
http://researchrepository.murdoch.edu.au/15195/

Copyright: © 2002 Elsevier B.V.

It is posted here for your personal use. No further distribution is permitted.
An $L_1$ Estimation Algorithm with Degeneracy and Linear Constraints

Mingren Shi * Mark A. Lukas *

Abstract

An implementation of the reduced gradient algorithm is proposed to solve the linear $L_1$ estimation problem (least absolute deviations regression) with linear equality or inequality constraints, including rank deficient and degenerate cases. Degenerate points are treated by solving a derived $L_1$ problem to give a descent direction. The algorithm is a direct descent, active set method that is shown to be finite. It is geometrically motivated and simpler than the projected gradient algorithm (PGA) of Bartels, Conn and Sinclair, which uses a penalty function approach for the constrained case. Computational experiments indicate that the proposed algorithm compares favourably, both in reliability and efficiency, to the PGA, to the algorithms ACM551 and AFK (which use an LP formulation of the $L_1$ problem) and to LPASL1 (which is based on the Huber approximation method of Madsen, Nielsen and Pinar). Although it is not as efficient as ACM552 (Barrodale-Roberts algorithm) on large scale unconstrained problems, it performs better on large scale problems with bounded variable constraints.

Key words: linear model, $L_1$ estimation, least absolute deviations, regression, degeneracy, linear constraints, reduced gradient algorithm, active set.

AMS (MOS) subject classifications: 65U05, 62G05, 62J05

Abbreviated title: An $L_1$ estimation algorithm

1. Introduction

Given a linear model

$$b_i = a_i^T x + \epsilon_i, \quad i = 1, \ldots, m_1,$$

with linear equality or inequality constraints on $x$, a robust estimate can be defined by $L_1$ estimation, also called the method of least absolute deviations. The general form of this estimation problem (denoted $LL_1$) can be written as

$$\text{minimize} \quad S(x) = \sum_{i=1}^{m_1} |a_i^T x - b_i|, \quad x \in \mathbb{R}^n,$$

subject to

$$a_i^T x - b_i = 0, \quad i \in \mathcal{E} = \{m_1 + 1, m_1 + 2, \ldots, m_1 + l\}, \quad (1.1A)$$

$$a_i^T x - b_i \leq 0, \quad i \in \mathcal{I} = \{m_1 + l + 1, \ldots, m_1 + m_2\}. \quad (1.1B)$$

Here $n \geq 2$ and $l < n < m_1 + m_2$. Let $UL_1$ denote the corresponding unconstrained regression problem. The $LL_1$ problem arises naturally in the solution of curve fitting problems and discretized inverse problems using the $L_1$ norm criterion, subject to constraints such as non-negativity or monotonicity [30,33,35].

It is well known that (1.1) can be formulated as a linear programming (LP) problem [13,34]. Efficient simplex-like methods that exploit the special structure of the LP problem have been developed (see [12,15,16,24] for a historical development). The algorithms ACM551 of Abdelmalek [1] (for $UL_1$) and ACM552 of Barrodale and Roberts [4,5,6] (for both $UL_1$ and $LL_1$), which are available from the Netlib Software Library, and the algorithm (denoted AFK) of Armstrong, Frome and Kung [2] (for $UL_1$), whose Fortran code is given in [2], are of this type (see also [3]). Other efficient methods are based on a direct

* Mathematics and Statistics, Murdoch University, Murdoch, WA 6150, Australia. E-mail: lukas@central.murdoch.edu.au
descent, active set approach (see [7,8,9,11,24,25]). This includes the projected gradient algorithm (PGA) of Bartels, Conn and Sinclair [7,8,9], which is available as the NAG library routine E02GBF or from Netlib as ACM563 (for both $UL_1$ and $LL_1$), and the reduced gradient algorithm (RGA) developed by Bloomfield and Steiger [12] and Osborne [24,25]. As shown in [12,24], except for the start-up phase, all the above algorithms are actually equivalent for nondegenerate unconstrained problems, in that they produce the same sequence of iterates. However, there are differences in their implementation. An interior point method has been proposed in [38]. Recent work on a continuation method with a Huber approximation of the $L_1$ function has achieved promising computational results (see [18,19,20,21,23,26]). In particular, the algorithm of Madsen, Nielsen and Pinar [21] is available (for the unconstrained case) as the routine LPASL1 [20] from the authors or the routine LAV in the SAS IML library.

In this paper, we adapt and implement the RGA for the general $LL_1$ problem, in particular, presenting a treatment for the degenerate case. The RGA is an active set method and was motivated by the corresponding algorithm (of the same name) for LP problems (see [10,24]), which can be regarded as more natural than the simplex algorithm. See [31] for an illustration of the simple geometry of the algorithm.

First, in Section 2, we state the main optimality conditions for the $LL_1$ problem and give the main stages of the RGA in the nondegenerate case. It is known (see [12]) that the treatment of degenerate $L_1$ problems is difficult. Direct application of the algorithm given in Section 2 may result in cycling at some points. Sadovski [27] is aware of the cycling problem for $L_1$ line fitting, but his code only gives a failure indication. Seneta and Steiger [29] characterize degeneracy but bypass the treatment of it when it is diagnosed. Osborne [24] and Bartels and Conn [7] treat degenerate points using a perturbation approach. In Section 3, we develop a method for dealing with degeneracy that uses the active set framework of the RGA. For each degenerate point, we solve a derived $LL_1$ problem (or sequence of these), which either shows the point is optimal or gives a descent direction. Section 4 describes the implementation of the RGA. To find a feasible starting point for the $LL_1$ problem, an auxiliary LP problem is solved using an active set algorithm from [36,37]. It is shown that the resulting algorithm for the $LL_1$ problem is finite. Results about the stability of solutions to the $LL_1$ problem are derived elsewhere (see [32]).

The proposed RGA algorithm has been implemented in a Fortran 77 routine and the code is available from the authors. Section 5 contains the results of computational experiments with the RGA routine on several examples from the literature, including a rank deficient problem, a highly degenerate problem and randomly generated, large scale problems of several sizes. These results are compared with those obtained using the PGA, ACM551, ACM552, AFK and LPASL1 algorithms. For the smaller problems, all the algorithms found correct solutions, except in the rank deficient problem, the PGA routine E02GBF terminated due to ill-conditioning. Also, AFK and LPASL1 could not be used directly on this problem. For the large scale unconstrained problems, PGA (ACM563) failed in several cases (possibly in its handling of degeneracy) and LPASL1 failed in the largest case. The computer user times indicate that RGA compares favourably with PGA, ACM551, AFK and LPASL1 (with their default starting points). However, using the $L_2$ solution as a starting point, ACM552 performed better than RGA on problems with large numbers of variables.

The last example in Section 5 involves large scale problems with both upper and lower bounds on the variables. Of the algorithms above, only RGA, PGA and ACM552 can be applied here, but PGA (ACM563)
failed on all problems with more than 15 variables. RGA and ACM552 were applied successfully using each of the upper bound vector and the midpoint of the bounds as the starting point, and it was found that in both cases RGA was more efficient. Also, unlike ACM552, the RGA routine has the advantage of allowing the user to directly choose the starting point without transforming the problem.

2. Optimality conditions and the reduced gradient algorithm

Optimality conditions for a general class of problems including (1.1) were proved in [24] using methods of convex analysis. For the $LL_1$ problem (1.1) itself, a simpler proof using the Farkas lemma can be derived, but here we only give the result.

Let the active set at a given point $x$ be

$$A = A(x) = \{i \mid r_i(x) \equiv a_i^T x - b_i = 0, \ 1 \leq i \leq m_1 + m_2\},$$

and denote

$$A_S = A \cap \{1, \ldots, m_1\}, \quad A_I = A \cap I, \quad A_S^c = \{1, \ldots, m_1\} \setminus A_S. \quad \text{(2.1B)}$$

**Theorem 2.1** Necessary conditions for $x^*$ to be a solution to (1.1), assuming that the corresponding active gradient vectors $\{a_i, \ i \in A^* = A(x^*)\} \text{ are linearly independent, are that there exist multipliers } \lambda_i^*, \ i \in A^*, \text{ such that}$

$$a_i^T x^* - b_i = 0, \ i \in E, \quad \text{and} \quad a_i^T x^* - b_i \leq 0, \ i \in I,$$

$$c \equiv \sum_{i \in A_S^*, i} \sigma_i a_i = \sum_{i \in A_S} \lambda_i^* a_i + \sum_{i \in I} \lambda_i^* a_i,$$

$$\lambda_i^* \leq 0, \ i \in A_I^c.$$  \quad \text{(2.3C)}$$

where $A_S^* = \{1, 2, \ldots, m_1\} \setminus A_S^c$ and $\sigma_i = \sigma_i(x) = \text{sign}(r_i(x))$. Conversely, (2.2) and (2.3) (without any requirement of linear independence) are sufficient conditions for $x^*$ to be a solution. □

The active set method for $L_1$ norm minimization relies on the fundamental, well-known fact that the function $S(x)$ attains its minimum at a special point, which is similar to a basic point for LP problems and will be called a base point. (In [12], it is called an extreme point and in [25], the corresponding point of the graph of $S(x)$ is called an exposed point.)

**Definition 2.1** Assuming that $A$ is the active set at a given point $x$ and $r \equiv \text{rank}\{a_i, \ 1 \leq i \leq m_1 + m_2\} \leq n$, we say $x$ is a base point of problem (1.1) if the rank of the active gradient vectors $\{a_i, \ i \in A\}$ is $r$.

**Theorem 2.2** The minimum value of problem (1.1) is achieved and can be found at a (feasible) base point of (1.1). (It is assumed that the feasible region is not empty.)

It is clear from the above that a solution to the $LL_1$ problem can be obtained from the set of base points. After finding a starting base point, we move along a descent direction to another base point, and repeat this procedure until an optimal base point is reached. The main stages of the RGA for the nondegenerate case, which is defined below, are given in Algorithm 2.1. See Section 4 for the details.

**Definition 2.2** A (feasible) base point $x$ is nondegenerate if the active gradient vectors $\{a_i, \ i \in A\}$ are linearly independent. If every (feasible) base point of problem (1.1) is nondegenerate, the problem (1.1) is said to be nondegenerate and otherwise it is degenerate.
Algorithm 2.1 (For simplicity, here we assume that $r = n$. It will be seen in Section 4 that this assumption can be removed.)

Suppose that $\tilde{x}$ is feasible for (1.1) and is a nondegenerate base point of (1.1). Let its active set be $\tilde{A} = \{j_1, j_2, \ldots, j_n\}$, and denote $\tilde{A}_S = \tilde{A} \cap \{1, \ldots, m_1\}$, $\tilde{A}_I = \tilde{A} \cap \mathcal{I}$ and $\tilde{A}_S^c = \{1, \ldots, m_1\} \setminus \tilde{A}_S$. Let the base matrix (with column vectors $a_i$, $i \in \tilde{A}$) and its inverse be, respectively, $\hat{A} = [a_{j_1}, a_{j_2} \ldots a_{j_n}]$ and $\hat{D} = \hat{A}^{-1} = [\hat{d}_1 \ \hat{d}_2 \ \ldots \ \hat{d}_n]^T$, where $\hat{d}_i^T$ are the row vectors of $\hat{D}$. Then the ‘cost’ vector $\hat{c}$ can be expressed as a linear combination of the columns of $\hat{A}$ and the multipliers $\hat{\lambda}_i$ are given by

$$\hat{c} = \sum_{i \in \tilde{A}_S} \hat{\lambda}_i a_i = \sum_{i=1}^{n} \hat{\lambda}_i \hat{a}_{j_i}, \quad \hat{\lambda}_i = \hat{d}_i^T \hat{c}, \quad i = 1, 2, \ldots, n. \quad (2.4)$$

(I) Test the optimality conditions. Let max{|$\hat{\lambda}$| : $j_i \in \tilde{A}_S$} = $|$|$\hat{\lambda}$|$_{q_1}$| and max{|$\hat{\lambda}$| : $j_i \in \tilde{A}_I$} = $|$|$\hat{\lambda}$|$_{q_2}$|. If $|$|$\hat{\lambda}$|$_{q_1}$| $\leq 1$ and $|$|$\hat{\lambda}$|$_{q_2}$| $\leq 0$, then the point $\tilde{x}$ is optimal (see (2.3B) and (2.3C)) and the algorithm is terminated.

(II) Choose a (feasible) descent direction. If the above optimality condition does not hold, then either of the choices

$$d = \begin{cases} -\text{sign}(\hat{\lambda}_{q_1})\hat{d}_{q_1} & \text{if } |\hat{\lambda}_{q_1}| > 1, \\ -\hat{d}_{q_2}, & \text{if } \hat{\lambda}_{q_2} > 0, \end{cases} \quad (2.5)$$

is a feasible descent direction at $\tilde{x}$, as we now show. Let $q$ be the index $q_1$ or $q_2$ that is used to define $d$. Then, since $a_{j_i}^T\hat{d}_q = \delta_{q_i}$, $j_i \in \tilde{A}$, it is easy to see that $d$ is feasible. Let the shortest step to the boundary planes $P_j = \{x \mid a_j^Tx - b_j = 0\}$ associated with inactive equations (i.e. nonzero residuals) and with inactive inequality constraints be, respectively,

$$\delta_{k_1} = \min\{d_j \mid d_j = -r_j(\tilde{x})/a_j^T d > 0, \ j \in \tilde{A}_S^c, \ a_j^T d \neq 0\},$$

$$\alpha_{p_1} = \min\{d_j \mid d_j = -r_j(\tilde{x})/a_j^T d > 0, \ j \in \mathcal{I} \setminus \tilde{A}_I, \ a_j^T d > 0\}. \quad (2.6)$$

(Note that $\delta_{k_1} = +\infty$ or $\alpha_{p_1} = +\infty$, if the set concerned is empty. However at least one of $\delta_{k_1}$ or $\alpha_{p_1}$ is finite.) If the step length satisfies $0 < \alpha < \theta_p = \min\{\alpha_{p_1}, \delta_{k_1}\}$, then clearly the point $\tilde{x} + \alpha d$ is feasible.

It is not hard to show that for any direction $d$ and $0 < \alpha < \delta_{k_1}$,

$$S(\tilde{x} + \alpha d) = S(\tilde{x}) + \alpha \Delta S_0,$$

where $\Delta S_0 \equiv \hat{c}^T d + \sum_{i \in \tilde{A}_S} |a_i^T d|$. \quad (2.7)

From $\tilde{A} = \tilde{A}_S \cup E \cup \tilde{A}_I$ and (2.4), for the two choices of $d$ in (2.5) we have

$$\Delta S_0 = \begin{cases} -|\hat{\lambda}_{q_1}| + 1 < 0, & \text{if } |\hat{\lambda}_{q_1}| > 1, \\ -\hat{\lambda}_{q_2} < 0, & \text{if } \hat{\lambda}_{q_2} > 0, \end{cases} \quad (2.8)$$

so $d$ is a descent direction for each case.

(III) Move from $\tilde{x}$ along $d$ to another base point, and then go to (I).

Theorem 2.3 Algorithm 2.1 has finite step convergence for nondegenerate $LL_1$ and $UL_1$ problems.

Proof. The result follows because the number of base points of (1.1) is finite and the algorithm is a descent method, so a base point cannot appear twice in the iteration sequence. ■

3. Treatment of degeneracy

(I) The problem and notation

Recall that the result that the minimum value of $S(x)$ occurs at a base point (Theorem 2.2) and the sufficient conditions for optimality (Theorem 2.1) are true in general, not merely for the nondegenerate case. Regarding the necessary conditions, (2.3A) is still valid, but (2.3B) and (2.3C) do not necessarily hold if the active gradient vectors are linearly dependent, since in that case the representation (2.3A) is not unique. Geometrically, a base point is degenerate if it is the intersection of more than $r$ active planes.
For a degenerate $L_1$ problem, a direct application of Algorithm 2.1 can result in cycling at a degenerate base point. A cycling example is given in [31].

One approach used to deal with degeneracy is the perturbation method. Geometrically, this method moves the surplus active planes out from the degenerate point (say by perturbing $b_i$, $1 \leq i \leq m_1 + m_2$), making the point nondegenerate. This approach is discussed in [7, 24] and is used in the PGA routine ACM563. However, as shown in Section 5, ACM563 appears to add unnecessary iterations in the degenerate case. Here we propose to treat degeneracy in a different way.

For simplicity, in this section we again assume that $r = n$. Let $\hat{x}$ be any base point (possibly degenerate) of (1.1) generated by Algorithm 2.1. In this section, we use the following notation.

1. Suppose the active set $\hat{A} = \{i | a_i^T \hat{x} - b_i = 0\}$ has more than $n$ elements and $\hat{A}^e = \{1, 2, \ldots, m_1 + m_2\} \setminus \hat{A}$.

2. Let the index set $\hat{I} = \{j_1, j_2, \ldots, j_m\}$ be the subset of $\hat{A}$ associated with $\hat{x}$ resulting from Algorithm 2.1 and $\hat{I}^c = \{1, 2, \ldots, m_1 + m_2\} \setminus \hat{I}$. Let $\hat{Z}_S = \hat{I} \cap \hat{A}_S$, $\hat{Z}_S^c = \{1, 2, \ldots, m_1\} \setminus \hat{Z}_S$, $\hat{I}_I = \hat{I} \cap \hat{A}_I$ and $\hat{I}_I^c = I \setminus \hat{I}_I$.

3. The corresponding base matrix and its inverse are $\hat{A} = [a_{j_1} a_{j_2} \ldots a_{j_m}], j_1, j_2, \ldots, j_m \in \hat{I}$, and $\hat{D} = \hat{A}^{-1} = [\hat{d}_1 \hat{d}_2 \ldots \hat{d}_n]^T$.

4. The corresponding multipliers are $\hat{\lambda}_i = d_i^T \hat{c}_0$, $i = 1, 2, \ldots, n$, where

$$\hat{c}_0 = \sum_{i \in \hat{Z}_S} \sigma_i a_i = \sum_{i \in \hat{A}_S} \sigma_i a_i = \hat{c}$$

since $\sigma_i \equiv \text{sign}(r_i(\hat{x})) = 0$, $\forall i \in \hat{A}_S \setminus \hat{Z}_S$.

(II) Optimality and descent direction

Firstly, from (2.7) we have the following.

Lemma 3.1 For a (feasible) direction $d$, let $\Delta(d) = \Delta S_0$, as defined in (2.7). Then $d$ is descent at $\hat{x}$ if and only if $\Delta(d) < 0$. ■

From (2.8), if $\hat{x}$ is nondegenerate, the direction (see (2.5)) $\hat{d} = -\text{sign}(\hat{\lambda}_{q_1}) \hat{d}_{q_1}$ is descent, provided $\max \{|\hat{\lambda}_i| : j_i \in \hat{Z}_S = \hat{A}_S\} = |\hat{\lambda}_{q_1}| > 1$. At a degenerate base point, however, if $\hat{A}_S \setminus \hat{Z}_S \neq \emptyset$, we have

$$\Delta(d) = \hat{c}^T d + \sum_{i \in \hat{Z}_S} |a_i^T d| + \sum_{i \in \hat{A}_S \setminus \hat{Z}_S} |a_i^T d| = -|\hat{\lambda}_{q_1}| + 1 + \sum_{i \in \hat{A}_S \setminus \hat{Z}_S} |a_i^T \hat{d}_{q_1}|. \quad (3.1)$$

Then the condition $|\hat{\lambda}_{q_1}| > 1$ does not guarantee that $\hat{d} = -\text{sign}(\hat{\lambda}_{q_1}) \hat{d}_{q_1}$ is a descent direction at $\hat{x}$. Similarly, if $\hat{A}_I \setminus \hat{I}_I \neq \emptyset$, the condition $\hat{\lambda}_{q_2} > 0$ does not guarantee that $\hat{d} = -\hat{d}_{q_2}$ is descent. Let

$$Q_1 = \{i : |\hat{\lambda}_i| > 1, j_i \in \hat{Z}_S\} \quad \text{and} \quad Q_2 = \{i : \hat{\lambda}_i > 0, j_i \in \hat{I}_I\}.$$

From the sufficient conditions for optimality (Theorem 2.1), we have:

Lemma 3.2 If $Q_1 \cup Q_2 = \emptyset$, then $\hat{x}$ is optimal.

This follows since (2.3) is still valid provided we choose $\hat{\lambda}_i = 0$, $\forall j_i \in \hat{A} \setminus \hat{Z}$. ■

From the above analysis and (3.1), we also have:

Lemma 3.3 If $\hat{A}_S \setminus \hat{Z}_S = \emptyset$ and $Q_1 \neq \emptyset$ (or $Q_2 \neq \emptyset$), then $\hat{x}$ is not optimal; the direction $\hat{d} = -\text{sign}(\hat{\lambda}_{q_1}) \hat{d}_{q_1}$ (or $\hat{d} = -\hat{d}_{q_2}$) is descent. ■

For the special case when $\hat{A}_S = \emptyset$, we have:
Lemma 3.4 Assume \( \hat{x} \) is a degenerate base point of (1.1), \( \hat{A}_S = \emptyset \) and \( \hat{Z} = \hat{Z}_I \cup \mathcal{E} \) (i.e. all of the surplus active gradients come from the constraints). If there is a \( j_i \in \hat{Z}_I \) such that \( \hat{\lambda}_i > 0 \), then \( d = -d_l \) is a feasible descent direction at \( \hat{x} \). Thus, \( \hat{x} \) is optimal if and only if \( \hat{\lambda}_i \leq 0 \), \( j_i \in \hat{Z}_I \).

**Proof.** Since \( a^T_l d_l = \delta_{d_l}, j_i \in \hat{Z}_I \), then \( a^T_l (\hat{x} - d_l) - b_j = -\delta_{d_l}, j_i \in \mathcal{E} \cup \hat{A}_I \). Hence \( d \) is feasible. Furthermore, \( \Delta(d) = -\hat{c}^T d_l + 0 = -\hat{\lambda}_l < 0 \) and it follows from Lemma 3.1 that \( -d_l \) is a descent direction. \( \blacksquare \)

When \( Q_1 \cup Q_2 \neq \emptyset \) and \( \hat{A}_S \setminus \hat{Z}_I \neq \emptyset \), we construct a corresponding \( LL_1 \) problem to test the optimality conditions or to find a descent direction. This problem, which we call the derived problem for (1.1) at \( \hat{x} \), is:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in \hat{A}_S} |a^T_l y|, \quad y \in \mathbb{R}^n, \quad n < |\hat{A}|, \\
\text{subject to} & \quad -\hat{c}^T y = 1, \quad a^T_l y = 0, \quad i \in \mathcal{E}, \quad a^T_l y \leq 0, \quad i \in \hat{A}_I,
\end{align*}
\]

Theorem 3.1 Suppose \( \hat{x} \) is a (feasible) degenerate base point of (1.1) (assumed to be of full-rank, which is the case after introduction of artificial constraints as in Section 4).

1. If the feasible region of (3.2) is empty, then \( \hat{x} \) is optimal for (1.1).

2. If the feasible region of (3.2) is not empty, it has an optimal feasible base point \( y^* \). Then the necessary and sufficient conditions for \( \hat{x} \) to be a minimal point of \( S(x) \) are that \( y^* \) is not a descent direction for (1.1) at \( \hat{x} \), i.e.

\[
\Delta(y^*) = \hat{c}^T y^* + \sum_{i \in \hat{A}_S} |a^T_l y^*| = -1 + \sum_{i \in \hat{A}_S} |a^T_l y^*| \geq 0.
\]

3. If \( \Delta(y) < 0 \) for any feasible base point \( y \) of (3.2), then \( y \) is a feasible descent direction for (1.1) and a move from \( \hat{x} \) along \( y \) will reach another base point (of (1.1)) with lower value of \( S(x) \).

**Proof.** (1) If the feasible region of (3.2) is empty, then there can be no feasible descent direction at \( \hat{x} \) for (1.1). For if \( d \) is such a direction, then Lemma 3.1 implies that \( \hat{c}^T d < 0 \) and, therefore, the vector \( y = -d/\hat{c}^T d \) satisfies (3.2B). Thus \( \hat{x} \) is optimal for (1.1).

(2) From Theorem 2.2, if the feasible region is not empty, (3.2) has an optimal feasible base point \( y^* \). Now if \( \hat{x} \) is an optimal point of \( S(x) \), then any direction, in particular \( y^* \), is not descent at \( \hat{x} \). Conversely, if \( \Delta(y^*) \geq 0 \), we must show that for any feasible direction \( \tilde{d} \neq 0 \) (satisfying conditions \( a^T_l y = 0, \quad i \in \mathcal{E} \) and \( a^T_l y \leq 0, \quad i \in \hat{A}_I \))

\[
\Delta(\tilde{d}) = \hat{c}^T \tilde{d} + \sum_{i \in \hat{A}_S} |a^T_l \tilde{d}| \geq 0.
\]

If \( \hat{c}^T \tilde{d} \geq 0 \), then (3.4) is obviously true. Otherwise, let \( \tilde{y} = \tilde{d}/|\hat{c}^T \tilde{d}| \) and then \( -\hat{c}^T \tilde{y} = 1 \), implying that \( \tilde{y} \) is a feasible point of (3.2). Moreover \( \Delta(\tilde{d}) = \Delta(\tilde{y})|\hat{c}^T \tilde{d}| \) and, since \( y^* \) is an optimal point of (3.2),

\[
\Delta(\tilde{y}) = -1 + \sum_{i \in \hat{A}_S} |a^T_l \tilde{y}| \geq -1 + \sum_{i \in \hat{A}_S} |a^T_l y^*| = \Delta(y^*) \geq 0,
\]

which yields (3.4).

(3) Suppose that \( \Delta(\tilde{y}) < 0 \) for some feasible base point \( \tilde{y} \) of (3.2). Then, since \( \tilde{y} \) satisfies the conditions \( a^T_l y = 0, \quad i \in \mathcal{E} \) and \( a^T_l y \leq 0, \quad i \in \hat{A}_I \), it is feasible and descent at \( \hat{x} \) for (1.1). Moreover, since \( \tilde{y} \) is a base point of (3.2), there are \( n \) active planes, one of which is the equality constraint \( -\hat{c}^T y = 1 \). The remaining \( n - 1 \) active planes (with \( \tilde{y} \) substituted) have the form \( a^T_l \tilde{y} = 0, \quad i = l_1, l_2, \ldots, l_{n-1} \in \hat{A} \). Hence, a move from \( \hat{x} \) along \( \tilde{y} \) will be along the common edge of \( n - 1 \) planes \( a^T_l \hat{x} - b_i = 0, \quad i = l_1, \ldots, l_{n-1} \). Therefore, the move will reach another base point (of (1.1)). \( \blacksquare \)
Remark 3.1 The derived problem (3.2) could be degenerate. However, we now show that it can be solved by defining a sequence of similar derived problems at any degenerate point encountered. In practice, it is likely that this procedure will terminate at an early stage.

Suppose \( \hat{y}^{(1)} \) is a degenerate base point of (3.2) and the corresponding active set and cost vector are \( \hat{A}^{(2)} \) and \( \hat{c}_2 \), respectively. If \( \hat{c}_2 = 0 \), which includes the case that all of the equations in (3.2A) are active at \( \hat{y}^{(1)} \), then \( \hat{y}^{(1)} \) is an optimal base point. Otherwise, we can form a derived problem at \( \hat{y}^{(1)} \) for (3.2), called the second derived problem at \( \hat{x} \) for (1.1): denoting \( \hat{A}_S^{(2)} = \hat{A}^{(2)} \cap \hat{A}_S \), \( \hat{A}_I^{(2)} = \hat{A}^{(2)} \cap \hat{A}_I \) and \( \hat{c}_1 = \hat{c} \),

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in \hat{A}_S^{(2)}} |a_i^T y|, \quad y \in \mathbb{R}^n, \quad n < |\hat{A}^{(2)}|, \\
\text{subject to} & \quad -\hat{c}_I^T y = 1, \quad \hat{c}_S^T y = 0, \quad a_i^T y = 0, \quad i \in \mathcal{E}, \quad a_i^T y \leq 0, \quad i \in \hat{A}_I^{(2)}. \tag{3.5A}
\end{align*}
\]

If (3.5) has a degenerate base point, we can repeat this step and so on, thus defining a sequence of derived problems.

Theorem 3.2 Suppose the \( k \)-th derived problem has a degenerate base point \( \hat{y}^{(k)} \) with active set \( \hat{A}^{(k+1)} \) and cost vector \( \hat{c}_{k+1} \), and \( \hat{A}_S^{(k)} \neq \emptyset \), \( k = 1, 2, \ldots \) (otherwise see Lemma 3.4).

1. If \( \hat{c}_{k+1} = 0 \), then \( \hat{y}^{(k)} \) is an optimal base point of the \( k \)-th derived problem. (Then \( \hat{y}^{(k)} \) is used as in Theorem 3.1 to test for optimality in the \((k-1)\)-th derived problem, and (3.3) can be reduced to \( \Delta(\hat{y}^{(k)}) = -1 + \sum_{i \in \hat{A}_S} |a_i^T \hat{y}^{(k)}| \), where \( \hat{A}_S = \hat{A}_S^{(k)} \cap \hat{A}_S^{(k+1)} \).

2. Otherwise, we can form the \((k+1)\)-th derived problem, with \( \hat{A}_S^{(k+1)} = \hat{A}_S^{(k+1)} \cap \hat{A}_S^{(k)} \), and will have \( \text{rank}\{a_i, \; i \in \hat{A}_S^{(k+1)}\} < \text{rank}\{a_i, \; i \in \hat{A}_S^{(k)}\} \).

3. The procedure will stop either in the case the cost vector at a degenerate base point is zero or in the case where the \( t \)-th derived problem has \( \text{rank}\{a_i, \; i \in \hat{A}_S^{(t)}\} = 1 \) and can easily be solved.

Proof (1) This follows since all its multipliers are zero which satisfies the sufficient optimality condition.

(2) If, to the contrary, \( \text{rank}\{a_i, \; i \in \hat{A}_S^{(k+1)}\} = \text{rank}\{a_i, \; i \in \hat{A}_S^{(k)}\} \), then all of the equations in the \( k \)-th derived problem are active at \( \hat{y}^{(k)} \) (equal to 0) and so \( \hat{c}_{k+1} = 0 \) (since there is no inactive equation).

(3) Assume \( \text{rank}\{a_{j_l}, \; j_l \in \hat{A}_S^{(k)}\}, \; l = 1, 2, \ldots, s \} = 1 \), and then \( a_{j_l} = \gamma_{j_l} a_{j_1}, \; l = 2, \ldots, s \). If there is a degenerate base point \( \hat{y}^{(t)} \) for this problem and \( a_{j_1} \) is an active equation gradient at \( \hat{y}^{(t)} \), then all of the equation gradients are active and so \( \hat{c}_{t+1} = 0 \). If these gradients are all inactive, we have

\[
\hat{c}_{t+1} = \sum_{l=1}^{s} \sigma_{j_l} a_{j_l} = \sum_{l=1}^{s} \sigma_{j_l} \gamma_{j_l} a_{j_1} = \beta a_{j_1} \quad (\gamma_{j_1} = 1).
\]

We can assume \( \beta \neq 0 \) (for if \( \beta = 0 \), then \( \hat{c}_{t+1} = 0 \)). Now the \((t+1)\)-th derived problem is:

\[
\begin{align*}
\text{minimize} & \quad \sum_{l=1}^{s} |\gamma_{j_l}| |a_{j_l}^T y| \\
\text{subject to} & \quad -\beta a_{j_1}^T y = 1, \tag{3.6A} \\
& \quad \hat{c}_k^T y = 0, \quad k = 1, \ldots, t, \quad a_i^T y = 0, \quad i \in \mathcal{E}, \quad a_i^T y \leq 0, \quad i \in \hat{A}_I^{(t+1)}. \tag{3.6C}
\end{align*}
\]

Note that the set \( \mathcal{C} \) of constraint gradients in (3.6C) is equal to the set of active constraint gradients at \( \hat{y}^{(t)} \) for the \( t \)-th derived problem. Hence rank \( \mathcal{C} = n \) (since \( \hat{y}^{(t)} \) is a base point and \( a_{j_1} \) is inactive) and so rank
{C \cup \{a_{j_1}\}} = n. Therefore, there is a unique feasible base point for (3.6) (with one of the active inequality constraints becoming inactive) and this is the solution to (3.6). ■

4. Implementation of the RGA

Based on Theorem 3.1, we can solve a general \( LL_1 \) problem by applying Algorithm 2.1 at nondegenerate base points and applying an appropriate version to the derived problem at any degenerate base point. We now briefly describe the implementations of these algorithms. See [31] for details.

(I) Nondegenerate problem

(0) Finding a starting point \( x^{(0)} \)

For the \( UL_1 \) problem, an arbitrary point \( \hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)^T \) (e.g. \( \hat{x} = 0 \)) can be taken as the starting point, since it can be considered as a base point of the augmented problem: minimize \( S(x) \) subject to \( e_i^T x - \hat{x}_i \equiv x_i - \hat{x}_i \leq 0, \ i = 1, 2, \ldots, n \). Note that these constraints, called artificial constraints, are only a computational device; they are not enforced by the algorithm. The corresponding gradients, active set at \( \hat{x} \) and the base matrix are \( a_{m_1+i} = e_i, \ i = 1, 2, \ldots, n \), \( \hat{A} = \{m_1 + 1, \ldots, m_1 + n\} \) and \( \hat{A} = I \), respectively. Note that the gradients \( \{a_i, \ i = 1, 2, \ldots, m_1 + n\} \) have full rank \( n \) even if \( \{a_i, \ i = 1, 2, \ldots, m_1\} \) do not.

For some constrained \( LL_1 \) problems it is easy to find a starting feasible base point. For example, if the constraints are simply upper and lower bounds on the variables, i.e. \( v \leq x \leq u \), then both \( v \) and \( u \) are feasible base points and so either can be set as a starting point \( x^{(0)} \) (see Example 5.1). For the general constrained case, we solve an auxiliary LP problem as in [36,37] to find a feasible base point (possibly involving some artificial constraints).

By introducing artificial constraints, the rank of the matrix of given gradient vectors together with the artificial gradient vectors will equal \( n \), and therefore the augmented problem is nondegenerate by construction. Then for every base point \( x \) of the augmented problem, there are precisely \( n \) linearly independent active gradients \( a_i \). The active set at \( x \) can be written as \( A = A_S \cup E \cup A_I \cup A_0 \), where \( A_0 \) denotes a set of indices of artificial constraints.

(i) Test of optimality (See (I) of Algorithm 2.1)

If there are some artificial constraints, there is an extra optimality condition to check for the multipliers in (2.4). Let \( \max\{|\hat{\lambda}_j| : \ j_i \in A_0\} = |\hat{\lambda}_{q_0}| \). If \( |\hat{\lambda}_{q_1}| \leq 1, \hat{\lambda}_{q_2} \leq 0 \) and \( |\hat{\lambda}_{q_0}| = 0 \) are all satisfied, then \( \hat{x} \) is optimal by Theorem 2.1.

(ii) Choice of (feasible) descent direction

In addition to the directions in (2.5), if there are some artificial constraints, then another option for the descent direction is \( d = -\text{sign}(\hat{\lambda}_{q_0})\hat{a}_{q_0} \), if \( |\hat{\lambda}_{q_0}| \neq 0 \). This follows since, from \( \hat{A} = \hat{A}_S \cup E \cup \hat{A}_I \cup \hat{A}_0, \) \( \hat{A}_I, \hat{A}_0 \) and \( \hat{a}_{j_i}^T \hat{d}_q = \delta_{iq} j_i \in \hat{A}, \) we have \( \Delta S_0 = -|\hat{\lambda}_{q_0}| < 0, \) if \( |\hat{\lambda}_{q_0}| \neq 0 \). Note that any direction is regarded as feasible with respect to the artificial constraints. Clearly, more than one of the three choices of \( d \) may be possible. But if \( |\hat{\lambda}_{q_0}| \neq 0 \), we choose \( d = -\text{sign}(\hat{\lambda}_{q_0})\hat{a}_{q_0} \), since we want to delete the artificial constraints. Let \( q \in \{q_0, q_1, q_2\} \) be the index corresponding to the choice of \( d \). A move in this direction will leave the boundary plane \( P_{j_i} \) but remain on the intersection of the other \( n - 1 \) boundary planes \( P_{j_i}, \ j_i \in \hat{A} \setminus \{j_q\} \). Hence the index \( j_q \) will become inactive.
(iii) Choice of the step length

Our line search to minimize $S(\bar{x} + \alpha d)$ is the same as that in [24] and equivalent to the multiple pivot sequence in the Barrodale-Roberts algorithm (see [4,12]). It is implemented as follows. If $\alpha_{pi} < \delta_{ki}$ (see (2.6)), then we choose the step length $\alpha = \alpha_{pi}$. Otherwise, assume that the positive numbers in the set

$$\{\delta_j \mid 0 < \delta_j = -\frac{\bar{x}_j}{a_{pj}^T d} \leq \alpha_{pi}, \ j \in \hat{A}_S, \ a_{j}^T d \neq 0\}$$

are ordered as $0 < \delta_{k_1} \leq \delta_{k_2} \leq \ldots \leq \delta_{k_L}$, $t \geq 1$, so the $l$-th encountered base point on the inactive plane $P_{k_l}$, $k_l \in \hat{A}_S^c$, is $\bar{x} + \delta_{k_l} d$, $l = 1, 2, \ldots, t$. Let $\Delta S_0$ be as in (2.7) and let $\Delta S_l = -2\sigma_{k_l} a_{k_l}^T d = 2|a_{k_l}^T d|$, $l = 1, 2, \ldots, t$, (where the second equality follows since $a_{k_l}^T (\bar{x} + \delta_{k_l} d) - b_{k_l} = 0$ implies $\delta_{k_l} a_{k_l}^T d = -r_{k_l} (\bar{x})$ so $\text{sign}(a_{k_l}^T d) = -\sigma_{k_l}$). It is not hard to show that for $\alpha \in [\delta_{k_L}, \delta_{k_{L+1}}]$, $L = 0, 1, \ldots, t - 1$,

$$S(\bar{x} + \alpha d) = S(\bar{x}) + \alpha \Delta S_0 + \sum_{l=1}^L (\alpha - \delta_{k_l}) \Delta S_l.$$ 

Now, substituting $\alpha = \delta_{k_L}$ and $\alpha = \delta_{k_{L+1}}$, the difference of the results is

$$S(\bar{x} + \delta_{k_{L+1}} d) - S(\bar{x} + \delta_{k_L} d) = (\delta_{k_{L+1}} - \delta_{k_L}) \sum_{l=0}^L \Delta S_l.$$ 

If the sum on the right hand side is negative, then $S$ is smaller for a step length of $\delta_{k_{L+1}}$ compared to $\delta_{k_L}$. Applying this test for increasing $L$, $L = 0, 1, \ldots, t - 1$, as long as it succeeds, we find the step length $\delta_{k_l}$ that achieves the greatest reduction in the value of $S$. If $\delta_{k_l}$ is equal to one or more previous $\delta_{k_{l-1}}, \ldots, \delta_{k_{l-s}}$, then the next base point is degenerate with $s$ surplus active equations.

(iv) Update of the inverse matrix and vector $c$

Let $\bar{x}$ be the new base point resulting from the step in (iii) and let $p$ be the index of the new active equation or constraint. The new multipliers are defined by the inverse of the base matrix $\tilde{A}$ at $\bar{x}$ and the new vector $\bar{c}$ as in (2.4). To keep the implementation simple, we use the following direct update of the inverse (rather than a factored form of the base matrix). If the original coefficient matrix is ill-conditioned, the use of this update can lead to an inaccurate inverse, so it may be beneficial to employ a re-inversion strategy (see [31]).

From above, $\tilde{A}$ is obtained by replacing the $q$-th column vector of $\hat{A}$ by the vector $a_p$, where $q$ is one of $q_i$, $i = 0, 1, 2$, according to the choice of $d$. So $\tilde{A}$ can be represented as $\tilde{A} = \hat{A} + (a_p - a_{q_i}) e_q^T$, where $e_q$ is the $q$-th coordinate vector. Consequently, the inverse matrix $\tilde{A}^{-1}$ can be updated from $\hat{A}^{-1}$ using the Sherman-Morrison formula [10], rather than from $\tilde{A}$ itself. Letting the $i$-th row of $\hat{A}^{-1}$ and $\tilde{A}^{-1}$ be $\hat{d}_i^T$ and $\tilde{d}_i^T$, respectively, this gives

\[
\bar{d}_i = \begin{cases} 
\frac{1}{a_{p}^T d} \hat{d}_q, & i = q, \\
\frac{1}{a_{p}^T d} a_{q_i}^T \hat{d}_q \bar{d}_i - (a_{q_i}^T \hat{d}_q) \frac{1}{a_{p}^T d} \hat{d}_q, & i \neq q.
\end{cases}
\]

The vector $\bar{c}$ in (2.4) can also be efficiently updated; see [31] for details.

(II) Derived problem

We now describe a method to solve the derived problem at a degenerate base point.

(i) Starting point for the derived problem

Assume that $\Delta(-\text{sign}(\lambda_{q}) \bar{d}_i) \geq 0$, $\forall i \in Q_1$ (if $Q_1 \neq \emptyset$) and $\Delta(-\bar{d}_i) \geq 0$, $\forall i \in Q_2$ (if $Q_2 \neq \emptyset$); otherwise a descent direction at $\bar{x}$ has already been found. Let $\bar{d}$ be defined by either $\bar{d} = -\text{sign}(\lambda_{q}) \bar{d}_q$, for some $q \in Q_1$,
or \( \tilde{d} = -\tilde{d}_q \), for some \( q \in Q_2 \), and let \( y^{(0)} = \tilde{d}/|\tilde{c}^T\tilde{d}| = \tilde{d}/|\lambda_q| \). Then, since \(-\tilde{c}^Ty^{(0)} = 1\) and \( a_j^T\tilde{d}_q = 0, i \neq q \), the vector \( y^{(0)} \) is a base point for (3.2) in the set (which contains the constraint set of (3.2))

\[
D = \{ y | -\tilde{c}^Ty = 1, a_i^T y = 0, i \in \mathcal{E}, a_i^T y \leq 0, i \in \tilde{\mathcal{E}}_I \}. 
\]

Therefore, \( y^{(0)} \) is a starting feasible base point for (3.2) if and only if \( y^{(0)} \) satisfies \( a_i^T y^{(0)} \leq 0, \forall i \in \tilde{\mathcal{E}}_I \). If this difference set is empty (which occurs often in computational experiments and of course is true for the unconstrained case), we have a starting point in hand.

If \( y^{(0)} \) violates \( a_i^T y^{(0)} \leq 0, \forall i \in \tilde{\mathcal{E}}_I \), we use the same technique as in [36,37] to find a feasible base point for (3.2) starting from \( y^{(0)} \). Let \( \{ i | a_i^T y^{(0)} > 0, i \in \tilde{\mathcal{E}}_I \} = \{ i_1, i_2, \ldots, i_t \}, t \geq 1. \) We process the sequence of LP problems \( LP(k), k = 1, 2, \ldots, t, \) defined as follows:

\[
\begin{align*}
\text{minimize} & \quad a_i^T y, \\
\text{subject to} & \quad y \in D \quad \text{(see (4.2)),} \quad a_i^T y \leq 0, i \in \mathcal{I}^{(k)} = \{ i_1, \ldots, i_{k-1} \}.
\end{align*}
\]

Note that \( \mathcal{I}^{(1)} = \emptyset \). It follows from (4.2) that \( y^{(0)} \) is a feasible vertex for \( LP^{(1)} \). Since \( y^{(0)} \) does not satisfy \( a_i^T y \leq 0 \), we want to decrease the value of \( a_i^T y \) until it becomes less than or equal to zero, which explains the choice of objective function in (4.3A). We solve \( LP^{(k)} \) by using the active set method in [36,37] and terminate the iterations after finding either an optimal vertex \( y_k^* \) satisfying \( a_i^T y_k^* > 0 \) or a vector \( y_k \) satisfying \( a_i^T y_k \leq 0 \) (which is a starting feasible vertex for \( LP^{(k+1)} \)). If the former case occurs, we stop processing the LP problems and conclude that the problem (3.2) has no solution (see [36,37]) and therefore, from Theorem 3.1, \( \hat{x} \) is optimal for (1.1). If the latter case occurs, we start the iterations for \( LP^{(k+1)} \). If the procedure ends by finding \( y_k \) satisfying \( a_i^T y_k \leq 0 \), then \( y_k \) is a starting feasible base point for (3.2).

Having found a starting point for (3.2), we apply the algorithm in (I) to solve it. If during the iteration we find \( \Delta(y^*) < 0 \) at a feasible base point \( y^* \), then, from Theorem 3.1, we can claim that \( \hat{x} \) is not optimal for (1.1), stop the iteration for (3.2) and start the iteration for (1.1). If the algorithm applied to (3.2) yields an optimal solution \( y^* \) that satisfies \( \Delta(y^*) \geq 0 \), then, from Theorem 3.1, we conclude that \( \hat{x} \) is optimal for (1.1). If a degenerate point of (3.2) is encountered, we use the algorithm in (I) to solve the corresponding sequence of derived problems. From Theorems 3.1 and 3.2, it is clear that with the above extension to the algorithm in (I) to deal with degenerate points, there can be no cycling and the algorithm has finite convergence.

Because of the special form of the derived problem (3.2), the algorithm in (I) applied to (3.2) simplifies in certain ways as shown below. (See [31] for the detailed derivation of formulae and the proofs of Theorems 4.1 and Corollary 4.1.) Let \( y^{(k)} \) and \( V^{(k)} = [v_1^{(k)} \ldots v_n^{(k)}]^T, k = 0, 1, 2, \ldots, \) be the sequence of feasible base points of (3.2) and the corresponding inverse matrices resulting from the use of the algorithm in (I).

(ii) Inverse matrix at the starting point \( y^{(0)} \)

By using the Sherman-Morrison formula (see (4.1)) and \( \hat{D} = \hat{A}^{-1} = [\hat{d}_1 \hat{d}_2 \ldots \hat{d}_n]^T \), we obtain the inverse matrix at the starting base point \( y^{(0)} \):

\[
\hat{v}_q^{(0)} = \frac{1}{(-\hat{c})^T \hat{d}_q} \hat{d}_q = \frac{1}{-\lambda_q} \hat{d}_q \quad \text{and} \quad \hat{v}_i^{(0)} = \hat{d}_i - \frac{1}{(-\hat{c})^T \hat{d}_q} \hat{d}_q = \hat{d}_i + \lambda_i \hat{v}_q^{(0)}, \ i \neq q.
\]
(iii) The step length

From (2.6), the step length from $y^{(k)}$ to $y^{(k+1)}$ along the direction $v^{(k)}$ is

$$\gamma^{(k)} = \min\{\gamma_1^{(k)}, \gamma_2^{(k)}\} = -a^T y^{(k)}/a^T v^{(k)},$$

where $\gamma_1^{(k)}$ and $\gamma_2^{(k)}$ are the shortest steps to the boundary planes associated with inactive equations and inactive constraints, respectively.

From the following theorem, it can be seen that no extra computation is required to find the base points.

**Theorem 4.1** If $y^{(0)} = \hat{d}/|\hat{\lambda}_q|$ (see II(i)) is the starting base point for (3.2), then

$$y^{(k)} = v^{(k)q}, k = 0, 1, 2, \ldots.$$  

**Corollary 4.1** The formula for calculating $\Delta(y^{(k)})$ can be simplified to

$$\Delta(y^{(k)}) = -1 + \sum_{i \in \mathcal{A} \setminus \mathcal{Y}^{(k)}} |a^T_i y^{(k)}|,$$

where $\mathcal{Y}^{(k)}$ is the active set at $y^{(k)}$.

5. Numerical results

The RGA algorithm above was implemented in Fortran 77 in double precision, making use of BLAS subroutines from LINPACK. The RGA routine was applied to several examples on a Sun Sparc Ultra 10 and a DEC alpha computer at Murdoch University. The results are compared with those obtained using the NAG library routine E02GBF or the Netlib routine ACM563 (which use the PGA [7,8,9]), the Netlib routines ACM551 of Abdelmalek [1] (for $UL_1$) and ACM552 of Barrodale and Roberts [6] (for $UL_1$ and $LL_1$), the AFK routine of Armstrong, Frome and Kung [2] (for $UL_1$), with Fortran code given in [2], and the Huber approximation algorithm of Madsen, Nielsen and Pinar [21] (for $UL_1$) implemented in the Fortran 77 package LPASL1 [20]. Note that LPASL1 was actually designed to solve LP problems, using the dual $UL_1$ problem, so a given $UL_1$ problem must be entered as the corresponding dual LP problem. Example 5.4 compares the RGA routine with ACM552 on large scale $LL_1$ problems. In Examples 5.3 and 5.4, all the routines were compiled with the -O optimization option and run on the Sun Sparc Ultra 10, which has machine precision $2^{-52} \approx 2.22 \times 10^{-16}$. Some more numerical results are reported in [31].

**Example 5.1** (a problem (from [9], Example 4) with matrix of less than full rank) Compute the best $L_1$ solution to the overdetermined system of linear equations $Mx = b$, where $m_1 = 9$, $n = 5$ and

$$M = \begin{bmatrix} 5 & 3 & 4 & 12 & 4 \\ 9 & 7 & 3 & 19 & 13 \\ 6 & 6 & 0 & 12 & 12 \\ 9 & 9 & 7 & 25 & 11 \\ 3 & 0 & 1 & 4 & 2 \\ 8 & 1 & 8 & 17 & 1 \\ 1 & 9 & 8 & 18 & 2 \\ 3 & 1 & 1 & 5 & 3 \\ 0 & 9 & 3 & 12 & 6 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 4 \\ 2 \\ 7 \\ 2 \\ 7 \\ 3 \\ 5 \\ 3 \end{bmatrix}.$$  

Also solve the corresponding constrained problem with the bounds $-10^5 \leq x_i \leq 10^5$, $i = 1, 2, \ldots, 5$.

We used our routine on both problems, with starting point 0 for the former and starting feasible base point $10^5(1, 1, \ldots, 1)^T$ for the latter. The same starting points were used for the PGA routine E02GBF, but for ACM551 and ACM552 (which use an LP approach) we used the default starting points.
All the routines converged successfully, except for the following. For the unconstrained case, the routine E02GBF stopped after 5 steps with the message “abnormal exit from E02GBF: IFAIL=3”, implying that the problem is too ill-conditioned. The routine LPASL1 cannot be used directly on this problem since \( \text{rank}(M) < n \), and the same for AFK. The optimal value of \( S \) for the unconstrained and constrained problems is \( S = 15.945578231293 \), which was obtained by the other routines. The iteration numbers (steps) and solution vectors obtained are shown in Table 5.1. Note that RGA and ACM552 have the same iteration number and solution. This is as expected in the unconstrained (nondegenerate) case, since the algorithms are equivalent in this case [12,24] and the same starting point 0 was used for both.

### Table 5.1 Results for Example 5.1 (unconstrained U, bounded B)

<table>
<thead>
<tr>
<th>Steps</th>
<th>RGA, ACM552 (U&amp;B)</th>
<th>ACM551 (U)</th>
<th>E02GBF (B)</th>
<th>ACM563 (U&amp;B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>0.00000</td>
<td>0.2653</td>
<td>100000.0000</td>
<td>0.16599</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>-0.26531</td>
<td>0.00000</td>
<td>99999.7347</td>
<td>-0.09932</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0.00000</td>
<td>0.00000</td>
<td>-77904.0082</td>
<td>0.19048</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>0.43537</td>
<td>0.30272</td>
<td>-71047.5605</td>
<td>0.25714</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>-0.13605</td>
<td>-0.26871</td>
<td>-88952.1492</td>
<td>-0.12381</td>
</tr>
</tbody>
</table>

**Remark 5.1** Note that the solutions to the constrained problem are also solutions to the unconstrained problem for this example. So the bounds are in fact redundant. The rank of the coefficient matrix \( M \) is 3. It can be shown (see [31]) that the solutions to the unconstrained problem form an affine set \( \{ x | x = x^* + \beta_1 d_{i_1} + \beta_2 d_{i_2} \} \), where \( d_{i_1} = (1, 1, 0, -1/2, -1/2)^T \) and \( d_{i_2} = (0, 0, 1, -1/2, 1/2)^T \). This explains why the solutions found in Table 5.1 are so very different.

**Example 5.2** (a degenerate problem) Compute the best \( L_1 \) solution to the overdetermined system of linear equations \( Mx = b \), where \( m_1 = 12 \), \( n = 3 \) and

\[
M = \begin{bmatrix} -1.00 & 1.00 & 0.00 \\ -0.50 & 1.00 & -0.50 \\ 0.50 & -1.00 & 1.00 \\ 1.00 & -1.00 & 1.00 \\ 1.00 & -0.50 & 0.50 \\ 2.00 & -1.00 & 1.00 \\ 1.00 & -1.00 & 1.00 \\ 0.50 & -1.00 & 1.00 \\ 0.50 & -0.75 & 1.00 \\ 2.00 & -2.00 & 3.00 \\ 0.00 & -1.00 & 1.00 \\ 1.00 & -1.00 & 3.00 \end{bmatrix}, \quad b = \begin{bmatrix} -1.00 \\ -0.50 \\ 1.00 \\ 1.00 \\ 0.50 \\ 1.00 \\ 0.00 \\ -0.50 \\ -0.25 \\ -3.00 \\ 0.00 \\ -2.00 \end{bmatrix}.
\]

This problem is highly degenerate as there are 30 degenerate base points, including the optimal point, out of 61 base points in total. Using our routine, with the starting point \( (3, -3, -5)^T \), the problem was solved after 5 iterations. The last three of these involved solving the derived problem at degenerate base points. The computed optimal solution is \( x^* = (0.7142857, -0.7142857, -1.1428571)^T \) with \( S(x^*) = 5.78571 \). The same solution was obtained using ACM551 with 7 iterations, ACM552 with 3 iterations, AFK with 5 iterations and LPASL1 with 6 iterations.

The problem was also solved using the PGA routines E02GBF and ACM563. E02GBF terminated after 7 iterations but the optimal point \( x^* \) appeared four times (as \( x^{(k)}, k = 3, 5, 6, 7 \)) and the direction from
\( \mathbf{x}^{(3)} \) to \( \mathbf{x}^{(4)} \) was not descent since \( S(\mathbf{x}^{(4)}) > S(\mathbf{x}^{(3)}) \). The behaviour of ACM563 (both in single and double precision) was similar to that of E02GBF; the optimal point was repeated 3 times in 5 iterations for the single precision case and repeated 9 times in 13 iterations for the double precision case.

**Example 5.3** Compute the \( L_1 \) regression estimate for the model \( \mathbf{b} = M \mathbf{x} + \epsilon \) with several different sizes \( (m_1 \times n) \) of \( M \). The elements of \( M \) were pseudo-random numbers uniformly distributed on \([0, 1]\), as generated by the UNIX function \texttt{rand()}. The data values \( b_i \) were generated as the sum of the entries in the \( i \)-th row of the matrix plus an independent normal \( N(0, \sigma^2) \) pseudo-random error, with the two cases \( \sigma = 0 \) (i.e. no error) or \( \sigma = 3.0 \). We considered two types of sizes of matrix \( M \): (I) large number of equations \( m_1 \), from 500 to 5000, as in [17], and (II) large number of variables \( n \) with double the number of equations, as in [18].

In [17], problems of type (I) were used to compare several existing routines, including ACM551, AFK and a Barrodale-Roberts routine, and it was concluded that AFK performed the best. Here, we compare our RGA routine with ACM551, ACM552, AFK, ACM563 and LPASL1. Each of the systems of type (I) was replicated 100 times using different random elements in the matrix and different data values. Apart from ACM563, the routines obtained the same correct solution for every problem except one, in which the AFK routine did not terminate. The average iteration numbers and user times over the 100 replicates are listed in Table 5.2 (but the AFK entries with \# are averages over 99 replicates). ‘ACM1’, ‘ACM2’ and ‘ACM3’ stand for ACM551, ACM552 and ACM563, respectively. Note that (for both types (I) and (II)) the user time does not include the random data generation. In the case of LPASL1, it begins after the dual LP problem is entered and finishes when this is solved. For RGA and ACM552, the user time includes the time for computing the starting point.

<table>
<thead>
<tr>
<th>Size</th>
<th>Average iteration number</th>
<th>σ = 0</th>
<th>σ = 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RGA</td>
<td>ACM1</td>
<td>ACM2</td>
</tr>
<tr>
<td>500×15</td>
<td>0</td>
<td>60.2</td>
<td>23.4</td>
</tr>
<tr>
<td>1000×15</td>
<td>0</td>
<td>79.92</td>
<td>83.8</td>
</tr>
<tr>
<td>5000×5</td>
<td>0</td>
<td>25.81</td>
<td>30.98</td>
</tr>
<tr>
<td>5000×15</td>
<td>0</td>
<td>95.23</td>
<td>110.79</td>
</tr>
<tr>
<td></td>
<td>Average user time (seconds)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500×15</td>
<td>0.002</td>
<td>0.085</td>
<td>0.045</td>
</tr>
<tr>
<td>1000×15</td>
<td>0.006</td>
<td>0.192</td>
<td>0.128</td>
</tr>
<tr>
<td>5000×5</td>
<td>0.001</td>
<td>0.615</td>
<td>1.078</td>
</tr>
<tr>
<td>5000×15</td>
<td>0.050</td>
<td>3.122</td>
<td>2.260</td>
</tr>
</tbody>
</table>

As a starting point for the RGA routine, we used the \( L_2 \) (least squares) regression estimate \( \hat{\mathbf{x}} \) (with artificial constraints \( \mathbf{x} \leq \hat{\mathbf{x}} \)), which was computed using LINPACK routines for QR decomposition of \( M \). In the case with \( \sigma = 0 \), the \( L_2 \) solution \( \hat{\mathbf{x}} \) is the same as the \( UL_1 \) solution. This means the starting point for RGA was the optimal point, which is why the iteration number for RGA was always zero. Note that the \( UL_1 \) problems with \( \sigma = 0 \) are degenerate since the optimal point is a degenerate base point, but no other degenerate points were detected.

For ACM552, we used the initialization procedure in [22,5] of transforming the problem into finding the \( L_1 \) solution to the overdetermined system \( M \mathbf{y} = \mathbf{b} - M \hat{\mathbf{x}} \), where \( \mathbf{x} = \mathbf{y} + \hat{\mathbf{x}} \), which effectively makes the \( L_2 \) problems degenerate.
solution $\hat{x}$ the starting point (since the default starting point for $y$ is 0). Note that RGA and ACM552 have the same iteration numbers in the case with $\sigma = 3.0$ (as expected) but not in the case with $\sigma = 0$.

For the other routines, we used the default starting point, which for the PGA routine ACM563 is 0. In the case with $\sigma = 0$, no results for ACM563 are given since for most replicates the routine failed after reaching the maximum iteration number (6000) (though for the size $5000 \times 5$ it did succeed for all the replicates with average iteration number 24.9 and average user time 1.15). A check of the ACM563 objective function values in a failed problem showed that, in fact, the optimal solution was obtained after a small number of iterations, but the algorithm did not stop there, possibly due to cycling at the degenerate solution. Also ACM563 required much more memory than the other routines.

For the AFK routine, we used the default initial active set, which is $\{1, 2, \ldots, n\}$. In the case with $\sigma = 0$, this means the initial point was, in fact, the optimal base point $e = (1, 1, \ldots, 1)^T$, but AFK did not terminate at this point. For one replicate of size $5000 \times 5$ it failed to terminate completely. Another disadvantage of the AFK routine is that it requires that the initial active gradients be linearly independent.

In the case with $\sigma = 3.0$, the average user times for RGA, ACM552 and AFK are similar, and significantly better than for ACM551, ACM563 and LPASL1. In the case with $\sigma = 0$, RGA is the most efficient.

<table>
<thead>
<tr>
<th>Table 5.3</th>
<th>Results for type (II) problems</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\sigma = 0$</td>
</tr>
<tr>
<td>Size</td>
<td>RGA</td>
</tr>
<tr>
<td>480 x 240</td>
<td>0</td>
</tr>
<tr>
<td>720 x 360</td>
<td>0</td>
</tr>
<tr>
<td>1080 x 540</td>
<td>0</td>
</tr>
<tr>
<td>1620 x 810</td>
<td>0</td>
</tr>
<tr>
<td>2430 x 1215</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>coef</td>
</tr>
<tr>
<td></td>
<td>exp</td>
</tr>
<tr>
<td>Average iteration number</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>RGA</td>
</tr>
<tr>
<td>720 x 360</td>
<td>2.95</td>
</tr>
<tr>
<td>1080 x 540</td>
<td>12.1</td>
</tr>
<tr>
<td>1620 x 810</td>
<td>46.3</td>
</tr>
<tr>
<td>2430 x 1215</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>coef</td>
</tr>
<tr>
<td></td>
<td>exp</td>
</tr>
</tbody>
</table>

In [18], problems of type (II) were used to compare the Huber approximation algorithm of Madsen and Nielsen with the Barrodale-Roberts algorithm, and it was observed that the former is more efficient. Here we compare RGA with ACM551, ACM552, AFK, ACM563 and LPASL1, using the same starting points as for the type (I) problems. Each problem of type (II) was replicated and solved 10 times and the results are given in Table 5.3. In the case with $\sigma = 0$, no results are given for ACM563 since for all the problems it failed after reaching the maximum iteration number, again possibly due to cycling at the degenerate solution. For the larger problems, no results are given for ACM551 and AFK since their user times would have been too large. For the largest problem with $\sigma = 0$, LPASL1 failed on some replicates, and when it succeeded, the user time was about 3400 seconds.
In [18], it was observed that for each of the two algorithms studied there, the average iteration number could be approximately modelled by

\[
\text{iteration number} \approx A(m_1/n) n^\alpha = A(2) n^\alpha
\]  

(5.1)

with \( \alpha = 0.5 \) for the Huber approximation algorithm and \( \alpha = 1.25 \) for the Barrodale-Roberts algorithm. Our results show that this is a good model for all the routines tested in this example (as a log-log plot reveals) and it also serves as a good model for the average user times. Table 5.3 contains the coefficients (coef.) and exponents (exp.) for the model obtained by a least squares fit of the data on a log-log plot. The results for LPASL1 on the random data problems created here are not as good as those reported in [18] and [21]. In the case with \( \sigma = 3.0 \), ACM552 performed the best, followed by RGA, while in the case with \( \sigma = 0 \), RGA was the best. If the routines were compiled without optimization, RGA performed the best in both cases, followed by LPASL1.

**Example 5.4** (constrained problems) Compute the \( L_1 \) regression estimate for the model \( b = Mx + \epsilon \) under bounded variable constraints. The problems were of type (II) above with \( \sigma = 3.0 \), and each was replicated 10 times. The constraints were \( 0 \leq x_i \leq 2 \) for all \( i \). In this example we tested RGA, ACM552 and ACM563, and ACM563 failed on all problems of size greater than \( 30 \times 15 \). For the RGA routine, we tried two different choices for the starting point: the upper bound vector \( (2, 2, \ldots, 2)^T \) and the midpoint \( (1, 1, \ldots, 1)^T \) of the upper and lower bounds. In our code, for the case of upper (or lower) bounded variable constraints with the upper (or lower) bound vector as the starting point, we use the second choice of \( d \) in (2.5) for the first \( n \) iterations (since the solution is likely to have only a few of these bounds as active). For ACM552 we used the initialization procedure in [5] giving the same two starting points. The average iteration numbers and user times are given in Table 5.4.

<table>
<thead>
<tr>
<th>Size</th>
<th>Iter. no. RGA</th>
<th>Iter. no. ACM2</th>
<th>User time RGA</th>
<th>User time ACM2</th>
<th>Iter. no. Both</th>
<th>User time Both</th>
<th>Iter. no. RGA</th>
<th>User time RGA</th>
<th>Iter. no. ACM2</th>
<th>User time ACM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>200×100</td>
<td>399.9</td>
<td>439.5</td>
<td>0.61</td>
<td>0.85</td>
<td>274.8</td>
<td>0.46</td>
<td>0.54</td>
<td>0.54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>320×160</td>
<td>878.8</td>
<td>954.1</td>
<td>3.38</td>
<td>5.00</td>
<td>624.8</td>
<td>2.36</td>
<td>3.32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>480×240</td>
<td>1711.3</td>
<td>1838.0</td>
<td>16.27</td>
<td>22.84</td>
<td>1082.3</td>
<td>10.80</td>
<td>13.95</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>720×360</td>
<td>3114.9</td>
<td>3270.7</td>
<td>69.42</td>
<td>121.03</td>
<td>1888.3</td>
<td>41.26</td>
<td>70.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1080×540</td>
<td>5332.0</td>
<td>5624.5</td>
<td>293.65</td>
<td>494.01</td>
<td>3029.0</td>
<td>172.54</td>
<td>266.36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1620×810</td>
<td>7799.5</td>
<td>8308.9</td>
<td>1096.43</td>
<td>1650.01</td>
<td>4110.5</td>
<td>613.27</td>
<td>812.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2430×1215</td>
<td>10794.2</td>
<td>11510.3</td>
<td>3615.36</td>
<td>5107.17</td>
<td>5267.0</td>
<td>1915.49</td>
<td>2348.70</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that with the midpoint start, RGA and ACM552 have the same iteration numbers, while with the upper bound start, RGA has fewer iterations. This can be explained by the fact that the upper bound point is degenerate for the LP formulation used in ACM552 and some iterations are needed to overcome this. In this example, only a few of the upper (or lower) bounds are active at the solution, so the upper bound start requires more iterations than the midpoint start. For either starting point, RGA performed better than ACM552. It is interesting that for this example, (5.1) is not a good model for either RGA or ACM552, since a log-log plot of the iteration numbers or user times against \( n \) shows a significant downward curvature.
Acknowledgement
The authors are grateful to M. C. Pinar for sending the LPASL1 code and for his advice in using it.

References


