm formulation of a linear program and for solving primal LP problem, use exterior penalty function. Minimization of the exterior penalty function provided on an exact least 2-norm solution to the dual problem. Instead of penalty function, Evtushenko-Yu *et al.* (2004) proposed to use augmented lagrangian. They can obtain an exact solution of dual problem and an exact least 2-norm solution of primal problem. Ketabchi *et al.* (2009) used minimization of the augmented lagrangian function with the Armijo step size regulation and they solved large scale linear programming problems. Here we use exterior penalty function of dual for generate the exact least 2-norm solution to the primal problem in the standard form and we help strong wolf conditions for finding step-size in each iteration. Our approach is to find an exact least 2-norm solution of primal problem in the standard form and an exact solution of dual problem. We give encouraging comparative test results with MATLAB on a class of generated large scale linear program in the standard form and a subset of problems from the Netlib collection.

Notation: For a vector x in the n-dimensional real space R^n where $x_{i+} = \max\{0, x_i\}$, I = 1, ..., n, replaces negative components of the vector x, by zero and $x_{s^k} = (x_{1^*}, ..., x_{n^*})$ where, x_{i^*} if $x_i>0$, x_{i^*} if $x_i>0$, $x_{i^*} = 0$ if $x_{i^*}\leq 0$, i = 1, ..., n. The 2-norm of x will be denoted by $\|x\|$ and for a matrix $A\epsilon R^{m\times n}$, $\|A\|$ is the 2-norm of A. a_i is the ith column of A and A_S is the sub matrix of A consisting of columns a_i where, $i\epsilon S = \{i = 1, ..., n\}$.

THE EXACT LEAST 2-NORM SOLUTION

Consider the primal linear programming problem in the standard form:

$$\text{Min } c^T x$$
$$\text{s.t.} \quad Ax = b, x\geq 0 \qquad (1)$$

Corresponding Author: Hamidreza Navidi, Department of Mathematics, Shahed University, P.O. Box 18151-159, Tehran, Iran
Tel:+ 982151212629 Fax: +982151212601

Together with its dual:

$$\begin{aligned} &\text{Min} \quad -b^T u \\ &\text{s.t.} \quad A^T u \le c \end{aligned} \tag{2}$$

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

Let us denote by $L = \inf \{c^T x: Ax = b, x \ge 0\}$ the optimal value of the primal Problem (1). This study aims to develop an exact least 2-norm solution of the primal problem (1). The minimization problem corresponding to it is; Find x that satisfies the following problem:

$$\begin{aligned} &\text{Min} \quad \frac{1}{2} x^T x \\ &\text{s.t} \quad c^T x = L, Ax = b, x \ge 0 \end{aligned} \tag{3}$$

The standard method for finding a minimum norm solution of a convex program is based on the Tikhonov regularization (Luca *et al*., 1996). In order to solve (3), the Tikhonov regularization generates a sequence of iterates $\{x_k\}$, with x_k being the unique solution of the regularized program:

$$\begin{aligned} &\text{Min} \quad c^T x + \frac{\varepsilon_k}{2} x^T x \\ &\text{s.t.} \quad Ax = b, x \ge 0 \end{aligned} \tag{4}$$

where, ε_k and the sequence $\{\varepsilon_k\}$ tends to zero. Due to the special properties of linear programs, it is known that a solution of a single quadratic program (4) with a sufficiently small but positive parameter ε_k already gives a solution of (3), (Fischer and Kanzow, 1996). Thus, from now we focus to the solution of the following program:

$$\begin{aligned} &\text{Min} \quad c^T x + \frac{\varepsilon_k}{2} x^T x \\ &\text{s.t.} \quad Ax = b, x \ge 0 \end{aligned} \tag{5}$$

Since, the objective function of (5) is strongly convex, its solution is unique. The necessary and sufficient Karush-Kuhn-Tucker optimality conditions for problem (5) are that $\exists w \in R_m$ such that:

$$\varepsilon x = (A^T w - c)_+, Ax - b = 0 \tag{6}$$

or equivalently:

$$x = \frac{1}{\varepsilon}(A^T w - c)_+ \tag{7}$$

$$-\varepsilon b + A(A^T w - c)_+ = 0 \tag{8}$$

Motivated by (8), let us introduce the function:

$$f(w) = -\varepsilon b^T w + \frac{1}{2}\left\|(A^T w - c)_+\right\|^2 \tag{9}$$

That f(w) is exterior penalty function of the dual linear program (2). Thus, the optimally conditions (7, 8) are in the form:

$$x = \frac{1}{\varepsilon}(A^T w - c)_+, \nabla f(w) = -\varepsilon b + A(A^T w - c)_+ = 0 \tag{10}$$

The primal least 2-norm solution corresponding to (10) is: Find $x \in R^m$ such that:

$$x = \frac{1}{\varepsilon}(A^T w - c)_+, \forall w \in \arg\ \min(-\varepsilon b^T w + \frac{1}{2}\left\|(A^T w - c)_+\right\|^2) \tag{11}$$

which is precisely the necessary and sufficient condition that make w to be a minimum solution of the parametric exterior penalty function f(w) for the dual program (2). Hence, minimization of the function f(w) provides a solution: $x = 1/2(A^T w\text{-}c)_+$ to '(5)' If in addition $\varepsilon \in (0, \bar{\varepsilon}]$ for some positive $\bar{\varepsilon}$ it follows by (Mangasarian and Meyer, 1979), that x is a least 2-norm solution to the primal linear program (1). The following theorem provides a criterion for the exact least 2-norm solution.

Theorem 1: The exact least 2-norm solution of linear program (1) is obtained by $x = 1/2(A^T w\text{-}c)$, for any positive $\varepsilon \in (0, \bar{\varepsilon}]$ such that ϵ for some positive $\bar{\varepsilon}$, where, w is a solution of the dual penalty problem:

$$\text{Min} f(w) = \min(-\varepsilon b^T w + \frac{1}{2}\left\|(A^T w - c)_+\right\|^2)$$

AN EXACT SOLUTION TO THE DUAL PROGRAM

A minimum solution of the exterior penalty function f(w) called w, will always locate in the exterior of the boundary of feasible region for dual program (2) (Rao, 1984). In order to generate an exact solution to the dual program (2), we must force w to locate in the boundary of the corresponding. For this to happen we must find a vector $v \in R^m$ such that:

$$(a_j)^T v - c_j = 0, j \in \{j: (a_j)^T w - c_j > 0\}$$

Equivalently we can write:

$$(A_s)^T v = c_s, S = \{j: ((a_j)^T w - c_j)_+ > 0\}$$

From (7), since $x_j = 1/\varepsilon\ ((a_j)^T\ w\text{-}cj)_+$ it follows that:

$$(A_s)^T v = c_s, S = \{j : x_j > 0\} \quad (12)$$

Therefore, the exact least 2-norm solution x for the primal linear program (1) can be utilized to generate an exact solution to the dual program (2) by solving (12) and (12) yields an exact solution of the dual program (2) if we make the additional assumption that the sub matrix A_S has linearly independent rows.

GENERALIZED NEWTON METHOD

For the piecewise quadratic function f_w the ordinary Hessian does not exist because its gradient:

$$\nabla f(w) = -\varepsilon b + A(A^T w - c)_+$$

is not differentiable. However, one can define its generalized Hessian (Mangasarian, 2002), which is the m×m positive semi-definite symmetric matrix:

$$\delta^2 f(w) = A.diag(A^T w - c)_*.A^T \quad (13)$$

where, diag(A^T w-c) denotes a n×n diagonal matrix with diagonal elements $((a_i)^T w - c_i)_*$, i = 1, ..., n. The generalized Hessian (13) has many properties of the regular Hessian (Mangasarian, 2002). The matrix $\delta^2 f(w)$ will be used to generate the Newton direction, since the generalized Hessian may be singular. In fact the direction that will be used is the following one:

$$d = -(\delta^2 f(w) + \sigma I)^{-1} \nabla f(w)$$

where, σ is some small positive number.

ALGORITHM

In order to guarantee global convergence, we utilize a strong wolf step-size:

- Choose $w^0 \in R^m$ and give ϵ tol and σ small positive numbers
- For $d^i = -(\delta^2 f(w^i) + \sigma I)^{-1} \nabla f(w^i)$ and the strong wolf step-size a_i:

$$a_i = \max\{\frac{1}{2^k} : k \in N, f(w^{i+1}) - f(w^i) \le \frac{1}{4}\alpha_i \nabla f(w^i)^T d^i, \left|\nabla f(w^{i+1})^T d^i\right| \le \frac{1}{2}\left|\nabla f(w^i)^T d^i\right|\}$$

find $w^{i+1} = w^i + \alpha_i d^i$

- If $w^{i+1} = w^i + \alpha_i d^i$, stop
- Put $x = 1/\varepsilon(A^T w^{i+1} - c)$ and find a solution v from (12)

We state a convergence result for this algorithm now.

Theorem 2: Each accumulation point w^* of the sequence w^i generated by above algorithm solves the exterior penalty problem:

$$Min f(w) = Min(-\varepsilon b^T w + \frac{1}{2}\left\|(A^T w - c)_+\right\|^2)$$

The corresponding x^* obtained by setting w to w^i in (7) is the exact least 2-norm solution to the primal program (1), provided ε is sufficiently small. An exact solution v^* to the dual linear program (2) is obtained from (12), provided that the sub matrix A_S of A has linearly independent rows (Mangasarian, 1995).

NUMERICAL TEST

Here, we present some numerical results for a subset of the problems given in the Netlib collection (www.netlib.com) and a subset of randomly generated problems. For the latter we use a MATLAB m-file code1 that is a linear programming test problem generator:

m = input(m:);n = input(n:); d = input(d:)
pl = inline(abs(x)+x)/2)
a = sprand(m, n, d); A = 100*(A-0.5* spones(A))
x = spdiags((sign)pl(rand(m, 1)-rand(m.1)))), 0, m, m* (rand(m, 1))-rand(m, 1:)
b = A* x
c = A'*u=spdiags((ones(n, 1)-sign(pl(x))), 0, n, n)* 10* ones (n, 1)

To test the proposed algorithm we solved a number of linear programs generated by LP generator, a 3.06 GHz Pentium IV machine with 1 GB of memory running Windows XP-SP2 and compared it with Linprog in MATLAB 7.6. The starting vector used in the all following examples is $W^0 = 0$. Results are presented in Table 1 where m and n show the size of the matrix A and d stands for its density. The total time of computations in each example is given in the last column of the Table. $\|Ax^*-b\|$, $\|(-x^*)_+\|$, $\|(A^tv^*-c)_+\|$ show the accuracies of feasible conditions of LP problems and $|b^Tv^*-c^Tx^*|$ shows the accuracies of optimality conditions of LP problems. Analysis of the Table1 shows that proposed algorithm in this paper finds normal solutions for large scale problems. It is also shown that for some problems, Linprog gives no solution, where it states Out Of Memory (O.O.M).

Table 2 shows that strong Wolf conditions are an important toll in this algorithm. For some problems, the proposed algorithm without strong wolf conditions does not work well. We present a number of these problems in Table 2.

Various problems from Netlib solved by proposed algorithm and Linprog in MATLAB. The comparison between two methods is given in Table 3.

In the study, we use penalty function for finding least 2-norm solution to the primal problem. Problems were solved by fast generalized Newton method and we

Table 1: Comparison between the proposed algorithm and the *Linprog* solver for various randomly generated problem for different size and densities

(m, n, d)	Method	$\|b^Tv^*-c^Tx^*\|$	$\|x^*\|$	$\|Ax^*-b\|$	$\|(-x^*)_+\|$	$\|(A^Tv^*-c)_+\|$	Time (sec)
$(10^5, 10^6, 10^{-6})$	New algorithm	3.1×10^{-7}	$1.47\times10^{+7}$	8.9×10^{-7}	0	1.4×10^{-17}	971
	Linprog	7.1×10^{-7}	$1.63\times10^{+4}$	1.5×10^{-11}	0	1.6×10^{-11}	32
$(10^4, 10^6, 10^{-5})$	New algorithm	1.3×10^{-8}	$6.76\times10^{+2}$	9.1×10^{-9}	0	3.1×10^{-17}	76
	Linprog	3×10^{-6}	1.90×10^{64}	3.1×10^{-11}	0	1.1×10^{-9}	27
$(10^3, 10^6, 10^{-3})$	New algorithm	1.5×10^{-7}	$2.54\times10^{+2}$	1.7×10^{-7}	0	1.5×10^{-10}	893
	Linprog	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
$(10^2, 10^6, 10^{-2})$	New algorithm	1.2×10^{-7}	$7.93\times10^{+1}$	5.3×10^{-7}	0	9.1×10^{-11}	38
	Linprog	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
$(10^4, 10^5, 10^{-4})$	New algorithm	6×10^{-9}	$6.58\times10^{+2}$	1.2×10^{-8}	0	1.5×10^{-11}	912
	Linprog	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
$(10^3, 10^5, 10^{-2})$	New algorithm	6.2×10^{-8}	$2.35\times10^{+2}$	2×10^{-7}	0	7.6×10^{-11}	483
	Linprog	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
$(10^2, 10^5, 10^{-6})$	New algorithm	2.2×10^{-8}	$7.6\times10^{+1}$	6.1×10^{-8}	0	9.6×10^{-12}	30
	Linprog	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M
$(10^3, 10^4, 10^{-1})$	New algorithm	2.1×10^{-7}	$2.22\times10^{+2}$	6.5×10^{-7}	0	2.7×10^{-11}	75
	Linprog	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M	O.O.M

Table 2: Comparison between the proposed algorithm without strong wolf and with strong wolf for various randomly generated problem for different size and densities

(m, n, d)	Method	$\|b^Tv^*-c^Tx^*\|$	$\|Ax^*-b\|$	$\|(-x^*)_+\|$	$\|(A^Tv^*-c)_+\|$
$(10^5, 1.5\times10^5, 1/m)$	With strong wolf	2.6×10^{-5}	6.6×10^{-4}	0	3.3×10^{-13}
	Without strong wolf	$5.1\times10^{+8}$	$1.8\times10^{+9}$	0	$4.5\times10^{+1}$
$(4\times10^4, 6\times10^4, 1/m)$	With strong wolf	1.7×10^{-4}	1×10^{-3}	0	1.4×10^{-13}
	Without strong wolf	$4.4\times10^{+7}$	$1.4\times10^{+8}$	0	$1.2\times10^{+1}$
$(10^4, 2\times10^4, 1/m)$	With strong wolf	1.5×10^{-4}	1.8×10^{-4}	0	$1.1\times10^{+1}$
	Without strong wolf	$1.1\times10^{+8}$	2.8×10^{-8}	0	$3.9\times10^{+1}$
$(4\times10^3, 1.2\times10^4, 1/m)$	With strong wolf	1.7×10^{-8}	6.4×10^{-8}	0	$1.9\times10^{+13}$
	Without strong wolf	$1.3\times10^{+3}$	$1.7\times10^{+9}$	0	1.7×10^{-13}

Table 3: Comparison between the proposed algorithm and the *Linprog* solver for various problems from Netlib

Program	Method	$\|b^Tv^*-c^Tx^*\|$	$\|x^*\|$	$\|Ax^*-b\|$	$\|(-x^*)_+\|$	$\|(A^Tv^*-c)_+\|$	Time (sec)
beaconfd	New algorithm	8.2×10^{-2}	$5.5352\times10^{+3}$	6.1×10^{-8}	0	6.4×10^{-4}	6
	Linprog	2.7×10^{-4}	$3.0266\times10^{+5}$	1.2×10^{-10}	0	1.5×10^{-6}	1
capri	New algorithm	3.2×10^{-8}	$1.3717\times10^{+4}$	5.6×10^{-6}	0	2.8×10^{-14}	67
	Linprog	4.7×10^{-7}	$2.8410\times10^{+4}$	5.8×10^{-12}	0	5×10^{-11}	1
fit1p	New algorithm	1.1×10^{-6}	$8.1068\times10^{+2}$	7.5×10^{-3}	0	5.8×10^{-7}	113
	Linprog	1.5×10^{-6}	$8.1096\times10^{+2}$	5.7×10^{-10}	0	2.1×10^{-7}	3
fit2p	New algorithm	0	$5.0863\times10^{+4}$	1.5×10^{-6}	0	0	320
	Linprog	1.9×10^{-9}	$6.3890\times10^{+4}$	1×10^{-10}	0	8.7×10^{-10}	57
gfrd-pnc	New algorithm	0	$9.0058\times10^{+3}$	0	0	0	0.04
	Linprog	5.1×10^{-9}	$9.0058\times10^{+3}$	1.5×101^{-12}	0	0	1
lotfi	New algorithm	7.5×10^{-9}	$1.6253\times10^{+4}$	2.5×10^{-6}	0	3.8×10^{-2}	5
	Linprog	No solve $1.4\times10^{+2}$	$3.5015\times10^{+4}$	8.4×10^{-9}	0	4.8×10^{-2}	3
marosr7	New algorithm	6.2×10^{-8}	$1.5264\times10^{+5}$	4.4×10^{-8}	0	1.4×10^{-15}	6834
	Linprog	1.2×10^{-3}	$1.5264\times10^{+5}$	1.7×10^{-10}	0	1.3×10^{-8}	62
pilot	New algorithm	6.2×10^{-3}	$3.6169\times10^{+5}$	8.3×10^{-7}	0	9.8×10^{-2}	26
	Linprog	No solve $2.1\times10^{+6}$	$1.0526\times10^{+5}$	3.5×10^{-11}	0	2.4	75
pilot87	New algorithm	2.9×10^{-1}	$6.2928\times10^{+3}$	5.9×10^{-4}	0	5.6×10^{-1}	453
	Linprog	No solve $9\times10^{+2}$	$2.6915\times10^{+5}$	1.6×10^{-8}	0	2.4×10^{-5}	567
sc105	New algorithm	3.7×10^{-12}	$2.1874\times10^{+3}$	2.5×10^{-9}	0	1.4×10^{-2}	0.4
	Linprog	6.9×10^{-8}	$2.1874\times10^{+3}$	2.1×10^{-13}	0	1.7×10^{-9}	0.7
sc205	New algorithm	6.4×10^{-7}	$8.8483\times10^{+3}$	4.5×10^{-6}	0	1.4×10^{-2}	2
	Linprog	7.2×10^{-8}	$8.8483\times10^{+3}$	1.2×10^{-12}	0	4.4×10^{-11}	1
scagr7	New algorithm	7.2×10^{-1}	$1.549\times10^{+4}$	6.6×10^{-5}	0	2.8×10^{-3}	10
	Linprog	2×10^{-2}	$1.5531\times10^{+4}$	2.9×10^{-10}	0	3.3×10^{-7}	1
share2b	New algorithm	4.6×10^{-1}	$1.6238\times10^{+2}$	2.8×10^{-7}	0	1.3×10^{-1}	4
	Linprog	5.1×10^{-7}	1.7715×10^{-10}	6.5×10^{-10}	0	1.1×10^{-11}	1
shell	New algorithm	0	$6.4072\times10^{+4}$	0	0	0	0.04

Table 3: Continued

Program	Method	$\|b^Tv^*-c^Tx^*\|$	$\|x^*\|$	$\|Ax^*-b\|$	$\|(-x^*)_+\|$	$\|(A^Tv^*-c)_+\|$	Time (sec)
	Linprog	3.2×10^{-15}	$6.4072\times10^{+4}$	5.3×10^{-20}	0	0	2
ship04s	New algorithm	1.8×10^{-2}	$2.5885\times10^{+2}$	7×10^{-7}	0	2.9×10^{-12}	19
	Linprog	1.6×10^{-4}	$2.5885\times10^{+2}$	1.4×10^{-8}	0	4.3×10^{-11}	4^4
ship08s	New algorithm	2.3×10^{-2}	$1.9853\times10^{+2}$	4.4×10^{-7}	0	3.8×10^{-1}	15
	Linprog	9.9×10^{-4}	$1.9860\times10^{+2}$	1.1×10^{-11}	0	0	5
ship121	New algorithm	1.4×10^{-1}	$1.2982\times10^{+2}$	2.1×10^{-6}	0	7.2×10^{-2}	140
	Linprog	7.5×10^{-3}	$1.2983\times10^{+2}$	4.1×10^{-9}	0	3.9×10^{-9}	25
ship12s	New algorithm	1×10^{-1}	$2.4339\times10^{+2}$	1.5×10^{-6}	0	1.5×10^{-1}	110
	Linprog	4.9×10^{-6}	$2.4339\times10^{+2}$	2.2×10^{-11}	0	5.2×10^{-10}	5
Stair	New algorithm	6.8×10^{-1}	$4.1603\times10^{+2}$	2.7×10^{-12}	0	7.3×10^{-1}	1
	Linprog	2.8×10^{-7}	$7.5866\times10^{+3}$	7.6×10^{-10}	0	3.3×10^{-10}	2
standata	New algorithm	0	$2.5981\times10^{+3}$	7.2×10^{-6}	0	0	0.2
	Linprog	1×10^{-8}	$2.5981\times10^{+3}$	7.2×10^{-13}	0	6.6×10^{-12}	4
standgub	New algorithm	0	$2.5981\times10^{+2}$	7.2×10^{-6}	0	0	0.2
	Linprog	2.3×10^{-12}	$2.5981\times10^{+3}$	$1.8.\times10^{-12}$	0	2.8×10^{-15}	3
tuff	New algorithm	0	0	0	0	0	0.004
	Linprog	6.6×10^{-13}	$9.1350\times10^{+2}$	1.1×110^{-13}	0	1.5×10^{-11}	2
Woodw	New algorithm	4.4×10^{-8}	1.1600	3.9×10^{-13}	0	3.6×10^{-1}	249
	Linprog	1.3×10^{-8}	1.2470	6.1×10^{-10}	0	1×10^{-8}	536

investigated its finite global convergence with the strong wolf step size regulation. The method is analyzed for solution and convergence. A subset of problems from the Netlib collection and a subset of generated large scale linear programs are solved using this method. Comparison between two methods shows that proposed algorithm in this study finds normal solutions for large scale problems and Netlib problems. It is also shown that for some problems, MATLAB gives no solution, where it states out of memory.

REFERENCES

Evtushenko-Yu, G., A.I. Golikov, S. Ketabchi and N. Mollaverdi, 2004. Augmented Lagrangian Methods for Linear Programming. In: Dynamics of Non-Homogeneous Systems, Popkov, Y.S. (Ed.). Russian Academy of Sciences Institute for System Analysis, Russia, pp: 101-106.

Fischer, A. and C. Kanzow, 1996. On finite termination of an iterative method for linear complementarity problems. Math. Programm., 74: 279-292.

Golikov, A.I. and Y.G. Evtushenko, 2000. Searching for normal solutions in linear programming problems. Comput. Math. Math. Phys., 40: 1694-1714.

Ketabchi, S., H. Navidi and N. Mollaverdi, 2009. Modified Lagrangian function and Armijo rule for normal solution of the LP problem. Far East J. Applied Math., 35: 103-111.

Luca, T.D., F. Facchinei and C. Kanzow, 1996. A semismooth equation approach to the solution of nonlinear complementarily problems. Math. Programm., 75: 407-439.

Mangasarian, O.L. and R.R. Meyer, 1979. Nonlinear perturbation of linear programs. SIAM J. Control Optimizat., 17: 745-752.

Mangasarian, O.L., 1995. Parallel gradient distribution in unconstrained optimization. SIAM J. Control Optimizat., 33: 1916-1925.

Mangasarian, O.L., 2002. A finite newton method for classification problems. Optimizat. Methods Software, 17: 913-929.

Rao, S.S., 1984. Optimization: Theory and Applications. 2nd Edn., Wiley Eastern, New York, ISBN: 0-470-27483-2.