A basis-deficiency-allowing primal Phase-I algorithm using the most-obtuse-angle column rule ⋆

Wei Li [a,*] Pablo Guerrero-García [b] Ángel Santos-Palomo [b]

[a] *School of Science, Hangzhou Dianzi University, Hangzhou 310018, P. R. China*

[b] *Department of Applied Mathematics, University of Málaga, 29071 Málaga, Spain*

Abstract

The dual Phase-I algorithm using the most-obtuse-angle row pivot rule is very efficient for providing a dual feasible basis, in either the classical or the basis-deficiency-allowing context. In this paper, we establish a basis-deficiency-allowing Phase-I algorithm using the so-called most-obtuse-angle column pivot rule to produce a primal (deficient or full) basis. Our computational experiments with the smallest test problems from the standard NETLIB set show that a dense projected-gradient implementation largely outperforms that of the variation of the primal simplex method from the commercial code MATLAB LINPROG v1.17, and that a sparse projected-gradient implementation of a normalized revised version of the proposed algorithm runs 34% faster than the sparse implementation of the primal simplex method included in the commercial code TOMLAB LPSOLVE v3.0.

Key words: simplex method, deficient basis, pivoting rule, most-obtuse-angle rule.

1 Introduction

The pivoting rule used is crucial to the simplex method. As a result, in the past a variety of pivot rules have been proposed. It is noticeable that among

⋆ The first author of this work was supported by NSFC Grant 10371028 and NSF Grant of Hangzhou Dianzi University KYS091504025. Review of an early draft by the first author dated in October, 2002.

* Corresponding author.

Email addresses: weilihz@126.com (Wei Li), pablito@ctima.uma.es (Pablo Guerrero-García), santos@ctima.uma.es (Ángel Santos-Palomo).

them the most-obtuse-angle row pivot rule is very efficient for achieving dual feasibility in the classical simplex context [13,14].

On the other hand, Pan generalized the concept of basis to include the deficient case, and established primal and dual pivot algorithms based on it [15,16]. Since the basis concept is crucial for pivot algorithms, this generalization provides us with a further improvement possibility. Computational results do show that the proposed basis-deficiency-allowing algorithms perform very favorably.

It is therefore very attractive to combine the most-obtuse-angle rule and the basis-deficiency-allowing algorithms. Along this line, Pan, Li and Wang recently developed a new dual Phase-I algorithm using the most-obtuse-angle row rule in the basis-deficiency-allowing context [18], and demonstrated its promise of success. On the other hand, similar effort has been made with primal cases by Santos-Palomo and Guerrero-García [19,6].

To make further progress, this paper develops a (primal) Phase-I algorithm using the most-obtuse-angle column pivot rule in the basis-deficiency-allowing context. For completeness, in the next section we briefly present the basis-deficiency-allowing pivot algorithms [15,16]. Then in Section 3, we describe the most-obtuse-angle column rule, and establish the basis-deficiency-allowing Phase-I procedure which uses it. In Section 4, we make some comments on the new algorithm. Finally, in Section 5, we report our computational results obtained with a set of standard test problems from NETLIB. These results show that a dense projected-gradient implementation largely outperforms that of the variation of the primal simplex method from the commercial code MATLAB LINPROG V1.17 [3], and that a sparse projected-gradient implementation of a normalized revised version of the proposed algorithm runs 34% faster than the sparse implementation of the primal simplex method included in the commercial code TOMLAB LPSOLVE V3.0 [10].

For unillustrated terminologies and symbols, we refer the reader to [15,16].

2 Preliminaries

Consider the following linear program in standard form:

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

where $A \in R^{m\times n}$ with $m < n$, and $b \in R^m, x, c \in R^n, 1 \leq rank(A) \leq m$. It is assumed that the cost vector c, the right-hand side b, and the columns and rows of A are nonzero, and that $Ax = b$ is consistent.

Conventionally, a basis is defined as a square nonsingular submatrix from the coefficient matrix A. The basis-deficiency-allowing variation of the simplex method generalized the basis as follows [16]:

Definition 1 [16] A basis is a submatrix consisting of any linearly independent subset of columns of A, whose range space includes b.

According to definition 1, the bases may be classified into two categories.

Definition 2 [16] If the number of basic columns equals the number of rows of the coefficient matrix, it is a normal basis; else, it is a deficient basis. Clearly, traditional simplex variants use normal bases only.

Let B be a basis with $m1$ columns and let N be the corresponding nonbasis, consisting of the remaining $n - m1$ columns. Define the ordered basic and nonbasic index sets respectively by

$$J_B = \{j_1, \ldots, j_{m1}\} \quad \text{and} \quad J_N = \{k_1, \ldots, k_{n-m1}\}$$

where $j_i, i = 1, \ldots, m1$, is the index of the i^{th} column of B, and $k_j, j = 1, \ldots, n - m1$, the index of the j^{th} column of N. The subscript of a basic index j_i is called a row index, and that of a nonbasic index k_j is called a column index. Components of x and c, and columns of A, corresponding to a basis and a nonbasis are subscripted with B and N, respectively. Hereafter, for simplicity of exposition, components of vectors and columns of matrices will always be arranged, and partitioned conformably, as the J_B, J_N changes. Thus we have

$$A = [B, N] = [a_{j_1}, \cdots, a_{j_{m1}}; a_{k_1}, \cdots, a_{k_{n-m1}}]$$

$$c^T = [c_B^T, c_N^T] = [c_{j_1}, \cdots, c_{j_{m1}}; c_{k_1}, \cdots, c_{k_{n-m1}}]$$
$$x^T = [x_B^T, x_N^T] = [x_{j_1}, \cdots, x_{j_{m1}}; x_{k_1}, \cdots, x_{k_{n-m1}}].$$

It is pedagogically convenient to use the tableau form for linear programming. Assume that the tableau form of (1) is

$$[B \quad N \quad b]\,. \tag{2}$$

Assuming $m1 < m$, after a series of appropriate Gauss or orthogonal transformations (we use orthogonal transformations here) we obtain:

$$[B \quad N \quad b] \rightsquigarrow \left[\overline{B} \quad \overline{N} \quad \overline{b}\right] := \begin{bmatrix} \overline{B}_1 & \overline{N}_1 & \overline{b}_1 \\ 0 & \overline{N}_2 & 0 \end{bmatrix} \tag{3}$$

where $\overline{B}_1 \in R^{m1\times m1}$ is an upper triangular matrix with nonzero diagonal entries.

The reduced cost $\overline{z}_N = c_N - \overline{N}_1^T \overline{B}_1^{-T} c_B$ and associated primal basic solution is then

$$\overline{x}_N = 0, \qquad \overline{x}_B = \overline{B}_1^{-1}\overline{b}_1, \tag{4}$$

with corresponding objective value $f = c_B^T \overline{B}_1^{-1}\overline{b}_1$. These entities are assumed to be updated after each iteration of the algorithm.

(3) is termed a canonical tableau. Since it is different from a corresponding basis B by only a non-singular matrix factor, $\overline{B}$ will be called basis as well.

By using the notation above we have:

Theorem 1 [16] If $\overline{x}_B \geq 0$ and $\overline{z}_N \geq 0$, then $\overline{x}$ is an optimal solution to the linear program (1).

3 The most-obtuse-angle column rule

The primal simplex procedure with a deficient basis needs a primal feasible basis to get itself started. Once an initial basis is available, which is neither primally nor dually feasible, a Phase-I procedure is then needed to achieve primal (or dual) feasibility. Recently, a dual Phase-I algorithm using the most-obtuse-angle rule with a deficient basis was described to achieve dual feasibility by Pan, Li and Wang [18]. In this section, we propose a primal version of the algorithm.

Define the row index sets I and H by

$$I = \{i \mid \overline{x}_{j_i} < 0, i = 1, \ldots, m1\}$$

$$H = \{j \mid \overline{z}_{k_j} < 0, j = 1, \ldots, n - m1\}$$

respectively. Suppose now that tableau (3) is primally infeasible, i.e., the set I is nonempty. If it is dually feasible, i.e., the set H is empty, then the basis-deficiency-allowing dual simplex algorithm is immediately applicable; otherwise, $H \neq \phi$ and Phase-I steps should be taken to achieve primal feasibility.

Now assume $\bar{x}_B \ngeq 0$ and $\bar{z}_N \ngeq 0$, i.e., $I \neq \phi$ and $H \neq \phi$. Select the pivot row index p such that

$$p = Argmin\{\overline{x}_{j_i} \mid i \in I\}. \tag{5}$$

Denote by

$$\left[\tilde{B}, \tilde{N}, \bar{b}\right] := \begin{bmatrix} \tilde{B}_1 & \tilde{N}_1 & \bar{b}_1 \\ 0 & \tilde{N}_2 & 0 \end{bmatrix} \tag{6}$$

the matrix resulting from the $[\bar{B}, \bar{N}, \bar{b}]$ by bringing the p^{th} column of $\bar{B}$ to the end of $\bar{N}$, with J_B and J_N adjusted conformably. If $p < m1$, then $\tilde{B}_1 \in R^{m1\times(m1-1)}$ is upper Hessenberg with nonzero subdiagonal entries in its p through $(m1-1)^{th}$ columns. A sequence of Givens rotations $G_j \in R^{m1\times m1}$, $j = p, \ldots, m1-1$, can be determined such that $Q_1^T\tilde{B}_1 \in R^{m1\times(m1-1)}$ is upper triangular, where $Q_1^T = G_{m1-1}\ldots G_p$. Consequently, we have

$$diag(Q_1^T, I_{m-m1})\left[\tilde{B}, \tilde{N}, \bar{b}\right] = \begin{bmatrix} Q_1^T\tilde{B}_1 & Q_1^T\tilde{N}_1 & Q_1^T\bar{b}_1 \\ 0 & \tilde{N}_2 & 0 \end{bmatrix} \tag{7}$$

where $I_{m-m1} \in R^{(m-m1)\times(m-m1)}$ is the identity matrix.

Denote with

$$d := \bar{N}_1^T\bar{B}_1^{-T}e_p. \tag{8}$$

Clearly d is a search direction in z_N-space [16]. However, we shall not compute d by (8), because an orthogonal transformation technique proposed by Pan [16] can be used to compute d at a lower cost as follows.

If we now redefine

$$[B, N, b] := diag(Q_1^T, I_{m-m1})\left[\tilde{B}, \tilde{N}, \bar{b}\right]$$

then from [16] we know that we can easily compute $\tilde{d}$, the same direction of d, by the following formula

$$\tilde{d} = -Sign(b_{m1})N^Te_{m1}. \tag{9}$$

When the set

$$\tilde{J} = \{j \mid \tilde{d}_j < 0, j = 1, \ldots, n-m1\} \tag{10}$$

is empty, the dual program is unbounded above, and hence program (1) has no feasible solution; otherwise, a column index q is chosen by

Rule 1 (most-obtuse-angle column select rule)

$$q = Argmin\{\tilde{d}_j \mid j \in \tilde{J}\}. \tag{11}$$

We then bring the q^{th} column of N to the end of B, with J_B and J_N adjusted conformably. Thus, after the m through $(m1+1)^{th}$ components of the column are zeroed using an appropriate sequence of Givens rotations $G_j^T, j = m - 1, \ldots, m1$, the iteration is completed. This is referred to as a *full iteration*, since the number of basic columns remains unchanged. The next can be either a full or a *rank-increasing iteration* (the number of basic columns grows by one), depending on whether b_{m1+1} is equal to zero or not. Clearly, a rank-increasing one will not include the steps prior to the computation of $\tilde{d}$ by (9).

The overall process is summarized as follows:

Algorithm 1

Given an initial canonical tableau (3) and associated sets J_B and J_N, $\overline{x}_B = \overline{B}_1^{-1}\overline{b}_1$, $\overline{x}_N = 0$:

1^o Stop if $\overline{x}_B \geq 0$;

2^o Determine row index p by (5);

3^o Bring the p^{th} column of B to the end of N, and adjust J_B and J_N conformably;

4^o If $p < m1$, annihilate nonzero subdiagonal entries in the p through $(m1-1)^{th}$ columns of B by premultiplying $[B, N, b]$ by appropriate Givens rotations;

5^o Compute $\widetilde{d}$ by (9);

6^o Stop if the set defined by (10) is empty;

7^o Determine column index q by (11);

8^o Bring the q^{th} column of N to the end of B, and adjust J_B and J_N conformably;

9^o If $m1 < m$, annihilate the m through $(m1+1)^{th}$ components of $m1^{th}$ column of B by premultiplying $[B, N, b]$ by appropriate Givens rotations;

10^o Go to 1^o if $m1 = m$ or $b_{m1+1} = 0$;

11^o Set $m1 := m1 + 1$;

12^o Goto 5^o.

It is noted that there is no assumption on set J at all in the above algorithm, making a key difference from the original dual algorithm with deficient basis

[16, §4] in which $H = \phi$ is assumed. Once primal feasibility is achieved by algorithm 1, the main procedure described in [16, §3] can then be used to complete the whole computation.

Theorem 2 Assuming termination of algorithm 1, it must take place at either
(1) Step 1^o, with primal basic feasible solution reached; or
(2) Step 6^o, detecting the infeasibility of program (1).

Proof The correctness of the first statement is obvious and hence is omitted. Let us discuss the second one. From the canonical tableau of (1) we have

$$x_p + \sum_{j \in J_N} d_j x_j = \bar{b}_p \tag{12}$$

where $\bar{b}_p < 0$ and $min\{d_j \mid j = 1, \ldots, n - m1\} \geq 0$. Thus (12) has no nonnegative solution. This completes the proof.

4 Comments on the pivoting rule

Let us take a look at the finiteness of the algorithm. Since there are only finitely many bases, the algorithm does not terminate if and only if cycling occurs. Furthermore, since the number of columns of a basis never decreases in the process, a cycle never involves any rank-increasing iteration. In other words, cycling can only occur in full iterations. This algorithm belong to the class of 'infinite' algorithms, since the possibility of cycling cannot be ruled out theoretically at present; in fact, a cycling example has been recently given by Guerrero-García and Santos-Palomo [9] in the classical context. From a practical point of view, finiteness is not a serious problem: first, it is well known that computational performance of existing 'finite' simplex variants is unsatisfactory whereas successful simplex variants are actually 'infinite', such as Dantzig's conventional simplex method. Second, as degeneracy occurs in practice very frequently, finiteness proofs under non-degeneracy assumption is only of conceptual or pedagogical interest. It might not be wise to confine ourselves to develop finite algorithms.

Algorithm 1 has some attractive features. First of all, due to the use of rule 1, a ratio-test-free pivot rule, it needs fewer computation time per iteration than conventional algorithms. Second, it can get started from any initial basis and hence, there will be no need to introduce artificial variables. It is clear that the problem size and hence the computational effort would increase significantly if the the process of finding an initial feasible solution was treated in the usual manner by introducing artificial variables. In addition, the column selection rule in algorithm 1 chooses a pivot candidate possessing the maximum absolute

value, hence this will improve numerical behaviour. Indeed, it is more than that, the most obtuse angle rule is favorable from the following geometrical point of view. The direction $\tilde{d}$ is computed in terms of an ascent direction with respect to the dual objective. Clearly, the gradient of the left-hand side of the constraint $z_{k_q} \geq 0$ makes the most obtuse angle with $\tilde{d}$ among all nonnegative constrains $z_{k_i} \geq 0, i = 1, \ldots, n - m1$. It is clear that, if the direction $\tilde{d}$ is closer to the ascent direction b, the gradient of the dual objective, then the gradient of the left-hand side of the constraint $z_{k_q} \geq 0$ also makes the most obtuse angle with b. Under the spirit of Pan's geometrical characterization of an optimal basis (or nonbasis) [12], we know that z_{k_q} is therefore eligible to be used as an optimal nonbasic variable for the dual problem. Thus, x_{k_q} is an optimal basic variable with respected to the primal problem according to the complementary slackness conditions.

All these remarks makes algorithm 1 a promising Phase-1 for the primal basis-deficiency allowing (BDA) simplex algorithm.

The algorithm is closely related with that called "dual-then-primal" in [7, §3, p. 8], which consists in first using a dual BDA algorithm [16, §4] but with its min-ratio test replaced by a most-obtuse-angle rule, i.e., *normalizing* the ratio-test-free rule obtained when

$$\min_j \left\{ \frac{\bar{z}_{k_j}}{-\tilde{d}_j} \right\} \quad \text{is replaced by} \quad \min_j \left\{ \tilde{d}_j \right\} \quad \text{where} \quad \tilde{d}_j \doteq a_{k_j}^T d^P < 0$$

to obtain the most-obtuse-angle rule in which

$$\min_j \left\{ \frac{\bar{z}_{k_j}/\|a_{k_j}\|}{-\tilde{d}_j/\|a_{k_j}\|} \right\} \text{ is replaced by } \min_j \left\{ \tilde{d}_j/\|a_{k_j}\| \right\} \text{ where } \tilde{d}_j \doteq a_{k_j}^T d^P < 0,$$

and then using a primal BDA algorithm [16, §3] but with a normalized criterion to determine the entering variable. Note that within this framework, the crash heuristic [15, §4] employed to obtain the initial basis and the Phase-I given here form an unique dual phase: the dual BDA algorithm adds constraints for which $a_{k_j}^T d^P < 0$ one after another in accordance with the normalized ratio-test-free rule described above until a first basis is available, a suitable deletion is performed to obtain a new search direction d^P and then a sequence of constraint additions takes place but now in accordance with the unnormalized ratio-test-free rule. Furthermore, the column rule used to select the entering variable in the primal BDA algorithm is not usually normalized. When the normalized rules are used everywhere, the algorithm obtained is essentially the sagitta method given by Santos-Palomo in [19] with its restarting procedure done before entering its primal-feasibility search loop (see [21], where other possibilities are explored); as we shall illustrate in the next section, the differences in performance obtained with the unnormalized and normalized sparse versions are appreciable.

#	Name	Optimal value	n	m	nnz(A)	nnz(c)	nnz(b)	nnz	dns
1	AFIRO	$-.46475314286e+3$	51	27	102	5	7	114	22
2	SC50B	$-.70000000000e+2$	78	50	148	1	5	154	11
3	SC50A	$-.64575077059e+2$	78	50	160	1	10	171	12
4	SC105	$-.52202061212e+2$	163	105	340	1	20	361	7
8	STOCFOR1	$-.41131976219e+5$	165	117	501	27	8	536	13
6	ADLITTLE	$.22549496316e+6$	138	56	424	82	37	543	19
9	BLEND	$-.30812149846e+2$	114	74	522	30	8	560	23
7	SCAGR7	$-.23313898243e+7$	185	129	465	133	53	651	8
10	SC205	$-.52202061212e+2$	317	205	665	1	38	704	4
12	SHARE2B	$-.41573224074e+3$	162	96	777	36	24	837	10
14	LOTFI	$-.25264706062e+2$	366	153	1136	8	49	1193	31
15	SHARE1B	$-.76589318579e+5$	253	117	1179	31	103	1313	13
17	SCORPION	$.18781248227e+4$	466	388	1534	282	76	1892	4
22	BRANDY	$.15185098965e+4$	303	220	2202	2	54	2258	32
19	SCAGR25	$-.14753433061e+8$	671	471	1725	475	179	2379	3
20	SCTAP1	$.14122500000e+4$	660	300	1872	360	154	2386	6
16	DUISRAEL	$.89664482186e+6$	316	142	2411	171	89	2671	28
23	ISRAEL	$-.89664482186e+6$	316	174	2443	89	171	2703	22
29	BANDM	$-.15862801845e+3$	472	305	2494	165	118	2777	17
31	SCFXM1	$.18416759028e+5$	600	330	2732	23	116	2871	10
30	E226	$-.18751929066e+2$	472	223	2768	189	99	3056	17
26	SCSD1	$.86666666743e+1$	760	77	2388	760	1	3149	49
28	AGG	$-.35991767287e+8$	615	488	2862	131	432	3425	4

Table 1
Linear programming test problems in standard form from NETLIB

#	Optimal value	Its	CntSc	MinRed	MinVar	DuaGap
1	$-4.647531428571369e+2$	34	14	$-1e-14$	$-1e-13$	$-6e-12$
2	$-6.999999999998724e+1$	48	17	$0e+00$	$3e+01$	$-1e-11$
3	$-6.457507705855319e+1$	49	16	$0e+00$	$-1e-14$	$-1e-11$
4	$-5.220206121152414e+1$	155	331	$-4e-17$	$-1e-13$	$-2e-10$
6	$2.254949631623806e+5$	109	45	$-6e-14$	$4e-15$	$3e-11$
7	$-2.331389824033978e+6$	291	967	$2e-03$	$1e+01$	$-3e-04$
8	$-4.113197621943637e+4$	138	425	$4e+00$	$-6e-15$	$0e+00$
9	$-3.081214984555769e+1$	102	86	$-1e-15$	$-2e-16$	$-3e-10$
10	$-5.220205511471001e+1$	383	4374	$-5e-17$	$-3e-12$	$-6e-06$
12	$-4.157322407413797e+2$	192	203	$-1e-13$	$-9e-14$	$-4e-11$
14	$-2.526463392655504e+1$	586	2072	$1e-19$	$-4e-11$	$-7e-05$
15	$-7.658931857917045e+4$	244	361	$7e-04$	$8e-01$	$-2e-08$
16	$8.966448218630457e+5$	351	327	$1e+00$	$-4e-16$	$2e-09$
17	$1.878124822738103e+3$	396	14380	$0e+00$	$-2e-14$	$7e-12$
19_a	$1.069507249599776e+7$	1185	81699	$-3e-01$	$-9e-13$	$-2e+07$
20	$1.412249999999994e+3$	849	38545	$-3e-12$	$-5e-14$	$6e-12$
22_b	$0.000000000000000e+0$	1101	36550	$0e+00$	$0e+00$	$0e+00$
23_a	$-8.724890524549816e+5$	494	1786	$-5e+01$	$1e-03$	$-1e+03$
26	$8.666666674333362e+0$	113	341	$-1e-15$	$-6e-16$	$-7e-15$
28_{cd}	$-9.714485304728061e+7$	0	1145	$0e+00$	$0e+00$	$0e+00$
29_{ab}	$1.079112845981221e+7$	1526	62109	$0e+00$	$0e+00$	$0e+00$
30_a	$-1.874248957270827e+1$	1044	30575	$-5e-15$	$-5e-14$	$-9e-03$
31_a	$3.059466008444887e+9$	1267	120208	$-1e+05$	$-8e-11$	$-3e+09$

Table 2
10657 dense iterations in 66.1 minutes with MATLAB LINPROG v1.17: (a) The problem is badly conditioned (the solution may not be reliable); (b) Maximum number of iterations exceeded; (c) Divide by zero; (d) The constraints are overly stringent (no feasible starting point found)

#	Optimal value	Its	CntSc	MinRed	MinVar	DuaGap
1	$-4.64753142857143e+2$	23	2	$-7e-32$	$-9e-16$	$-2e-13$
2	$-7.00000000000000e+1$	60	9	$0e+00$	$3e+01$	$1e-13$
3	$-6.45750770585645e+1$	60	8	$0e+00$	$0e+00$	$7e-14$
4	$-5.220206121170 72e+1$	132	39	$-1e-34$	$-1e-15$	$2e-13$
6	$2.25494963162380e+5$	138	34	$-1e-13$	$2e-15$	$2e-10$
7	$-2.33138982433098e+6$	188	141	$2e-03$	$1e+01$	$9e-10$
8	$-4.11319762194364e+4$	140	36	$4e+00$	$-1e-14$	$9e-11$
9	$-3.08121498458282e+1$	119	31	$0e+00$	$-1e-15$	$-1e-14$
10	$-5.22020612117072e+1$	263	247	$-3e-33$	$2e-17$	$-6e-13$
12	$-4.15732240741418e+2$	172	86	$-4e-16$	$0e+00$	$2e-12$
14	$-2.52647060618800e+1$	316	292	$-1e-18$	$-8e-14$	$5e-13$
15	$-7.65893185791856e+4$	305	220	$7e-04$	$8e-01$	$-4e-10$
16	$8.96644821863046e+5$	265	169	$1e+00$	$-2e-17$	$1e-09$
17	$1.87812482273811e+3$	376	1017	$0e+00$	$-2e-15$	$7e-13$
19	$-1.47534330607685e+7$	665	5063	$5e-02$	$-9e-12$	$-1e-08$
20	$1.41225000000000e+3$	369	995	$-6e-14$	$-7e-16$	$1e-12$
22	$1.51850989648813e+3$	310	375	$-1e-16$	$-1e-13$	$9e-13$
23	$-8.96644821863046e+5$	386	402	$-6e-14$	$1e-03$	$5e-10$
26	$8.66666667433337e+0$	143	100	$-9e-16$	$-6e-16$	$-1e-14$
28	$-3.59917672865765e+7$	556	3241	$-2e-15$	$-2e-11$	$8e-06$
29	$-1.58628018450121e+2$	570	1678	$0e+00$	$-3e-15$	$-6e-13$
30	$-1.87519290663705e+1$	601	1041	$0e+00$	$-5e-16$	$-2e-14$
31	$1.84167590283489e+4$	481	1628	$-2e-15$	$-9e-14$	$2e-11$

Table 3
6638 dense iterations in 2.8 minutes (unnormalized rule)

#	Optimal value	Its	CntSc	MinRed	MinVar	DuaGap
1	$-4.647531428571428e+2$	23	72	$-5e-32$	$1e+01$	$-6e-14$
2	$-7.000000000000000e+1$	54	30	$0e+00$	$3e+01$	$-4e-14$
3	$-6.457507705856452e+1$	59	31	$-6e-02$	$3e+00$	$0e+00$
4	$-5.220206121170725e+1$	122	156	$-1e-02$	$2e+00$	$-9e-14$
6	$2.254949631623804e+5$	151	88	$-2e-13$	$5e-13$	$-5e-10$
7	$-2.331389824330984e+6$	196	311	$2e-03$	$1e+01$	$1e-09$
8	$-4.113197621943641e+4$	154	186	$4e+00$	$2e-15$	$-5e-11$
9	$-3.081214984582823e+1$	150	98	$0e+00$	$2e-16$	$-5e-14$
10	$-5.220206121170725e+1$	258	938	$-1e-02$	$1e-16$	$-2e-13$
12	$-4.157322407414187e+2$	169	175	$-8e-02$	$2e-16$	$-2e-12$
14	$-2.526470606187998e+1$	345	778	$-1e-18$	$-3e-14$	$1e-14$
15	$-7.658931857918580e+4$	407	492	$7e-04$	$8e-01$	$1e-07$
16	$8.966448218630460e+5$	579	1056	$1e+00$	$4e-16$	$2e-09$
17	$1.878124822738106e+3$	414	5066	$0e+00$	$-3e-16$	$2e-12$
19	$-1.475343306076853e+7$	952	15216	$5e-02$	$-3e-13$	$-6e-09$
20	$1.412250000000000e+3$	549	4195	$-1e+01$	$-8e-16$	$2e-12$
22	$1.518509896488128e+3$	588	1958	$-2e-16$	$-3e-14$	$8e-12$
23	$-8.966448218630455e+5$	585	1400	$-6e-14$	$1e-03$	$1e-09$
26	$8.666666674333365e+0$	583	803	$-2e-08$	$1e-18$	$0e+00$
28	$-3.599176728657644e+7$	523	9591	$-8e+00$	$-9e-11$	$-1e-08$
29	$-1.586280184501208e+2$	1191	8436	$0e+00$	$-1e-14$	$-1e-12$
30	$-1.875192906637055e+1$	864	3058	$-2e-31$	$-3e-15$	$1e-13$
31	$1.841675902834895e+4$	670	5938	$-2e-15$	$-4e-14$	$3e-11$

Table 4
9586 sparse iterations in 10.0 minutes with Tomlab LpSolve v3.0

#	Optimal value	Its	CntSc	MinRed	MinVar	DuaGap
1	$-4.647531428571428e+2$	23	13	$-1e-31$	$5e-30$	$-6e-14$
2	$-7.000000000000000e+1$	60	27	$0e+00$	$3e+01$	$3e-14$
3	$-6.457507705856450e+1$	60	22	$0e+00$	$-2e-29$	$1e-14$
4	$-5.220206121170725e+1$	128	100	$0e+00$	$0e+00$	$0e+00$
6	$2.254949631623804e+5$	139	122	$-2e-13$	$1e-27$	$9e-11$
7	$-2.331389824330984e+6$	237	328	$2e-03$	$1e+01$	$0e+00$
8	$-4.113197621943641e+4$	142	178	$4e+00$	$-2e-17$	$0e+00$
9	$-3.081214984582824e+1$	114	130	$0e+00$	$-2e-30$	$4e-14$
10	$-5.220206121170725e+1$	265	405	$-1e-31$	$-1e-27$	$1e-14$
12	$-4.157322407414193e+2$	200	181	$-2e-15$	$-3e-28$	$3e-13$
14	$-2.526470606187999e+1$	326	3047	$-3e-17$	$-1e-14$	$-7e-15$
15	$-7.658931857918580e+4$	368	573	$7e-04$	$8e-01$	$2e-10$
16	$8.966448218630462e+5$	280	839	$1e+00$	$-2e-17$	$-5e-10$
17	$1.878124822738106e+3$	401	1066	$0e+00$	$-2e-28$	$7e-13$
19	$-1.475343306076853e+7$	748	3939	$5e-02$	$-1e-26$	$9e-09$
20	$1.412250000000000e+3$	379	1113	$-4e-14$	$-5e-17$	$-2e-13$
22	$1.518509896488128e+3$	373	2925	$-5e-16$	$-2e-27$	$0e+00$
23	$-8.966448218630456e+5$	381	1259	$-1e-14$	$1e-03$	$-1e-10$
26	$8.666666674333365e+0$	165	13774	$-1e-15$	$-5e-17$	$2e-15$
28	$-3.599176728657644e+7$	591	1364	$-2e-15$	$-1e-13$	$-7e-08$
29	$-1.586280184501208e+2$	598	11769	$0e+00$	$-3e-29$	$2e-13$
30	$-1.875192906637055e+1$	601	5156	$-2e-28$	$-3e-28$	$4e-15$
31	$1.841675902834894e+4$	508	6175	$-8e-31$	$-3e-15$	$-4e-12$

Table 5
7087 sparse iterations in 9.1 minutes (unnormalized rule)

#	Optimal value	Its	CntSc	MinRed	MinVar	DuaGap
1	$-4.647531428571428e+2$	24	6	$-1e-31$	$2e-30$	$-6e-14$
2	$-7.000000000000000e+1$	63	30	$0e+00$	$3e+01$	$3e-14$
3	$-6.457507705856450e+1$	60	22	$0e+00$	$-2e-29$	$1e-14$
4	$-5.220206121170725e+1$	130	98	$0e+00$	$0e+00$	$0e+00$
6	$2.254949631623803e+5$	143	133	$-1e-13$	$-1e-28$	$0e+00$
7	$-2.331389824330984e+6$	220	295	$2e-03$	$1e+01$	$0e+00$
8	$-4.113197621943641e+4$	141	163	$4e+00$	$-2e-17$	$0e+00$
9	$-3.081214984582824e+1$	119	125	$-4e-16$	$-3e-31$	$2e-14$
10	$-5.220206121170725e+1$	287	503	$-2e-32$	$-3e-27$	$1e-14$
12	$-4.157322407414200e+2$	218	208	$-7e-16$	$0e+00$	$1e-12$
14	$-2.526470606188001e+1$	278	1720	$-3e-17$	$-9e-16$	$1e-14$
15	$-7.658931857918580e+4$	305	425	$7e-04$	$8e-01$	$2e-10$
16	$8.966448218630463e+5$	304	980	$1e+00$	$-8e-17$	$-7e-10$
17	$1.878124822738107e+3$	393	1103	$0e+00$	$-6e-28$	$5e-13$
19	$-1.475343306076852e+7$	745	4078	$5e-02$	$-2e-26$	$7e-09$
20	$1.412250000000000e+3$	382	1289	$-4e-14$	$-2e-16$	$0e+00$
22	$1.518509896488128e+3$	389	3152	$-2e-16$	$-4e-27$	$5e-13$
23	$-8.966448218630457e+5$	434	1594	$-7e-15$	$1e-03$	$-2e-10$
26	$8.666666674333365e+0$	118	2131	$-4e-10$	$-3e-18$	$-4e-15$
28	$-3.599176728657644e+7$	582	1345	$-7e-15$	$-3e-20$	$-7e-08$
29	$-1.586280184501207e+2$	556	10675	$0e+00$	$0e+00$	$3e-14$
30	$-1.875192906637055e+1$	546	4173	$-2e-30$	$-4e-28$	$4e-15$
31	$1.841675902834894e+4$	526	5056	$-2e-15$	$-2e-15$	$0e+00$

Table 6
6963 sparse iterations in 6.6 minutes (normalized rule)

5 Computational results

We have carried out a dense implementation using projected gradient techniques based on an orthogonal factorization of A_B, in particular, that obtained with the classical Gram-Schmidt method with reorthogonalization [2, §2.4]. A systematic update and downdate of the factor $Q \in R^{m \times m1}$ with orthonormal columns and the triangular factor $\overline{B}_1 \in R^{m1 \times m1}$ is carried out in this revised implementation, see details in [22].

We have conducted some dense computational experiments using MATLAB v5.3 (with an Intel Pentium 4, 3.0 Ghz, 512 Mb RAM) to compare the unnormalized revised version of the Phase-I given above against the MATLAB LINPROG v1.17 [3] dense implementation of a variation of the usual primal simplex method. Our computational experiment has been performed by using a subset of the standard NETLIB benchmark [5]; all problems with less than 3500 nonzeros in which neither BOUNDS nor RANGES sections occur has been selected. The details of a quite similar sparse computational experience can be found in [6, §5] and [8], where the ordering and scaling techniques applied to the test problems in table 1 are described. The density (in percentage) of the Cholesky factor of $A^T A$ for the ordering chosen (not relevant for dense tables) has been included as the right-most column of the table. No additional preprocessing techniques have been used.

The information that has been included in each row of the tables 2–6 (from left to right) is: the number in table 1 of the test problem, the optimal value, the number of iterations performed, the elapsed time (in hundredths of a second), the minimum reduced cost, the minimum value of the computed solution, and the duality gap.

The main outcome obtained in the dense case is that 6638 (in 2.8 minutes, cf. table 3) iterations were performed by the unnormalized version, versus 10657 (in 66.1 minutes, cf. table 2) needed by LINPROG. Furthermore, LINPROG was unable to solve several problems in less than $5m$ iterations, and some of them faced with numerical problems. Hence the unnormalized version largely outperforms LINPROG.

A suitable sparse orthogonal approach is to adapt the methodology of Björck [1] and Oreborn [11] to be able to apply the sparse NNLS algorithm (via corrected seminormal equations (CSNE) with the Cholesky factor R_B of $A_B^T A_B$) with a "short-and-fat" matrix A. They proposed an active set algorithm for the sparse least squares problem

$$\text{minimize} \qquad 1/2 \cdot x^T C x + d^T x, \quad x \in \mathbb{R}^n \qquad \text{subject to} \qquad l \leq x \leq u$$

with $C > 0$. It turns out that our Phase-I is also related with the problem in

which

$$C = A^T A \quad \wedge \quad d = -A^T b \quad \wedge \quad \forall i \in 1{:}\, n \ , \ l_i = 0 \ \wedge \ u_i = +\infty$$

but $C \geq 0$, hence to maintain a sparse QR factorization of A_B we have had to adapt [20] the proposed technique as in [4], but without forming C. The column ordering of A dictates that of A_B, although this fact does not prevent us to maintain the order of arrival in the basic index set.

We have also conducted some sparse computational experiments using MATLAB v5.3 (with an Intel Pentium 4, 2.8 Ghz, 512 Mb RAM) to compare both the unnormalized and the normalized revised versions of the Phase-I's given above against the TOMLAB LPSOLVE v3.0 [10] sparse implementation of the usual primal simplex method.

The main outcome obtained in the sparse case is that 7087 (in 9.1 minutes, cf. table 5) and 6963 (in 6.6 minutes, cf. table 6) iterations were performed by the unnormalized and normalized versions respectively, versus 9586 (in 10.0 minutes, cf. table 4) needed by LPSOLVE. A non-trivial implementation [22] of the primal min-ratio test for the Phase-II allowed us to improve the run time of the unnormalized version until 8.9 minutes (7161 iterations), but it is worth noting that:

(1) Excluding SCAGR25 (#19) from the test set, only the normalized version outperforms LPSOLVE, turning out to run 21% faster.
(2) Excluding SCSD1 (#26) from the test set, both the unnormalized and the normalized version outperforms LPSOLVE considerably, turning out to run 31% and 37% faster, respectively.
(3) With the full test set, both the unnormalized and the normalized version outperforms LPSOLVE, turning out to run 11% and 34% faster, respectively.

These preliminary experiments lead us to conclude that a clear advantage was obtained in number of iterations, quality of solutions and execution time in both the dense and sparse case when suitable pivot strategies are used and with no special anti-cycling tools.

Acknowledgements

The authors thank Ping-Qi Pan for many helpful comments and suggestions that greatly improve the quality of the paper.

References

[1] Å. Björck. A direct method for sparse least squares problems with lower and upper bounds. *Numer. Math.*, 54:19–32, 1988.

[2] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM Pubs., Philadelphia, 1996.

[3] T. Coleman and M. A. Branch and A. Grace. Optimization Toolbox v2.1 for use with MATLAB, user's guide. Technical report, The Math Works Inc., 1999.

[4] T. F. Coleman and L. A. Hulbert. A direct active set algorithm for large sparse quadratic programs with simple lower bounds. *Math. Progr.*, 45:373–406, 1989.

[5] D. M. Gay. Electronic mail distribution of linear programming test problems. *Committee on Algorithms (COAL) Newsletter*, 13:10–12, 1985.

[6] P. Guerrero-García. *Range-Space Methods for Sparse Linear Programs* (Spanish). Ph. D. thesis, Department of Applied Mathematics, University of Málaga, Spain, July 2002.

[7] P. Guerrero-García and Á. Santos-Palomo. A non-simplex active-set framework for basis-deficiency-allowing simplex variations. In D. Griffiths and G. A. Watson, editors, *Proceedings of the 20th Biennial Conference on Numerical Analysis*, page 19, Dundee, Scotland, June 2003. Submitted for publication.

[8] P. Guerrero-García and Á. Santos-Palomo. A comparison of three sparse linear program solvers. Technical report, Department of Applied Mathematics, University of Málaga, October 2003. Submitted for publication.

[9] P. Guerrero-García and Á. Santos-Palomo. Phase-I cycling under the most-obtuse-angle pivot rule. *Europ. J. Operl. Res.* 167(1):20–27, November 2005.

[10] K. Holmström. The TOMLAB optimization environment v3.0 user's guide. Technical report, Mälardalen University, Sweden, April 2001.

[11] U. Oreborn. *A Direct Method for Sparse Nonnegative Least Squares Problems*. Licentiat thesis, Department of Mathematics, Linköping University, Sweden, 1986.

[12] P.-Q. Pan. Practical finite pivoting rules for the simplex method. *OR Spektrum*, 12:219–225, 1990.

[13] P.-Q. Pan. New non-monotone procedures for achieving dual feasibility. *Journal of Nanjing University Mathematical Biquarterly*, 12(2):155–162, November 1995.

[14] P.-Q. Pan. The most-obtuse-angle row pivot rule for achieving dual feasibility: A computational study. *Europ. J. Operl. Res.* 101(1):167–176, August 1997.

[15] P.-Q. Pan. A dual projective simplex method for linear programming. *Computers Math. Applic.*, 35(6):119–135, March 1998.

[16] P.-Q. Pan. A basis-deficiency-allowing variation of the simplex method for linear programming. *Computers Math. Applic.*, 36(3):33–53, August 1998.

[17] P.-Q. Pan and Y.-P. Pan. A Phase-1 approach for the generalized simplex algorithm. *Computers Math. Applic.*, 42(10/11):1455–1464, November 2001.

[18] P.-Q. Pan and W. Li and Y. Wang. A Phase-I algorithm using the most-obtuse-angle rule for the basis-deficiency-allowing dual simplex method. *OR Transactions* (Chinesse), 8(3):88–96, 2004.

[19] Á. Santos-Palomo. The sagitta method for solving linear programs. *Europ. J. Operl. Res.* 157(3):527–539, September 2004.

[20] Á. Santos-Palomo and P. Guerrero-García. Solving a sequence of sparse least squares problems. Technical report, Department of Applied Mathematics, University of Málaga, September 2001. Submitted for publication.

[21] Á. Santos-Palomo and P. Guerrero-García. Sagitta method with guaranteed convergence. Technical report, Department of Applied Mathematics, University of Málaga, February 2005. Submitted for publication.

[22] Á. Santos-Palomo and P. Guerrero-García. Computational experiences with dense and sparse implementations of the sagitta method. Technical report, Department of Applied Mathematics, University of Málaga, February 2005. Submitted for publication.