J. Austral. Math. Soc. Ser. B 41(2000), 312–328

FINAL ITERATIONS IN INTERIOR POINT METHODS – PRECONDITIONED CONJUGATE GRADIENTS AND MODIFIED SEARCH DIRECTIONS

WEICHUNG WANG[1]

(Received 16 May 1997; revised 15 April 1998)

Abstract

In this article we consider modified search directions in the endgame of interior point methods for linear programming. In this stage, the normal equations determining the search directions become ill-conditioned. The modified search directions are computed by solving perturbed systems in which the systems may be solved efficiently by the preconditioned conjugate gradient solver. A variation of Cholesky factorization is presented for computing a better preconditioner when the normal equations are ill-conditioned. These ideas have been implemented successfully and the numerical results show that the algorithms enhance the performance of the preconditioned conjugate gradients-based interior point methods.

1. Introduction

The development of interior point methods has led to many successful implementations that may efficiently solve linear-programming problems

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

where c and x are real n-vectors, b is a real m-vector, and $A \in \mathbb{R}^{m\times n}$ is a real matrix of rank m with $m \leq n$. These methods eliminate the inequalities in (1) by applying a logarithmic barrier function with a barrier parameter μ and then forming the Lagrangian of the barrier subproblem. A sequence of Lagrangians corresponding to a sequence of barrier parameters $\{\mu_k\}$, with μ_k decreasing to zero, are solved for

[1]Department of Mathematics and Science Education, National Tainan Teachers College, Tainan 700, Taiwan

iterates converging to the optimal solution to the linear-programming problem. The first order optimality conditions of the Lagrangian are

$$\begin{cases} XZe - \mu e = 0, \\ c - A^T y - z = 0, \\ Ax - b = 0. \end{cases} \tag{2}$$

The vector z here is a dual slack variable, y contains Lagrangian multipliers, and e is a vector with all 1's. The diagonal matrices X and Z contain x and z in their main diagonals respectively.

Newton's method is used to solve the nonlinear system (2) and the search direction is then determined by solving the KKT (Karush-Kuhn-Tucker) system

$$\begin{pmatrix} X & Z & 0 \\ -I & 0 & A^T \\ 0 & A & 0 \end{pmatrix} \begin{pmatrix} \Delta z \\ \Delta x \\ -\Delta y \end{pmatrix} = - \begin{pmatrix} Xz - \mu e \\ c - A^T y - z \\ Ax - b \end{pmatrix}. \tag{3}$$

Equivalently, we may first solve the normal equations

$$(A\Theta A^T)\Delta y = A\Theta(r_d + Ze - \mu X^{-1}e) + r_p, \tag{4}$$

where $r_p = b - Ax$, $r_d = c - A^T y - z$ and $\Theta = Z^{-1}X$, and then compute

$$\Delta x = \Theta(A^T \Delta y - r_d - Ze + \mu X^{-1} e) \tag{5}$$

and

$$\Delta z = r_d - A^T \Delta y. \tag{6}$$

Either direct methods or iterative methods may be used to solve the systems to determine the search directions. The computation of the search directions is the bulk of the computational effort for interior point methods, and thus accelerating this computation is a key problem.

In this article, we focus on solving the normal equations by the preconditioned conjugate gradient method for determining the search directions. Good preconditioners are necessary to make it competitive, but they are difficult to find: the requirements of accuracy for beginning and later stages are greatly different, the matrix Θ may change wildly, and Θ becomes very ill-conditioned when iterates become close to an optimum. To overcome these difficulties, Wang and O'Leary [17] recently proposed an algorithm that adaptively chooses either a direct method, or preconditioned conjugate gradients. They also discussed adaptive preconditioning strategies that either recompute a Cholesky factorization $A\Theta A^T = LPL^T$, where L is an $m \times m$ unit lower triangular matrix and P is diagonal, or apply α rank-1 updates. That is, the current preconditioner is computed as

$$LPL^T + \sum_{\alpha \text{ largest } |\Delta\Theta_{ii}|} \Delta\Theta_{ii} a_i a_i^T,$$

where $\Delta\Theta$ is the difference between the current Θ and the previous $\hat{\Theta}$ satisfying $A\hat{\Theta}A^T = LPL^T$, and a_i is the i-th column of A. The adaptive algorithm switches to a direct method whenever P contains a zero element in its main diagonal. This situation is due to ill-conditioning in Θ and may be found in the endgame of many linear-programming problems. Consequently, though the computational results reported in [17] are promising, there is room for improvement.

We improve the algorithm in [17] by considering modified search directions in the endgame. When the iterates are close to optimal solutions, we perturb small entries in the slack variables z in the left-hand side of (3), so that preconditioned conjugate gradients converge rapidly.

We survey some other related works. Many papers, for example [7, 10, 18], address theoretical and implementation aspects of interior point methods. Direct methods relying on sparse Cholesky factorization were used by Lustig, Marsten, and Shanno (OB1-R) [10], Czyzyk, Mehrotra, and Wright (PCx) [2], Zhang (LIPSOL) [19], and other researchers, to solve the normal equations. Iterative methods, in contrast, were also considered, since iterative methods may take advantage of the fact that approximate solutions are allowed in the early stage of an interior point method. See, for example, Freund and Jarre [4], Portugal, Resende, Veiga, and Júdice [13], and Mehrotra and Wang [11]. Mizuno and Jarre [12] proposed, and further analyzed, an infeasible interior point algorithm using inexact solutions of the reduced KKT system as search directions. On the other hand, many recent studies concentrate on the stability of highly ill-conditioned systems which may be found in the endgame of interior point methods. Hough and Vavasis [8] consider weighted least-squares problems with a highly ill-conditioned weight matrix. They propose a complete orthogonal decomposition algorithm which is stable in the sense that its forward error bound is independent of the matrix Θ. In [3], Forsgren, Gill, and Shinnerl present a perturbation analysis of a class of symmetric diagonally ill-conditioned systems and give a rounding-error analysis for symmetric indefinite matrix factorization.

In the next section, we discuss ideas for perturbing the normal equations to obtain modified search directions, and then propose an algorithm based on the ideas. The modification is closely akin to that proposed by Karmarkar [9] in order to reduce the complexity of his interior point method to $O(n^{2.5})$ by updating a matrix rather than recomputing it. The differences in formulation are that his was a primal algorithm, while ours is primal-dual; our choice of parameters is somewhat different, and we solve the linear systems iteratively, taking advantage of the fact that the modified systems are much easier to solve than the original ones. Section 3 discusses implementation issues for finding the modified search directions. Numerical results are presented in Section 4.

We introduce the following notation to be used throughout the article. Let e be the vector in which all elements are 1's, and let e_i be the vector with all 0's except that the i-th component is equal to 1. Let K denote the matrix $A\Theta A^T$. If C is a square matrix, diag (C) is the vector formed from the main diagonal of C; if v is a vector, diag (v) is a diagonal matrix with the elements of v on the main diagonal.

The variables x_j, y_j and z_j denote the j-th vector in the sequence $\{x_j\}$, $\{y_j\}$ and $\{z_j\}$, respectively. The Greek variable χ_i denotes the i-th component of the vector x_j, where the index of x will be clear from the context, that is, $x_j = (\chi_1, \cdots, \chi_n)^T$. Similarly, we let $y_j = (\eta_1, \cdots, \eta_m)^T$ and $z_j = (\zeta_1, \cdots, \zeta_n)^T$ for $y_j \in \mathbb{R}^m$ and $z_j \in \mathbb{R}^n$.

The solution of (2) for a fixed μ is denoted as $x^*(\mu)$, $y^*(\mu)$ and $z^*(\mu)$. Capital letters X, Y and Z denote diagonal matrices containing vectors x, y and z on the main diagonals respectively. Let $S_Y = \{y \in \mathbb{R}^m \mid \|y\| \leq \Lambda_Y\}$ and $S_Z = \{z \in \mathbb{R}^n \mid 0 < \Omega_Z e \leq z \leq \Lambda_Z e\}$, where Λ_Y, Ω_Z, Λ_Z are positive numbers.

2. Modified search directions for the endgame

We consider the course of the algorithm in the endgame, where iterates x, y, and z are close to the solution of (2). The strict complementarity implies that, for each i, either χ_i or ζ_i is close to zero in the relative interior of a non-singleton solution set (see, for example, [19]). The resulting diagonal matrix Θ, in which the i-th diagonal entry is $\Theta_{ii} = \chi_i/\zeta_i$, consequently contains some very small positive entries and some irregularly distributed large entries corresponding to small ζ_i's. Moreover, these wildly changing entries may cause troubles for the preconditioned conjugate gradient solver using the updated preconditioner. The observation, however, that

$$A\Theta A^T = \sum_{i=1}^{n} \Theta_{ii} a_i a_i^T,$$

suggests that only large diagonal elements in Θ are significant. We further observe that a slight perturbation in the small ζ_i's may result in a significant change in the corresponding large Θ_{ii}'s. The following question is then raised: is it possible to slightly perturb those small ζ_i's, such that the preconditioned conjugate gradient method may benefit? In other words, we hope to find a modified search direction by solving perturbed normal equations where the new system can be easily solved by preconditioned conjugate gradients. At the same time, the outer iterations of the interior point method can still converge and the performance will not be degraded.

The answer to the question is positive and we propose a method to achieve this goal. Let $\hat{\Theta}$ be a previous diagonal matrix for which we have a preconditioner

$C_{\hat{\Theta}} = LPL^T = A\hat{\Theta}A^T$. We may partition the diagonal entries of $\hat{\Theta}$ as $[\hat{\Theta}^B, \hat{\Theta}^S]$, where $\hat{\Theta}^B$ and $\hat{\Theta}^S$ contain the big and small entries in $\hat{\Theta}$ respectively. The matrix Θ is then partitioned compatibly as $[\Theta^B, \Theta^S]$. The main idea is that we slightly perturb the small ζ's so that the resulting perturbed matrix $\overline{\Theta}^B = \kappa\hat{\Theta}^B$. Rather than perturb all the Θ^B entries, however, we may wish to perturb only Θ^{B_1}, the part of Θ^B containing really small ζ's and fairly large χ's. Therefore, the corresponding perturbation sizes remain small. Let

$$\Theta^B = \left[\Theta^{B_1}, \Theta^{B_2}\right]. \tag{7}$$

We may choose $\varepsilon_i \in \mathbb{R}, \forall\, i \in B_1$, such that

$$\bar{\zeta}_i = \zeta_i + \varepsilon_i, \quad \text{where} \quad \bar{\zeta}_i > 0 \tag{8}$$

and

$$\left(\overline{\Theta}^{B_1}\right)_{ii} = \frac{\chi_i}{\bar{\zeta}_i} = \frac{\chi_i}{\zeta_i + \varepsilon_i} = \kappa\left(\hat{\Theta}^{B_1}\right)_{ii}. \tag{9}$$

The perturbed system is then

$$\begin{aligned} A\overline{\Theta}A^T &= \sum \overline{\Theta}^{B_1}_{ii} a_i a_i^T + \sum \Theta^{B_2}_{ii} a_i a_i^T + \sum \Theta^S_{ii} a_i a_i^T \\ &= \kappa(LPL^T) + \sum \Delta\Theta^{B_2}_{ii} a_i a_i^T + \sum \Delta\Theta^S_{ii} a_i a_i^T, \end{aligned} \tag{10}$$

where $\Delta\Theta^{B_2}_{ii} = \Theta^{B_2}_{ii} - \kappa\hat{\Theta}^{B_2}_{ii}$. Using LPL^T as a preconditioner of (10),

$$\begin{aligned} (LPL^T)^{-1}(A\overline{\Theta}A^T) &= \kappa I + (LPL^T)^{-1}\sum \Delta\Theta^{B_2}_{ii} a_i a_i^T \\ &\quad + (LPL^T)^{-1}\sum \Delta\Theta^S_{ii} a_i a_i^T. \end{aligned} \tag{11}$$

If we perturb most of the small ζ's, the second term in (11) will be a small rank matrix and the third term has small rank or norm. Consequently, the preconditioned conjugate gradient method will converge rapidly. For further improving the performance of the preconditioned conjugate gradients, we may apply rank-1 updates on some largest $|\Delta\Theta^{B_2}_{ii}|$'s.

The discussion above leads to a *theoretical* algorithm that solves a sequence of perturbed 3×3 block KKT systems. The j-th in the sequence is

$$\begin{pmatrix} X_j & \overline{Z}_j & 0 \\ -I & 0 & A^T \\ 0 & A & 0 \end{pmatrix} \begin{pmatrix} \Delta z_j \\ \Delta x_j \\ -\Delta y_j \end{pmatrix} = -\begin{pmatrix} X_j z_j - \mu e \\ c - A^T y_j - z_j \\ A x_j - b \end{pmatrix}. \tag{12}$$

The system (12) contains the perturbed matrix $\overline{Z}_j$ and this is the only difference between (3) and (12). In [15], we describe the theoretical algorithm and prove its global convergence. By assuming primal feasibility, (that is, the starting point satisfies $Ax = b$), we further establish the superlinear convergence rate of the iteration in the inner loop of the algorithm [15].

3. Implementation

We now discuss a practical way to implement the idea of using a perturbed system for determining a modified search direction. We consider the perturbed normal equations

$$\left(A\overline{\Theta}A^T\right)\Delta y = A\overline{\Theta}\left(r_d + Ze - \mu X^{-1}e\right) + r_p. \tag{13}$$

The search directions are

$$\Delta x = \overline{\Theta}\left(A^T\Delta y - r_d - Ze + \mu X^{-1}e\right) \tag{14}$$

and

$$\Delta z = r_d - A^T\Delta y. \tag{15}$$

We now focus on how we modify the current diagonal matrix Θ. Suppose that we have a Cholesky factorization of $A\hat{\Theta}A^T = LPL^T$. Recall that our goal is to determine the index set B_1 and the proportionality factor κ such that the corresponding $\Theta^{B_1} = \kappa\hat{\Theta}^{B_1}$. See (7) for the definition of Θ^{B_1}. After the modified matrix $\overline{\Theta}$ has been determined, we update the preconditioner using $\overline{\Theta}$ and then use the preconditioned conjugate gradient method to solve the normal equations involving $A\overline{\Theta}A^T$.

We first set the index set B_1 containing all the large entries of the current Θ. We then find the ratios of $\hat{\Theta}_{ii}$ to Θ_{ii}, for every $i \in B_1$. The mean and the variance of those ratios are computed and κ is assigned as the mean value. If the variance value is small, we calculate $\overline{\Theta}_{ii} = \kappa\Theta_{ii}$, for $i \in B_1$. Otherwise, we take out the indices corresponding to some largest and smallest ratios from B_1 to form a new B_1. The mean value of the ratios corresponding to the new B_1 is re-computed to obtain a new κ and then the current Θ is perturbed using the new κ.

The ideas discussed above are implemented by modifying OB1-R, which considers linear-programming problems with simple upper bounds

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \\ & x + s = u, \\ & x \geq 0, \\ & s \geq 0, \end{aligned} \tag{16}$$

where $u \in \mathbb{R}^n$ contains upper bounds for the entries of x and some of them may be infinite. Since the problem considered by OB1-R is slightly different from our model problem (1), we elaborate the corresponding equations for clarity. The Newton search directions may be determined by solving the equations

$$\Delta y = (A\Theta A^T)^{-1}(A\Theta(\widetilde{\psi}(\mu) + r_d) + (b - Ax)), \tag{17}$$

$$\Delta x = \Theta(A^T\Delta y - (\widetilde{\psi}(\mu) + r_d)), \tag{18}$$

$$\Delta w = \mu S^{-1}e - We - S^{-1}W\Delta s, \tag{19}$$

$$\Delta z = \mu X^{-1}e - Ze - X^{-1}Z\Delta x, \tag{20}$$

$$\Delta s = x + s - u - \Delta x, \tag{21}$$

where $\widetilde{\psi}(\mu) = \mu(S^{-1} + X^{-1})e - (W - Z)e$ and $\Theta^{-1} = (S^{-1}W + X^{-1}Z)$.

The modified search directions, perturbing the diagonal matrix Z to $\overline{Z}$, may be written as

$$\Delta\bar{y} = (A\overline{\Theta}A^T)^{-1}(A\overline{\Theta}(\widetilde{\psi}(\mu) + r_d) + (b - Ax)), \tag{22}$$

$$\Delta\bar{x} = \overline{\Theta}(A^T\Delta y - (\widetilde{\psi}(\mu) + r_d)), \tag{23}$$

$$\Delta\bar{w} = \mu S^{-1}e - We - S^{-1}W\Delta s, \tag{24}$$

$$\Delta\bar{z} = \mu X^{-1}e - Ze - X^{-1}\overline{Z}\Delta x, \tag{25}$$

$$\Delta\bar{s} = x + s - u - \Delta x, \tag{26}$$

where $\overline{\Theta}^{-1} = (S^{-1}W + X^{-1}\overline{Z})$ and $\Delta\bar{w}$ and $\Delta\bar{s}$ are the same as (19) and (21), respectively. All the discussion may be easily extended to the problems (16). However, it is worth mentioning that the perturbed matrix $\overline{Z}$ is needed for determining $\Delta\bar{z}$ and may be computed by solving

$$\overline{\Theta}^{-1} = (S^{-1}W + X^{-1}\overline{Z}),$$

after $\overline{\Theta}$ has been determined.

We now present computational results of some test problems to show the modified search direction may improve performance in the endgame. In other words, in the endgame, we determine the perturbed matrix $\overline{\Theta}$ and then solve (22) to (26) to find the search directions. Table 1 illustrates the performance of the preconditioned conjugate gradient solver in the last μ values. The original and the perturbed normal equations are solved for the problems `pilot` and `pilot87` from the NETLIB collection [5]. The number of preconditioned conjugate gradient iterations and the time for forming and solving the normal equations (in seconds) are compared for both approaches. The preconditioned conjugate gradient solvers use the same stopping criterion. Complete Cholesky factorization is performed to determine the preconditioners at the 73rd and 77th iterations in `pilot` and at the 75th, 79th and 83rd iterations in `pilot87`. All other iterations use updated preconditioners. From the table, we observe that, by perturbing the normal equations, we may improve both the preconditioned conjugate

TABLE 1. Solving the normal equations with and without perturbing

Problem : pilot						
		Outer iteration				
Prtrb.		73	74	75	76	77
Yes	PCG iter.	3	26	38	40	3
	Time (s)	6.86	3.34	4.71	4.96	6.89
No	PCG iter.	3	31	45	57	3
	Time (s)	6.71	3.81	5.38	6.73	6.75

Problem : pilot87										
		Outer iteration								
Prtrb.		75	76	77	78	79	80	81	82	83
Yes	PCG iter.	1	32	50	49	1	22	25	33	1
	Time (s)	22.50	8.51	14.68	17.58	22.50	6.04	6.28	8.26	22.50
No	PCG iter.	1	37	72	100	1	35	43	54	1
	Time (s)	22.51	9.68	20.34	35.00	22.61	9.23	10.80	13.36	22.54

gradient iteration numbers and timing. Furthermore, the number of outer iterations remains the same for using OB1-R.

We combine these ideas in an algorithm to solve the (perturbed) normal equations using preconditioned conjugate gradients. To factor an ill-conditioned matrix, we use a variant of the standard Cholesky factorization. See Algorithm 1 in the Appendix for details.

The preconditioned conjugate gradient solver is used through the whole interior point method, except for the first μ, with one exception. If we factor the matrix $A\Theta A^T$ and preconditioned conjugate gradients converge in more than, for example, 50 iterations, even if the hybrid modified Cholesky factorization is used, we switch to a direct method in the next μ iteration. This situation occurs in the case when the matrix $A\Theta A^T$ is too ill-conditioned to make the refactored Cholesky factors an efficient preconditioner. This is an unusual occurrence: only one problem (`dfl001`) met the criterion among all the problems we tested using all the default parameters; but we include the criterion as a “safe guard” for efficiency.

If the ratio of the last barrier parameter to the current barrier parameter is large near the endgame, we refactor the matrix $A\Theta A^T$ to obtain the preconditioner for the current iteration. Since the barrier parameter is proportional to the duality gap, a large change in the two successive barrier parameters implies a large change in the corresponding duality gaps. In this case, the iterates made a “big” improvement and

TABLE 2. Statistics for the test problems from NETLIB

	LP size and nonzeros			Nonzeros		Density	
Problem	Rows	Columns	Nonzeros	AA^T	L	AA^T	L
More costly problems							
cre-b	7240	72447	256095	194579	940374	.01	.04
cre-d	6476	69980	242646	181670	853300	.01	.04
dfl001	6071	12230	35632	38098	1634257	.00	.09
maros-r7	3136	9408	144848	330472	1195107	.07	.24
pds-06	9881	28655	62524	39061	582158	.00	.01
pds-10	16558	48763	106436	66550	1674872	.00	.01
pilot	1441	3652	43167	59540	193137	.06	.19
Smaller cost problems							
pilot87	2030	4883	73152	115951	421194	.06	.20
80bau3b	2237	9799	21002	9972	40895	.00	.02
d2q06c	2171	5167	32417	26991	165676	.01	.07
d6cube	404	6184	37704	13054	54445	.16	.67
degen3	1503	1818	24646	50178	119403	.04	.11
fit2d	25	10500	129018	296	299	.99	1.00
greenbea	2389	5405	30877	33791	81914	.01	.03
greenbeb	2389	5405	30882	33766	80503	.01	.03
ken-11	14694	21349	49058	33880	118869	.00	.00
ken-13	28632	42659	97246	66586	315642	.00	.00
maros	845	1443	9614	11409	24839	.03	.07
osa-07	1118	23949	143694	52466	54783	.08	.09
osa-14	2337	52460	314760	113843	116160	.04	.04
scfxm3	990	1371	7777	8749	13520	.02	.03
seba	515	1028	4352	51400	53748	.39	.41
ship12l	1042	5427	16170	10673	11137	.02	.02
stoch3	16675	15695	64875	103360	206731	.00	.00
truss	1000	8806	27836	12561	52509	.03	.11
vtp.base	198	203	909	1575	2121	.08	.11
wood1p	244	2594	70215	18046	18082	.61	.61
woodw	1098	8405	37474	20421	47657	.03	.08

TABLE 3. Computational results for the test problems from NETLIB

	IPM ite.		Rel. dual gap		Time		
Problem	OB1-R	Adap2	OB1-R	Adap2	OB1-R	Adap2	Diff
More costly problems							
cre-b	91	91	-.16e-07	-.16e-07	5020.10	4954.07	66.03
cre-d	92	92	-.69e-08	-.56e-08	3872.00	3839.33	32.67
dfl001	98	98	.27e-06	.27e-06	19844.37	14436.65	5407.72
maros-r7	29	29	.31e-09	.31e-09	1952.93	1743.57	209.36
pds-06	102	102	.48e-09	.44e-09	2817.62	2742.75	74.87
pds-10	128	128	.19e-08	.29e-08	19650.00	14320.02	5329.98
pilot	77	77	.71e-08	.61e-08	485.08	388.92	96.16
pilot87	82	83	.94e-08	.16e-08	1948.82	1430.97	517.85
Smaller cost problems							
80bau3b	78	78	.44e-08	.44e-08	46.15	47.15	-1.00
d2q06c	55	55	.25e-08	.46e-08	257.13	245.23	11.90
d6cube	77	77	.67e-06	.85e-08	113.90	110.80	3.10
degen3	30	30	.16e-09	.16e-09	66.22	75.12	-9.90
fit2d	54	54	.21e-08	.21e-08	46.80	49.03	-2.23
greenbea	52	52	-.62e-04	-.62e-04	52.03	53.78	-1.75
greenbeb	74	74	.80e-09	.22e-09	69.15	71.20	-2.05
ken-11	33	33	.16e-09	.16e-09	53.28	68.50	-15.22
ken-13	51	51	.43e-08	.43e-08	244.95	277.92	-32.97
maros	45	45	.11e-08	.12e-08	11.17	11.62	-0.45
osa-07	53	53	.11e-05	.11e-05	80.68	91.08	-10.40
osa-14	55	55	-.81e-06	-.81e-06	191.52	225.02	-33.50
scfxm3	39	39	.21e-08	.21e-08	4.25	4.63	-0.38
seba	30	30	.15e-08	.18e-09	43.05	35.79	7.26
ship12l	26	26	.53e-08	.53e-08	4.15	5.22	-1.07
stoch3	87	87	.70e-09	.70e-09	142.22	159.12	-16.90
truss	30	30	.80e-09	.80e-09	19.55	21.23	-1.68
vtp.base	26	26	.33e-08	.38e-08	0.45	0.58	-0.13
wood1p	18	18	.35e-08	.35e-08	12.95	18.23	-5.28
woodw	37	37	.27e-08	.27e-08	25.30	27.05	-1.75

TABLE 4. Results for the problems keeping artificial variables

	IPM ite.		Rel. dual gap		Time		
Problem	OB1-R	Adap2	OB1-R	Adap2	OB1-R	Adap2	Diff
cre-b	102	103	-.99e-09	.14e-08	5365.32	4157.33	1207.99
cre-d	103	103	-.60e-08	.10e-08	4415.48	3511.63	903.85
pds-06	117	116	.50e-08	.76e-08	3216.82	2470.22	746.60
pds-10	131	131	.69e-08	.32e-08	19978.53	12965.85	7012.68

thus the variables and the current resulting matrix Θ may change widely. The update strategy is thus not suitable.

More algorithmic details are given in the Appendix.

4. Numerical results

We modify OB1-R, by Lustig, Marsten and Shanno [10] to implement our ideas. In Section 4.1, we make comparisons with the computational results of OB1-R. In Section 4.2 we further compare our algorithm with the adaptive algorithm reported in [17] (Adap1).

All the algorithms are coded in FORTRAN using double precision arithmetic. The codes are compiled on a SUN SPARCstation 20 containing 64 megabytes main memory and running SunOS Release 4.1.3. Optimization level `-O3` is turned on for compiling the programs. Numerical experiments are performed on the same platform. The timings reported are CPU time in seconds. Since all three codes use the same preprocessor HPREP, we omit the preprocessing time.

Test problems are chosen from the NETLIB problem collection [5], a standard linear-programming test-problem set. Small problems have a relatively small cost for forming and factoring the coefficient matrix in the normal equations, as mentioned in [4] and [17], so we do not expect an interior point algorithm based on iterative solvers to prevail over a direct solver based algorithm. We consequently run a few small problems (`maros`, `scfxm3`, `seba`, `ship12l`, and `vtp.base`) from the NETLIB collection, but concentrate on the larger problems, those containing more than 25 000 nonzero entries in the coefficient matrix A. The problem `fit2p`, however, is neglected since all three codes fail to solve the problem on our workstation in a reasonable time, due to the problem containing a large dense matrix $A\Theta A^T$. Another large problem set found in the NETLIB site is the "Kennington" problems used by Carolan, Hill, Kennington, Niemi and Wichmann [1]. We also present the problems from the set containing 25 000 to 370 000 nonzero elements in the matrix A.

Table 2 shows the characteristics of the tested problems. The numbers of rows,

TABLE 5. Computational results for Adap1 and our algorithm Adap2

	IPM ite.		Rel. dual gap		Time		
Problem	Adap1	Adap2	Adap1	Adap2	Adap1	Adap2	Diff
Without artificial variables							
maros-r7	29	29	.31e-09	.31e-09	1825.87	1743.57	82.30
pilot87	82	83	.94e-08	.16e-08	1626.55	1430.97	195.58
cre-b	91	91	-.16e-07	-.16e-07	4872.30	4954.07	-81.77
cre-d	92	92	-.69e-08	-.56e-08	3761.92	3839.33	-77.41
pds-06	102	102	.48e-09	.44e-09	2781.30	2742.75	38.55
pds-10	128	128	.19e-08	.29e-08	18718.57	14320.02	4398.55
Keeping artificial variables							
dfl001	98	98	.27e-06	.27e-06	16644.35	14436.65	2207.70
cre-b	102	103	-.10e-08	.14e-08	4472.75	4157.33	315.42
cre-d	103	103	-.60e-08	.10e-08	3698.33	3511.63	186.70
pds-06	116	116	.64e-08	.76e-08	2554.35	2470.22	84.13
pds-10	131	131	.69e-08	.32e-08	13546.32	12965.85	580.47
NET0416	53	53	.50e-09	.45e-09	9265.85	8281.15	984.70

columns, and nonzeros of coefficient matrix A are reported. The numbers are obtained from output of the preprocessor HPREP and may not be identical to the data in [5]. Only the nonzero elements in the lower sub-diagonal part of AA^T and L are counted and tabulated. We calculate the density of the matrix AA^T and L by computing the ratio of the number of nonzeros to the number of the entries in the lower sub-diagonal parts of the matrices.

4.1. Comparison with a direct solver based algorithm Numerical results of OB1-R and our Adap2 code on the NETLIB problems are shown in Table 3. The table indicates the name of the problem and compares the number of μ values needed by the interior point methods for both codes, final relative duality gaps and CPU time in seconds used by both codes. The time differences between the two programs are shown in the last column. Our Adap2 codes are faster for the problems with positive time difference.

Both OB1-R and Adap2 take the same number of μ numbers to achieve similar small relative duality gaps, except on problems `pilot87` and `greenbea`. Adap2 takes one additional μ value in `pilot87`, achieves a slightly smaller relative duality gap, and uses less time. In the problem `greenbea`, both algorithms stop unsuccessfully since they fail to converge with small duality gaps. The problem, as mentioned in [14], is difficult to solve by interior point methods. On the problem `d6cube`, our algorithm attains a duality gap two orders smaller and is a little quicker.

Performance of the two algorithms is similar for the problems taking about four minutes or less. On the costly problems, like `maros-r7` and `pilot87`, our algorithm tends to outperform OB1-R. In the most expensive problems `df1001` and `pds-10`, our algorithm is significantly faster than OB1-R.

We keep the artificial variables, the slack variables of the equality constraints, to prevent rank deficiency on `df1001`. Without doing so, neither method terminates successfully, since the matrix $A\Theta A^T$ is very ill-conditioned. We further observe that Adap2 may benefit from keeping the artificial variables. Keeping the artificial variables ensures that the rows in A are independent, and thus may lead the matrix $A\Theta A^T$ to smaller condition number. Table 4 shows that Adap2 significantly outperforms OB1-R on the costly problems if we keep all artificial variables to prevent rank deficiency. Even if OB1-R eliminates the artificial variables and solves the smaller problems, the cost of Adap2 keeping the artificial variables is still less than that of OB1-R.

4.2. Comparison with the adaptive algorithm We compare the numerical performance of our algorithm (Adap2) with the adaptive algorithm (Adap1) of [17]. The main differences between the two approaches are as follows.

- Adap2 uses modified search directions in the endgame; however, Adap1 does not.
- Adap2 uses the OB1-R Cholesky factorization first until the OB1-R Cholesky factorization fails to generate a good preconditioner, in the sense that the preconditioned conjugate gradient solver does not converge within 5 iterations by using the refactored preconditioner. We then switch to the hybrid modified Cholesky factorization. In contrast, Adap1 uses only the OB1-R Cholesky factorization.
- Adap2 allows zero in the diagonal Cholesky factor P while Adap1 can not handle the situation. Adap2 uses a portion of the modified Cholesky factor by Gill and Murray [6] (see [16, Chap. 5] for details).

Table 5 compares Adap1 and Adap2 in the costly problems that take Adap1 more than 1 500 seconds to solve. Both algorithms perform similarly for other cheaper problems not listed. Adap2 outperforms in all the problems except the problems `cre-b` and `cre-d` without artificial variables. These two problems are not suitable for iterative solvers since the Θ's are ill-conditioned and change wildly in the first μ values. Consequently, Adap2 detects two successive μ values at which the number of preconditioned conjugate gradient iterations exceed the maximum number of iterations allowed. We thus decide to use a direct method at the 9th and 14th μ value in the problem `cre-b` and `cre-d`, respectively. In contrast, Adap1 detects zero in the diagonal Cholesky factor P in the second μ value and thus switches to a direct method for the two problems.

For the problems that switch to a direct method in the later phases in Adap1 (for example, `df1001` and `pds-10`), Adap2 achieves significant saving.

5. Conclusion

We have presented an algorithm using the preconditioned conjugate gradient solver through the whole process of interior point methods for linear programming problems. If the algorithm recomputes the preconditioners in later phases, we adopt the hybrid modified Cholesky factorization as an alternative to the Cholesky factorization used by OB1-R. The hybrid modified Cholesky factorization generates a more efficient preconditioner. We modify the preconditioned conjugate gradient solver and rank-1 update and downdate procedure to handle a zero component in the diagonal Cholesky factor P.

In the endgame, we perturb the diagonal matrix Θ for determining modified search directions. The resulting coefficient matrix $A\Theta A^T$ is thus more closely related to the preconditioner. We discuss the motivation of the modified search directions. Numerical results show that the algorithms enhance the performance of OB1-R and the adaptive algorithm in [17].

Acknowledgments

I thank Dianne P. O'Leary for many helpful discussions and useful suggestions during the preparation of this article. I am also grateful to the referee whose valuable comments helped me to improve the content of this paper. This work was supported in part by the National Science Council under grant NSC-88-2115-M-024-001.

Appendix

ALGORITHM 1. (A hybrid modified Cholesky factorization)

```
Initialize pii_tiny $\leftarrow$ a positive tiny number.
do ($i = 1 : m$)
    Determine $P_{ii}$ by the modified Cholesky factorization in [6].
    if ($P_{ii} \leq$ pii_tiny)
        Set $P_{ii} = 0$.
        Skip computation of the $i$th column of $L$.
    else
        Determine the $i$th column of $L$ using $P_{ii}$.
    end if
end do
```

ALGORITHM 2. (Interior point algorithm with adaptive solver)

```
Initialize $k \leftarrow 1$; $\mu_0 > 0$; $x_0, y_0, z_0 > 0$; UseDirect $\leftarrow$ False.
while (not convergent)
    if [($k > 1$) and (UseDirect = False)] then
        Solve using PCG. (See Algorithm 3 for details.)
        Determine the preconditioner.
        Iterate the PCG method using Algorithm 3.
    end if
    if [($k = 1$) or (UseDirect = True)] then
        Solve using direct solver.
        Form the matrix $A\Theta A^T$.
        Factor $A\Theta A^T = LPL^T$.
        Solve the normal equations using $LPL^T$,
                applying iterative refinement if necessary.
        Compute drct_cost as the elapsed time of the direct solver.
    end if
        Update the primal and dual variables.
        Compute  $x_{k+1} \leftarrow x_k + \alpha_p \Delta x$; $y_{k+1} \leftarrow y_k + \alpha_d \Delta y$;
                 $z_{k+1} \leftarrow z_k + \alpha_d \Delta z$.
    Choose $\mu_{k+1} < \mu_k$.
    Set $k \leftarrow k + 1$.
end while
```

ALGORITHM 3. (The modified PCG solver)

```
Solve using PCG

Determine the preconditioner
updt_nmbr ← the number of rank-one updates to be performed.
pred_cost ← predicted cost of iterative solution.
if [(prev_cost > 0.8× drct_cost) or (pred_cost > drct_cost) or
(the ratio of $\mu_{k-1}$ to $\mu_k$ is large near endgame)] then
    UseRefact ← True.
    Form the matrix $A\Theta A^T$.
    if [UseStdChol = True] then
        Factor $A\Theta A^T$ using the standard Cholesky algorithm.
    else
        Factor $A\Theta A^T$ using the hybrid Cholesky algorithm (Algorithm 1).
    end if
else
    UseRefact ← False.
    Compute the modified diagonal matrix $\overline{\Theta}$.
    Perform updt_nmbr rank-one updates based on $A\overline{\Theta}A^T$.
end if

Iterate the PCG method
pcg_itn ← 0.
until (convergent)
    Execute a PCG iteration using the modified $A\overline{\Theta}A^T$.
    pcg_itn ← (pcg_itn + 1).
    if (pcg_itn > max_pcg_itn) then
        if (this happened for the previous $\mu$) then (UseDirect ← True).
        UseRefact ← True.
        Factor $(A\Theta A^T)$ to reinitialize the preconditioner.
        Restart the PCG iteration.
    end if
end until
if [(UseRefact = True) and (pcg_itn ≥ 5)] then
    UseStdChol ← False.
else if [(UseRefact = True) and (pcg_itn ≥ 50)] then
    UseDirect ← True.
end if
```

References

[1] W. Carolan, J. Hill, J. Kennington, S. Niemi and S. Wichmann, "An empirical evaluation of the KORBX algorithms for military airlift applications", *Operations Research* **38** (2) (1990) 240–248.

[2] J. Czyzyk, S. Mehrotra and S. J. Wright, "PCx user guide", Technical Report OTC 96/01, Optimization Technology Center, Argonne National Laboratory and Northwestern University, 1996.

[3] A. Forsgren, P. E. Gill and J. R. Shinnerl, "Stability of symmetric ill-conditioned systems arising in interior methods for constrained optimization", *SIAM J. Matrix Anal. Appl.* **17** (1996) 187–211.

[4] R. W. Freund and F. Jarre, "A QMR-based interior-point algorithm for solving linear programs", Technical Report, AT&T Bell Laboratories and Institut für Angewandte Mathematik und Statistik, 1995.

[5] D. M. Gay, "Electronic mail distribution of linear programming test problems", *Mathematical Programming Soc. COAL Newsletter* (1985).

[6] P. E. Gill and W. Murray, "Newton-type methods for unconstrained and linearly constrained optimization", *Mathematical Programming* **7** (1974) 311–350.

[7] C. C. Gonzaga, "Path-following methods for linear programming", *SIAM Review* **34** (2) (June 1992) 167–224.

[8] P. D. Hough and S. A. Vavasis, "Complete orthogonal decomposition for weighted least squares", Center of Applied Mathematics and Department of Computer Science, Cornell University, May 1996.

[9] N. K. Karmarkar, "A new polynomial-time algorithm for linear programming", *Combinatorica* **4** (1984) 373–395.

[10] I. J. Lustig, R. E. Marsten and D. F. Shanno, "Computational experience with a primal-dual interior point method for linear programming", *Linear algebra and its application* **152** (1991) 191–222.

[11] S. Mehrotra and J.-Sh. Wang, "Conjugate gradient based implementation of interior point methods for network flow problems", Technical Report 95-70.1, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208-3119, U.S.A., October 1995.

[12] Sh. Mizuno and F. Jarre, "Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation", Technical Report Research Memorandum 605, The Institute of Statistical Mathematics, Tokyo, Japan, April 1996.

[13] L. F. Portugal, M. G. C. Resende, G. Veiga and J. J. Júdice, "A truncated primal-infeasible dual-feasible network interior point method", November 1994.

[14] R. J. Vanderbei, "LOQO : An interior point code for quadratic programming", Program in Statistics and Operations Research, Princeton University. `rvdb@princeton.edu`, 1995.

[15] W. Wang, "Final iterations in interior point methods — preconditioned conjugate gradients and modified search directions", Technical Report CS-TR-3674, Computer Science Department Report, University of Maryland, August 1996.

[16] W. Wang, "Iterative methods in interior point algorithms for linear programming", Ph.D. dissertation, Applied Mathematics Program, University of Maryland, August 1996.

[17] W. Wang and D. P. O'Leary, "Adaptive use of iterative methods in interior point methods for linear programming", Technical Report CS-TR-3560, Computer Science Department Report, University of Maryland, November 1995.

[18] M. H. Wright, "Interior methods for constrained optimization", in *Acta Numerica 1992* (ed. A. Iserles), (Cambridge University Press, New York, USA, 1992) 341–407.

[19] Y. Zhang, "Solving large-scale linear programs by interior-point methods under the MATLAB environment", Technical Report TR 96–01, Department of Mathematics and Statistics, University of Maryland Baltimore County, February 1996.