104/55

Lp NORM ESTIMATION PROCEDURES AND AN L1 NORM ALGORITHM FOR UNCONSTRAINED AND CONSTRAINED ESTIMATION FOR LINEAR MODELS

by

Buyong Kim

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Statistics

APPROVED:

August, 1986

Blacksburg, Virginia

Lp NORM ESTIMATION PROCEDURES AND AN L1 NORM ALGORITHM FOR UNCONSTRAINED AND CONSTRAINED ESTIMATION FOR LINEAR MODELS

by

Buyong Kim

Bradley O. Skarpness, Chairman

Statistics

(ABSTRACT)

When the distribution of the errors in a linear regression model departs from normality, the method of least squares seems to yield relatively poor estimates of the coefficients. One alternative approach to least squares which has received a great deal of attention of late is minimum L_p norm estimation. However, the statistical efficiency of a L_p estimator depends greatly on the underlying distribution of errors and on the value of p. Thus, the choice of an appropriate value of p is crucial to the effectiveness of L_p estimation.

Previous work has shown that L_1 estimation is a robust procedure in the sense that it leads to an estimator which has greater statistical efficiency than the least squares estimator in the presence of outliers, and that L_1 estimators have some desirable statistical properties asymptotically. This dissertation is mainly concerned with the development of a new algorithm for L_1 estimation and constrained L_1 estimation. The mainstream of computational procedures for L_1 estimation has been the simplex-type algorithms via the linear programming formulation. Other procedures are the reweighted least squares method, and nonlinear programming technique using the penalty function approach or descent method.

A new computational algorithm is proposed which combines the reweighted least squares method and the linear programming approach. We employ a modified Karmarkar algorithm to solve the linear programming problem instead of the simplex method. We prove that the proposed algorithm converges in a finite number of iterations. From our simulation study we demonstrate that our algorithm requires fewer iterations to solve standard problems than are required by the simplex-type methods although the amount of computation per iteration is greater for the proposed algorithm. The proposed algorithm for unconstrained L_1 estimation is extended to the case where the L_1 estimates of the parameters of a linear model satisfy certain linear equality and/or inequality constraints. These two procedures are computationally simple to implement since a weighted least squares scheme is adopted at each iteration. Our results indicate that the proposed L_1 estimation procedure yields very accurate and stable estimates and is efficient even when the problem size is large.

Acknowledgements

The author wishes to express his most sincere appreciation to Dr. Bradley O. Skarpness who, as advisory committee chairman, offered continued guidance and assistance during the course of research.

Special appreciation is also extended to Dr. Hanif D. Sherali for his inspiring comments and proofreading, and to other members of the advisory committee, Dr. Robert V. Foutz, Dr. Richard G. Krutchkoff, Dr. Marion R. Reynolds, Jr., for their assistance and suggestions throughout the entire graduate program.

The author expresses special gratitude to Prof. Ki-Joong Yoon, Dr. Jongbin Kim, Dr. Suk-Bum Yoon for their encouragement and concern throughout the undergraduate and graduate study.

The author is forever indebted to his parents, parents-in-law, wife whose love, guidance, patience and understanding was a constant source of encouragement during the entire graduate program.

Table of Contents

List of Illustrations

List of Tables

I. INTRODUCTION

The method of least squares is usually regarded as the most suitable technique for estimating the coefficients in a linear regression model when the errors follow a normal distribution and the regressors are orthogonal. In fact, the principle of least squares is usually defended on the basis of the assumption that the errors are normally distributed. In many practical situations, however, it may happen that some model assumptions fail to hold. Furthermore, the assumption of normality of errors in a regression model can not always be taken for granted. When the distributions of errors are non-normal, the least squares method may yield relatively poor estimates of the regression coefficients. A particular problem is estimation in the presence of outliers. If one has a problem with any significant number of outliers, then the underlying error distribution may not be normal, and it has been shown that the least squares method is rather inadequate.

This has led to the development of an alternative approach which is called minimum L_p norm estimation by Rice and White (1964). The family of L_p norm estimates which minimize the sum of the p-th powers of the absolute deviations of the observations from the regression hyperplane is considered as a generalization of the least squares method. If we choose $p = 2$, then minimum L_p norm estimation is the least squares method. Two other values of p that are commonly used are 1 and ∞. For $p = 1$ the criterion is least absolute deviations, and for $p = \infty$ it is the least maximum deviation criterion.

During the past few years several computational methods have been proposed which allow computer solution of problems in L_p estimation, especially in the area of L_1 estimation for a linear model. Subsequently, we first review the theory of minimum L_p norm estimation including the computational formulation in Chapter II. In addition, several properties of L_p estimators are examined for various choices of p.

We next focus our attention to L_1 estimation which we feel is a viable alternative to least squares (L_2) estimation for certain situations. In Chapter III, we outline these situations, and review the statistical properties of the L_1 estimator and some statistical inference procedures for large samples. Also, recent advances in computational algorithms for unconstrained L_1 estimation are surveyed, and a new algorithm based on a modified Karmarkar's algorithm for linear programming is proposed. We then prove that the proposed algorithm convergences and a strong duality theorem holds. In Chapter IV, several computational methods for constrained L_1 estimation are examined, and a new procedure is proposed which is an extension of our proposed algorithm for unconstrained L_1 estimation. A proof of the convergence and strong duality are given. A comparison of the computational behavior of various algorithms is conducted in Chapter V, from the viewpoint of computational efficiency, and numerical accuracy and stability, for the case of full rank design matrices. We conclude with a summary of our results derived from this study in Chapter VI.

II. Lp NORM ESTIMATION

Consider the problem of estimating the parameters of a linear regression model,

$$y = X\beta + \varepsilon \tag{2.1}$$

where y is an n-dimensional vector of observations, X is an $n \times m$ matrix of known quantities on m independent variables, β is a vector containing m unknown parameters, and ε is a vector of n random errors which are independently and identically distributed with distribution f.

It is well known the least squares estimator, $\hat{\beta} = (X'X)^{-1}X'y$, of the linear model (2.1) is unbiased under the assumption that error vector has mean zero. In fact, if $E(\varepsilon) = 0$ and $E(\varepsilon\varepsilon') = \sigma^2 I$, that is, the elements of ε are uncorrelated and have the same variance, the least squares estimator has the property of being the best linear unbiased estimator (BLUE) by the Gauss-Markov theorem. In general, the L_p norm estimator ($p \neq 2$) is not a linear estimator, but it may have a smaller variance than the least squares estimator when the errors do not follow a normal distribution.

2.1. COMPUTATIONAL PROCEDURES

Although other computational methods are available and have been investigated with success, we shall concentrate our efforts in this dissertation on minimum L_p norm estimation procedures based on mathematical programming technique. This method has three main advantages over other techniques in this area; relatively high speed, good degree of flexibility, and easy constraints imposition [Kiountouzis(1973)].

The following preliminary results are used throughout this dissertation.

Definition 2.1: A real valued function $\|.\|$ defined on a vector space X is called a norm on X if it satisfies the following three properties;

(*i*) $\|x\| \geq 0$ for each $x \in X$, and $\|x\| = 0$ if and only if $x = 0$.

(*ii*) $\|ax\| = |a|.\|x\|$ for all $x \in X$ and $a \in R$.

(*iii*) $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in X$.

Definition 2.2: L_p norm $(1 \leq p < \infty)$ of an n vector x is, $\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$.

Definition 2.2.1: L_∞ norm of an n vector is, $\|x\|_\infty = \max |x_i|$ for $i = 1, ..., n$.

Lemma 2.1: If $1 \leq p \leq q$, then $\|x\|_p \geq \|x\|_q$.

Definition 2.3: The L_p norm estimate of β for the linear model (2.1) is denoted by $\hat{\beta}$. It is the value of $\beta \in E^m$ which minimizes the discrete L_p norm,

$$\|y - X\beta\|_p = (\sum_{i=1}^{n} |y_i - X_i\beta|^p)^{1/p}$$

where X_i is the i-th row of X. In particular, when $p = \infty$

$$\|y - X\beta\|_\infty = \text{maximum } |y_i - X_i\beta|$$

In this dissertation we shall only consider values of $p \geq 1$. Some noted examples are, the absolute value estimator or L_1 estimator of β, $\hat{\beta}$ = arg min $\|y - X\beta\|_1$, the least squares estimator or L_2 estimator of β, $\hat{\beta}$ = arg min $\|y - X\beta\|_2$, and the Chebyshev estimator or L_∞ estimator of β , $\hat{\beta}$ = arg min $\|y - X\beta\|_\infty$.

The exponent $1/p$ in the L_p norm can be ignored and the computing problem becomes one of minimizing the nonlinear function

$$\Phi(\beta) = \sum_{i=1}^{n} |y_i - X_i\beta|^p \tag{2.2}$$

The problem (2.2) can be converted into a convex programming problem by letting e_i^+ denote the positive vertical deviations and e_i^- denote the negative vertical deviations about any fitted hyperplane as follows,

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{n} (e_i^{+p} + e_i^{-p}) \\ &\text{subject to} && X\beta + Ie^+ - Ie^- = y \\ & && e^+, e^- \geq 0 \\ & && \beta \text{ unrestricted} \end{aligned} \tag{2.3}$$

An important decision to be made at this point is the choice of L_p norm to be used to define the parameter estimates. It should be noted that the statistical efficiency of the estimators depends greatly on the value of p. The problem of choosing a p value is discussed in detail later in this chapter.

2.2. UNBIASEDNESS

Harvey (1978) showed the unbiasedness of L_p estimator for $1 < p < \infty$, under the assumption that its first moment exists and the errors in the model are symmetrically distributed. In this situation uniqueness is assured. Hence, the L_p estimator will be unbiased regardless of the computational procedure used to estimate it.

In a recent paper, Sposito (1982) investigated the unbiasedness of the L_p estimator ($p \geq 1$) under the assumptions that the error distribution is symmetric with mean zero, and the model is of arbitrary rank. To obtain the unbiased L_p estimators, Sposito considered any initial antisymmetrical estimator, β^0, such that

$$\beta - \beta^0(\varepsilon) = -[\beta - \beta^0(-\varepsilon)] \tag{2.4}$$

Now if β and β^0 are each written as the differences of two nonnegative quantities, say, $\beta = \beta^+ - \beta^-$ and $\beta^0 = \beta^0_1 - \beta^0_2$, then we have the following two equivalent convex programming problems from problem (2.3),

P_1; minimize $\sum_{i=1}^{n}(e_i^{+p} + e_i^{-p})$

subject to

$$[X \; -X \; I \; -I]\begin{bmatrix} B_1 \\ B_2 \\ e^+ \\ e^- \end{bmatrix} = [y - X\beta^0]$$

$$B_1,\; B_2,\; e^+,\; e^- \geq 0$$

P_2; minimize $\sum_{i=1}^{n}(e_i^{+p} + e_i^{-p})$

subject to

$$[X \; -X \; I \; -I]\begin{bmatrix} B_2 \\ B_1 \\ e^- \\ e^+ \end{bmatrix} = -\,[y - X\beta^0]$$

$$B_1,\; B_2,\; e^+,\; e^- \geq 0$$

where $B_1 = \beta^+ + \beta_2^0$ and $B_2 = \beta^- + \beta_1^0$. After β^0 has been determined, the remainder of the algorithm is,

(*i*) Select one of the problems P_1 and P_2 with respective probabilities 1/2.

(*ii*) Determine an optimal solution of the problem selected by any convex simplex procedure, and then obtain $\bar{\beta} = \hat{B}_1 - \hat{B}_2$.

(*iii*) Estimate β by $\hat{\beta} = \bar{\beta} + \beta^0$.

Under the assumption that errors are symmetrically distributed and from the relationship (2.4), we know that $y(-\varepsilon) - X\beta^0(-\varepsilon) = -[y(\varepsilon) - X\beta^0(\varepsilon)]$. And hence $P_1(-\varepsilon)$ and $P_2(\varepsilon)$ are identical except for the ordering of the variables, also $P_1(\varepsilon)$ and $P_2(-\varepsilon)$ are identical. Now if we let $\bar{\beta} = \hat{B}_1 - \hat{B}_2$ denote an optimal solution obtained by any procedure, then it follows that

$$\begin{aligned} \hat{B}_1\{P_1(-\varepsilon)\} - \hat{B}_2\{P_1(-\varepsilon)\} &= -[\hat{B}_1\{P_2(\varepsilon)\} - \hat{B}_2\{P_2(\varepsilon)\}] \\ \hat{B}_1\{P_1(\varepsilon)\} - \hat{B}_2\{P_1(\varepsilon)\} &= -[\hat{B}_1\{P_2(-\varepsilon)\} - \hat{B}_2\{P_2(-\varepsilon)\}]. \end{aligned} \tag{2.5}$$

Therefore, it can be shown that $\hat{\beta}$ is unbiased.

$$\begin{aligned} E(\hat{\beta}) &= E(\hat{B}_1 - \hat{B}_2 + \beta^0) \\ &= E(\bar{\beta} + \beta^0) \\ &= E\{(\bar{\beta} \mid \varepsilon) + (\bar{\beta} \mid -\varepsilon)\} + \beta \\ &= E\{(\bar{\beta} \mid \varepsilon, P_1) \times 1/2 + (\bar{\beta} \mid \varepsilon, P_2) \times 1/2\} \\ &\quad + E\{(\bar{\beta} \mid -\varepsilon, P_1) \times 1/2 + (\bar{\beta} \mid -\varepsilon, P_2) \times 1/2\} + \beta \\ &= \beta \end{aligned}$$

Money et al.(1982), in their simulation study, showed that the L_p norm estimators are unbiased for all $p \geq 1$ when the error distribution is symmetric.

2.3. STATISTICAL EFFICIENCY

Statistical efficiency of the L_p estimator was investigated by Rice and White (1964). The results of their experiments show that (*i*) L_2 estimator is efficient when error distributions are normal, triangle or square. (*ii*) L_∞ estimator is efficient when errors follow uniform distribution. (*iii*) L_1 estimator is efficient when error distributions are Laplace or Cauchy.

Money et al.(1982) examined the sample variance of each of the regression coefficients for several choice of p, and for several error distributions. From their simulation study it is apparent that for $p = 2$, the sample variances are approximately the same for all error distributions with the exception of the Cauchy. The reason is that the true covariance matrix of the L_2 estimate is given by $\sigma^2(X^tX)^{-1}$ which is independent of the error distribution, provided $E(\varepsilon) = 0$ and $E(\varepsilon\varepsilon^t) = \sigma^2 I$. Also, they show that as the kurtosis of the error distribution increases, a choice of $p < 2$ results in estimates with smaller sample variances than those obtained using least squares. Examination of their simulation studies reveals that as the sample size increases, L_2 estimator becomes less efficient if the error distribution is not normal.

2.4. CHOICE OF p VALUE

It is very important to choose an appropriate value of p for various types of error distributions. Several criteria for the choice of p have been proposed and examined through simulation studies. Rice and White (1964) showed that the statistical efficiency of L_p estimation depends greatly on the underlying distribution of errors and on the p value, and that there is a large variation in the effectiveness of various norms and no single norm is the best in all situations. These facts are demonstrated in Table 1. They considered the one parameter problem for L_1, L_2 and L_∞, and for five error distributions; uniform, triangle, normal, Laplace and Cauchy. Table 1 gives the variances of the resulting estimates for these combinations [the entries for the L_1 norm are from Siddiqui (1962), and the entries for the L_∞ norm are from Cramer (1946)].

Table 1. Asymptotic values of

the variances of L_p estimators

Error distribution	L_1 norm	L_2 norm	L_∞ norm
Uniform	$\frac{1}{4n}$	$\frac{1}{12n}$	$\frac{1}{2n^2}$
Triangle		$\frac{1}{6n}$	$\frac{4-\pi}{4n}$
Normal	$\frac{\pi}{2n}$	$\frac{1}{n}$	$\frac{\pi^2}{12 \ln n}$
Laplace	$\frac{1}{2n}$	$\frac{2}{n}$	$\frac{\pi^2}{12}$
Cauchy	$\frac{\pi^2}{4n}$	∞	∞

It should be noted from this table that for an error distribution with sharply defined extremes one should use the L_∞ norm, for error distribution with long tails one should use the L_1 norm.

Forsythe (1972) stated that a reasonably fast method for estimating regression coefficients by minimizing the L_p norm is available, and that the L_p $(p = 1.5)$ estimator is better than least squares when the error distribution is contaminated which can produce long tailed or skewed error distributions, and is close to least squares when the errors have a normal distribution.

Harter (1977) suggested that if the sample kurtosis k of the error distribution is larger than 3.8, then use L_1 estimation, if $2.2 \le k \le 3.8$, then use L_2 estimation, and if $k < 2.2$, then use L_∞ estimation.

Money et al.(1982) examined the L_p estimators for the values of $p = 1.00, 1.25, 1.50, 1.75, 2.00$, and ∞ in their simulation study. They used statistical symmetric distributions to generate the random errors; uniform distribution $(k = 1.8)$, normal distribution $(k = 3)$, contaminated normal distributions $(k = 3.5, 4.0, 4.5, 5.0, 5.5)$, Laplace distribution $(k = 6)$, Cauchy distribution (k is undefined). From their work, we can see the following relationships; (*i*) The L_p norm estimators are unbiased for all $p \ge 1$ when the error distribution is symmetric. (*ii*) For $p \ne 2$ the sample variances are not constant but vary according to the error distribution. In addition, for any given error distribution, these variances are dependent on the choice of p. As the kurtosis of the error distribution increases, a choice of $p < 2$ results in estimates with smaller sample variances than those obtained using L_2 estimation. (*iii*) If minimum generalized variance is assumed to be a suitable criterion for choosing p , the longer tailed the error distribution (that is, the larger the kurtosis), the smaller the optimal p. (*iv*) It is proposed that the functional relationship, $p = \frac{9}{k^2} + 1$, be used in practice to determine a suitable p, provided the error distribution is symmetric.

Sposito, Hand and Skarpness (1983) examined the efficiency of the sample kurtosis in obtaining L_p estimates and establish guidelines for determining an appropriate value of p based on the kurtosis of the error distribution. They suggested the following; (*i*) For large sample sizes ($n \geq 200$), the relationship $p = \frac{6}{k}$ yields a value for p which is reasonably close to the optimal value of $p \in [1, 2]$. (*ii*) For small sample sizes and $p \in [1, \infty)$, use the L_∞ estimator for $k < 2.2$, L_2 estimator for $2.2 \leq k \leq 3.0$, and Forsythe's compromise rule, $p = 1.5$, for $3 < k < 6$.

III. L1 NORM ESTIMATION

It is well known that if the distribution of errors is normal, the maximum likelihood estimates of the regression coefficients are the L_2 estimates for a fixed value of σ. On the other hand, it has been shown that if the elements of ε are independently and identically distributed according to a double exponential distribution, then the L_1 estimates are equivalent to the maximum likelihood estimates and are unbiased, efficient and asymptotically normally distributed. This property of the L_1 estimate along with the results of Rice and White (1964) on the asymptotic variances for the error distributions discussed in section 2.4 would strongly suggest that L_1 estimation is preferable to least squares estimation in certain situations. For example, when the distribution of the errors departs from normality and follows some long-tailed distribution, then L_1 estimation yields estimates with smaller variance than least squares. In particular, in situations where outlying observations are present, we can estimate the coefficients of a linear model by the L_1 estimation procedure which is robust in the sense that it leads to an estimator which has greater statistical efficiency than the least squares estimator. Given that L_1 estimation is a reasonable alternative to least squares, we examine its sampling properties, statistical inference procedures and some computational algorithms.

3.1. STATISTICAL PROPERTIES

3.1.1. *Small Sample Properties*

Statistical properties of L_1 estimators in small samples are not known exactly. However, a number of simulation studies have shown that L_1 estimators are more efficient than least squares estimators in small samples when the errors have Cauchy, Laplace, stable Paretian distribution with the characteristic exponent less than or equal to 1.5, and certain contaminated normal distributions. These studies on the efficiency of L_1 estimation have been done by Rice and White (1964), Blattberg and Sargent (1971), Kiountouzis (1973), Rosenberg and Carlson (1977), and Pfaffenberger and Dinkel (1978).

Wilson (1978) showed that when the least squares assumptions hold, the efficiency of L_1 estimators relative to least squares estimators is about 80% in terms of relative efficiency defined by $\hat{\sigma}_{L2}(\hat{\beta}_j)/\hat{\sigma}_{L1}(\hat{\beta}_j)$. Glahe and Hunt (1970) and Hunt, Dowling and Glahe (1974) demonstrated that the L_1 estimator performs well in simultaneous equation models with long-tailed error distributions. Brecht (1976) proved that the L_1 estimator is relatively more robust than least squares when a model has a normal error distribution and unusually large errors in the explanatory variables.

Rosenberg and Carlson (1977) revealed that the L_1 estimator has a significantly smaller standard error than the least squares estimator when the error distribution has high kurtosis, and it is almost exactly normally distributed in the presence of high-kurtosis errors. They also showed that if the errors follow a symmetric distribution, then $(\hat{\beta} - \beta)$, is approximately normally distributed with mean zero and covariance matrix $\lambda^2(X'X)^{-1}$, where $\frac{\lambda^2}{n}$ is the variance of the median of a sample of size n from the error distribution.

3.1.2. Large Sample Properties

Bassett and Koenker (1978) noted that the sampling distributions of L_1 estimators are asymptotically normal with mean β and variance $\lambda^2(X'X)^{-1}$ (as suggested by Rosenberg and Carlson). They proved the following theorem.

Theorem 3.1: Consider the linear regression model where the errors are independently and identically distributed with cdf F. Let $\{\hat{\beta}_\tau\}$ denote a sequence of unique solutions to the L_1 norm problem with the matrix of independent variables denoted by X_τ . Assume that (*i*) F is continuous and has continuous and positive density f at the median, (*ii*) $\lim(n^{-1}X_\tau^t X_\tau) = Q$ as $n \to \infty$, where Q is a positive definite matrix. Then, $\sqrt{n}\,(\hat{\beta}_\tau - \beta)$ converges in distribution to a normal random vector with mean zero and variance matrix $\lambda^2 Q^{-1}$, where $\frac{\lambda^2}{n}$ is the asymptotic variance of the sample median of random samples from the distribution F.

They also proved that L_1 estimators are consistent. The only assumption required in this theorem is that the errors are independently and identically distributed. This result implies that L_1 estimation is preferable to least squares for any error distribution for which the sample median is superior to the mean as an estimator of location.

3.1.3. Unbiasedness

A computational method for obtaining unbiased L_1 estimators, when the error distribution is symmetric with mean zero and the estimates may not be unique, was given by Sielken and Hartley (1973). Also, the procedure of Sposito (1982) discussed in section 2.2 can be employed to show the unbiasedness of an L_1 estimator. Under the assumption that errors are symmetrically distributed, it is necessary to employ an initial antisymmetrical estimator of β such as (2.4). McCormick and Sposito (1976) showed that if the least squares estimator is available, then this estimator can be used to reduce the number of iterations in L_1 estimation. Since $y = X\beta + \varepsilon$, it follows that if β^0 is the least squares estimator, then

$$\beta^0(\varepsilon) = (X^tX)^{-1}X^t(X\beta + \varepsilon) = \beta + (X^tX)^{-1}X^t\varepsilon$$
$$\beta^0(-\varepsilon) = (X^tX)^{-1}X^t(X\beta - \varepsilon) = \beta - (X^tX)^{-1}X^t\varepsilon.$$

Therefore, the relationship (2.4) is verified and $E(\beta^0) = \beta$.

From the convex programming problems for the unbiasedness of L_p estimation, two equivalent linear programming problems for L_1 estimation can be formulated,

P_1; minimize $\ell^t(e^+ + e^-)$
subject to

$$[X \ -X \ I \ -I]\begin{bmatrix} B_1 \\ B_2 \\ e^+ \\ e^- \end{bmatrix} = [y - X\beta^0]$$

$$B_1,\ B_2,\ e^+,\ e^- \geq 0$$

P_2; minimize $\ell^t(e^+ + e^-)$
subject to

$$[X \ -X \ I \ -I]\begin{bmatrix} B_2 \\ B_1 \\ e^- \\ e^+ \end{bmatrix} = -\,[y - X\beta^0]$$

$$B_1,\ B_2,\ e^+,\ e^- \geq 0$$

where $B_1 = \beta^+ + \beta_2^0$ and $B_2 = \beta^- + \beta_1^0$, and ℓ is an n-dimensional vector of all ones.

Now the unbiased L_1 estimator can be obtained by using the unbiased L_p estimation procedure. And the unbiasedness of $\hat{\beta}$ can be shown by the relationship (2.5). Note that the unbiasedness of $\hat{\beta}$ is independent of the actual algorithm used to solve the linear programming problems P_1 or P_2.

3.1.4. Nonuniqueness

As with least squares, the L_1 criterion does not lead to unique estimates for non-full rank models. This problem of nonuniqueness in L_1 estimation is more serious than the nonuniqueness that may arise in least squares problems. In least squares the concept of estimability makes nonuniqueness less disturbing, and the uniqueness of the residuals gives a heuristic appeal to the ensuring inferential procedures. Uniqueness in the least squares estimation depends on the design matrix X, and additional observations can not destroy uniqueness once it exists for a given data set. On the other hand, in L_1 estimation there does not appear to be any simple characteristic of X that determines uniqueness. A given set of data can yield a unique L_1 estimate, and then an additional observation can give nonunique estimates for the same model. Furthermore, the residuals from L_1 estimation are not necessarily unique. Harter (1977) discussed the question of nonuniqueness of the L_1 estimate, and proposed a method for finding a compromise regression line when the L_1 regression is not unique.

3.2. STATISTICAL INFERENCES

Statistical inferences on L_1 estimators which are mainly based on Theorem 3.1 have been proposed by Dielman and Pfaffenberger (1982). We first consider the estimation of λ . Cox and Hinkley (1974) suggested one approach to estimating λ in large samples as follows. It follows from Cramer (1946) that

$$\lambda = \frac{f_m}{2}, \text{ asymptotically}$$

where $f_m \doteq \dfrac{1}{f(m)}$, $f(m)$ is the value of the distribution function for the true residuals, e_t, evaluated at the median, m, of the distribution. A consistent estimate of f_m is

$$\hat{f}_m = \frac{\tilde{e}_t - \tilde{e}_s}{(t - s)/n}$$

where the $\tilde{e}_t$ are the ordered residuals from $e = y - X\hat{\beta}$, and $t = [n/2] + \nu$, $s = [n/2] - \nu$, [.] indicates the integer portion of the numbers and ν is an integer chosen to spread t and s symmetrically about the index of the median sample residual, $(t - s)$ should be kept quite small, and $\tilde{e}_t \neq 0$ or $\tilde{e}_s \neq 0$. Subsequently, a consistent estimator of λ is

$$\hat{\lambda} = \frac{\tilde{e}_t - \tilde{e}_s}{2(t - s)/n}.$$

Now, confidence interval formulations and hypothesis testing procedures can be developed on the basis of the large sample properties of the L_1 estimator. A point estimate of $r'\beta$ is $r'\hat{\beta}$ and a $(1 - \alpha)$ 100% confidence interval for $r'\beta$ is given by,

$$r^t\hat{\beta} \pm z_{\alpha/2}\hat{\lambda}[r^t(X^tX)^{-1}r]^{1/2}$$

where $z_{\alpha/2}$ denotes the percentile from the standard normal distribution. In particular, the confidence interval for a single component of β is

$$\hat{\beta}_i \pm z_{\alpha/2}\hat{\lambda}(X^tX)^{-1/2}_{ii}$$

where $(X'X)^{-1/2}_{ii}$ denotes the square root of the i-th diagonal element of the $(X'X)^{-1}$ matrix.

The point estimate of the mean response, y_0 , corresponding to input vector x_0 is given by

$$\hat{y}_0 = x_0^t\hat{\beta}$$

and a $(1 - \alpha)$ 100% confidence interval for the mean response is

$$\hat{y}_0 \pm z_{\alpha/2}s(\hat{y}_0)$$

where $s^2(\hat{y}_0) = \hat{\lambda}^2[x_0^t(X^tX)^{-1}x_0]$.

A prediction interval for a new observation, y', corresponding to an observed input vector x_0 is

$$\hat{y}' \pm z_{\alpha/2}s(\hat{y}')$$

where $s^2(\hat{y}') = \hat{\lambda}^2[1 + x_0^t(X^tX)^{-1}x_0]$. Note that the confidence interval for y_0 and y' depend on the asymptotic normality of y, and this assumption may not be warranted in many applications of L_1 estimation.

Consider testing the null hypothesis

$$H_0; r^t\beta = h \text{ vs. } H_1; r^t\beta \neq h$$

The test statistic is

$$z = \frac{r^t\hat{\beta} - h}{\hat{\lambda}[r^t(X^tX)^{-1}r]^{1/2}}$$

In particular, the test statistic for the null hypothesis $H_0; \beta_i = 0$ is

$$z = \frac{\hat{\beta}_i}{\hat{\lambda}(X^t X)_{ii}^{-1/2}}$$

The test statistic for the null hypothesis H_0; $\beta_2 = \beta_3 = , \ldots , \beta_m = 0$ is

$$\chi^2 = \frac{(\hat{\beta}' - \beta')X'^t X'(\hat{\beta}' - \beta')}{\hat{\lambda}^2}$$

where ['] denotes that the first column of ones in the X matrix and β_1 have been deleted. This statistic is asymptotically a χ^2 distribution with $m - 1$ degrees of freedom, since $\hat{\beta}' - \beta'$ is asymptotically normal with mean zero and covariance matrix $\lambda^2(X^t X)^{-1}$ using Corollary 2.2 of Searle (1971,p58) and the fact that $\hat{\lambda}$ is a consistent estimator of λ .

3.3. COMPUTATIONAL PROCEDURES

There are two general types of methods for computing L_1 estimates. One type of method makes use of linear programming; some examples are Wagner (1959), Fisher (1961), Robers and Robers (1973), Barrodale and Roberts (1973), Armstrong, Frome and Kung (1979), and Abdelmalek (1980). These algorithms use the following linear programming problem to compute L_1 estimate,

$$\begin{aligned} \text{minimize} \quad & \ell^t e^+ + \ell^t e^- \\ \text{subject to} \quad & X\beta + Ie^+ - Ie^- = y \\ & e^+, e^- \geq 0 \\ & \beta \text{ unrestricted} \end{aligned} \tag{3.1}$$

where ℓ is an n-dimensional vector of all ones, and I is the n-th order identity matrix. The components of e^+ are vertical deviations above the fitted hyperplane and the components of e^- are vertical deviation below. Since β is not constrained to be nonnegative, we can write it as the difference of two nonnegative variables. This adds more variables to the problem. However, since nonnegativity restrictions actually simplify the solution of linear programs, it is well worth it. Rice (1964) verified the following; (*i*) Under a certain general condition a solution to the problem (3.1) always exists. (*ii*) The uniqueness of this solution can not, in general, be guaranteed. (*iii*) The sum of the absolute deviation, together with vector β, form a convex polytope in the parametric space. Therefore, any simplex-type algorithm which descends to a minimum via a sequence of extreme points will terminate either when the lowest vertex of the polytope is reached or when one of the many vertices at the flat bottom is reached.

Other types of methods are weighted least squares technique, nonlinear programming approach or descent method; some examples are Schlossmacher (1973), Bartels and Conn (1980), and Wesolowsky (1981).

Meketon (1985) suggested an algorithm which combines the two methods; he employs the iterative method of Schlossmacher (1973) and the linear programming method based on a modified Karmarkar (1984) algorithm. However, his algorithm has two shortcomings which will be discussed in section 3.4. Using Meketon's approach we have developed a new algorithm which employs orthogonal triangularization to solve the current L_1 estimates at each iteration. This algorithm can compute the L_1 estimates even when data sets are of rank deficiency. And the termination criteria of this algorithm is different from Meketon's algorithm in that the infinity norm of the vector of projected gradient is compared with a tolerance value at each iteration. In the following section, a brief outline of the computational procedures is presented and a comparison is conducted in terms of CPU time, number of iterations, and numerical accuracy and stability using the results of simulation studies in Chapter V.

3.3.1. Robers and Robers' Algorithm

This algorithm computes the L_1 estimates using the suboptimization method of interval linear programming which was developed by Robers and Ben-Israel (1970). The dual problem of the linear programming (3.1) is,

$$\begin{aligned} &\text{maximize} \quad y^t w \\ &\text{subject to} \quad X^t w = 0 \\ &\qquad\qquad -\ell \le w \le \ell \end{aligned} \tag{3.2}$$

This dual problem can be solved by any linear programming algorithm. However, the suboptimization method of interval linear programming is applicable to solve this problem because of its special structure. The details of this algorithm are described in Robers and Robers (1973).

A computational evaluation of this algorithm was presented by Robers and Robers, and they demonstrated that this algorithm is more efficient than the simplex method applied to the primal problem (3.1), and the bounded variables simplex method applied to the dual problem (3.2). However, this algorithm is the least efficient among the algorithms which are discussed in this dissertation.

3.3.2. Schlossmacher's Algorithm

This algorithm uses an iterative technique for L_1 estimation. Schlossmacher (1973) argued that this procedure is mathematically and computationally simple, requires less storage, and is faster than the linear programming algorithm proposed by Wagner (1959) and Fisher (1961). The algorithm minimizes the criterion

$$\Psi = \sum_{i=1}^{n} w_i e_i^2 \tag{3.3}$$

where the weighting factor, w_i, can be chosen by an investigator to meet the requirements of the specific system under investigation. By considering the $(k + 1)$st iteration of a specially constructed iterative process, the criterion (3.3) can be written as,

$$\Psi^{k+1} = \sum_{i=1}^{n} \frac{(e_i^{k+1})^2}{| e_i^k |}$$

where the weighting factors, $\frac{1}{|e_i^k|}$ are determined by the k-th iteration. If

$$|e_i^k - e_i^{k+1}| \cong 0$$

then it follows that

$$\Psi^{k+1} \cong \sum_{i=1}^{n} |e_i^{k+1}|$$

which approximates the desired absolute deviation index. Therefore, the following iterative algorithm is proposed;

(*i*) Solve the least squares problem with the weighting factors, $w_i = 1$, and set $k = 1$.

(*ii*) With the generated least squares coefficients, calculate the residuals, $e_i^k = y_i - X_i\hat{\beta}^k$.

(*iii*) Resolve the least squares problem with the weighting factors, $w_i = \frac{1}{|e_i^k|}$. If any $e_i^k \cong 0$, then set $w_i = 0$. This is justified since the effect of an almost zero residual on the total sum will be negligible.

(*iv*) Set $k = k + 1$, and repeat (*ii*) and (*iii*) until $|e_i^{k+1} - e_i^k| \cong 0$.

It should be noted that the convergence of this algorithm has not been proved.

3.3.3. *Barrodale and Roberts' Algorithm*

This algorithm is based on the idea that a direct application of the simplex method to the L_1 problem does not yield an efficient algorithm. This modified algorithm of the linear programming simplex method saves storage space and is more efficient in computation since it can skip over simplex vertices. In other words, this algorithm allows one to reduce significantly the total number of iterations required, by passing through several neighboring simplex vertices in a single iteration. The problem (3.1) can be rewritten as,

$$\begin{aligned} \text{minimize} \quad & \ell^t e^+ + \ell^t e^- \\ \text{subject to} \quad & X(\beta^+ - \beta^-) + Ie^+ - Ie^- = y \\ & \beta^+, \beta^-, e^+, e^- \geq 0 \end{aligned} \tag{3.4}$$

An algorithm for this problem is implemented in two stages; (*i*) In the first stage, the pivotal column for each of the first k iterations is chosen as that corresponding to the largest nonnegative marginal cost associated with β_j^+ or β_j^-. The pivot element is chosen from among the basic vectors e_i^+ and e_i^- so that the vector that leaves the basis causes the maximum reduction in the objective function. Thus the pivot element is not necessarily the same as in the usual linear programming simplex algorithm. At the end of the first stage the number of vectors corresponding to a β_j^+ or β_j^- in the basis will be at least as great as the rank of the matrix X. (*ii*) In the second stage, nonbasic columns corresponding to the largest positive marginal costs associated with e_i's are interchanged with basic columns corresponding to e_i's that will cause maximum reduction in the objective function. The basis vectors corresponding to β_j^+'s or β_j^-'s are not allowed to leave the basis. At each iteration in the second stage the solution goes through a number of data points at least as great as the rank of X. The algorithm terminates when all the marginal costs are nonpositive.

The main modification to the simplex method is in choosing the vector e^+ or e^- to leave the basis. In both stages this vector is chosen as that one which causes the maximum reduction in the objective function. Geometrically, the simplex transformation which then follows is usually equivalent to a movement through several neighboring simplex vertices. A direct search over all the basic vectors e^+ and e^- is inefficient. This algorithm does not require that the matrix X be of full rank.

Barrodale and Roberts (1973) compared the computational behavior of four algorithms and showed the superiority of their algorithm to the descent method of Usow (1967), the algorithm of Robers and Robers (1973), and the bounded variable simplex method of Dantzig (1963) applied to the dual formulation of the L_1 estimation problem.

3.3.4. Armstrong, Frome and Kung's Algorithm

This algorithm is a modification of Barrodale and Roberts's algorithm. Instead of a full tableau method, Armstrong, Frome and Kung (1979) used the revised simplex method with the principle of *LU* decomposition in maintaining the current basis.

Comparison of CPU times was made with the algorithm of Barrodale and Roberts (1973), and this algorithm is competitive with Barrodale and Roberts'. However, one drawback of this algorithm is that the matrix X must have full column rank.

3.3.5. Abdelmalek's Algorithm

Usow's (1967) algorithm for solving the discrete L_1 problem was generalized so that it can also solve an overdetermined system of linear equations in the L_1 norm sense. It is shown that this algorithm is completely equivalent to a dual simplex method applied to a linear programming formulation in nonnegative bounded variables. However, one iteration in the former is equivalent to one or more iterations in the latter, i,e, certain intermediate iterations are skipped.

Recall that L_1 estimation in a linear regression model is to determine β which minimizes the L_1 norm

$$\Phi(\beta) = \|y - X\beta\|_1 = \sum_{i=1}^{n} |e_i| .$$

Let the set A be

$$A = \{(\beta, \delta) \mid (\beta, \delta) \in E^{m+1}, \ \Phi(\beta) \leq \delta\}$$

then A is a convex polytope. This algorithm uses a descent method on A from vertex to vertex along connecting edges of the polytope in such a way that certain intermediate vertices are bypassed. This descent continues until the lowest vertex is reached. The primal form of the above problem is the same as (3.1), and the dual problem is the same as (3.2). By defining $h_i = w_i + 1$, we get the formulation

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} y_i(h_i - 1) \\ \text{subject to} \quad & X^t h = X^t \ell \\ & 0 \le h_i \le 2 \end{aligned}$$

This is a linear programming problem in nonnegative bounded variables. It may be solved by the simplex algorithm as a problem with $(m + n)$ constraints instead of the m constraints. In the first part of the algorithm, a dual feasible solution is obtained. For the purpose of numerical stability, a triangular decomposition method is applied to the basis matrix. The second part is a dual simplex algorithm.

Abdelmalek (1980) compared the number of iterations and the CPU time of this algorithm with Barrodale and Roberts′ algorithm. The numerical results indicate that this algorithm is comparable with Barrodale and Roberts′, in both speed and number of iterations. It also has the advantage that in the case of ill-conditioned problems, the basis matrix can lend itself to triangular factorization and can thus ensure a stable solution.

3.3.6. Bartels and Conn's Algorithm

This algorithm considers the overdetermined system of linear equations in the L_1 sense, and finds a real vector β having m components which solves the problem

$$\text{minimize } \{\Phi(\beta) = \|y - X\beta\|_1\}$$

Bartels, Conn and Sinclair (1978) proved that this problem is precisely equivalent to the linear programming problem. However, this algorithm does not make use of linear programming technique, but a direct geometric approach. The algorithm moves from a trial point β^0 to a displaced point $\beta^0 + \alpha p$, $p \in R^n$ so that $\Phi(\beta^0 + \alpha p) < \Phi(\beta^0)$ for all $\alpha > 0$ sufficiently small. Such a p is called a descent direction. After having determined a descent direction, we can take a step, a particular value of α , which minimizes Φ on the ray $\{\beta^0 + \alpha p \,|\, \alpha \geq 0\}$ since Φ is a nonnegative convex piecewise linear function of the single variable α on this ray. The point β may be reset to $\beta^0 + \alpha p$ and a descent direction for Φ can be found from this new point.

This algorithm can easily be extended to handle the L_1 problem with linear equality and inequality constraints. On implementation, Givens transformations are employed for numerical stability. The details of this algorithm are in Bartels Conn and Sinclair (1978) and Bartels and Conn (1980).

3.3.7. Wesolowsky's Algorithm

This algorithm is closely related to that of Bartels and Conn (1980) in that it descends the surface created by the objective function. However, the rules for descent are different in that the descent is reduced to univariate optimization. For the simple linear regression model, the problem is

$$\text{minimize } \{\Phi = \sum_{i=1}^{n} |y_i - \beta_1 - \beta_2 x_i|\} \tag{3.5}$$

This procedure determines estimates directly rather than reforming the linear programming problem. Consider a single term in (3.5),

$$\Phi_j = |y_j - \beta_1 - \beta_2 x_j|$$

In the β_1, β_2 space,

$$\begin{aligned} \Phi_j &= y_j - \beta_1 - \beta_2 x_j, \quad \text{if } \beta_1 \le y_j - \beta_2 x_j \\ &= -y_j + \beta_1 + \beta_2 X_j, \quad \text{otherwise} \end{aligned}$$

These two planes form the edge whose projection is,

$$\beta_1 = y_j - \beta_2 x_j$$

Since Φ is convex, a minimum in (3.5) must occur not only on an edge but at the intersection of at least two edges. Suppose the edge is defined by the point (x_j, y_j), then

$$\beta_2 = \frac{y_j}{x_j} - \frac{\beta_1}{x_j} \tag{3.6}$$

Substituting (3.6) in (3.5), we obtain

$$\underset{(\beta_1)}{\text{minimize}} \left\{\Phi_j = \sum_{i=1}^{n} \left|1 - \frac{x_i}{x_j}\right| \left| \frac{y_i - \frac{y_j x_i}{x_j}}{1 - \frac{x_i}{x_j}} - \beta_1 \right|, \quad x_i \neq x_j \right\} \tag{3.7}$$

Wesolowsky (1981) showed how to solve problem (3.7). This algorithm can be stated as;

(*i*) Set $k = 1$. Choose initial value β_1^0, β_2^0 such as the least squares estimates. Choose j so that $|y_j - \beta_1 - \beta_2 x_j|$ is the minimum.

(*ii*) Set $k = k + 1$. Solve problem (3.7), recording the index i at which the term $(y_i - y_j x_i/x_j)/(1 - x_i/x_j)$ is the lowest weighted median. Set β_1^k equal to this median.

(*iii*) a) If $\beta_1^k - \beta_1^{k-1} = 0$; then, if $k \geq 3$, go to step (*iv*), if $k < 3$, set $j = i$ and go to step (*ii*). b) If $\beta_1^k - \beta_1^{k-1} \neq 0$; set $j = i$ and go to step (*ii*).

(*iv*) Let $\hat{\beta}_2 = \frac{y_j}{x_j} - \frac{\beta_1^k}{x_j}$, and $\hat{\beta}_1 = \beta_1^k$.

This algorithm travels along edge segments and it only changes location if there is an improvement in the objective function. There is a finite number of edge segments and hence the algorithm must converge to an optimal solution in a finite number of iterations. However, when there are three or more points on a line, a collinear problem is encountered. In this situation, a small random value δ_j should be added to each y_j, before beginning the algorithm to avoid this collinear problem. This algorithm is extended to the multiple linear regression model, and the details are in Wesolowsky (1981). A comparison of this algorithm with Armstrong, Frome and Kung's algorithm has been made. The results show that the latter is clearly superior, but the former is comparable or better in computational efficiency to current algorithm for roughly 4 or fewer independent variable and a large number of observations.

3.4. NEW COMPUTATIONAL PROCEDURE

3.4.1. Karmarkar's Algorithm for Linear Programming

This algorithm considers the linear programming problem of standard form,

$$\begin{aligned} \text{minimize} \quad & c^t x \\ \text{subject to} \quad & Ax = 0 \\ & \ell^t x = 1 \\ & x \geq 0 \end{aligned} \tag{3.8}$$

where c and x are n-dimensional vector, A is $m \times n$ matrix, and ℓ is an n-dimensional vector of all ones. It is assumed that the minimum of the objective function is zero and there exists a solution to this problem with the property that $x^0 > 0$. We define a simplex $S_x = \{x \mid \ell^t x = 1, x \geq 0\}$. Under these assumptions, let $D = \mathrm{diag}[x_1^0, x_2^0, \ldots, x_n^0]$, and employ a projective transformation and its inverse as follows,

$$x' = \frac{D^{-1}x}{\ell^t D^{-1} x}, \qquad x = \frac{Dx'}{\ell^t Dx'}$$

Now the problem (3.8) can be transformed into a fractional program (3.9) on a simplex in x'-space by the projective transformation $T(x^0, x_0)$ of the simplex S_x that maps the interior point x^0 to the center x_0,

$$\begin{aligned} \text{minimize} \quad & \frac{c^t Dx'}{\ell^t Dx'} \\ \text{subject to} \quad & ADx' = 0 \\ & \ell^t x' = 1 \\ & x' \geq 0 \end{aligned} \tag{3.9}$$

Here, the point x^0 in the x-space is mapped onto the point $x_0 = [1/n, \ldots, 1/n]$ in the x'-space. Since the optimal value of objective function is assumed to be zero, the denominator of the objective function can be ignored and an improving step is taken from the center of the simplex to a new point in the x'-space. This new point is transformed back into x-space by the inverse of the projective transformation and the result is evaluated. This algorithm creates a sequence of points x^0, $x^1, \dots, x^k$. These points are computed by the following steps;

(*i*) Define $D = \mathrm{diag}[x_1^k, \dots, x_n^k]$ as the diagonal matrix whose *ii*-th entry is x_i^k, an interior feasible point at the *k*-th iteration. Augment the matrix AD with a row of all ones,

$$B = \begin{bmatrix} AD \\ \ell^t \end{bmatrix}$$

(*ii*) Project $c' = Dc$ orthogonally onto null space of its constraint matrix B,

$$c_p = [I - B^t(BB^t)^{-1}B]Dc$$

(*iii*) Normalize c_p and scale it by the radius, r, of the largest sphere which can be inscribed in the simplex to produce the direction vector,

$$\hat{c} = \frac{c_p}{|c_p|}, \quad p = r\hat{c}$$

where $r = \dfrac{1}{\sqrt{n(n-1)}}$.

(*iv*) Take a descent step of length α to a new feasible point,

$$y' = \frac{1}{n}\ell - \alpha p$$

where $\alpha \in (0, 1)$ is a step length parameter which can be chosen appropriately.

(v) Project y' back into x-space to obtain the new point, $x^{k+1} = \dfrac{Dy'}{\ell^t Dy'}$. If the new point satisfies the termination criterion, then stop. Otherwise, set $k = k + 1$ and go to step (i).

3.4.2. Modified Karmarkar's Algorithm for Linear Programming

Karmarkar (1984) stated that any type of linear programming problem can be converted to the standard form (3.8) by a pre-process, and his approach can be applied to the problem with unknown optimal objective values. One variant of Karmarkar's original algorithm has been proposed by Cavalier and Soyster (1985) and Vanderbei et al. (1985). It was demonstrated that this algorithm performs competitively with Karmarkar's algorithm on some small to moderately sized test problems. Consider the general linear programming problem,

$$\begin{aligned} &\text{minimize} \quad c^t x \\ &\text{subject to} \quad Ax = b \\ &\qquad\qquad\quad x \geq 0 \end{aligned} \tag{3.10}$$

where A has full rank, and an initial interior feasible point is available. The algorithm for this form of the linear programming problem (3.10) is as follows;

(i) Let $D = \text{diag}[x_1^k, \ldots, x_n^k]$ where x_i^k is an interior feasible point at the k-th iteration, and consider the linear transformation $y = D^{-1}x$. The transformed problem, in terms of the y coordinates, is

$$\begin{aligned} \text{minimize} \quad & c^t D y \\ \text{subject to} \quad & ADy = b \\ & y \geq 0 \end{aligned}$$

(*ii*) Determine the projected gradient,

$$p = PDc, \quad \text{where } P = [I - (AD)^t\{(AD)(AD)^t\}^{-1}(AD)]$$

(*iii*) Step 97% toward the nearest inequality constraint,

$$y' = \ell - \alpha p, \quad \alpha = \frac{0.97}{\max\{p_i\}}.$$

(*iv*) Map back to the x coordinates, $x^{k+1} = Dy'$.

This is a very simple descent procedure which is repeated until some suitable criterion is satisfied. And there is no requirement that the optimal value of the objective function be zero. Vanderbei et al. (1985) proved the convergence of this algorithm under the assumption that every vertex is nondegenerate in both the primal and dual problem. Sherali (1986) developed a variant of this scaling algorithm by exhibiting relationships of Karmarkar types of algorithms with barrier function methods, and subgradient optimization procedures employing space dilation techniques. He also proved the convergence of his algorithm under certain regularity condition. His assumption set is weaker than Vanderbei et al.'s, and does not require nondegeneracy. Kortanek and Shi (1985) also proved the convergence of this algorithm under similar assumptions.

3.4.3. Meketon's Algorithm

Meketon (1985) employed the iterative technique of Schlossmacher (1973), and applied a modification of Karmarkar's algorithm to the dual of the linear programming problem to obtain an L_1 estimate. From the primal formulation (3.1), the dual problem can be written as,

$$\begin{aligned} &\text{maximize} \quad y^t w \\ &\text{subject to} \;\; X^t w = 0 \\ &\qquad\qquad -\ell \leq w \leq \ell \end{aligned} \tag{3.11}$$

where X has full rank, and w is an n-dimensional vector of dual variables. While Robers and Robers (1973) applied the suboptimization method of interval linear programming to this problem, Meketon suggested the following algorithm using the iterative reweighted least squares technique,

(*i*) Set $k = 0$, and $w^k = 0$.

(*ii*) Define $D = \text{diag}[\min\{1 + w_i^k, 1 - w_i^k\}]$, and compute the current estimate, $\hat{\beta}^k = (X^t D^2 X)^{-1} X^t D^2 y$, and residuals, $e^k = y - X\hat{\beta}^k$.

(*iii*) If $\|e^k\|_1 - y^t w^k < 0.00001$, then stop. Otherwise,

(*iv*) Let $p = D^2 e^k$ (projected gradient), set $\omega = \max_i[\max\{\frac{p_i}{1 - w_i^k}, \frac{-p_i}{1 + w_i^k}\}]$.

Update w, $w^{k+1} = w^k + \frac{0.97}{\omega} p$. Set $k = k + 1$, and go to step (*ii*).

3.4.4. Proposed Algorithm

We found that Meketon's algorithm did not terminate for a wide variety of sample problems and the solutions obtained were not very accurate. Upon closer examination of the algorithm we determined that modifying a couple of steps of the algorithm it was possible to improve its performance. (*i*) In computing $\hat{\beta}^k = (X^t D^2 X)^{-1} X^t D^2 y$ in his algorithm, we noticed severe numerical accuracy problems associated with the inversion of $(X^t D^2 X)$, since the matrix X or $X^t D^2 X$ can often be a highly ill-conditioned matrix. (*ii*) The termination rule, $\|e^k\|_1 - y^t w^k < 0.00001$, did not work for several data sets unless the computations were extremely accurate.

Therefore, to avoid problem (*i*), we employ an orthogonal triangularization approach used in the least squares methods to compute the current estimate and the residuals. This approach can avoid many numerical problems associated with the computation of $\hat{\beta}^k$ and the projected gradient, and yields stable solutions even though the matrix X is highly ill-conditioned. Moreover, in each step of the algorithm, the residuals and hence the projected gradient can be computed without explicitly computing $\hat{\beta}^k$. Thus, this implementation reduces the number of computations at each iteration.

Before proceeding, we first consider a least squares method based on orthogonal triangularization. Assume that the matrix X has full rank, and consider a decomposition of X as,

$$X = QR \tag{3.12}$$

where Q is an $n \times n$ orthogonal matrix and R is an $n \times p$ matrix of the form

$$R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

where R_1 is an $p \times p$ nonsingular and upper-triangular matrix. Using the decomposition (3.12), the sum of squared residuals can be written as,

$$\begin{aligned} \|y - X\beta\|_2 &= \|y - QR\beta\|_2 \\ &= \|Q^t y - R\beta\|_2 \\ &= \|c_1 - R_1\beta\|_2 + \|c_2\|_2 \end{aligned} \tag{3.13}$$

where c_1 is an p vector and c_2 is an $n - p$ vector such that

$$Q^t y = c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

The sum of squared residuals (3.13) is minimized when β is chosen to satisfy the system of linear equations,

$$R_1\beta = c_1$$

This method is an alternative to forming the normal equations for computing $\hat{\beta}$. Furthermore, the residuals can be obtained as,

$$\begin{aligned} y - X\hat{\beta} &= Q(c - R\hat{\beta}) \\ &= Q\left[\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - \begin{bmatrix} c_1 \\ 0 \end{bmatrix}\right] \\ &= Q\begin{bmatrix} 0 \\ c_2 \end{bmatrix} \end{aligned} \tag{3.14}$$

Now, assume that X has rank $r < p$, and consider the decomposition of X,

$$X = QTP \tag{3.15}$$

where Q and P are orthogonal matrices and the matrix T is of the form

$$T = \begin{bmatrix} T_1 & T_2 \\ 0 & 0 \end{bmatrix}$$

with T_1 being $r \times r$ upper-triangular and nonsingular matrix. Now let

$$Q^t y = c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

where c_1 is an r vector and c_2 is an $n - r$ vector. Consider an additional Householder transformation applied to (3.15) from the right to zero the elements of T_2, then

$$X = QRU^t \tag{3.16}$$

where

$$R = \begin{bmatrix} R_2 & 0 \\ 0 & 0 \end{bmatrix}$$

and R_2 is an $r \times r$ lower-triangular. Introducing a change of variables such that

$$U^t\beta = a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{3.17}$$

where a_1 is an r vector, we get the sum of squared residuals from (3.16) as follows,

$$\begin{aligned} \|y - X\beta\|_2 &= \|Q^t y - RU^t\beta\|_2 \\ &= \|c_1 - R_2 a_1\|_2 + \|c_2\|_2 \end{aligned}$$

The least squares solution can be found by solving the system of equations,

$$R_2 a_1 = c_1 \tag{3.18}$$

and substituting (3.18) in (3.17) gives,

$$\hat{\beta} = U\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

where a_2 is arbitrary, and the solution has minimum length when $a_2 = 0$. Furthermore, the residuals can be obtained as,

$$\begin{aligned} y - X\hat{\beta} &= Q[c - RU^t\hat{\beta}] \\ &= Q\left[\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - \begin{bmatrix} R_2 a_1 \\ 0 \end{bmatrix}\right] \\ &= Q\begin{bmatrix} 0 \\ c_2 \end{bmatrix} \end{aligned} \tag{3.19}$$

It can be shown that the computation of $c_p = [I - B^t(BB^t)^{-1}B]Dc$ in Karmarkar's and the modified Karmarkar's algorithm is equivalent to finding the residuals of the least squares problem,

$$\begin{aligned} \tilde{\beta} &= \arg\min \|Dc - B^t\beta\|_2 \\ &= (BB^t)^{-1}BDc \end{aligned}$$

and

$$c_p = Dc - B^t\tilde{\beta}.$$

Therefore, we can compute c_p through (3.14) or (3.19). This is the main idea of the proposed algorithm which combines the iterative reweighted least squares method and the projective transformation approach.

The matrix B in the primal linear programming problem (3.1) for L_1 estimation is usually large and a very sparse matrix. However, in the dual problem (3.11) we have a dense matrix so that the least squares method based on Householder transformation can be employed to compute the projected gradient.

To handle problem (*ii*) in Meketon's algorithm, we derive an alternative termination rule,

$$\|p^k\|_\infty < \delta$$

where δ denotes the tolerance level. This stopping rule is justified by Theorem 3.3.

The L_1 estimation problem can be solved by applying Karmarkar's algorithm or the modification to the primal formulation (3.1). However, it has been shown that the direct application of these algorithms to the primal linear programming problem is not very efficient because the dimension of problem (3.1) become large when a large number of observations are involved. So we start with the dual formulation (3.11).

By defining $D = \text{diag}[v_i]$ where $v_i = \min\{1 - w_i, 1 + w_i\}$, and putting $w = D\gamma$, we can transform the problem (3.11) to the problem (3.20).

$$\begin{aligned} &\text{maximize} \quad (Dy)^t\gamma \\ &\text{subject to} \quad (DX)^t\gamma = 0 \\ &\qquad\qquad -\ell \le D\gamma \le \ell \end{aligned} \tag{3.20}$$

This transformation does not affect the solution of the problem (3.11). The motivation for considering the linear transformation $\gamma = D^{-1}w$, where $D = \text{diag}[v_i]$ is the following. If the current feasible point is located close to the boundary of the polytope, the movement in the gradient direction may not give a substantial reduction in the objective function. Therefore, the current feasible point is linearly transformed to a point which makes the smaller slacks in the inequality ($\gamma = \ell$) so that all of the boundaries are sufficiently distant from that point in the new space. In the transformed space, the current feasible point can be moved sufficiently to a new point along the projected gradient.

Now we compute $e'^k = [I - (D^kX)\{(D^kX)^t(D^kX)\}^{-1}(D^kX)^t](D^ky)$, projection of D^ky onto the null space of $(D^kX)^t\gamma = 0$ at each iteration. Denoting λ be a step length, we can form the following,

$$\gamma^{k+1} = \gamma^k + \lambda e'^k, \quad \text{i.e.,}\ w^{k+1} = w^k + \lambda D^k e'^k$$

Let $p^k = D^k e'^k$ and $\lambda = \alpha/\omega^k$, then $w^{k+1} = w^k + (\alpha/\omega^k)p^k$. Since we want $-1 < w_i^{k+1} < 1$, ω^k should be chosen as,

$$\omega^k = \max_i[\max\{\frac{p_i^k}{1 - w_i^k}, \frac{-p_i^k}{1 + w_i^k}\}]$$

This result is shown in the proof of Theorem 3.2. This centering scheme requires fewer iterations and less computation per iteration than transforming the variables and including the boundary constraints within the constraint matrix X in (3.11) by employing slack variables. The algorithm proceeds as follows;

(*i*) Initialization; set $w^k = 0$, and $k = 0$.

(*ii*) Define $D^k = \text{diag}[v_i^k]$, where $v_i^k = 1 + w_i^k$ if $w_i^k \leq 0$, $v_i^k = 1 - w_i^k$ otherwise, and consider the linear transformation $X' = D^k X$ and $y' = D^k y$. Compute $e'^k = [I - X'(X'^t X')^{-1} X'^t] y'$, the projection of y' onto the null space of X'^t, by least squares method.

(*iii*) Compute the projected gradient, $p^k = D^k e'^k$ in the w-space.

(*iv*) If $\|p^k\|_\infty < \delta$, then compute $\hat{\beta} = (X'^t X')^{-1} X'^t y'$ and stop. Otherwise,

(*v*) Let $\omega^k = \max_i\{u_i^k\}$, where $u_i^k = \dfrac{p_i^k}{1 - w_i^k}$ if $p_i^k \geq 0$, $u_i^k = \dfrac{-p_i^k}{1 + w_i^k}$ if $p_i^k < 0$. Update w^k,

$$w^{k+1} = w^k + \frac{\alpha}{\omega^k} p^k.$$

Set $k = k + 1$ and go to step (*ii*).

A flow chart for this algorithm is given in Appendix C. We note here that in our simulation runs, $\delta = 0.00001$ and $\alpha = 0.97$ appear to work well in our algorithm [Appendix A].

The computation of $\hat{\beta} = (X'^t X')^{-1} X'^t y'$ in step (*iv*) can be obtained as follows. From the proof of Proposition 1 in Vandervei et al. (1985) and the problem (3.20), it follows that there exists a vector β such that

$$\beta^{k^t} (D^k X)^t = (D^k y)^t \tag{3.21}$$

It can be shown that β^k is the vector of dual variables corresponding to the constraint $(D^k X)^t \gamma = 0$ of the problem (3.20). And the scaling leaves the duals with respect to the problem (3.11) unchanged. Since $(D^k X)$ has full rank from the assumption, $(X^t D^{k^2} X)^{-1}$ exists for all k. Hence, (3.21) can be rewritten as,

$$\beta^k = (X^t D^{k^2} X)^{-1} X^t D^{k^2} y$$

This is the vector of current estimates of β which is the reweighted least squares solution.

Theorem 3.2: Each iterate w^k of this algorithm is feasible for the dual linear programming problem (3.11).

proof: In the initial step we set $w^0 = 0$, therefore $X^t w^0 = 0$ and $-1 \le w_i^0 \le 1$, i.e., w^0 is feasible. Assume that feasibility holds at the k-th iteration, then

$$\begin{aligned}
X^t w^{k+1} &= X^t[w^k + (\alpha/\omega^k)p^k] \\
&= X^t w^k + (\alpha/\omega^k)X^t[D^{k^2}\{y - X(X^t D^{k^2} X)^{-1} X^t D^{k^2} y\}] \\
&= X^t w^k + (\alpha/\omega^k)[X^t D^{k^2} y - X^t D^{k^2} X(X^t D^{k^2} X)^{-1} X^t D^{k^2} y] \\
&= X^t w^k \\
&= 0
\end{aligned}$$

And ω^k must satisfy the inequality

$$-1 \le w_i^k + \frac{p_i^k}{\omega^k} \le 1, \ \textit{i.e.,} \ \ -1 < w_i^{k+1} < 1.$$

Therefore, it follows that

$$\begin{aligned}
&(i) \text{ if } p_i^k \ge 0, \text{ then } \ \omega^k \le \frac{-p_i^k}{1 + w_i^k} \ \text{ or } \ \omega^k \ge \frac{p_i^k}{1 - w_i^k} \\
&(ii) \text{ if } p_i^k < 0, \text{ then } \ \omega^k \le \frac{p_i^k}{1 - w_i^k} \ \text{ or } \ \omega^k \ge \frac{-p_i^k}{1 + w_i^k}
\end{aligned}$$

Consequently, we can choose $\omega^k = \max_i[\max\{\frac{p_i^k}{1 - w_i^k}, \frac{-p_i^k}{1 + w_i^k}\}]$. Thus, the feasibility holds for each iteration.

Corollary 3.2: Weak duality holds between the primal (3.1) and dual (3.11) problem.

proof: From the feasibility of w^k, and Definition 2.2 it follows that

$$\begin{aligned} y^t w^k &= e^{k^t} w^k \\ &\le \|e^k\|_1 . \|w^k\|_\infty \\ &\le \|e^k\|_1 \quad \text{since } \|w^k\|_\infty \le 1 \\ &= \ell^t e^+ + \ell^t e^- \end{aligned}$$

Therefore, weak duality holds.

Lemma 3.1: If $p^k = 0$, then w^k is optimal. Otherwise assume that $p^k \neq 0$ for all k. Then $\{y^t w\}$ converges in problem (3.11).

proof: If $p^k = 0$, then $e'^k = 0$ which means that the objective function is a constant for all feasible solution. Hence, in particular, w^k is optimal. Therefore, suppose $p^k \neq 0$ for all k. We first show that $\{y^t w\}$ is a strictly increasing sequence. The projected gradient in the transformed space is obtained as,

$$e'^k = D^k y - D^k X\{(D^k X)^t (D^k X)\}^{-1} (D^k X)^t (D^k y)$$

and it is transformed to the original space by,

$$p^k = D^{k^2} y - D^{k^2} X\{(D^k X)^t (D^k X)\}^{-1} (D^k X)^t (D^k y) \tag{3.22}$$

It follows from equation (3.22) that

$$[y - (D^{k^2})^{-1}p^k]^t = y^t D^{k^2} X\{X^t D^{k^2} X\}^{-1} X^t \tag{3.23}$$

Now the following relationship can be obtained from Theorem 3.2 and (3.23),

$$X^t(w^{k+1} - w^k) = 0$$
$$y^t D^{k^2} X\{X^t D^{k^2} X\}^{-1} X^t(w^{k+1} - w^k) = 0$$
$$y^t(w^{k+1} - w^k) = p^{k^t}(D^{k^2})^{-1}(w^{k+1} - w^k)$$

Furthermore, since $(D^{k^2})^{-1}$ is positive definite and $p^k \neq 0$ it follows that

$$\begin{aligned} p^{k^t}(D^{k^2})^{-1}(w^{k+1} - w^k) &= p^{k^t}(D^{k^2})^{-1}\{w^k + (\alpha/\omega^k)p^k - w^k\} \\ &= (\alpha/\omega^k)p^{k^t}(D^{k^2})^{-1}p^k \\ &> 0 \end{aligned}$$

Therefore, $y^t w^{k+1} > y^t w^k$. Since $\{y^t w\}$ is bounded above by the weak duality theorem, $\{y^t w\}$ converges.

Theorem 3.3: The proposed algorithm terminates in a finite number of iterations for any given tolerance $\delta > 0$.

proof: In Lemma 3.1, we know that

$$\begin{aligned} y^t w^{k+1} - y^t w^k &= (\alpha/\omega^k)p^{k^t}(D^{k^2})^{-1}p^k \\ &= (\alpha/\omega^k)\, \|D^{k^{-1}}p^k\|_2^2 \end{aligned}$$

The convergence of $\{y^t w\}$ implies that its difference sequence tends to zero,

$$\frac{\alpha}{\omega^k} \|D^{k^{-1}} p^k\|_2^2 \to 0 \tag{3.24}$$

From step (*v*) of the proposed algorithm and Lemma 2.1 it follows that

$$\begin{aligned} 0 \le \omega^k &= \max_i[\max\{\frac{p_i^k}{1 - w_i^k}, \frac{-p_i^k}{1 + w_i^k}\}] \\ &\le \max_i\{|p_i^k| / v_i^k\} \\ &= \|D^{k^{-1}} p^k\|_\infty \end{aligned}$$

and hence we know that

$$\begin{aligned} 0 \le \alpha\|D^{k^{-1}} p^k\|_2 &= \alpha\|D^{k^{-1}} p^k\|_2^2 \;/\; \|D^{k^{-1}} p^k\|_2 \\ &\le \alpha\|D^{k^{-1}} p^k\|_2^2 \;/\; \|D^{k^{-1}} p^k\|_\infty \\ &\le (\frac{\alpha}{\omega^k})\|D^{k^{-1}} p^k\|_2^2 \end{aligned}$$

Therefore, the necessary condition for (3.24) is that $\|D^{k^{-1}} p^k\|_2 \to 0$. Now, since

$$\begin{aligned} \max_i |p_i^k| &= \|p^k\|_\infty \\ &\le \|D^{k^{-1}} p^k\|_\infty \\ &\le \|D^{k^{-1}} p^k\|_2 \end{aligned}$$

it follows that $p_i^k \to 0$, for all $i = 1, \dots, n$. Hence, the algorithm terminates in a finite number of iterations for any given tolerance $\delta > 0$.

Theorem 3.4: (Strong Duality) Let the primal optimal solution be $\beta^\circ, e^{+\circ}$ and $e^{-\circ}$. Then the algorithm produces w° with no duality gap.

proof: Since $e'^{k} = D^{k}y - (D^{k}X)\beta^{k}$ where $\beta^{k} = (X^{t}D^{k^2}X)^{-1}X^{t}D^{k^2}y$,

$e^{k} = D^{k^{-1}}e'^{k} = y - X\beta^{k}$. Now, $\{e^{k}\}_K \to e^{\circ}$, $\{\beta^{k}\}_K \to \beta^{\circ}$ as $k \to \infty$.

Given that $\{p^{k}\}_K = \{D^{k^2}e^{k}\}_K$, it follows from Theorem 3.3 that

$$p^{\circ} = D^{\circ 2}e^{\circ} = 0$$

where $e_i^{\circ} = e_i^{+\circ}$ when $e_i^{\circ} \geq 0$, $e_i^{\circ} = -e_i^{-\circ}$ when $e_i^{\circ} < 0$.

Let w_i° be a limit of a subsequence of $\{w^{k}\}$.

(*i*) If $-1 < w_i^{\circ} < 1$, then $D_{ii}^{\circ 2} \neq 0$. Therefore, $e_i^{\circ} = 0$.

(*ii*) If $w_i^{\circ} = -1$, then $D_{ii}^{\circ 2} = 0$. Since $w_i^{k} > -1$ for any k over the converging subsequence, we have $\{w_i^{k}\}_K \to -1$. Suppose $e_i^{\circ} > 0$. Then $e_i^{k} > 0$ for k large enough, $k \in K$, and since $(\alpha/\omega^{k}) > 0$ and $p_i^{k} > 0$,

$$\begin{aligned} w_i^{k+1} &= w_i^{k} + \frac{\alpha}{\omega^{k}}p_i^{k} \\ &> w_i^{k} \end{aligned}$$

This contradicts that $w_i^{k} \to -1$. Hence, $e_i^{\circ} \leq 0$.

(*iii*) Likewise, if $w_i^{\circ} = 1$, then it follows that $e_i^{\circ} > 0$.

Subsequently, it can be shown that

$$\begin{aligned}
y^t w^\circ &= (y - X\hat{\beta}^\circ)^t w^\circ + (X\hat{\beta}^\circ)^t w^\circ \\
&= e^{\circ t} w^\circ \qquad \text{by feasibility} \\
&= \sum_{\substack{i=1 \\ \{i \mid w_i^\circ = -1\}}}^{n} w_i^\circ e_i^\circ + \sum_{\substack{i=1 \\ \{i \mid -1 < w_i^\circ < 1\}}}^{n} w_i^\circ e_i^\circ + \sum_{\substack{i=1 \\ \{i \mid w_i^\circ = 1\}}}^{n} w_i^\circ e_i^\circ \\
&= \sum_{i=1}^{n} | e_i^\circ | \\
&= \ell^t e^{+\circ} + \ell^t e^{-\circ}
\end{aligned}$$

Therefore, the strong duality principle holds.

Example 3.1: We now illustrate the steps of the proposed algorithm on the computer generated example. This set of data was generated by Hoffman and Shier's (1980) test problem generator. There are 7 observations and 3 parameters including a intercept term. The 2nd and 3rd column of X were generated from a normal distribution with mean [5.0, 10.0] and variance [1.3, 1.9], respectively. The optimal estimates of the regression coefficients were specified as [2.0, -2.0, 5.0]. The errors were generated from a normal distribution with mean 0.0 and variance 2.3.

$$X = \begin{bmatrix} 1.000000 & 5.766515 & 9.235767 \\ 1.000000 & 4.661123 & 11.439430 \\ 1.000000 & 2.970308 & 9.238118 \\ 1.000000 & 2.740973 & 11.706110 \\ 1.000000 & 6.769230 & 9.862975 \\ 1.000000 & 4.075700 & 7.034439 \\ 1.000000 & 4.157894 & 12.830360 \end{bmatrix} \qquad y = \begin{bmatrix} 38.55223 \\ 49.57025 \\ 45.27223 \\ 55.04866 \\ 37.77638 \\ 25.13447 \\ 57.83601 \end{bmatrix}$$

The L_1 estimates of the coefficients in the above linear regression model are computed in 6 iterations. A summary of each step follows.

(1) Initial iteration ($k = 0$); (*i*) We set $w = 0$, so $D^0 = \text{diag}[1, 1, ..., 1, 1]$. This implies that the weighting factors are ones for all observations. Therefore, (*ii*) we compute $\hat{\beta}^0$ and e'^0 by least squares without any transformation of X and y.

$\hat{\beta}^{0'} = [\ -1.222812 \quad -1.981091 \quad 5.318323]$

$e'^{0'} = [2.0832 \quad -0.8115 \quad 3.2482 \quad -0.5553 \quad -0.0448 \quad -2.9798 \quad -0.9400]$

And L_1 norm $= 10.662885$. Since $p^0 = D^0 e'^0$, (*iii*) we obtain

$p^{0'} = [2.0832 \quad -0.8115 \quad 3.2482 \quad -0.5553 \quad -0.0448 \quad -2.9798 \quad -0.9400]$

Here, we notice that the 3rd observation is the largest in absolute value and signifies the most outlying observation. (*iv*) Since $\|p^0\|_\infty = 3.24820 > \delta$, we go to step (*v*) then the next iteration.

$\omega^0 = \max\{u_i\}$

$\quad = \max[2.0832 \quad 0.8115 \quad 3.2482 \quad 0.5553 \quad 0.0448 \quad 2.9798 \quad 0.9400] = 3.2482$

$w^{1'} = \frac{0.97}{3.2482} p^{0'}$

$= [0.6221 \quad -0.2423 \quad 0.9700 \quad -0.1658 \quad -0.0134 \quad -0.8898 \quad -0.2807]$

Note that the 3rd observation is downweighted the most, and $y'w^1 = 7.651545$.

(2) 1st iteration ($k = 1$); (*ii*) The weighting factors are obtained as, $D^1 = \text{diag}[0.3779, 0.7577, 0.0300, 0.8342, 0.9866, 0.1102, 0.7193]$. Therefore, the linear transformation of X and y are as follows,

$$X' = D_1 X = \begin{bmatrix} 0.37789 & 2.17909 & 3.49007 \\ 0.75768 & 3.53162 & 8.66737 \\ 0.03000 & 0.08911 & 0.27714 \\ 0.83416 & 2.28642 & 9.76480 \\ 0.98661 & 6.67860 & 9.73093 \\ 0.11015 & 0.44894 & 0.77485 \\ 0.71929 & 2.99072 & 9.22872 \end{bmatrix} \quad y' = D_1 y = \begin{bmatrix} 14.56950 \\ 37.55812 \\ 1.13517 \\ 45.91948 \\ 37.27062 \\ 2.76859 \\ 41.60072 \end{bmatrix}$$

The current least squares estimates and residuals are computed on the transformed data, X' and y', as follows,

$\hat{\beta}^{1'} = [3.415632 \quad -2.022338 \quad 4.883875]$

$e'^{1'} = [0.64056 \quad -0.21807 \quad 0.08237 \quad 0.00415 \quad -0.11752 \quad -0.48401 \quad 0.12024]$

We see that the 1st observation is the most outlying one at this iteration, and the L_1 norm $= 9.413973$. (*iii*) Compute the projected gradient,

$p^{1'} = (D^1 e'^1)'$

$= [0.24206 \quad -0.16522 \quad 0.00247 \quad 0.00346 \quad -0.11595 \quad -0.05331 \quad 0.08649]$

(*iv*) Since $\|p^1\|_\infty = 0.24206 > \delta$, we go to step (*v*) then the next iteration.

$\omega^1 = \max[0.6406\ \ 0.2181\ \ 0.0824\ \ 0.0030\ \ 0.1175\ \ 0.4840\ \ 0.0675] = 0.6406$

$w^{2t} = [0.9887\ \ -0.4925\ \ 0.9737\ \ -0.1606\ \ -0.1890\ \ -0.9706\ \ -0.1497]$

(3) A summary of further iterations is given in Table 2.

It can be shown that the proposed procedure requires fewer iterations to reach the optimal point than are required by the simplex-type methods. The reason for this is because our procedure cuts across the interior of a convex region while the simplex-type methods move from vertex to vertex of a convex polyhedral region. On the other hand, the amount of computation required per iteration is usually greater for the proposed procedure. One important point to be considered is the updating of the orthogonal triangularization in the implementation of this algorithm. If we can update the previous orthogonal decomposition of $X' = D^k X$ in the next iteration, this would significantly reduce the number of computations and hence CPU time required by the proposed algorithm.

Table 2. Illustration of steps of the proposed L_1 algorithm

itera.	initial	1-st	2-nd	3-rd	4-th	5-th	6-th
$\hat{\beta}_i^k$	−1.222812	3.415632	1.899648	2.005962	1.999910	1.999998	2.000001
	−1.981091	−2.022338	−1.999285	−2.000271	−2.000001	−2.000009	−2.000009
	5.318323	4.883875	5.005493	4.999597	5.000007	5.000003	5.000003
e'^k_i	2.083245	0.640561	0.022162	0.018250	0.008792	0.000264	0.000234
	−0.811465	−0.218066	−0.137289	−0.004640	−0.004027	−0.002238	−0.000067
	3.248198	0.082373	0.080605	0.034147	0.001025	0.000909	0.000551
	−0.555336	0.004152	0.028612	−0.000491	0.000023	0.000005	−0.000000
	−0.044833	−0.117521	0.033490	−0.000187	−0.000002	0.000006	0.000000
	−2.979799	−0.484008	−0.112598	−0.023387	−0.007846	−0.001054	−0.000573
	−0.940010	0.120244	0.022861	0.000328	0.000007	0.000005	−0.000000
p_i^k	2.083245	0.242059	0.000251	0.000175	0.000041	0.000000	0.000000
	−0.811465	−0.165223	−0.069671	−0.000071	−0.000053	−0.000016	−0.000000
	3.248198	0.002471	0.002117	0.000386	0.000000	0.000000	0.000000
	−0.555336	0.003463	0.024017	−0.000486	0.000023	0.000005	−0.000000
	−0.044833	−0.115947	0.027161	−0.000187	−0.000002	0.000006	0.000000
	−2.979799	−0.053314	−0.003312	−0.000141	−0.000016	−0.000000	−0.000000
	−0.940010	0.086490	0.019438	0.000324	0.000007	0.000005	−0.000000
w_i^k	0.000000	0.622114	0.988663	0.990439	0.995396	0.999862	0.999878
	0.000000	−0.242324	−0.492522	−0.984776	−0.986782	−0.992655	−0.999780
	0.000000	0.970000	0.973742	0.988696	0.999661	0.999699	0.999818
	0.000000	−0.165838	−0.160594	0.009095	−0.004717	−0.002149	−0.000028
	0.000000	−0.013388	−0.188967	0.002939	−0.002361	−0.002625	−0.000030
	0.000000	−0.889849	−0.970582	−0.993985	−0.997981	−0.999729	−0.999853
	0.000000	−0.280713	−0.149741	−0.012408	−0.003214	−0.002403	−0.000060
$\|e^k\|_1$	10.662885	9.413973	9.224974	9.123723	9.122742	9.122725	9.122708
$y^t w^k$	0.000000	7.651545	8.752752	9.042275	9.101018	9.118243	9.121284

IV. CONSTRAINED L1 NORM ESTIMATION

The problem of minimizing a least absolute deviation subject to linear equality and inequality constraints imposed on the parameters is considered in this chapter. Special purpose algorithms for constrained L_1 estimation have been developed by Armstrong and Hultz (1977), Bartels and Conn (1980) and Barrodale and Roberts (1977, 1978). We first review aspects of these algorithms which are pertinent to the development of our proposed algorithm. We next present our algorithm for the L_1 constrained problem, prove the convergence, and finally illustrate the steps for a simple example.

4.1. COMPUTATIONAL PROCEDURES

4.1.1. Armstrong and Hultz's Algorithm

This algorithm considers a primal formulation of the constrained L_1 estimation problem as a linear programming problem, and solve this problem by extending the techniques of interval linear programming to include variables of interval constraints. Armstrong and Hultz (1977) incorporated a technique for bypassing simplex vertices. However, their computer implementation does not solve rank deficiency problems.

4.1.2. Bartels and Conn's Algorithm

Bartels and Conn (1980) proposed an extension of their algorithm for unconstrained L_1 estimation to include any mixture of linear equality and inequality constraints. The formulation of this problem is,

$$\begin{aligned} \text{minimize} \quad & \|y - X\beta\|_1 \\ \text{subject to} \quad & C\beta = d \\ & E\beta \geq f \end{aligned} \tag{4.1}$$

where C and E are given $k \times m, r \times m$ matrix, and d and f are given k-dimensional and r -dimensional vector, respectively. In the problem (4.1), equality constraints are handled by direct elimination, and inequality constraints are included in the objective function via a penalty function approach. Methods using penalty functions transform a constrained problem into a unconstrained problem, that is, the following unconstrained piecewise linear function is minimized,

$$\Phi(\beta) = \gamma \sum_{i=1}^{n} | y_i - X_i\beta | + \sum_{j=1}^{k} | d_j - C_j\beta | - \sum_{k=1}^{r} \min(0, f_k - E_k\beta)$$

where γ is a positive penalty parameter. It is shown that the minimum of Φ is achieved in a finite number of steps for each value of γ, and only a few values of γ are considered to obtain a solution for this problem. For each value of γ the computational details required to minimize Φ is the same as unconstrained L_1 estimation of Bartels, Conn and Sinclair (1978). It differs from the simplex method in that it does not require a feasible starting point. In addition, it can take more general steps in approaching an optimal point than those 'from vertex to vertex' movement of simplex method. Their computer implementation does not require that the matrices involved be of full rank.

4.1.3. Barrodale and Roberts' Algorithm

Barrodale and Roberts (1977,1978) described a new algorithm which is a generalization of their procedure for unconstrained L_1 estimation. This algorithm is considered to be extremely effective in determining constrained L_1 estimates. The minimization problem (4.1) can be expressed as the following linear programming problem if we consider the inequality constraint such that $E\beta \leq f$,

$$\begin{aligned} \text{minimize} \quad & \ell^t(e^+ + e^-) \\ \text{subject to} \quad & X(\beta^+ - \beta^-) + Ie^+ - Ie^- = y \\ & C(\beta^+ - \beta^-) = d \\ & E(\beta^+ - \beta^-) + Ih = f \\ & \beta^+,\ \beta^-,\ e^+,\ e^-,\ h \geq 0 \end{aligned} \tag{4.2}$$

where h is an r -dimensional column vector of slack variables. This linear programming problem can be solved by applying the simplex method to (4.2) or to its dual. Barrodale and Roberts' algorithm is a modification of the standard simplex method applied to a condensed tableau corresponding to the primal problem (4.2). The modification is designed to eliminate many intermediate simplex iterations, and so the algorithm is efficient both in speed and storage requirements. In order to start the simplex iterations, we introduce artificial vectors g^+, g^-, and h^- . Then the problem is

$$\begin{aligned} \text{minimize} \quad & \ell^t(e^+ + e^-) + M\ell^t(g^+ + g^-) + M\ell^t h^- \\ \text{subject to} \quad & X(\beta^+ - \beta^-) + Ie^+ - Ie^- = y \\ & C(\beta^+ - \beta^-) + Ig^+ - Ig^- = d \\ & E(\beta^+ - \beta^-) + Ih^+ - Ih^- = f \\ & \beta^+, \beta^-, e^+, e^-, g^+, g^-, h^+, h^- \geq 0 \end{aligned} \tag{4.3}$$

where g^+ and g^- are k-dimensional column vectors and h^- is an r-dimensional column vector. The quantity M in the objective function is a large positive number which represents the cost of each artificial vector. The algorithm is implemented using the usual two-phase simplex method, and makes no assumptions about the rank of matrices X, C and E. The details of the algorithm are in Barrodale and Roberts (1978).

4.2. PROPOSED ALGORITHM

We consider the linear equality and inequality constraints imposed on β ,

$$C\beta = d \text{ and/or } E\beta \leq f$$

Then, we obtain the primal linear programming problem such as (4.2). This primal problem can be solved by any simplex-type algorithm or modified Karmarkar-type algorithm. If the modified Karmarkar's algorithm is applied, we can use Paige and Saunder (1982) algorithm for sparse matrix to compute the projection matrix. However, when the problem has a large number of observations, the dimension of the problem (4.2) is so large that the direct application of these algorithms to this problem is not efficient. Thus, we consider the dual problem

$$\begin{aligned} \text{maximize} \quad & y^t w_{(1)} + d^t w_{(2)} + f^t w_{(3)} \\ \text{subject to} \quad & X^t w_{(1)} + C^t w_{(2)} + E^t w_{(3)} = 0 \\ & -\ell \leq w_{(1)} \leq \ell \\ & w_{(2)} \text{ unrestricted} \\ & w_{(3)} \leq 0 \end{aligned} \tag{4.4}$$

where $w_{(1)}^t = [w_1, \dots, w_n]$, $w_{(2)}^t = [w_{n+1}, \dots, w_{n+k}]$ and $w_{(3)}^t = [w_{n+k+1}, \dots, w_{n+k+l}]$ are the transposes of row vectors of dual variables, k and l are the numbers of equality and inequality constraints, respectively.

We can convert the problem (4.4) to a standard format of linear programming by defining $w_{(1)} = w'_{(1)} - \ell$, $w_{(2)} = w'_{(2)1} - w'_{(2)2}$, $w_{(3)} = - w'_{(3)}$, and introducing slack variables. Improvement in computational efficiency can be achieved by considering the dual problem (4.4) instead of primal problem (4.2). But this procedure is no more efficient than other algorithms for constrained L_1 estimation since this procedure is a sort of direct application of the simplex method or the modified Karmarkar's algorithm.

Now, we consider an extension of the proposed algorithm for unconstrained L_1 estimation to this constrained L_1 estimation problem. For this purpose, we use the big-M method, and obtain the primal linear programming problem (4.3). This problem can be converted to the dual problem (4.5) which has the same form of bounded constraints on w as (3.11). The transformed dual problem is,

$$\begin{aligned} \text{maximize} \quad & y^t w_{(1)} + d^t w_{(2)} + f^t w_{(3)} \\ \text{subject to} \quad & X^t w_{(1)} + C^t w_{(2)} + E^t w_{(3)} = 0 \\ & -\ell \le w_{(1)} \le \ell \\ & -M \le w_{(2)} \le M \\ & -M \le w_{(3)} \le 0 \end{aligned} \tag{4.5}$$

where $w'_{(1)}$, $w'_{(2)}$, and $w'_{(3)}$ are defined as in (4.4).

In order to extend the algorithm proposed in section 3.4.4 to problem (4.5), we need an initial feasible, strictly interior, point. The two-phase method may be employed to obtain an initial feasible point. But the difficulty is that we still must obtain an initial feasible point in the first phase. So we need to convert problem (4.5) to a standard format of linear programming by defining $w_{(1)} = w'_{(1)} - \ell$, $w_{(2)} = w'_{(2)} - M$, $w_{(3)} = w'_{(3)} - M$. Then an initial feasible point for problem (4.5) is obtained by applying the modified Karmarkar's algorithm so that our proposed algorithm for L_1 estimation can be easily extended to this constrained L_1 problem.

However, it is very cumbersome and takes a significant number of steps to compute an initial feasible solution in the two-phase method since the problem size becomes large by explicitly including the boundary constraints within the constraints matrix in (4.5). Furthermore, it is not guaranteed that the initial feasible point obtained in the first phase is a strictly interior point. Thus, we consider an alternative procedure to solve problem (4.5).

We can initialize the algorithm with $w = 0$ which is a feasible point for this problem. But since $w_{(3)} = 0$ are boundary values, we employ a special technique to maintain the feasibility of w at each iteration. This can be done if we focus on the constraints, $-M \le w_{(3)} \le 0$. Starting at the point $w^k_{(3)} = 0$ and taking a step toward a new point $w^{k+1}_{(3)}$, it is clear that not all directions are allowed if the new point is to be in the feasible region. For instance, if $w^k_{(3)i} = 0$ and the i-th inequality constraint is inactive, i.e., $e'^k_i > 0$, then the new point $w^{k+1}_{(3)i} = w^k_{(3)i} + \frac{\alpha}{\omega^k} p^k_i$ violates the constraint $-M \le w_{(3)} \le 0$ since $p^k = D^k e'^k$ and $\alpha/\omega^k > 0$. Therefore, moving in the direction i is not permitted. Thus, such a variable $w^k_{(3)i}$ must be detected and the inequality constraint corresponding to that variable must be temporarily deleted from the problem when the projected gradient is computed.

In the proposed algorithm, we first examine the feasibility of new point, and delete the most inactive constraint from among the inequality constraints which violate the feasibility, i.e., $w^k_{(3)i} = 0$ and $e'^k_i > 0$, and repeat until only the variables which satisfy the feasibility remain. Then we compute the projected gradient from the remaining variables and update $w^{k+1} = w^k + \frac{\alpha}{\omega^k} p^k$. At the next iteration, if we have any inequality constraints which were deleted in the previous iteration but now satisfy the feasibility, then we add the most active one back into our problem and repeat until all the variables which satisfy the feasibility are included. It is shown, Theorem 4.1, that this procedure of temporarily dropping a plane corresponding to the inequality constraints which violate the feasibility and projecting on the remaining constraints leads to a usable feasible direction.

Our proposed algorithm for the constrained L_1 estimation is given below using the notation for matrices;

$$S = \begin{bmatrix} X \\ C \\ E \end{bmatrix}, \quad b = \begin{bmatrix} y \\ d \\ f \end{bmatrix}, \quad w = \begin{bmatrix} w_{(1)} \\ w_{(2)} \\ w_{(3)} \end{bmatrix}, \quad p = \begin{bmatrix} p_{(1)} \\ p_{(2)} \\ p_{(3)} \end{bmatrix}.$$

Let $\Lambda = \{1, \ldots, l\}$ and $H = \{i \in \Lambda \mid w^k_{(3)i} = 0, e'^k_{(3)i} > 0\}$. Then S_T denotes the matrix S with the i-th row deleted if $i \in H$, and D_T denotes the diagonal matrix D with the i-th row and the i-th column deleted if $i \in H$. Similarly, b_T, p_T and w_T are the vectors b, p and w with the i-th element deleted if $i \in H$, respectively.

The steps of the proposed algorithm for the constrained problem are as follows;

(*i*) Initialization; set $w^k = 0$, and $k = 0$.

(*ii*) Define $D^k = \text{diag}[v^k_{(1)i}, v^k_{(2)i}, v^k_{(3)i}]$, where $v^k_{(1)i} = \min\{1 - w^k_{(1)i}, 1 + w^k_{(1)i}\}$, $v^k_{(2)i} = \min\{1 - w^k_{(2)i}/M, 1 + w^k_{(2)i}/M\}$, $v^k_{(3)i} = 1 + w^k_{(3)i}/M$, and consider the linear transformation $S' = D^kS$ and $b' = D^kb$. Compute $e'^k = [I - S'(S'^tS')^{-1}S'^t]\,b'$, the projection of b' onto the null space of S'^t, by least squares method.

(*iii*) (a) When $k = 0$; If any $w^0_{(3)i} = 0$ and $e'^0_{(3)i} > 0$, then delete the most inactive inequality constraint and repeat until only the variables which satisfy the feasibility remain. (b) When $k \geq 1$; If any $w^k_{(3)i} = 0$ and $e'^k_{(3)i} < 0$ among the inequality constraints which were deleted at the $(k - 1)$th iteration, then add the most active one and repeat. And take the same step as (a).

(*iv*) If any inequality constraints are deleted or added in step (*iii*), then compute e'^k by D^k_T, S_T, b_T.

(*v*) Compute $p^k = D^ke'^k$. If $\|p^k\|_\infty < \delta$, then $\hat{\beta} = (S'^tS')^{-1}S'^tb'$ and stop. Otherwise,

(*vi*) Let $\omega^k = \max\{\omega^k_1, \omega^k_2, \omega^k_3\}$ where

$$\omega^k_1 = \max_i[\max\{\frac{p^k_{(1)i}}{1 - w^k_{(1)i}}, \frac{-p^k_{(1)i}}{1 + w^k_{(1)i}}\}]\,,\quad \omega^k_2 = \max_i[\max\{\frac{p^k_{(2)i}}{M - w^k_{(2)i}}, \frac{-p^k_{(2)i}}{M + w^k_{(2)i}}\}]\,,$$

$$\omega^k_3 = \max_i[\max\{\frac{-p^k_{(3)i}}{w^k_{(3)i}}, \frac{-p^k_{(3)i}}{M + w^k_{(3)i}}\} \text{ if } w^k_{(3)i} \neq 0,\ \max\{0, \frac{-p^k_{(3)i}}{M}\} \text{ if } w^k_{(3)i} = 0]\,.$$

Update w^k,

$$w^{k+1} = w^k + \frac{\alpha}{\omega^k} p^k$$

Set $k = k + 1$ and go to step (*ii*).

A flow chart of this algorithm is given in Appendix D. This procedure will minimize the sum of artificial variables in the constraints of problem (4.3) as it does in the simplex method. The additional termination criterion $\|p^k_{(2)}\|_\infty < \delta$ and $\|p^k_{(3)}\|_\infty < \delta$ imply that the artificial variables are close to zero, which is required for the big-M method.

Theorem 4.1: Each iterate w^k of this algorithm is feasible for the dual linear programming problem (4.5).

proof : In the initial step we set $w^0 = 0$, so $S^t w^0 = 0$, and $-1 \le w^0_{(1)i} \le 1$, $-M \le w^0_{(2)i} \le M$ and $-M \le w^0_{(3)i} \le 0$. Assume that w is feasible at the k-th iteration, then, since p^k_i corresponding to the i-th inequality constraint dropped is equal to zero, it follows that

$$\begin{aligned} S^t w^{k+1} &= S^t w^k + (\alpha/\omega^k)[X^t p^k_{(1)} + C^t p^k_{(2)} + E^t p^k_{(3)}] \\ &= S^t w^k + (\alpha/\omega^k) S^t_T p^k_T \\ &= S^t w^k + (\alpha/\omega^k) S^t_T [D^{k^2}_T \{b_T - S_T (S^t_T D^{k^2}_T S_T)^{-1} S^t_T D^{k^2}_T b_T\}] \\ &= S^t w^k \\ &= 0 \end{aligned}$$

And ω^k must satisfy the following inequalities

$$-1 \le w_{(1)i}^k + \frac{p_{(1)i}^k}{\omega_1^k} \le 1, \ i.e., \ -1 < w_{(1)i}^{k+1} < 1$$

$$-M \le w_{(2)i}^k + \frac{p_{(2)i}^k}{\omega_2^k} \le M, \ i.e., \ -M < w_{(2)i}^{k+1} < M$$

$$-M \le w_{(3)i}^k + \frac{p_{(3)i}^k}{\omega_3^k} \le 0, \ i.e., \ -M < w_{(3)i}^{k+1} < 0$$

Therefore, it follows that

(*i*) if $p_{(1)i}^k \ge 0$, then $\omega_1^k \le \dfrac{-p_{(1)i}^k}{1 + w_{(1)i}^k}$ or $\omega_1^k \ge \dfrac{p_{(1)i}^k}{1 - w_{(1)i}^k}$

(*ii*) if $p_{(1)i}^k < 0$, then $\omega_1^k \le \dfrac{p_{(1)i}^k}{1 - w_{(1)i}^k}$ or $\omega_1^k \ge \dfrac{-p_{(1)i}^k}{1 + w_{(1)i}^k}$

(*iii*) if $p_{(2)i}^k \ge 0$, then $\omega_2^k \le \dfrac{-p_{(2)i}^k}{M + w_{(2)i}^k}$ or $\omega_2^k \ge \dfrac{p_{(2)i}^k}{M - w_{(2)i}^k}$

(*iv*) if $p_{(2)i}^k < 0$, then $\omega_2^k \le \dfrac{p_{(2)i}^k}{M - w_{(2)i}^k}$ or $\omega_2^k \ge \dfrac{-p_{(2)i}^k}{M + w_{(2)i}^k}$

(*v*) if $p_{(3)i}^k > 0$, then $\omega_3^k \le \dfrac{-p_{(3)i}^k}{M + w_{(3)i}^k}$ or $\omega_3^k \ge \dfrac{-p_{(3)i}^k}{w_{(3)i}^k}$

(*vi*) if $p_{(3)i}^k \le 0$, then $\omega_3^k \le \dfrac{-p_{(3)i}^k}{w_{(3)i}^k}$ or $\omega_3^k \ge \dfrac{-p_{(3)i}^k}{M + w_{(3)i}^k}$ when $w_{(3)i}^k \ne 0$,

$\omega_3^k \ge \dfrac{-p_{(3)i}^k}{M}$ when $w_{(3)i}^k = 0$

Consequently, we can choose $\omega^k = \max[\max(\frac{p^k_{(1)i}}{1 - w^k_{(1)i}}, \frac{-p^k_{(1)i}}{1 + w^k_{(1)i}}),$
$\max(\frac{p^k_{(2)i}}{M - w^k_{(2)i}}, \frac{-p^k_{(2)i}}{M + w^k_{(2)i}}), \{ \max(\frac{-p^k_{(3)i}}{w^k_{(3)i}}, \frac{-p^k_{(3)i}}{M + w^k_{(3)i}})$ if $w^k_{(3)i} \neq 0,$
$\max(0, \frac{-p^k_{(3)i}}{M})$ if $w^k_{(3)i} = 0 \}]$. Thus, the feasibility holds for each iteration.

Lemma 4.1: If $p^k = 0$, then w^k is optimal. Otherwise assume that $p^k \neq 0$ for all k. Then $\{y^t w_{(1)} + d^t w_{(2)} + f^t w_{(3)}\}$ converges in problem (4.5).

proof: If $p^k = 0$, then $e'^k = 0$ which implies that the objective function is a constant for all feasible solution. Hence, in particular, w^k is optimal. Therefore, suppose $p^k \neq 0$ for all k. We first show that $\{y^t w_{(1)} + d^t w_{(2)} + f^t w_{(3)}\}$ is a strictly increasing sequence. Since the matrix S has full column rank, the projected gradient on the original space is obtained as,

$$p^k_T = D^{k^2}_T b_T - D^{k^2}_T S_T\{(D^k_T S_T)^t (D^k_T S_T)\}^{-1} (D^k_T S_T)^t (D^k_T b_T) \tag{4.6}$$

It follows from the equation (4.6) that

$$[b_T - (D^{k^2}_T)^{-1} p^k_T]^t = b^t_T D^{k^2}_T S_T \{S^t_T D^{k^2}_T S_T\}^{-1} S^t_T \tag{4.7}$$

Now the following relationship can be obtained from Theorem 4.1 and (4.7),

$$S^t(w^{k+1} - w^k) = 0$$
$$b^t_T D^{k^2}_T S_T \{S^t_T D^{k^2}_T S_T\}^{-1} S^t_T (w^{k+1}_T - w^k_T) = 0$$
$$b^t_T (w^{k+1}_T - w^k_T) = p^{k^t}_T (D^{k^2}_T)^{-1} (w^{k+1}_T - w^k_T)$$

Furthermore, since $(D^{k^2}_T)^{-1}$ is positive definite and $p^k_T \neq 0$ it follows that

$$p_T^{k^t}(D_T^{k^2})^{-1}(w_T^{k+1} - w_T^k) = \frac{\alpha}{\omega^k} p_T^{k^t}(D_T^{k^2})^{-1} p_T^k$$
$$> 0$$

Now, it can be shown that

$$b^t w^{k+1} - b^t w^k = b_T^t(w_T^{k+1} - w_T^k)$$

Therefore, $b^t w^{k+1} > b^t w^k$. Since $\{b^t w\}$ is bounded above by the weak duality theorem, $\{y^t w_{(1)} + d^t w_{(2)} + f^t w_{(3)}\}$ converges.

Theorem 4.2: The proposed algorithm terminates in a finite number of iterations for any given tolerance $\delta > 0$.

proof: It follows from Lemma 4.1 that

$$\begin{aligned} b^t w^{k+1} - b^t w^k &= (\alpha/\omega^k) p_T^{k^t}(D_T^{k^2})^{-1} p_T^k \\ &= (\alpha/\omega^k)\, \|D_T^{k^{-1}} p_T^k\|_2^2 \end{aligned}$$

The convergence of $\{b^t w\}$ implies that its difference sequence tends to zero,

$$\frac{\alpha}{\omega^k} \|D_T^{k^{-1}} p_T^k\|_2^2 \rightarrow 0 \tag{4.8}$$

From step (*vi*) of the algorithm and Lemma 2.1 it follows that

$$\begin{aligned} 0 \le \omega^k &= \max[\omega_1^k, \omega_2^k, \omega_3^k] \\ &\le \max_i\{\,|p_i^k|\,/v_i^k\} \quad \text{if } \omega^k \neq \frac{-p_{(3)i}^k}{w_{(3)i}^k} \\ &= \|D_T^{k^{-1}} p_T^k\|_\infty \end{aligned}$$

and hence we know that

$$\begin{aligned} 0 \le \alpha \| D_T^{k^{-1}} p_T^k \|_2 &= \alpha \| D_T^{k^{-1}} p_T^k \|_2^2 \;/\; \| D_T^{k^{-1}} p_T^k \|_2 \\ &\le \alpha \| D_T^{k^{-1}} p_T^k \|_2^2 \;/\; \| D_T^{k^{-1}} p_T^k \|_\infty \\ &\le (\frac{\alpha}{\omega^k}) \| D_T^{k^{-1}} p_T^k \|_2^2 \end{aligned}$$

Therefore, the necessary condition for (4.8) is that $\|D_T^{k^{-1}} p_T^k\|_2 \to 0$. If $\omega^k = \dfrac{-p_{(3)l}^k}{w_{(3)l}^k}$ in particular, then this implies that $w_{(3)l}^{k+1} = 0$. At the next iteration, if $e'^{k+1}_{(3)l} > 0$, then this constraint is deleted. And if $e'^{k+1}_{(3)l} \le 0$, then $\omega^{k+1} \ne \dfrac{-p_{(3)l}^{k+1}}{w_{(3)l}^{k+1}}$. Therefore, this particular case does not affect the above result. Now, since

$$\begin{aligned} \max_i | p_{Ti}^k | &= \| p_T^k \|_\infty \\ &\le \| D_T^{k^{-1}} p_T^k \|_\infty \\ &\le \| D_T^{k^{-1}} p_T^k \|_2 \end{aligned}$$

it follows that $p_i^k \to 0$, for all $i = 1, \dots, n + k + l$. Hence, the algorithm terminates in a finite number of iterations for any given tolerance $\delta > 0$.

Theorem 4.3: (Strong Duality) Let the primal optimal solution be β°, $e^{+\circ}$, $e^{-\circ}$, $g^{+\circ}$, $g^{-\circ}$ and $h^{-\circ}$. Then the algorithm produces w° with no duality gap.

proof: Since $e'^k = D^k b - (D^k S)\beta^k$ where $\beta^k = (S^t D^{k^2} S)^{-1} S^t D^{k^2} y$,

$e^k = D^{k^{-1}} e'^k = b - S\beta^k$. Now, $\{e^k\}_K \to e^\circ$, $\{\beta^k\}_K \to \beta^\circ$ as $k \to \infty$.

Given that $\{p^k\}_K = \{D^{k^2} e^k\}_K$, it follows from Theorem 4.2 that

$$p^\circ = D^{\circ 2} e^\circ = 0$$

where $e_{(1)i}{}^\circ = e_i^{+\circ}$ when $e_{(1)i}{}^\circ \geq 0$, $e_{(1)i}{}^\circ = -e_i^{-\circ}$ when $e_{(1)i}{}^\circ < 0$, $e_{(2)i} = g_i^{+\circ}$ when $e_{(2)i}{}^\circ \geq 0$, $e_{(2)i}{}^\circ = -g_i^{-\circ}$ when $e_{(2)i}{}^\circ < 0$, $e_{(3)i}{}^\circ = -h_i^{-\circ}$.

Let w_i° be a limit of a subsequence of $\{w^k\}$.

(*i*) If $-1 < w_{(1)i}{}^\circ < 1$, $-M < w_{(2)i}{}^\circ < M$ or $-M < w_{(3)i}{}^\circ < 0$, then $D^{\circ 2}_{ii} \neq 0$. Therefore, $e_i^\circ = 0$.

(*ii*) If $w_{(1)i}{}^\circ = -1$, $w_{(2)i}{}^\circ = -M$ or $w_{(3)i}{}^\circ = -M$, then $D^{\circ 2}_{ii} = 0$. Since $w^k_{(1)i} > -1$, $w^k_{(2)i} > -M$ or $w^k_{(3)i} > -M$ for any k over the converging subsequence, we have $\{w^k_{(1)i}\}_K \rightarrow -1$, $\{w^k_{(2)i}\}_K \rightarrow -M$ or $\{w^k_{(3)i}\}_K \rightarrow -M$. Suppose $e_i^\circ > 0$. Then $e_i^k > 0$ for k large enough, $k \in K$, and since $\frac{\alpha}{\omega^k} > 0$ and $p_i^k > 0$,

$$\begin{aligned} w_i^{k+1} &= w_i^k + \frac{\alpha}{\omega^k} p_i^k \\ &> w_i^k \end{aligned}$$

This contradicts that $w_i^k \rightarrow w_i^\circ$. Hence, $e_i^\circ \leq 0$.

(*iii*) Likewise, if $w_{(1)i}{}^\circ = 1$ or $w_{(2)i}{}^\circ = M$, then it follows that $e_i^\circ > 0$. And if $w_{(3)i}{}^\circ = 0$, then $e_i^\circ = 0$.

Subsequently, it can be shown that

$$
\begin{aligned}
b^t w^\circ &= e_{(1)}^{\circ t} w_{(1)}^\circ + e_{(2)}^{\circ t} w_{(2)}^\circ + e_{(3)}^{\circ t} w_{(3)}^\circ \quad \text{by feasibility} \\
&= \sum_{\substack{i=1 \\ \{i \mid w_{(1)i}^\circ = -1\}}}^{n} w_{(1)i}^\circ e_{(1)i}^\circ + \sum_{\substack{i=1 \\ \{i \mid -1 < w_{(1)i}^\circ < 1\}}}^{n} w_{(1)i}^\circ e_{(1)i}^\circ + \sum_{\substack{i=1 \\ \{i \mid w_{(1)i}^\circ = 1\}}}^{n} w_{(1)i}^\circ e_{(1)i}^\circ \\
&+ \sum_{\substack{i=1 \\ \{i \mid w_{(2)i}^\circ = -M\}}}^{n+k} w_{(2)i}^\circ e_{(2)i}^\circ + \sum_{\substack{i=1 \\ \{i \mid -M < w_{(2)i}^\circ < M\}}}^{n+k} w_{(2)i}^\circ e_{(2)i}^\circ + \sum_{\substack{i=1 \\ \{i \mid w_{(2)i}^\circ = M\}}}^{n+k} w_{(2)i}^\circ e_{(2)i}^\circ \\
&+ \sum_{\substack{i=1 \\ \{i \mid w_{(3)i}^\circ = -M\}}}^{n+k+l} w_{(3)i}^\circ e_{(3)i}^\circ + \sum_{\substack{i=1 \\ \{i \mid -M < w_{(3)i}^\circ < 0\}}}^{n+k+l} w_{(3)i}^\circ e_{(3)i}^\circ + \sum_{\substack{i=1 \\ \{i \mid w_{(3)i}^\circ = 0\}}}^{n+k+l} w_{(3)i}^\circ e_{(3)i}^\circ \\
&= \sum_{i=1}^{n} | e_{(1)i}^\circ | + M \sum_{i=1}^{n+k} | e_{(2)i}^\circ | + M \sum_{i=1}^{n+k+l} | \bar{e}_{(3)i}^\circ | \\
&= \ell^t (e^{+\circ} + e^{-\circ}) + M \ell^t (g^{+\circ} + g^{-\circ}) + M \ell^t h^{-\circ}
\end{aligned}
$$

where $\bar{e}_{(3)i}^\circ$ denotes negative deviation of $e_{(3)}^\circ$. Thus, strong duality holds.

Example 4.1: We illustrate the steps of the proposed constrained L_1 estimation algorithm on the data set generated in Example 3.1, with the following linear equality and inequality constraints imposed on β,

$$
\begin{gathered}
\beta_0 + \beta_1 + \beta_2 = 5 \\
\beta_0 \geq 0 \\
\beta_1 \geq 0 \\
\beta_2 \geq 0
\end{gathered}
$$

$$S = \begin{bmatrix} 1.000000 & 5.766515 & 9.235767 \\ 1.000000 & 4.661123 & 11.439430 \\ 1.000000 & 2.970308 & 9.238118 \\ 1.000000 & 2.740973 & 11.706110 \\ 1.000000 & 6.769230 & 9.862975 \\ 1.000000 & 4.075700 & 7.034439 \\ 1.000000 & 4.157894 & 12.830360 \\ 1.000000 & 1.000000 & 1.000000 \\ -1.000000 & 0.000000 & 0.000000 \\ 1.000000 & -1.000000 & 0.000000 \\ 1.000000 & 0.000000 & -1.000000 \end{bmatrix} \qquad b = \begin{bmatrix} 38.55223 \\ 49.57025 \\ 45.27223 \\ 55.04866 \\ 37.77638 \\ 25.13447 \\ 57.83601 \\ 5.00000 \\ 0.00000 \\ 0.00000 \\ 0.00000 \end{bmatrix}$$

The constrained L_1 estimates of the coefficients for the above linear regression model are computed by the proposed algorithm in 12 iterations. The steps of the algorithm at each iteration are outlined below.

(1) Initial iteration ($k = 0$); (*i*) We set $w = 0$, so $D^0 = \text{diag}[1, ..., 1]$. This implies that the weighting factors are ones for all observations. Therefore, (*ii*) we compute $\hat{\beta}^0$ and e'^0 by least squares method without any transformation of S and b as follows,

$\hat{\beta}^{0\prime} = [0.808372 \quad -1.866066 \quad 5.0713341]$

$e'^{0\prime} = [1.6699 \quad -0.5533 \quad 3.1571 \quad -0.0105 \quad -0.4186 \quad -3.7423 \quad -0.2805$

$0.9864 \quad 0.8083 \quad -1.8661 \quad 5.0713\ ]$

(*iii*) Notice that the 1st and 3rd inequality constraints are inactive, and $w^0_{(3)1} = w^0_{(3)3} = 0$. Therefore, (*iv*) we delete these constraints and go to step (*ii*). Compute e'^0 and hence p^0 as follows,

$$p^{0'} = e'^{0'} = [1.8346 \quad -0.5961 \quad 2.8917 \quad -0.3530 \quad -0.1104 \quad -3.8004$$
$$-0.4242 \quad 0.5577 \quad 0.0000 \quad -2.0198 \quad 0.0000\,]$$

Here, we notice that the 6th observation has the largest absolute deviation from zero. Also, $y'w^0 = 0.0$, and the L_1 norm $= 2587.564$. (*v*) Since $\|p^0\|_\infty = 3.8004 > \delta$, we go to step (*vi*) and the next iteration.

$$\omega^0 = \max[\omega^0_1, \omega^0_2, \omega^0_3]$$
$$= \max[1.8346 \; 0.5961 \; 2.8917 \; 0.3530 \; 0.1104 \; 3.8004 \; 0.4242 \; 0.5577$$
$$0.0000 \; 0.0020 \; 0.0000\,] = 3.8004$$

$$w^{1'} = \frac{0.97}{3.8004} p^{0'}$$
$$= [0.4683 \quad -0.1521 \quad 0.7381 \quad -0.0901 \quad -0.0282 \quad -0.9700 \quad -0.1083$$
$$0.1423 \quad 0.0000 \quad -0.5155 \quad 0.0000\,]$$

(2) 1st iteration ($k = 1$); (*ii*) The weighting factors are obtained as,

$$D^1 = \text{diag}\,[0.531748, 0.847861, 0.261928, 0.909900, 0.971828, 0.030000,$$
$$0.891738, 0.999858, 0.000000, 0.999484, 0.000000\,].$$

Since $e^1_{(3)1} = 1.862689, e^1_{(3)3} = 4.932495$, we do not have any inequality constraints to be added. The current least squares estimates and residuals are computed on the transformed data of S and b as follows,

$$\hat{\beta}^{1'} = [1.862689 \quad -1.781051 \quad 4.932495]$$

$$e'^{1'} = [0.74851 \quad -0.35247 \quad 0.82058 \quad 0.29792 \quad -0.65990 \quad -0.12500 \quad 0.08299$$
$$-0.01413 \quad 0.00000 \quad -1.78013 \quad 0.00000\,]$$

$p^{1^t} = (D^1 e'^1)^t = [0.39802\ \ -0.29884\ \ 0.21493\ \ 0.27107\ \ -0.64131\ \ -0.00375$

$0.07400\ \ -0.01413\ \ 0.00000\ \ -1.77922\ \ 0.00000\ ]$

And $y^t w^1 = 7.971908$, L_1norm = 1805.4059. (*v*) Since $\|p^1\|_\infty = 1.77922 > \delta$, we go to step (*vi*) and the next iteration.

$\omega^1 = \max\,[0.7485\ 0.3525\ 0.8206\ 0.2487\ 0.6599\ 0.1250\ 0.0668\ 0.0000$

$0.0000\ 0.0018\ 0.0000\ ] = 0.8206$

$w^{2^t} = [0.9387\ \ -0.5054\ 0.9921\ 0.2303\ \ -0.7863\ \ -0.9744\ \ -0.0208\ 0.1256$

$0.0000\ \ -2.6187\ 0.0000\ ]$

(3) A summary of further iterations is given in Table 3.

In the implementation of this algorithm, we can employ the technique of updating the triangular decomposition of S since it is necessary to recompute the projected gradient when the set of constraints changes. This procedure is more efficient than triangular decomposition of S at each iteration.

In our numerical experiments, it appeared that Bartels and Conn's and the proposed algorithm yield correct estimates while Barrodale and Roberts' algorithm computes incorrect estimates for several data sets generated. We note that Bartels and Conn's unconstrained algorithm was among the slowest and Barrodale and Roberts' the most efficient.

Table 3. Illustration of steps of the proposed constrained L_1 algorithm

itera.	initial	2-nd	3-rd	6-th	10-th	11-th	12-th
$\hat{\beta}_l^k$	0.808327	1.387306	0.566360	0.925677	0.730720	0.730590	0.730590
	−1.866066	−1.205246	−0.050614	−0.000240	−0.000119	0.000000	0.000000
	5.071334	4.798043	4.477799	4.074551	4.269386	4.269410	4.269410
$e_l'^k$	1.669945	−0.011982	−0.191931	−0.000225	−0.009971	−0.000299	−0.000009
	−0.553334	−0.537230	−0.149143	0.004180	0.000186	0.000000	−0.000000
	3.157112	0.024673	0.026354	0.015586	0.000147	0.000145	0.000077
	−0.010473	0.614526	0.050876	0.012142	0.000063	0.000063	0.000050
	−0.418557	−0.593203	−0.089963	−0.001112	−0.000034	−0.000034	−0.000030
	−3.742317	−0.130193	−0.136594	−0.005844	−0.000121	−0.000119	−0.000073
	−0.280454	−0.098543	0.023418	0.004367	0.000083	0.000082	0.000060
	0.986405	0.019894	0.006453	0.000013	0.000013	0.000000	0.000000
	0.808327	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	−1.966066	−1.202089	−0.050385	−0.000238	−0.000117	−0.000000	−0.000000
	5.071334	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
p_l^k	1.669945	−0.000734	−0.011979	−0.000077	−0.000062	−0.0000001	−0.0000000
	−0.553334	−0.265715	−0.011213	0.000009	0.000049	0.0000000	−0.0000000
	3.157112	0.000194	0.000199	0.000036	0.000000	0.0000000	0.0000000
	−0.010473	0.472981	0.001175	0.000023	0.000000	0.0000000	0.0000000
	−0.418557	−0.126792	−0.001224	−0.000000	−0.000000	−0.0000000	−0.0000000
	−3.742317	−0.003329	−0.002775	−0.000008	−0.000000	−0.0000000	−0.0000000
	−0.280454	−0.096496	0.019365	0.000004	0.000000	0.0000000	0.0000000
	0.986405	0.019891	0.006452	0.000013	0.000013	0.0000000	0.0000000
	0.808327	0.000000	0.000000	0.000000	0.000000	0.0000000	0.0000000
	−1.966066	−1.198941	−0.050158	−0.000236	−0.000116	−0.0000001	−0.0000000
	5.071334	0.000000	0.000000	0.000000	0.000000	0.0000000	0.0000000
w_l^k	0.000000	0.938745	0.937587	−0.658049	−0.993791	−0.999814	−0.999994
	0.000000	−0.505397	−0.924816	−0.997946	−0.736814	−0.732063	−0.731940
	0.000000	0.992142	0.992448	0.997676	0.999971	0.999972	0.999985
	0.000000	0.230333	0.976910	0.998111	0.999985	0.999985	0.999988
	0.000000	−0.786258	−0.986394	−0.999666	−0.999993	−0.999993	−0.999994
	0.000000	−0.974433	−0.979687	−0.998687	−0.999979	−0.999979	−0.999987
	0.000000	−0.020780	−0.173094	0.999058	0.999964	0.999965	0.999974
	0.000000	0.125648	0.157045	0.659505	0.730656	0.731927	0.731968
	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	0.000000	−2.618737	−4.511208	−8.770844	−9.410332	−9.421646	−9.422023
	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$\lVert e^k \rVert_1$	2587.5638	1238.330	81.184452	27.842392	24.198173	24.066599	24.066474
$y^t w^k$	0.000000	13.969431	17.901316	23.485754	24.056044	24.065725	24.066169

V. SIMULATION STUDIES

Some computational experimentations are conducted to compare seven algorithms for L_1 estimation on the basis of computational effort required and numerical accuracy and stability of estimation. The main purpose of the experiment is to extensively analyze the performance indicators which in turn are affected by problem size and the condition number of the matrix X.

5.1. TEST PROBLEM GENERATOR

An evaluation of the performance of various algorithms requires representative data sets generated which can be varied greatly. Among two test problem generators; Kennedy, Gentle and Sposito (1977) and Hoffman and Shier (1980), the latter was used in this experiment. This generator has the desirable property of yielding problems whose solutions are known and unique in the case of a full-rank design matrix. This generator allows testing and collecting of the following influential variables; problem size (number of observations and parameters), statistical distributions (normal, uniform, zero-one) for each column of X, distributions (normal, uniform) of residuals, specification of solution vector, generation of solution from normal or uniform distribution, repetitions of rows, rank loss, mean and standard deviation of columns from normal, lower and upper limit of columns from uniform distribution, mean and standard deviation of residuals from normal, lower and upper limit of residuals from uniform, and random number seed used to initiate the random number generator.

5.2. PERFORMANCE INDICATORS

We discuss some important factors that are considered when assessing the effectiveness of algorithms and comparing them. Performance indicators which we consider are CPU time (the total processing time needed to execute a subroutine on a single problem), the number of iterations to compare computational efficiency, and numerical accuracy and stability to compare the quality of the results acquired. The CPU time was checked by calling a subroutine 'TIMEON'. Means and standard errors of CPU time and the number of iterations were obtained. The relative difference between the true parameter values and the estimated values, and between the known L_1 norms and estimated ones, can be compared. However, the numerical accuracy of the estimates is measured by the mean absolute deviations between true parameters (specified on the generator) and the estimates, and between true L_1 norm and the estimated ones, since the same unique parameter values and L_1 norm were specified for each algorithm.

Also, means and standard errors of estimates were computed to check the numerical accuracy and stability.

5.3. EXPERIMENTAL DESIGN

An appropriate experimental design is crucial in making statistically valid inferences about the performances of algorithms. In our experiment, we chose five values for N, the number of observations; 30, 50, 100, 200, 400, and eight values for P, the number of parameters; 2, 5, 10, 15, 20, 50, 100, 200. Only twenty five possible combinations as shown in Table 2 were tested, each with 25 replications per combination as suggested by Gilsinn et al.(1977). The values of the regression coefficients were arbitrarily specified. Each column of X was generated from a normal distribution with randomly selected mean and variance, and the errors were generated from a normal distribution with mean 0 and variance 5.0.

5.4. EXPERIMENTAL RESULTS

We generated twenty five problems for each of the twenty five combinations by Hoffman and Shier's test problem generator, and ran all 7 algorithms on each problem. Measures of CPU time, number of iterations and accuracy were obtained in the case of a full rank and usually highly ill-conditioned X.

5.4.1. Computational Efficiency

5.4.1.1. CPU time: Comparisons of computational efficiency were performed on 25 different problem sizes given in section 5.3. All CPU time quoted are on an IBM 370/158 using the FORTVCG compiler. Table 4 gives the means and standard errors of the CPU times (milliseconds). Also, Appendix B shows the relationships of average CPU time to the problem sizes. As we expect, Robers and Robers' algorithm is the slowest. This fact supports the assertion that direct application of the simplex method to L_1 estimation is not very efficient. Another significant result is that Wesolowsky's algorithm is competitive with Armstrong, Frome and Kung's and Barrodale and Roberts' algorithm only for problems with a few number of parameters and a large number of observations. Wesolowsky's algorithm did not terminate for several data sets when the number of parameters become large. Wesolowsky (1981) noted that, in the above situations, a small random value should be added to each y. Also, Bartels and Conn's algorithm did not terminate in considerably many number of iterations for problems with a large number of parameters and observations. Examination of Table 4 reveals that our proposed algorithm is in general faster than Bartels and Conn's and Robers and Robers' algorithm. For problems with a large number of parameters and observations it is faster than Abdelmalek's and Armstrong, Frome and Kung's algorithm. However, it is somewhat slower than Barrodale and Roberts' algorithm.

Table 4. Means and standard errors of CPU time

size	ABDEL	ARMST	BARRO	BARTE	PROPO	ROBER	WESOL
2 × 30	0.00	0.00	0.00	0.40	1.00	6.20	0.00
	(0.000)	(0.000)	(0.000)	(0.100)	(0.000)	(0.311)	(0.000)
2 × 50	0.24	0.04	0.00	1.00	2.00	18.28	0.00
	(0.087)	(0.040)	(0.000)	(0.000)	(0.000)	(1.096)	(0.000)
2 × 100	1.56	0.56	1.00	2.08	4.00	79.16	0.68
	(0.130)	(0.804)	(0.000)	(0.055)	(0.060)	(4.750)	(0.095)
2 × 200	4.52	1.80	2.72	4.76	8.68	345.84	1.88
	(0.352)	(0.141)	(0.092)	(0.119)	(0.170)	(13.228)	(0.066)
5 × 30	1.36	0.64	1.04	3.60	3.44	17.24	3.40 +
	(0.098)	(0.098)	(0.040)	(0.224)	(0.010)	(0.561)	
5 × 50	2.80	1.48	2.44	6.32	6.08	58.00	5.12 +
	(0.141)	(0.102)	(0.117)	(0.243)	(0.080)	(0.241)	
5 × 100	8.28	3.92	7.04	13.76	12.68	238.16	10.24 +
	(0.349)	(0.199)	(0.255)	(0.654)	(0.235)	(8.818)	
5 × 200	19.44	8.44	18.12	34.12	25.24	1156.72	18.36 +
	(0.866)	(0.370)	(0.552)	(1.777)	(0.345)	(50.513)	
10 × 30	4.60	3.28	3.28	9.40	7.44		15.20 +
	(0.216)	(0.169)	(0.092)	(0.265)	(0.115)		
10 × 50	9.32	7.04	7.84	19.44	14.04		18.12 +
	(0.390)	(0.308)	(0.206)	(0.578)	(0.325)		
10 × 100	25.44	14.68	20.12	45.20	29.52		45.36 +
	(0.785)	(0.541)	(0.628)	(1.323)	(0.565)		
10 × 200	58.64	31.60	50.68	99.88	60.40		118.56 +
	(2.270)	(1.178)	(1.267)	(3.285)	(1.235)		
15 × 30	8.84	8.60	6.12	17.96	17.64		
	(0.330)	(0.447)	(0.145)	(0.511)	(0.225)		
15 × 50	19.20	17.56	13.48	34.84	29.84		
	(0.603)	(0.627)	(0.306)	(0.875)	(0.440)		
15 × 100	52.40	33.92	39.00	83.08	61.36		
	(1.535)	(1.275)	(0.911)	(1.933)	(0.765)		
15 × 200	120.24	75.40	99.88	183.16	126.88		
	(3.915)	(2.518)	(1.926)	(4.234)	(1.575)		
20 × 30	12.92	13.56	8.64	24.96	20.04		
	(0.387)	(0.589)	(0.128)	(0.402)	(0.375)		
20 × 50	31.96	34.56	21.60	55.28	40.48		
	(1.148)	(1.657)	(0.392)	(1.530)	(0.830)		
20 × 100	83.00	74.12	60.72	123.56	87.72		
	(1.618)	(2.067)	(1.670)	(1.148)	(1.600)		
20 × 200	210.76	140.80	162.76	238.1 +	193.12		
	(5.265)	(2.995)	(3.779)		(3.330)		
50 × 100	406.12	1181.40	232.08	8621.7 +	414.48		
	(10.338)	(52.392)	(3.289)		(7.903)		
50 × 200	1363.08	2368.33	723.76	17243.3 +	979.00		
	(94.082)	(125.329)	(8.284)		(14.904)		
100 × 200	44152.80	39584.75	1995.28	23854.6 +	3457.00		
	(5688.513)	(797.605)	(20.053)		(60.232)		
100 × 400	224181.00	50422.00	6562.80	46480.1 +	7932.40		
		(1321.588)	(77.518)		(135.596)		
200 × 400	224181.0 +	50422.0 +	18815.00	248559.0 +	29106.20		
			(245.749)		(833.803)		

The relationships between CPU time and problem sizes were analyzed using a nonlinear regression approach to determine the effects of P and N on CPU time. The following nonlinear model was employed,

$$CPU = \alpha P^{\beta} N^{\gamma} \tag{5.1}$$

where P and N indicate the number of parameters and the number of observations, respectively. Table 5 shows the estimated coefficients of α, β, and γ. In this table, the coefficient $\hat{\beta}$, $\hat{\gamma}$ indicate the elasticities of CPU time with respect to P and N, respectively. Therefore, large coefficients of $\hat{\beta}$ and $\hat{\gamma}$ imply that the algorithm is sensitive to increase in P and N. In addition, if $\hat{\beta}$ is larger than $\hat{\gamma}$, then this means that the algorithm is more sensitive to increase in the number of parameters than in the number of observations. These results show that all algorithms, except for Robers and Robers′, are more sensitive to increase in P than increase in N. Armstrong, Frome and Kung's algorithm is the most sensitive to increase in P. We note that the proposed algorithm is the least sensitive to increase in problem size.

Table 5. Relationships between CPU time and problem size

coeff.	ABDEL	ARMST	BARRO	BARTE	PROPO	ROBER
$\hat{\alpha}$	0.000056	0.000053	0.000540	0.002596	0.004129	0.001667
$\hat{\beta}$	2.1493	2.5174	1.6580	1.7476	1.5118	1.2240
$\hat{\gamma}$	1.7852	1.4953	1.4370	1.2183	1.2210	2.1573

5.4.1.2. Number of iterations: Even though the number of iterations is not a 'good' measure of efficiency of an algorithm because the computational effort per iteration may vary considerably from one algorithm to another, it was considered as a performance indicator because of its close relationship with CPU time. Table 6 and Appendix B show that the proposed algorithm take on the average the least number of iterations to solve the different combinations of problems. This agrees with previous conjectures that algorithms which employ Karmarkar type algorithm take less iterations than algorithms which use a simplex type method. Note that the proposed algorithm takes significantly less iterations than the other algorithms for problems with more than 10 parameters. Subsequently, decreases in the average CPU time are only possible by improving the efficiency of this algorithm at each iteration.

The relationships between the number of iterations and problem sizes are analyzed using the nonlinear model (5.1). The results show that the proposed algorithm is slightly more sensitive to increase in N than increase in P . Abdelmalek's algorithm is the most sensitive to increase in P, and the proposed algorithm is the least sensitive to increase in problem size.

Table 6. Means and standard errors of the number of iterations

size	ABDEL	ARMST	BARRO	BARTE	PROPO	ROBER	WESOL
2 × 30	4.60	2.76	2.40	2.40	7.44	4.64	3.50
	(0.153)	(0.202)	(0.100)	(0.100)	(0.130)	(0.223)	(0.170)
2 × 50	4.96	3.56	2.36	2.32	8.04	4.80	4.00
	(0.204)	(0.277)	(0.098)	(0.095)	(0.145)	(0.294)	(0.208)
2 × 100	5.32	3.76	2.52	2.44	8.32	5.16	4.12
	(0.275)	(0.302)	(0.143)	(0.130)	(0.110)	(0.320)	(0.211)
2 × 200	5.32	4.12	2.72	2.76	8.68	5.60	4.72
	(0.229)	(0.313)	(0.123)	(0.119)	(0.170)	(0.224)	(0.169)
5 × 30	11.68	8.56	12.92	11.16	9.40	12.24	25.6 +
	(0.403)	(0.480)	(0.481)	(0.711)	(0.165)	(0.393)	
5 × 50	13.64	10.80	14.48	12.76	9.64	15.20	26.2 +
	(0.443)	(0.455)	(0.707)	(0.524)	(0.150)	(0.635)	
5 × 100	16.76	13.84	17.60	15.00	9.96	15.84	28.6 +
	(0.618)	(0.713)	(0.757)	(0.719)	(0.160)	(0.596)	
5 × 200	18.64	14.60	18.80	19.92	10.20	19.24	26.4 +
	(1.328)	(0.594)	(0.831)	(1.110)	(0.160)	(0.855)	
10 × 30	20.08	12.00	20.92	19.40	7.44		38.0 +
	(0.651)	(0.597)	(0.535)	(0.532)	(0.115)		
10 × 50	25.08	19.96	29.32	26.44	8.52		35.0 +
	(0.823)	(0.848)	(0.808)	(0.796)	(0.200)		
10 × 100	33.16	26.48	34.76	34.04	9.16		56.4 +
	(0.960)	(1.052)	(1.285)	(1.035)	(0.190)		
10 × 200	37.68	32.76	40.08	40.24	9.52		84.9 +
	(1.426)	(1.291)	(1.260)	(1.415)	(0.210)		
15 × 30	26.12	13.96	27.84	27.00	9.32		
	(0.689)	(0.725)	(0.596)	(0.771)	(0.110)		
15 × 50	34.56	25.16	36.88	36.64	9.72		
	(0.837)	(0.901)	(0.893)	(0.986)	(0.170)		
15 × 100	49.84	35.32	51.64	48.68	10.32		
	(1.360)	(1.406)	(1.315)	(1.196)	(0.140)		
15 × 200	58.40	50.24	63.04	57.88	10.76		
	(1.764)	(1.826)	(1.368)	(1.442)	(0.145)		
20 × 30	29.40	11.52	31.56	30.36	6.68		
	(0.517)	(0.504)	(0.425)	(0.483)	(0.125)		
20 × 50	42.96	27.00	46.48	46.68	8.36		
	(1.250)	(1.399)	(0.942)	(1.329)	(0.190)		
20 × 100	59.20	47.20	64.68	58.92	9.20		
	(1.100)	(1.273)	(1.930)	(0.661)	(0.185)		
20 × 200	80.24	62.20	83.84	60.0 +	10.24		
	(1.974)	(1.329)	(2.192)		(0.195)		
50 × 100	113.92	93.04	112.08	2000 +	9.28		
	(2.540)	(4.226)	(1.735)		(0.196)		
50 × 200	215.08	162.90	175.52	1000 +	10.52		
	(14.751)	(8.039)	(2.155)		(0.175)		
100 × 200	3061.20	438.25	250.88	1500 +	10.92		
	(388.813)	(7.554)	(2.550)		(0.210)		
100 × 400	8841.00	527.50	412.10	1500 +	11.70		
		(10.571)	(4.952)		(0.213)		
200 × 400	8841.0 +	527.5 +	577.60	4000 +	12.20		
			(7.580)		(0.374)		

Table 7. Relationships between number of iterations and problem size

coeff.	ABDEL	ARMST	BARRO	BARTE	PROPO
$\hat{\alpha}$	0.1161	0.1916	0.8259	0.3623	5.2571
$\hat{\beta}$	1.2816	0.9990	1.0968	1.2741	0.0314
$\hat{\gamma}$	0.6082	0.5369	0.1952	0.2995	0.1090

5.4.2. Numerical Accuracy and Stability

To compare the numerical accuracy and stability of the seven algorithms, the means and the standard errors of the estimates, and the mean absolute deviations of estimates were computed for each of the 25 combinations. The two sets of results given were exemplary of the other results obtained.

Table 8 $\sim$ 11 indicate that the estimates of the intercept terms are considerably inaccurate. This fact has been revealed by Ashar and Wallace (1963), Kiountouzis (1973) and Money et al. (1982). Moreover, Forsythe (1972) noted that when error distribution is not symmetric, the estimators of intercepts are biased while the estimators of slopes are still unbiased.

There is no significant difference in numerical accuracy among Abdelmalek's, Armstrong, Frome and Kung's, Barrodale and Roberts', Bartels and Conn's and the proposed algorithm. On the other hand, Robers and Robers' and Wesolowsky's algorithm yield slightly less accurate estimates than the others. Investigation of Table 9 reveals that the proposed algorithm is the most accurate. Further examination of Table 11 reveals that Barrodale and Roberts' and the proposed algorithm are more accurate than the others. Table 10 indicates that the proposed algorithm seems to yield the most stable estimates.

In the light of these experiments, the proposed algorithm appears to yield numerically accurate and stable estimates. In addition, it is more computationally efficient than other algorithms, except for Barrodale and Roberts', when problem size is considerably large.

Table 8. Means and standard errors of estimates (P = 5,N = 30)

coeff.	ABDEL	ARMST	BARRO	BARTE	PROPO	ROBER	WESOL
$\beta_0 = 2$	2.064196	1.974875	1.990085	1.948334	2.040825	1.867842	2.083836
	(0.05893)	(0.04596)	(0.05313)	(0.02696)	(0.06744)	(0.23977)	(0.03606)
$\beta_1 = 5$	5.000068	5.000311	4.999783	5.000138	4.999914	5.001966	4.992641
	(0.00014)	(0.00014)	(0.00015)	(0.00014)	(0.00011)	(0.00194)	(0.00546)
$\beta_2 = 8$	7.999809	8.000103	8.000021	8.000049	7.999895	8.005767	8.009417
	(0.00019)	(0.00018)	(0.00013)	(0.00016)	(0.00012)	(0.00509)	(0.01024)
$\beta_3 = 11$	10.99980	10.99995	11.00004	11.00001	10.99988	10.99634	10.99697
	(0.00013)	(0.00010)	(0.00013)	(0.00008)	(0.00015)	(0.00398)	(0.00327)
$\beta_4 = 14$	13.99992	13.99999	14.00006	14.00020	13.99993	14.00124	13.98773
	(0.00018)	(0.00013)	(0.00015)	(0.00011)	(0.00017)	(0.00061)	(0.01022)

Table 9. Mean absolute deviation of estimates (P = 5,N = 30)

coeff.	ABDEL	ARMST	BARRO	BARTE	PROPO	ROBER	WESOL
$\beta_0 = 2$	0.133194	0.147040	0.140255	0.118822	0.130449	0.575967	0.124455
$\beta_1 = 5$	0.000492	0.000479	0.000437	0.000470	0.000325	0.003965	0.007654
$\beta_2 = 8$	0.000679	0.000693	0.000463	0.000645	0.000342	0.006894	0.011047
$\beta_3 = 11$	0.000322	0.000325	0.000341	0.000326	0.000288	0.005070	0.004021
$\beta_4 = 14$	0.000515	0.000411	0.000445	0.000434	0.000383	0.001746	0.012458

Table 10. Means and standard errors of estimates (P = 10,N = 200)

coeff.	ABDEL	ARMST	BARRO	BARTE	PROPO
$\beta_0 = 2$	2.455741	1.784963	2.038342	1.816144	1.902342
	(0.63377)	(0.10345)	(0.07118)	(0.12339)	(0.06350)
$\beta_1 = 5$	5.007535	4.999822	5.000005	4.999698	4.999715
	(0.00738)	(0.00021)	(0.00008)	(0.00026)	(0.00014)
$\beta_2 = 8$	7.995537	7.999609	7.999981	7.999756	8.000102
	(0.00399)	(0.00030)	(0.00015)	(0.00039)	(0.00012)
$\beta_3 = 11$	11.00626	11.00044	11.00003	11.00035	11.00015
	(0.00591)	(0.00022)	(0.00015)	(0.00027)	(0.00016)
$\beta_4 = 14$	14.00022	13.99973	13.99981	14.00010	13.99984
	(0.00051)	(0.00025)	(0.00012)	(0.00033)	(0.00010)
$\beta_5 = 14$	13.99837	14.00044	14.00009	14.00033	14.00021
	(0.00227)	(0.00025)	(0.00007)	(0.00035)	(0.00009)
$\beta_6 = 11$	10.99802	11.00033	10.99997	11.00052	11.00035
	(0.00288)	(0.00034)	(0.00020)	(0.00041)	(0.00019)
$\beta_7 = 8$	7.995828	8.000307	7.999938	7.999869	8.000163
	(0.00479)	(0.00037)	(0.00016)	(0.00027)	(0.00014)
$\beta_8 = 5$	5.005937	4.999998	4.999952	4.999681	4.999794
	(0.00620)	(0.00025)	(0.00017)	(0.00034)	(0.00014)
$\beta_9 = 2$	1.988790	2.001391	1.999755	2.001494	2.000632
	(0.01098)	(0.00061)	(0.00049)	(0.00075)	(0.00043)

Table 11. Mean absolute deviation of estimates (P = 10,N = 200)

coeff.	ABDEL	ARMST	BARRO	BARTE	PROPO
$\beta_0 = 2$	1.099932	0.300805	0.162078	0.305997	0.175341
$\beta_1 = 5$	0.008271	0.000604	0.000310	0.000652	0.000433
$\beta_2 = 8$	0.005632	0.000925	0.000466	0.000886	0.000407
$\beta_3 = 11$	0.006889	0.000726	0.000411	0.000810	0.000482
$\beta_4 = 14$	0.001566	0.000816	0.000456	0.000823	0.000399
$\beta_5 = 14$	0.003287	0.000631	0.000264	0.000776	0.000351
$\beta_6 = 11$	0.004569	0.001024	0.000589	0.001100	0.000632
$\beta_7 = 8$	0.007730	0.001112	0.000553	0.000965	0.000529
$\beta_8 = 5$	0.007317	0.000884	0.000610	0.001073	0.000500
$\beta_9 = 2$	0.014948	0.002082	0.001378	0.002120	0.001478

VI. SUMMARY AND CONCLUSIONS

The main purposes of the literature review are (*i*) to introduce the idea of the minimum L_p norm estimation, and show the superiority of L_p estimates to least squares estimates in certain situations, (*ii*) to propose some criteria for the choice of the p value in L_p estimation, (*iii*) to investigate the statistical properties of L_1 estimators, and establish some statistical inference procedures, and (*iv*) to examine several computational procedures and algorithms for unconstrained and constrained L_1 estimation.

A number of experimental results demonstrate that, in certain cases, minimum L_p norm estimation is preferred to the least squares criterion. It is apparent that the L_1 estimator is considerably superior in estimation effectiveness for those distributions which have a long tail due to outliers. It should be noted that the L_p norm one should use for estimation depends greatly on the distribution of the errors. Furthermore, there is a large variation in the effectiveness of various norms and no single norm is preferable in all situations. For the choice of the p value we suggest that L_2 estimation be performed on the sample data and that the residuals from this regression be used to estimate the kurtosis of the error distribution. Then use the criteria of Sposito, Hand and Skarpness (1983) to determine the value of p in L_p norm estimation.

Two general types of computational methods for L_1 estimation are considered. However, it is well known that the linear programming technique has nice advantages. A number of algorithms for the linear programming problems have been reviewed. So far Barrodale and Roberts' algorithm has been verified to be the most efficient for unconstrained L_1 estimation. This algorithm allows us to reduce significantly the total number of iterations required by a standard simplex algorithm by bypassing several neighboring simplex vertices in a single iteration. Extension of this algorithm to the constrained L_1 estimation seems to be also efficient. However, this algorithm has given incorrect results for several data sets we generated and tested.

A procedure based on the modified Karmarkar algorithm for the linear programming problem is proposed for L_1 estimation, and L_1 estimation with linear equality and inequality constraints. Simulation studies were conducted to compare the algorithms, from the viewpoint of computational efficiency and numerical accuracy and stability of the estimators, under various problem sizes and condition numbers associated with the design matrix X. From the simulation studies, we know that the proposed algorithm is not as efficient in computing the L_1 estimate as Barrodale and Roberts' algorithm. However, the proposed algorithm is more efficient than Robers and Robers', Wesolowsky's and Bartels and Conn's algorithm. And it is faster than Armstrong, Frome and Kung's and Abdelmalek's algorithm when the problem size is large. Also the CPU time of our algorithm is more stable than the others. The proposed algorithm yields more accurate estimates than the others and has the same accuracy as Barrodale and Roberts' algorithm. However, estimates obtained by our algorithm were more stable than Barrodale and Roberts'. Moreover, it can handle the problem of a rank-deficient design matrix. Even though our algorithm is not the most efficient, substantial improvement are possible if we can update the orthogonal triangularization at each iteration.

Appendix A. Numerical Experiments on the Choice of Step Length

Table 12. Number of iterations

and mean deviation of estimates (P = 5,N = 100)

α	0.95	0.96	0.97	0.98	0.99
iteration	10.04	10.00	9.96	9.96	9.88
$\beta_0 = 2$	0.05068005	0.05068135	0.05067680	0.05057087	0.05048912
$\beta_1 = 5$	0.00023600	0.00023582	0.00023741	0.00023702	0.00023630
$\beta_2 = 8$	0.00030413	0.00030541	0.00030716	0.00030679	0.00030707
$\beta_3 = 11$	0.00018298	0.00018402	0.00018456	0.00018361	0.00018252
$\beta_4 = 14$	0.00018250	0.00018307	0.00018508	0.00018476	0.00018402

Table 13. Number of iterations

and mean deviation of estimates (P = 10,N = 200)

α	0.95	0.96	0.97	0.98	0.99
iteration	9.80	9.72	9.52	9.56	9.60
$\beta_0 = 2$	0.17523807	0.17787778	0.17534125	0.17432421	0.17290640
$\beta_1 = 5$	0.00045002	0.00048012	0.00043314	0.00043460	0.00043532
$\beta_2 = 8$	0.00042044	0.00043666	0.00040736	0.00041095	0.00041454
$\beta_3 = 11$	0.00049734	0.00052093	0.00048218	0.00048423	0.00048587
$\beta_4 = 14$	0.00040115	0.00040313	0.00039859	0.00039941	0.00039996
$\beta_5 = 14$	0.00035831	0.00036721	0.00035138	0.00035345	0.00035547
$\beta_6 = 11$	0.00063707	0.00064872	0.00063164	0.00063067	0.00062857
$\beta_7 = 8$	0.00053096	0.00054975	0.00052902	0.00052443	0.00051667
$\beta_8 = 5$	0.00051631	0.00054131	0.00050040	0.00050314	0.00050539
$\beta_9 = 2$	0.00149204	0.00153739	0.00147768	0.00147216	0.00146263

Appendix B. Results on the Computational Efficiency

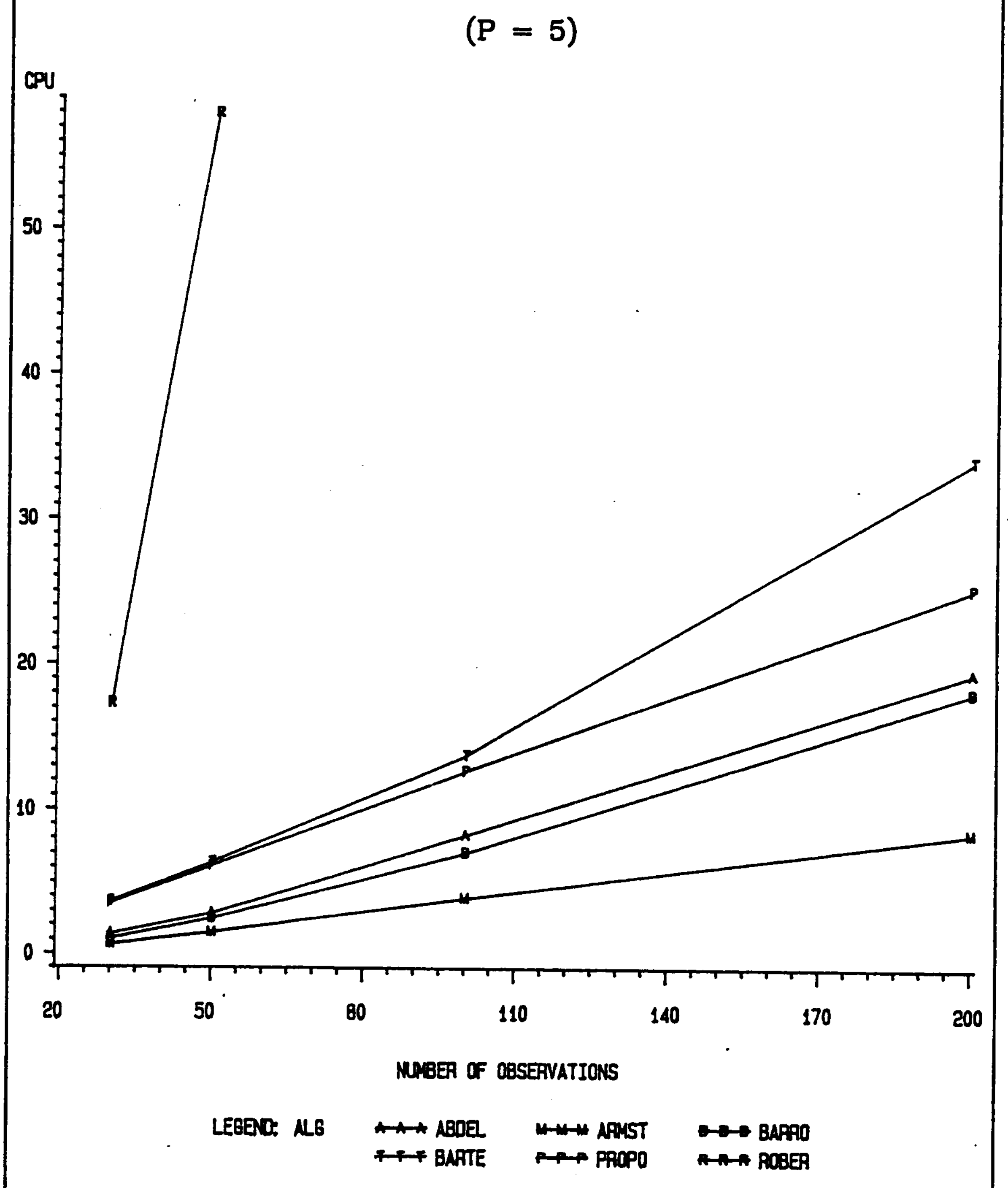


Figure 1. Variation in CPU time with problem size (P = 5)

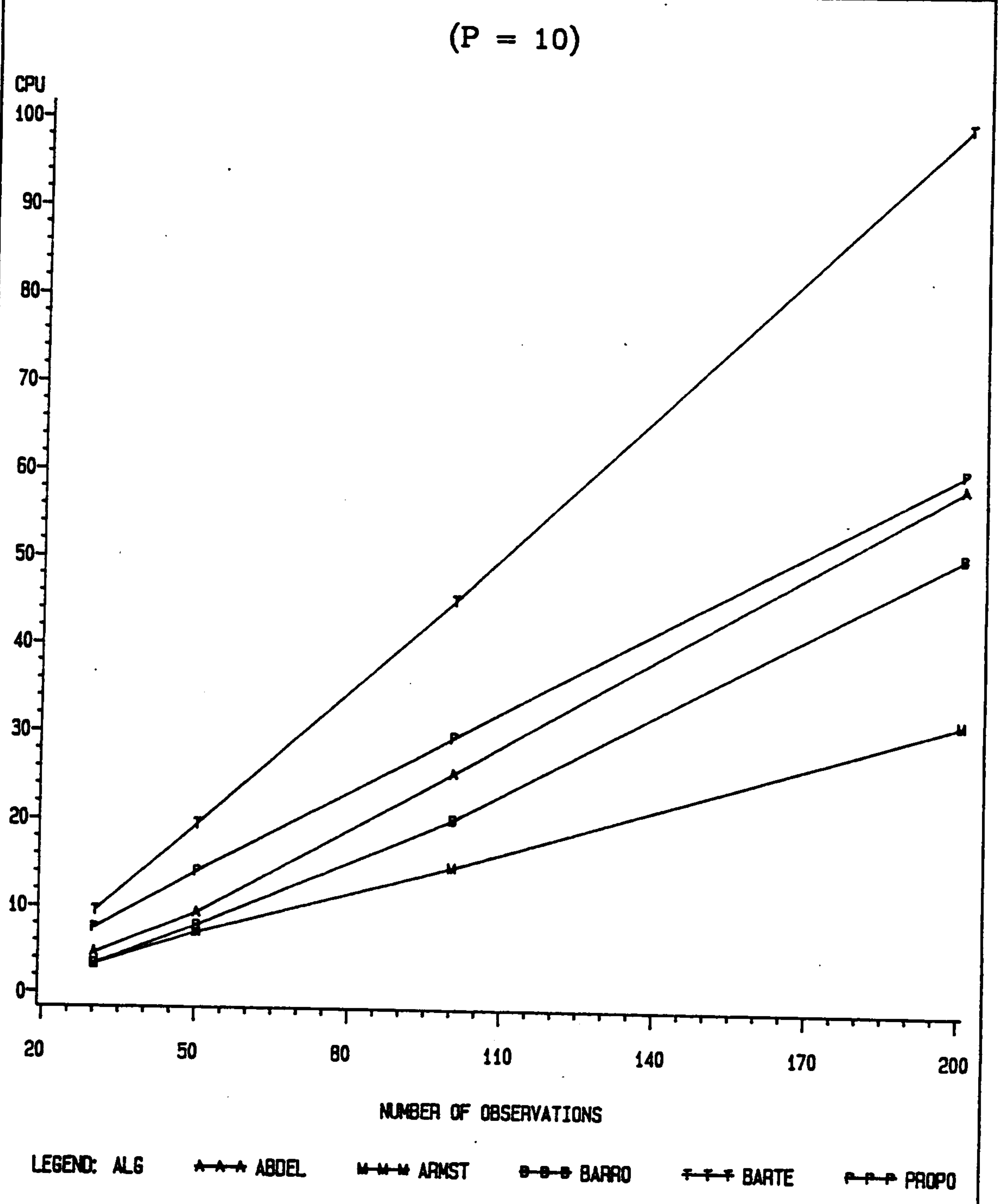


Figure 2. Variation in CPU time with problem size (P = 10)

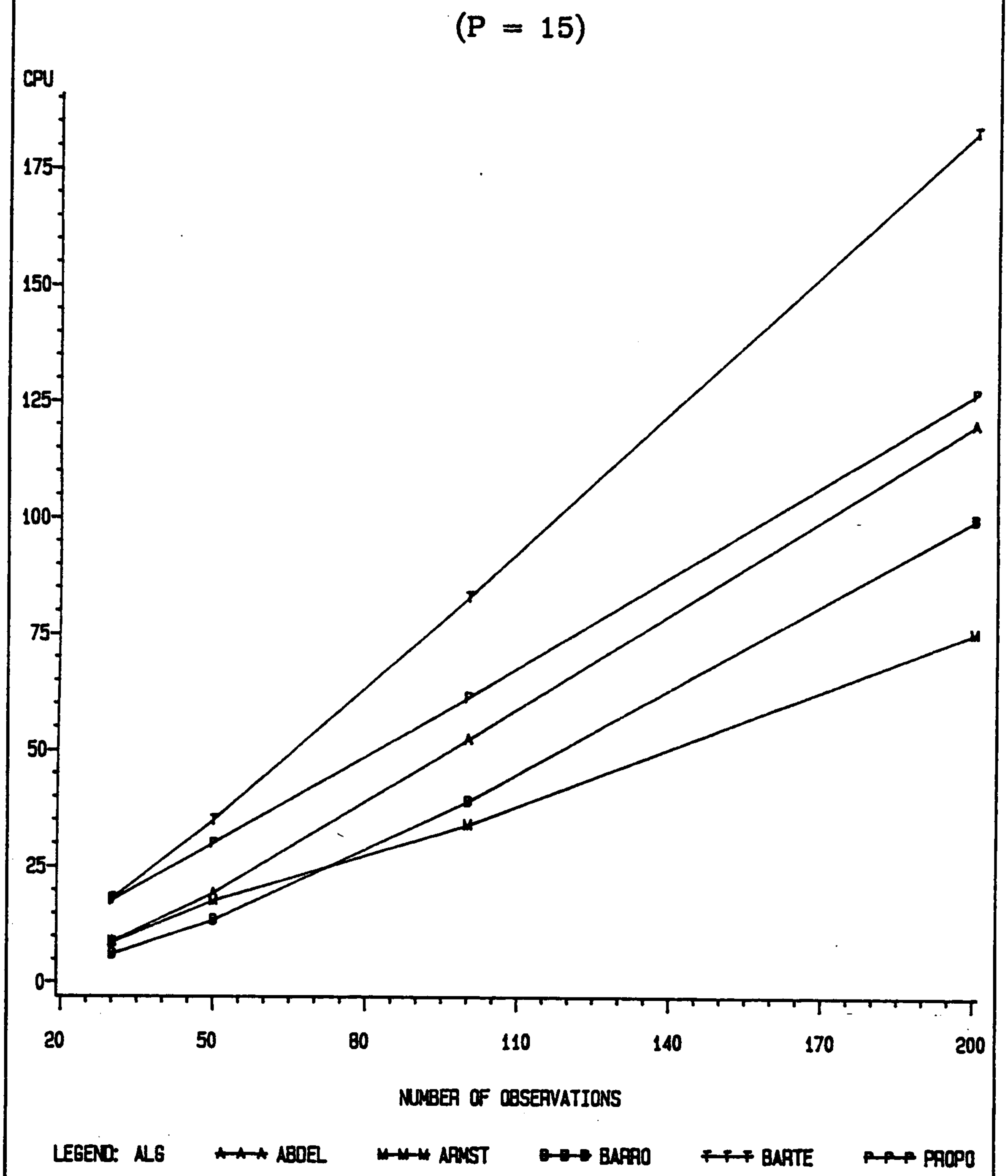


Figure 3. Variation in CPU time with problem size (P = 15)

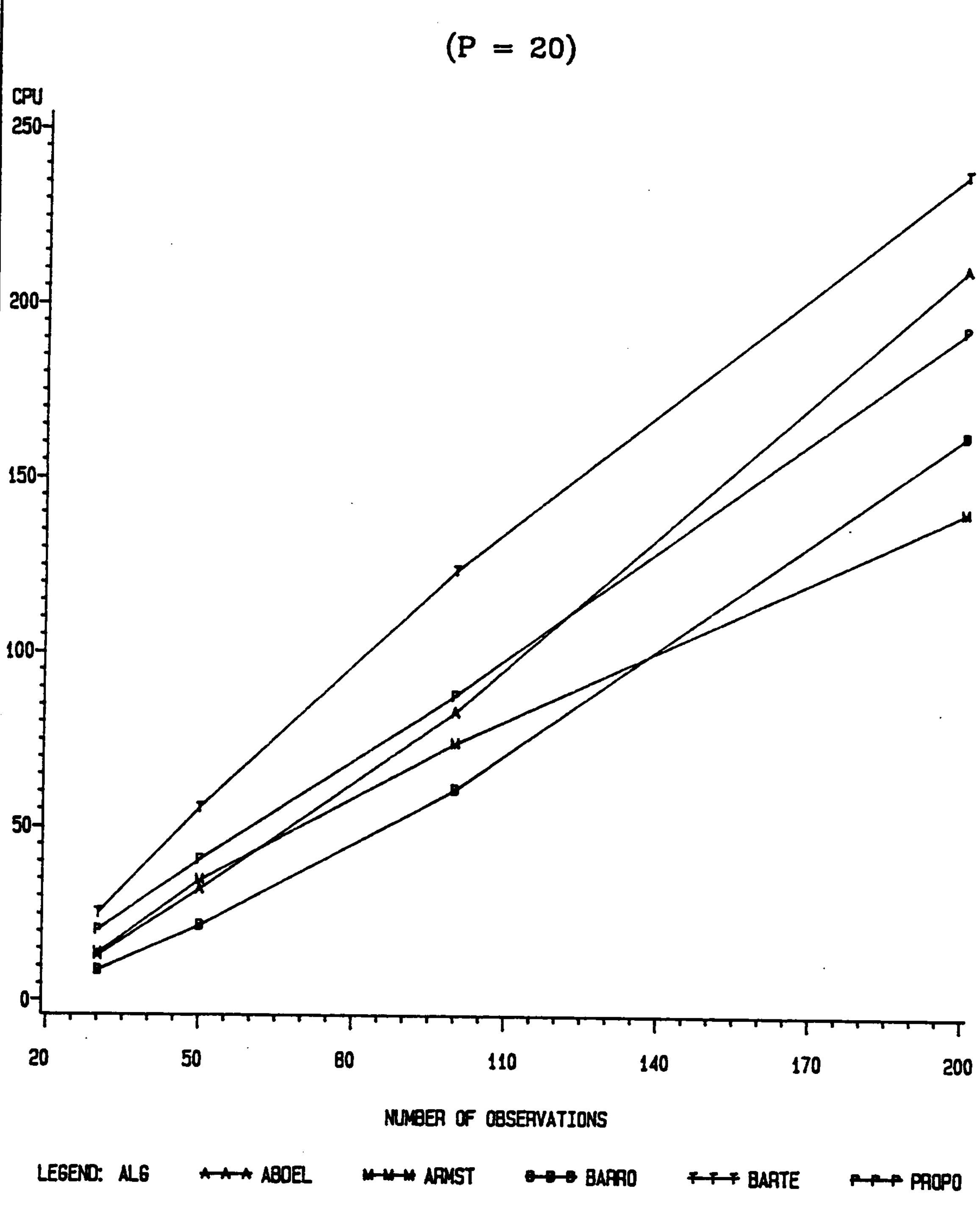


Figure 4. Variation in CPU time with problem size (P = 20)

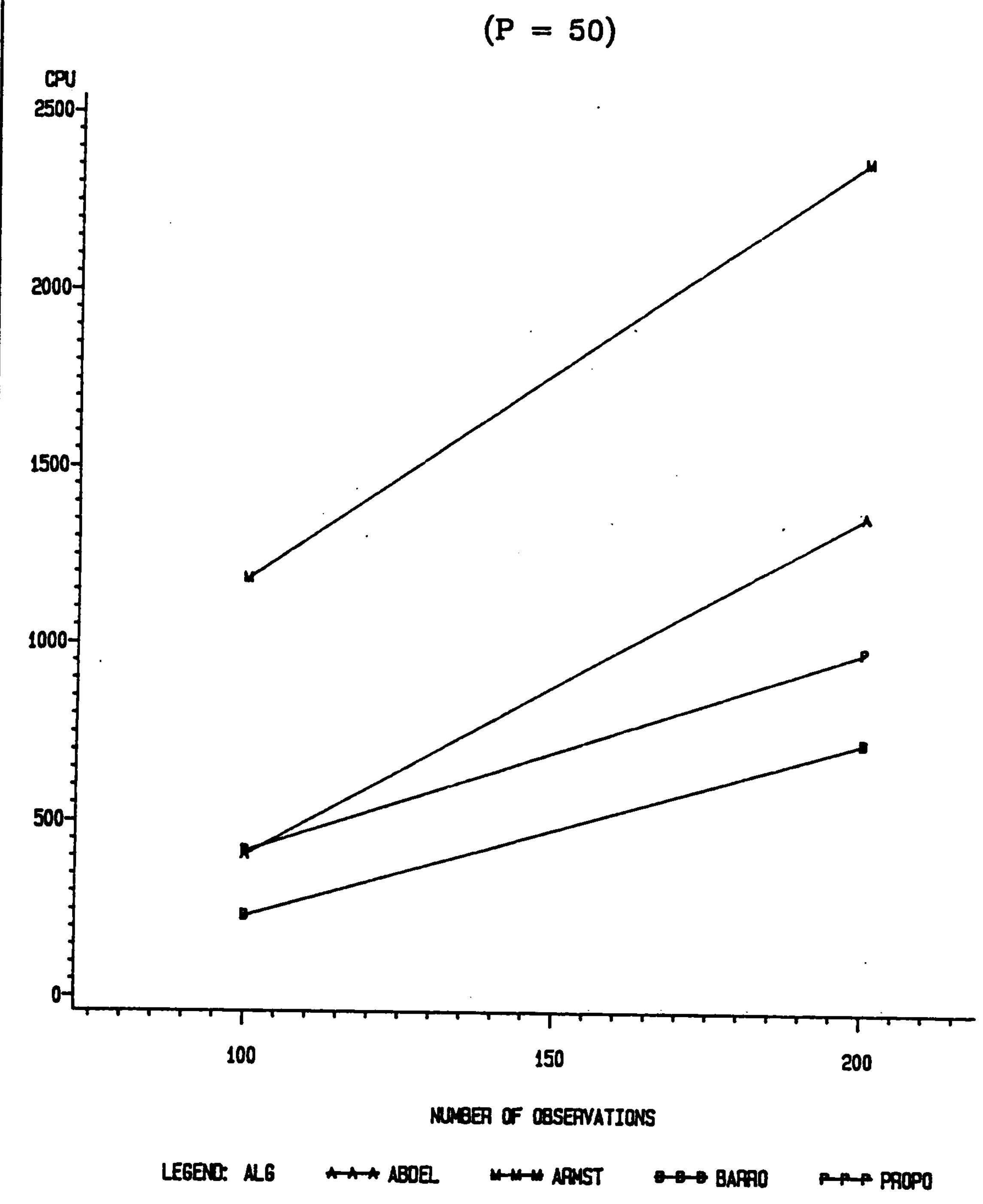


Figure 5. Variation in CPU time with problem size (P = 50)

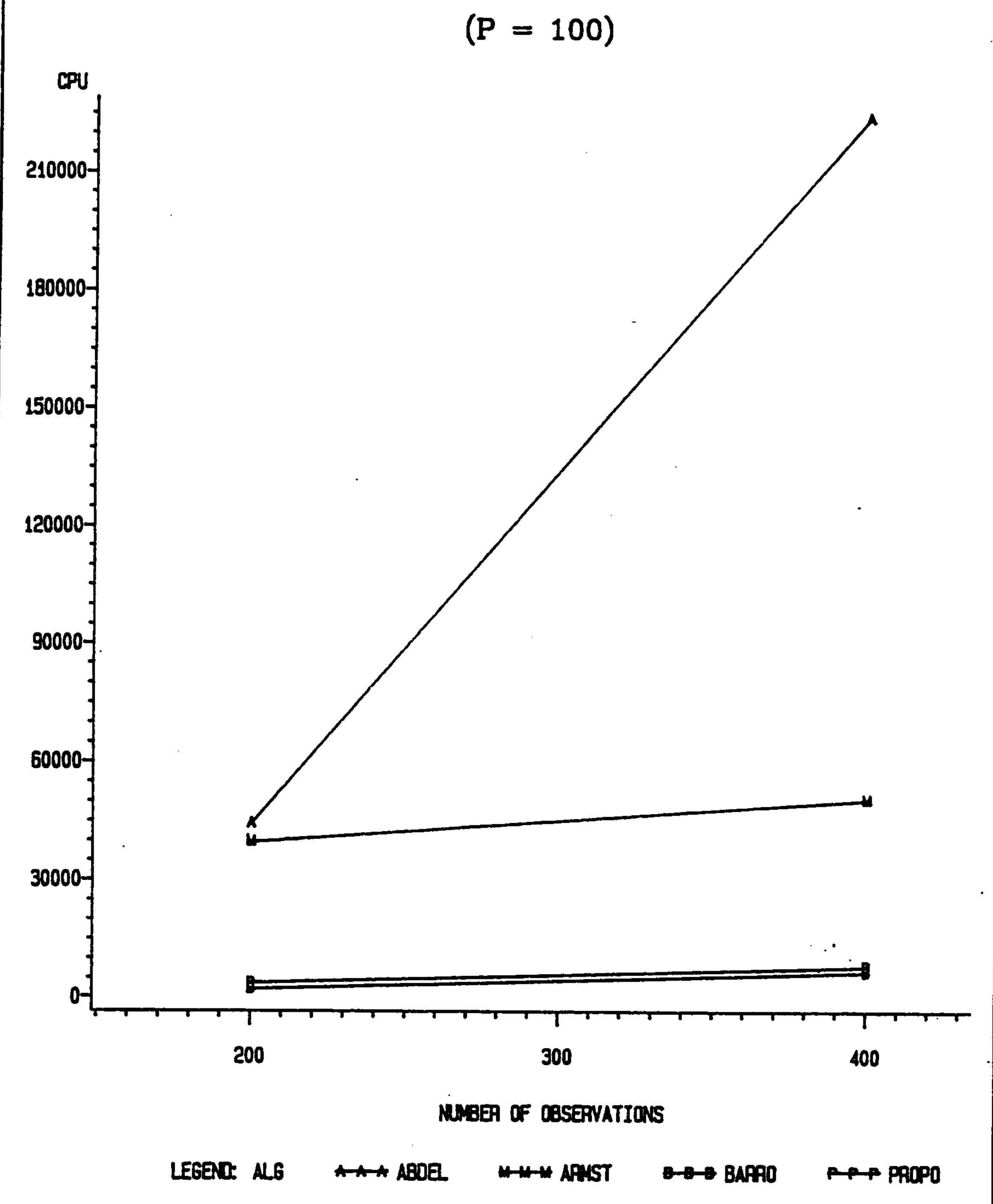


Figure 6. Variation in CPU time with problem size (P = 100)

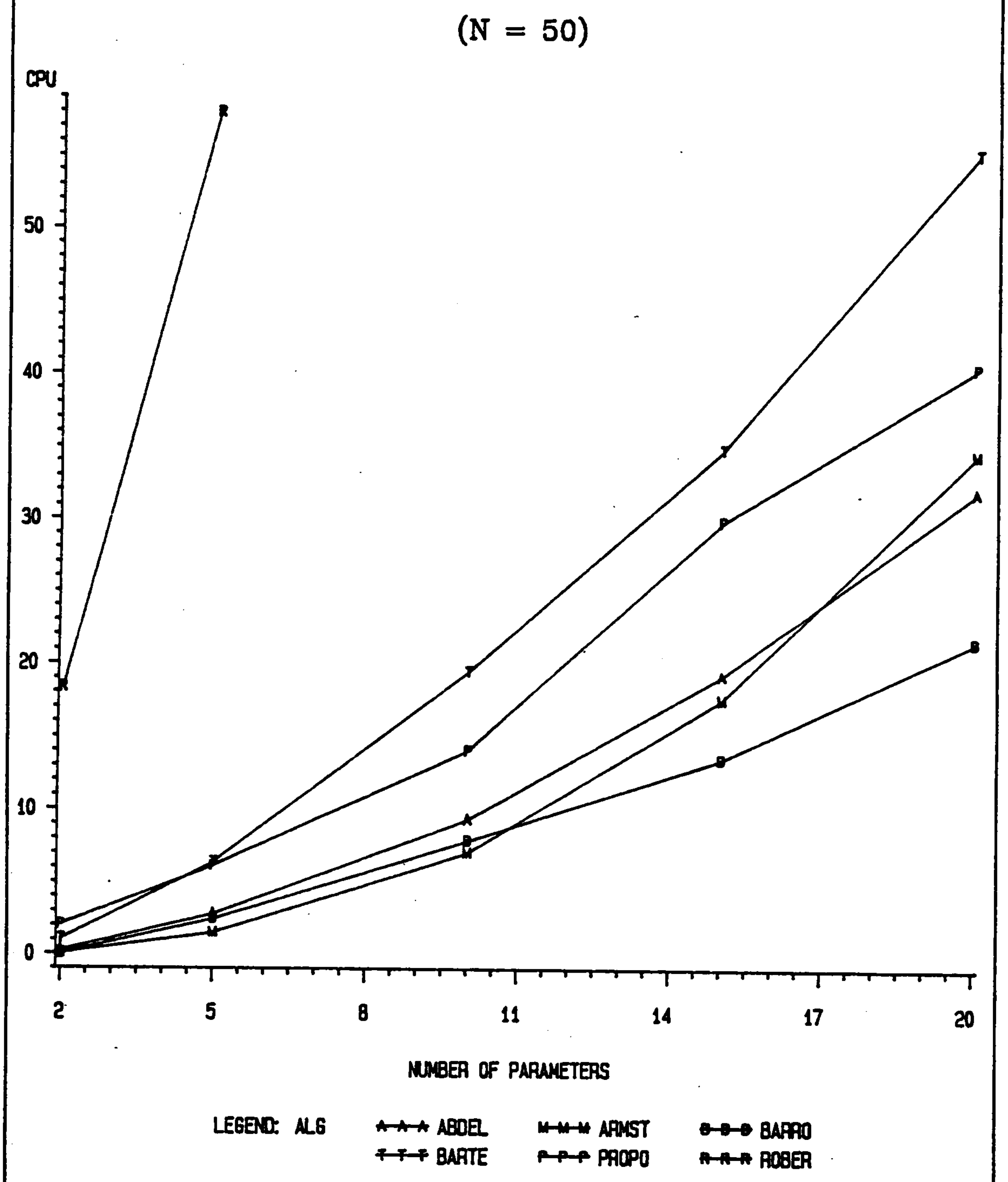


Figure 7. Variation in CPU time with problem size (N = 50)

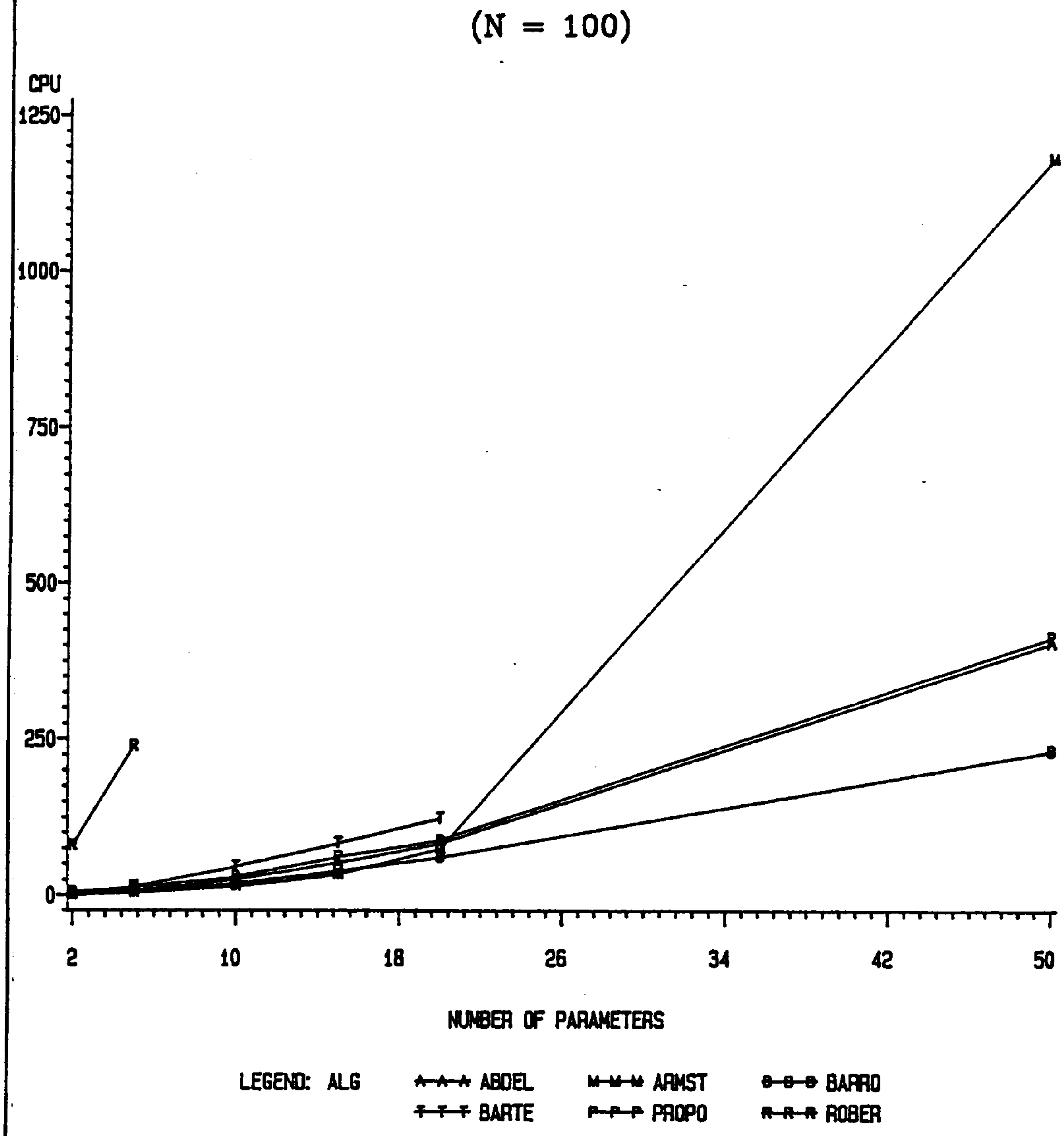


Figure 8. Variation in CPU time with problem size (N = 100)

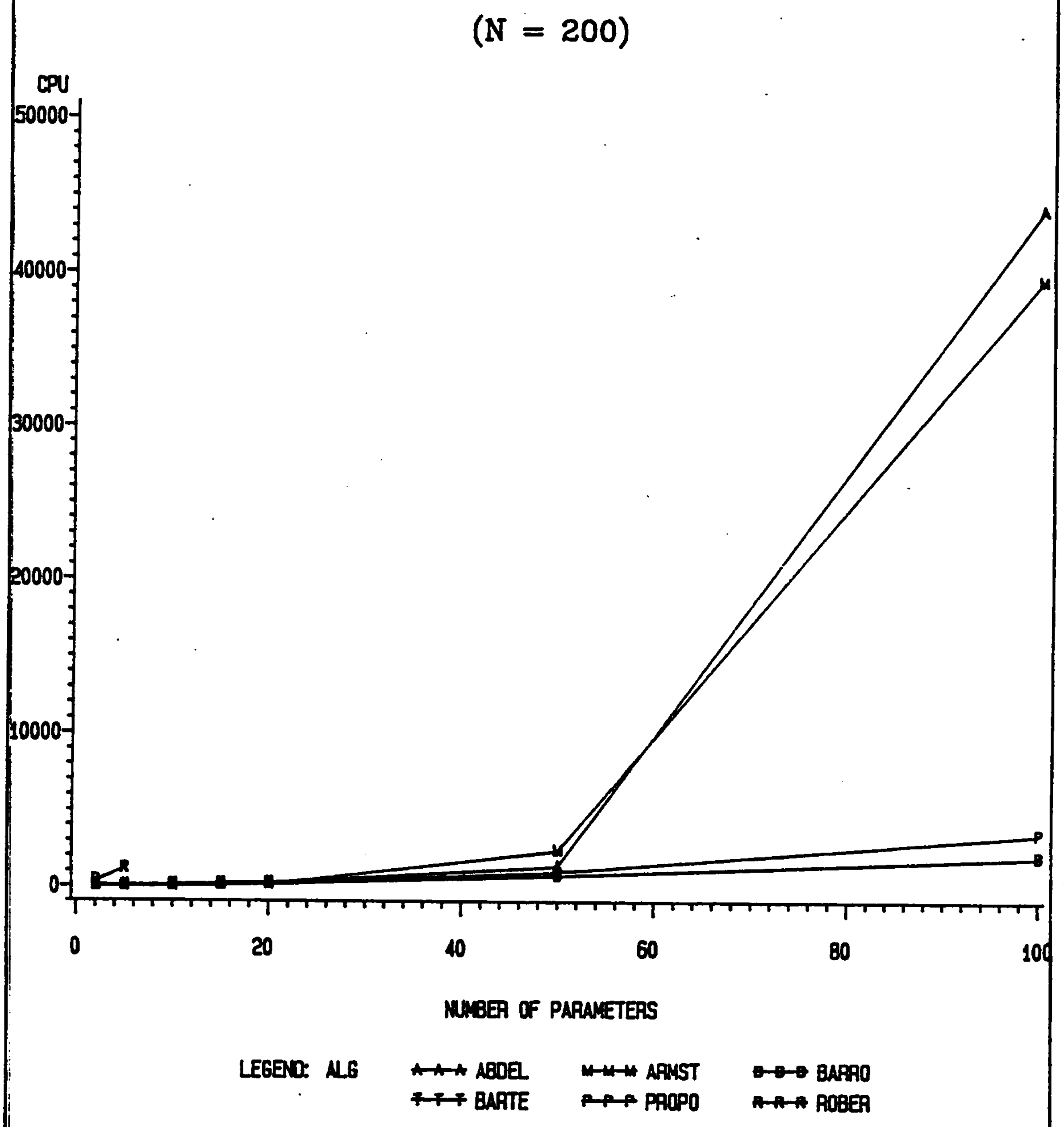


Figure 9. Variation in CPU time with problem size (N = 200)

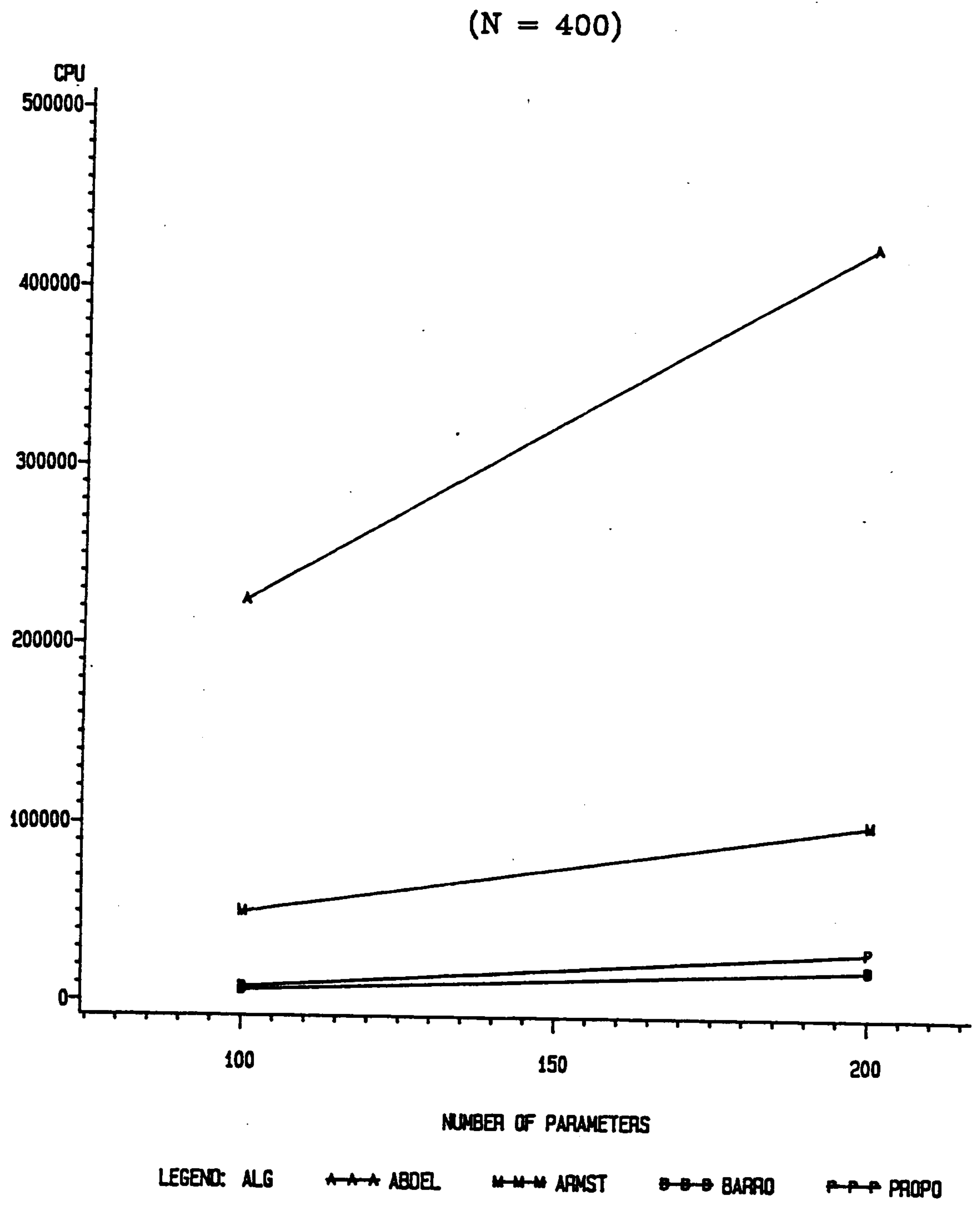


Figure 10. Variation in CPU time with problem size (N = 400)

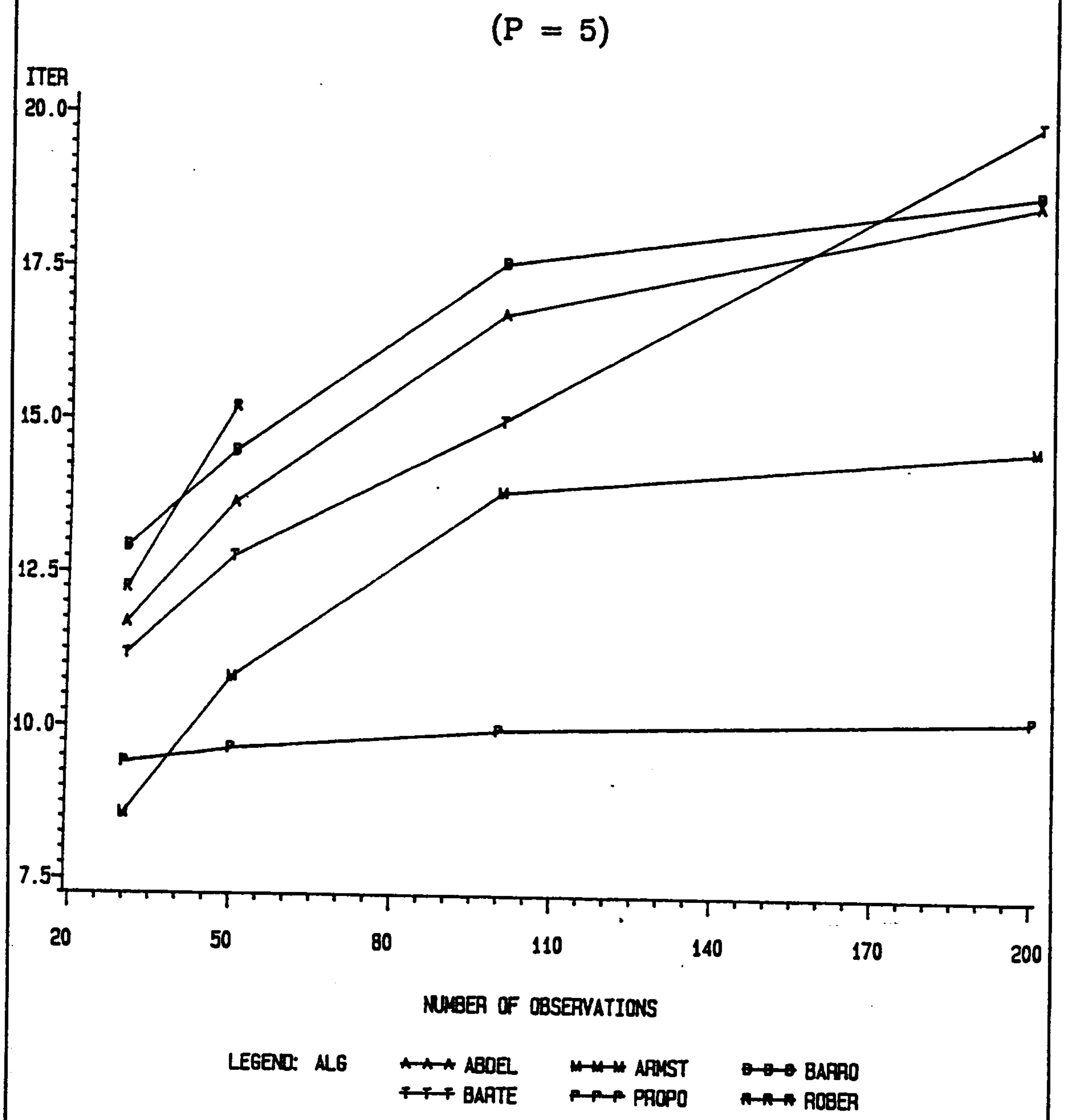


Figure 11. Variation in number of iterations with problem size (P = 5)

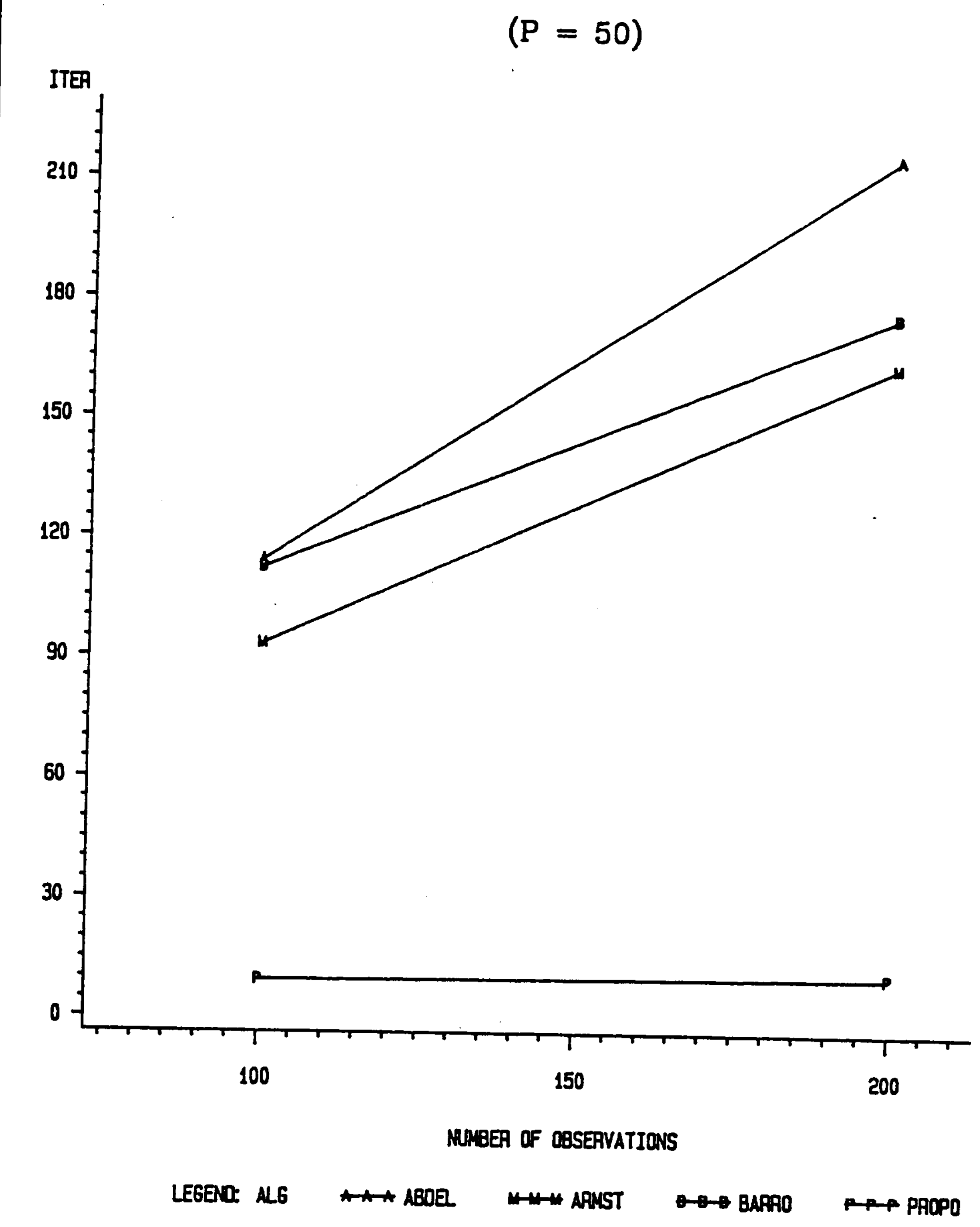


Figure 12. Variation in number of iterations with problem size (P = 50)

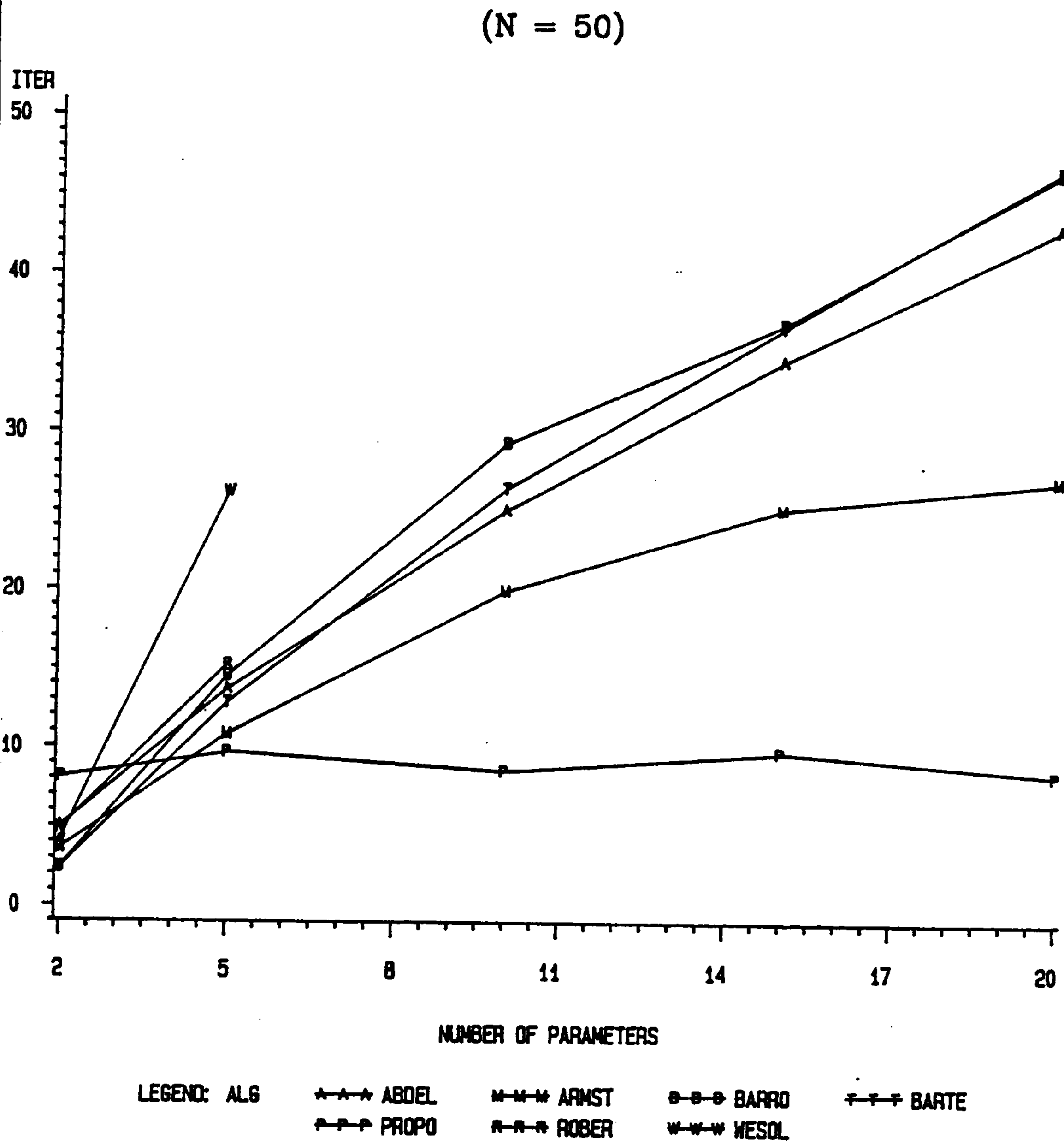


Figure 13. Variation in number of iterations with problem size (N = 50)

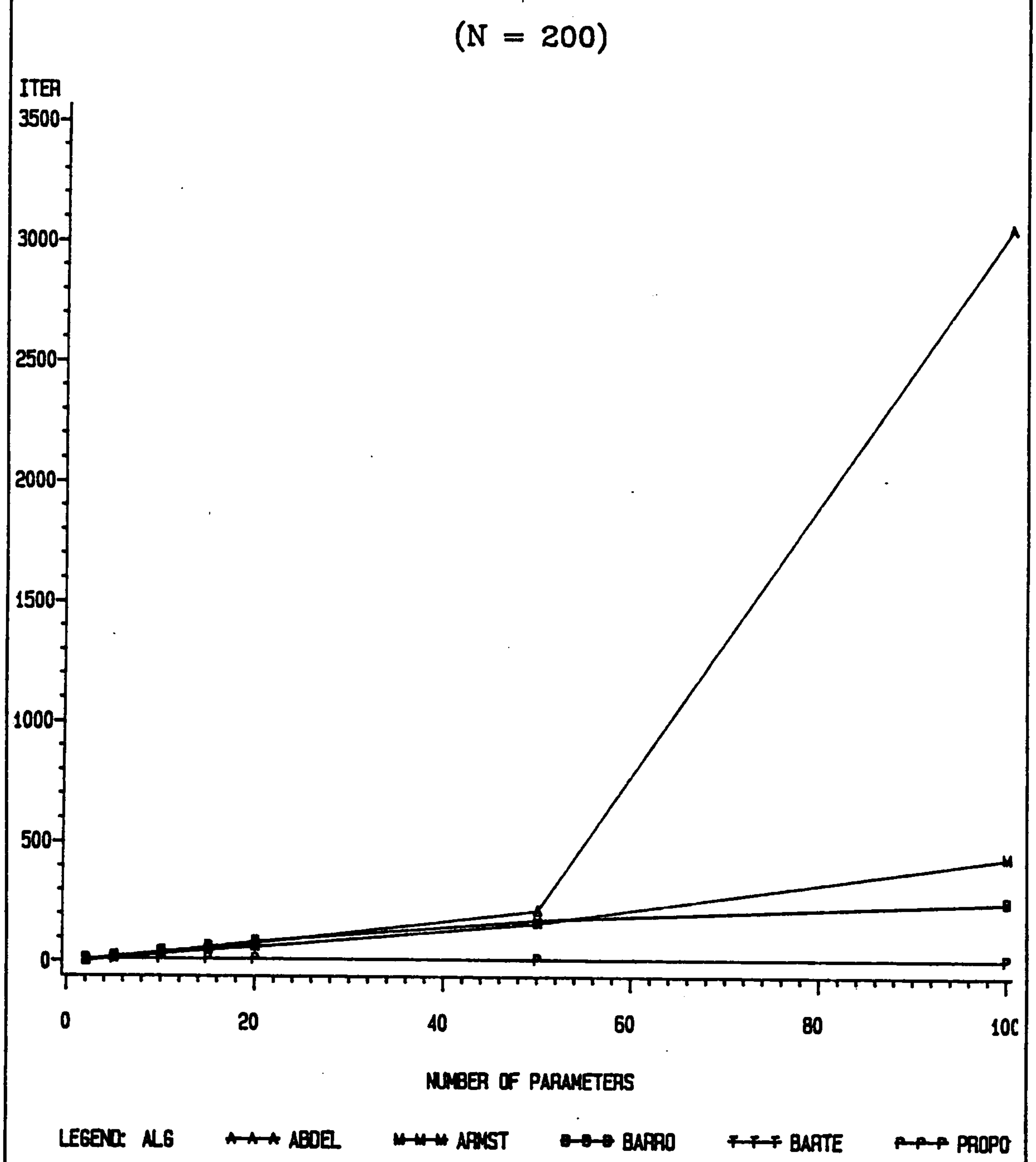
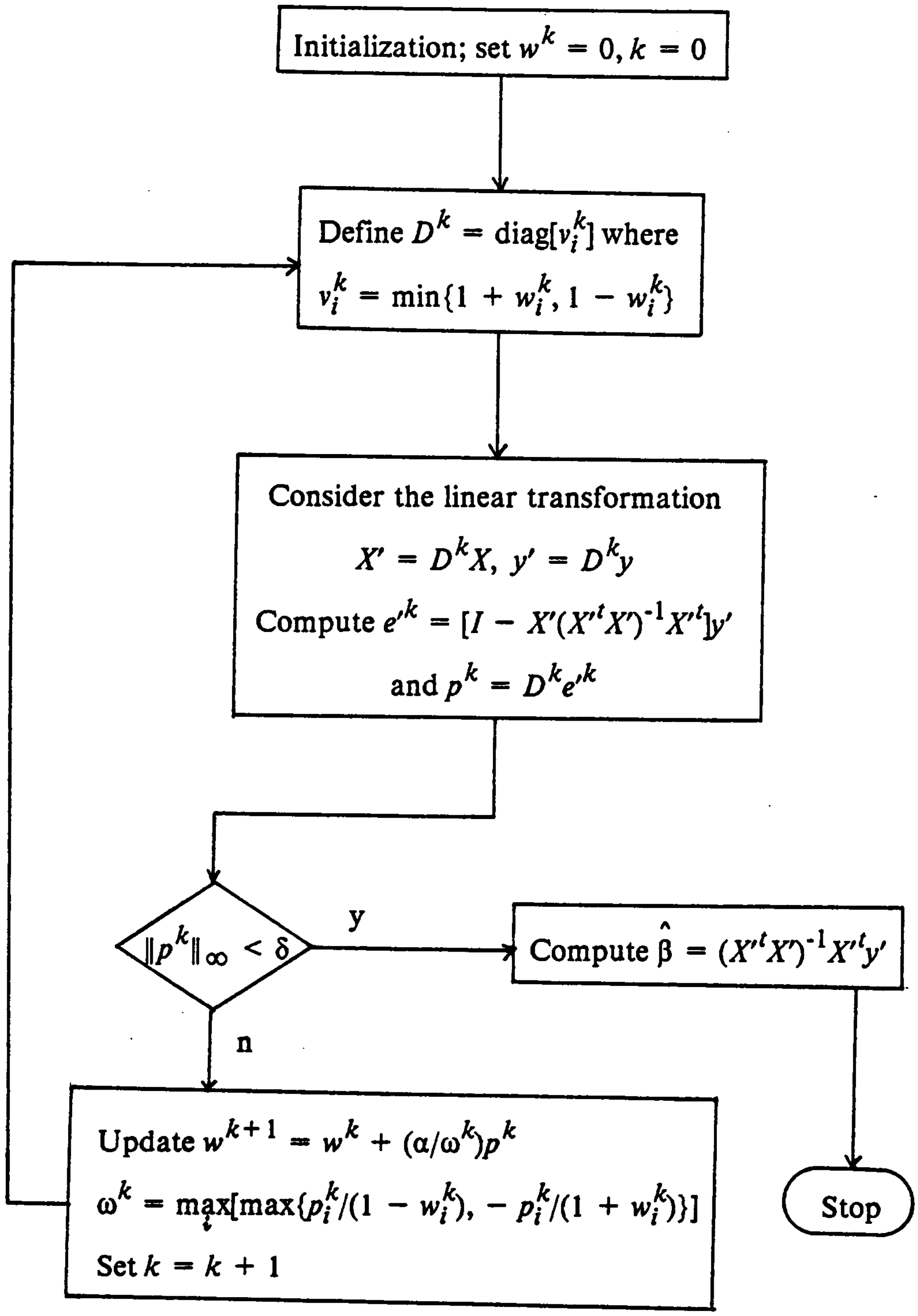


Figure 14. Variation in number of iterations with problem size (N = 200)

Appendix C. FORTRAN Program for the Proposed L1 Estimation


Initialization; set $w^k = 0, k = 0$
Define $D^k = \text{diag}[v_i^k]$ where
$v_i^k = \min\{1 + w_i^k, 1 - w_i^k\}$
Consider the linear transformation
$X' = D^k X,\ y' = D^k y$
Compute $e'^k = [I - X'(X'^t X')^{-1} X'^t] y'$
and $p^k = D^k e'^k$
$\|p^k\|_\infty < \delta$
y
Compute $\hat{\beta} = (X'^t X')^{-1} X'^t y'$
n
Update $w^{k+1} = w^k + (\alpha/\omega^k) p^k$
$\omega^k = \max_i[\max\{p_i^k/(1 - w_i^k), -p_i^k/(1 + w_i^k)\}]$
Set $k = k + 1$
Stop

```
C     **********************
C     SUBROUTINE PROJE1
C     **********************
      SUBROUTINE PROJE1(XY,X,LDX,N,M,M1,QRAUX,WORK,JOB,K,Y,DUM,B,INFO,
     *                 IR,PK,RNORM,W,W1,W2,RSD,EPS,MW)
C     ----------------------------------------------
C     A PROGRAM FOR THE L1 ESTIMATION IN A LINEAR REGRESSION MODEL USING
C     PROJECTIVE TRANSFORMATION METHOD VIA ORTHOGONAL TRIANGULAR
C     DECOMPOSITION.
C
C     +++++ ON ENTRY +++++
C     XY --- N X M1  INPUT MATRIX (X  MATRIX AUGMENTED BY  Y  VECTOR)
C     X ---- N X M MATRIX OF  X  WHICH WILL BE DESTROYED BY SUBROUTINE.
C     Y ---- M  VECTOR OF  Y  WHICH WILL BE DESTROYED BY SUBROUTINE.
C     N ---- NUMBER OF OBSERVATIONS
C     M ---- NUMBER OF PARAMETERS
C     M1 --- M + 1
C     LDX -- THE LEADING DEMENSION OF THE MATRIX  X.
C     K ---- INTEGER WHICH IS LESS THAN OR EQUAL TO  MIN(N,M).
C     EPS -- THE TOLERANCE LEVEL SMALL ENOUGH.
C     DUM -- DUMMY VARIABLE IN THE CALLING PROGRAM.
C     MIR -- MAXIMUM NUMBER OF ITERATIONS ALLOWED.
C
C     +++++ ON RETURN +++++
C     B ---- THE L1 ESTIMATES OF THE LINEAR REGRESSION MODEL.
C     RNORM - L1 NORM OF THE LINEAR REGRESSION MODEL.
C     IR ---- NUMBER OF ITERATIONS
C     INFO -- INDICATOR
C     RSD -- RESIDUAL VECTOR, SAME NAME IS USED TO SAVE STORAGE.
C
C     PROJE1  USES THE FOLLOWING SUBPROGRAM;  DQRDC  DQRSL.
C     ----------------------------------------------
      DIMENSION XY(N,M1)
      DOUBLE PRECISION X(N,M),QRAUX(N),Y(N),DUM(N),B(M),W1(N),W2(N),
     *                 W(N),WORK(M),PK(N),RSD(N),MW(N)
      INTEGER LDX,K,JOB,INFO
      DOUBLE PRECISION DZERO,OMEGA,AR,MXW
      DATA DZERO /0.D0/
C     ----------------
C     INITIALIZATION
C     ----------------
      CALL DQRDC(X,LDX,N,M,QRAUX,WORK,JOB)
      CALL DQRSL(X,LDX,N,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,10,INFO)
      OMEGA=DZERO
      DO 10 I=1,N
        AR=DABS(RSD(I))
        IF(AR .GE. OMEGA) OMEGA=AR
   10 CONTINUE
      OMEGA=0.97/OMEGA
      DO 13 I=1,N
        W(I)=OMEGA*RSD(I)
   13 CONTINUE
C     ------------------------
C     START THE 1ST ITERATION
C     ------------------------
      IR=1
   20 CONTINUE
      DO 25 I=1,N
        W2(I)=1-W(I)
        W1(I)=1+W(I)
        MW(I)=W1(I)
        IF(MW(I) .GE. W2(I)) MW(I)=W2(I)
        DO 24 J=1,M
   24     X(I,J)=MW(I)*XY(I,J)
   25   Y(I)=MW(I)*XY(I,M1)
C     ----------------------------------------------
C     COMPUTE PROJECTED GRADIENT AND OMEGA.
C     ----------------------------------------------
      CALL DQRDC(X,LDX,N,M,QRAUX,WORK,JOB)
      CALL DQRSL(X,LDX,N,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,10,INFO)
C     ------------
```

```
C    COMPUTE PK
C    ---------------
     INC=0
     DO 26 I=1,N
       PK(I)=RSD(I)*MW(I)
       AR=DABS(PK(I))
       IF (AR .GE. EPS) INC=INC+1
  26 CONTINUE
C    ------------------------------------------
C    CHECK CONVERGENCE OF ALGORITHM
C    ------------------------------------------
     IF (INC .EQ. 0) GO TO 30
C    ---------------
C    FIND OMEGA
C    ---------------
     OMEGA=DZERO
     DO 27 I=1,N
       W2(I)=PK(I)/W2(I)
       MXW=-PK(I)/W1(I)
       IF (MXW .LE. W2(I)) MXW=W2(I)
       IF (MXW .GE. OMEGA) OMEGA=MXW
  27 CONTINUE
C    -------------
C    UPDATE  W
C    -------------
     OMEGA=0.97/OMEGA
     DO 28 I=1,N
       W(I)=W(I) + OMEGA*PK(I)
  28 CONTINUE
     IF (IR .GE. MIR) GO TO 30
     IR=IR+1
     GO TO 20
  30 CONTINUE
C    ------------------------------------------
C    COMPUTE L1-ESTIMATES AND L1-NORM
C    ------------------------------------------
     RNORM=0.
     DO 32 I=1,N
       AR=DABS(RSD(I)/MW(I))
       RNORM=RNORM+AR
  32 CONTINUE
     CALL DQRSL(X,LDX,N,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,100,INFO)
     RETURN
     END
C    **********************
C    SUBROUTINE DQRSL
C    **********************
     SUBROUTINE DQRSL(X,LDX,N,K,QRAUX,Y,QY,QTY,B,RSD,XB,JOB,INFO)
C    ------------------------------------------
C    DQRSL  APPLIES THE OUTPUT OF  DQRDC  TO COMPUTE COORDINATE
C    TRANSFORMATIONS, PROJECTIONS, AND LEAST SQUARES SOLUTIONS.
C    FOR  K .LE. MIN(N,P), LET  XK  BE THE MATRIX
C
C         XK  =  (X(JPVT(1)), X(JPVT(2)),  ... , X(JPVT(K)))
C
C    FORMED FROM COLUMNNS  JPVT(1), ... ,JPVT(K)  OF THE ORIGINAL  NXP
C    MATRIX  X  THAT WAS INPUT TO  DQRDC (IF NO PIVOTING WAS DONE,
C    XK  CONSISTS OF THE FIRST  K  COLUMNS OF  X  IN THEIR ORIGINAL
C    ORDER).  DQRDC  PRODUCES A FACTORED ORTHOGONAL MATRIX  Q  AND AN
C    AN UPPER TRIANGULAR MATRIX  R  SUCH THAT
C
C          XK  =  Q * (R)
C                     (0)
C
C    THIS INFORMATION IS CONTAINED IN CODED FORM IN THE ARRAYS
C    X  AND  QRAUX.
C
C    + + + + + ON ENTRY + + + + +
C      X --- DOUBLE PRECISION(LDX,P), CONTAINS THE OUTPUT OF  DQRDC.
C      LDX -- INTEGER, IS THE LEADING DIMENSION OF THE ARRAY  X.
C      N --- INTEGER, IS THE NUMBER OF ROWS OF THE MATRIX  XK.  IT
C            MUST HAVE THE SAME VALUE AS  N  IN  DQRDC.
```

```
C       K — INTEGER, IS THE NUMBER OF COLUMNS OF THE MATRIX  XK.  K
C           MUST NOT BE GREATER THAN  MIN(N,P), WHERE  P  IS THE
C           SAME AS IN THE CALLING SEQUENCE TO  DQRDC.
C       QRAUX- DOUBLE PRECISION(P), CONTAINS THE AUXILIARY OUTPUT
C           FROM  DQRDC.
C       Y — DOUBLE PRECISION(N), CONTAINS AN N-VECTOR THAT IS TO BE
C           MANIPULATED BY  DQRSL.
C       JOB – INTEGER, SPECIFIES WHAT IS TO BE COMPUTED. JOB  HAS THE
C           DECIMAL EXPANSION  ABCDE, WITH THE FOLLOWING MEANING.
C               IF (A.NE.0), COMPUTE  QY.
C               IF (B,C,D, OR E .NE. 0), COMPUTE  QTY.
C               IF (C.NE.0), COMPUTE  B.
C               IF (D.NE.0), COMPUTE  RSD.
C               IF (E.NE.0), COMPUTE  XB.
C           NOTE THAT A REQUEST TO COMPUTE  B, RSD, OR  XB
C           AUTOMATICALLY TRIGGERS THE COMPUTATION OF QTY, FOR WHICH
C           AN ARRAY MUST BE PROVIDED IN THE CALLING SEQUENCE.
C
C     + + + + + ON RETURN + + + + +
C       QY — DOUBLE PRECISION(N), CONNTAINS  Q*Y, IF ITS COMPUTATION
C           HAS BEEN REQUESTED.
C       QTY – DOUBLE PRECISION(N), CONTAINS  TRANS(Q)*Y, IF ITS
C           COMPUTATION HAS BEEN REQUESTED.  HERE  TRANS(Q) IS THE
C           TRANSPOSE OF THE MATRIX Q.
C       B — DOUBLE PRECISION(K), CONTAINS THE SOLUTION OF THE LEAST
C           SQUARES PROBLEM
C
C               MINIMIZE NORM2(Y - XK*B),
C
C           IF ITS COMPUTATION HAS BEEN REQUESTED.  (NOTE THAT IF
C           PIVOTING WAS REQUESTED IN  DQRDC, THE J-TH COMPONENT OF
C           B  WILL BE ASSOCIATED WITH COLUMN  JPVT(J)  OF THE
C           ORIGINAL MATRX THAT WAS INPUT INTO  DQRDC.)
C       RSD – DOUBLE PRECISION(N), CONTAINS THE LEAST SQUARES RESIDUAL
C           Y - XK*B, IF ITS COMPUTATION HAS BEEN REQUESTED. RSD  IS
C           ALSO THE ORTHOGONAL PROJECTION OF  Y  ONTO THE ORTHOGONAL
C           COMPLEMENT OF THE COLUMN SPACE OF  XK.
C       XB — DOUBLE PRECISION(N), CONTAINS THE LEAST SQUARES
C           APPROXIMATION  XK*B, IF ITS COMPUTATION HAS BEEN
C           REQUESTED. XB  IS ALSO THE ORTHOGONAL PROJECTION OF  Y
C           ONTO THE COLUMN SPACE OF  X.
C      INFO – INTEGER, IS ZERO UNLESS THE COMPUTATION OF  B  HAS BEEN
C           REQUESTED AND  R  IS EXACTLY SINGULAR. IN THIS CASE,
C           INFO IS THE INDEX OF THE FIRST ZERO DIAGONAL ELEMENT OF
C           R  AND  B  IS LEFT UNALTERED.
C
C     THE PARAMETERS  QY, QTY, B, RSD, AND  XB  ARE NOT REFERENCED IF
C     THEIR COMPUTATION IS NOT REQUESTED AND IN THIS CASE CAN BE
C     REPLACED BY DUMMY VARIABLES IN THE CALLING PROGRAM. TO SAVE
C     STORAGE, THE USER MAY IN SOME CASES USE THE SAME ARRAY FOR
C     DIFFERENT PARAMETERS IN THE CALLING SEQUENCE. A FREQUENTLY OCCURING
C     EXAMPLE IS WHEN ONE WISHES TO COMPUTE ANY OF  B, RSD, OR XB  AND
C     DOES NOT NEED  Y  OR  QTY. IN THIS CASE ONE MAY IDENTIFY  Y, QTY,
C     AND ONE OF  B, RSD, OR  XB, WHILE PROVIDING SEPARATE ARRAYS FOR
C     ANYTHING ELSE THAT IS TO BE COMPUTED.  THUS THE CALLING SEQUENCE
C
C        CALL DQRSL(X,LDX,N,K,QRAUX,Y,DUM,Y,B,Y,DUM,110,INFO)
C
C     WILL RESULT IN THE COMPUTATION OF  B  AND  RSD, WITH  RSD
C     OVERWRITING  Y. MORE GENERALLY, EACH ITEM IN THE FOLLOWING LIST
C     CONTAINS GROUPS OF PERMISSIBLE IDENTIFICATIONS FOR A SINGLE
C     CALLINNG SEQUENCE.
C        1. (Y,QTY,B) (RSD) (XB) (QY)
C        2. (Y,QTY,RSD) (B) (XB) (QY)
C        3. (Y,QTY,XB) (B) (RSD) (QY)
C        4. (Y,QY) (QTY,B) (RSD) (XB)
C        5. (Y,QY) (QTY,RSD) (B) (XB)
C        6. (Y,QY) (QTY,XB) (B) (RSD)
C     IN ANY GROUP THE VALUE RETURNED IN THE ARRAY ALLOCATED TO THE GROUP
C     CORRESPONDS TO THE LAST MEMBER OF THE GROUP.
C
C     DQRSL USES THE FOLLOWING FUNCTIONS AND SUBPROGRAMS.
```

```
C
C    BLAS DAXPY,DCOPY,DDOT
C    FORTRAN DABS,MIN0,MOD
C    ------------------------------------------------
      INTEGER LDX,N,K,JOB,INFO
      DOUBLE PRECISION X(LDX,1),QRAUX(1),Y(1),QY(1),QTY(1),B(1),RSD(1),
     *                 XB(1)
      INTEGER I,J,JJ,JU,KP1
      DOUBLE PRECISION DDOT,T,TEMP
      LOGICAL CB,CQY,CQTY,CR,CXB
C    ---------------
C    SET INFO FLAG
C    ---------------
      INFO = 0
C    ------------------------------------------
C    DETERMINE WHAT IS TO BE COMPUTED
C    ------------------------------------------
      CQY = JOB/10000 .NE. 0
      CQTY = MOD(JOB,10000) .NE. 0
      CB = MOD(JOB,1000)/100 .NE. 0
      CR = MOD(JOB,100)/10 .NE. 0
      CXB = MOD(JOB,10) .NE. 0
      JU = MIN0(K,N-1)
C    ---------------------------
C    SPECIAL ACTION WHEN N=1
C    ---------------------------
      IF (JU .NE. 0) GO TO 40
      IF (CQY) QY(1) = Y(1)
      IF (CQTY) QTY(1) = Y(1)
      IF (CXB) XB(1) = Y(1)
      IF (.NOT.CB) GO TO 30
      IF (X(1,1) .NE. 0.0D0) GO TO 10
      INFO = 1
      GO TO 20
   10 CONTINUE
      B(1) = Y(1)/X(1,1)
   20 CONTINUE
   30 CONTINUE
      IF (CR) RSD(1) = 0.0D0
      GO TO 250
   40 CONTINUE
C    ---------------------------------
C    SET UP TO COMPUTE QY OR QTY
C    ---------------------------------
      IF (CQY) CALL DCOPY(N,Y,1,QY,1)
      IF (CQTY) CALL DCOPY(N,Y,1,QTY,1)
      IF (.NOT.CQY) GO TO 70
C    --------------
C    COMPUTE QY
C    --------------
      DO 60 JJ = 1, JU
        J = JU - JJ + 1
        IF (QRAUX(J) .EQ. 0.0D0) GO TO 50
        TEMP = X(J,J)
        X(J,J) = QRAUX(J)
        T = -DDOT(N-J+1,X(J,J),1,QY(J),1)/X(J,J)
        CALL DAXPY(N-J+1,T,X(J,J),1,QY(J),1)
        X(J,J) = TEMP
   50   CONTINUE
   60 CONTINUE
   70 CONTINUE
      IF (.NOT.CQTY) GO TO 100
C    ----------------------
C    COMPUTE TRANS(Q)*Y
C    ----------------------
      DO 90 J = 1, JU
        IF (QRAUX(J) .EQ. 0.0D0) GO TO 80
        TEMP = X(J,J)
        X(J,J) = QRAUX(J)
        T = -DDOT(N-J+1,X(J,J),1,QTY(J),1)/X(J,J)
        CALL DAXPY(N-J+1,T,X(J,J),1,QTY(J),1)
        X(J,J) = TEMP
```

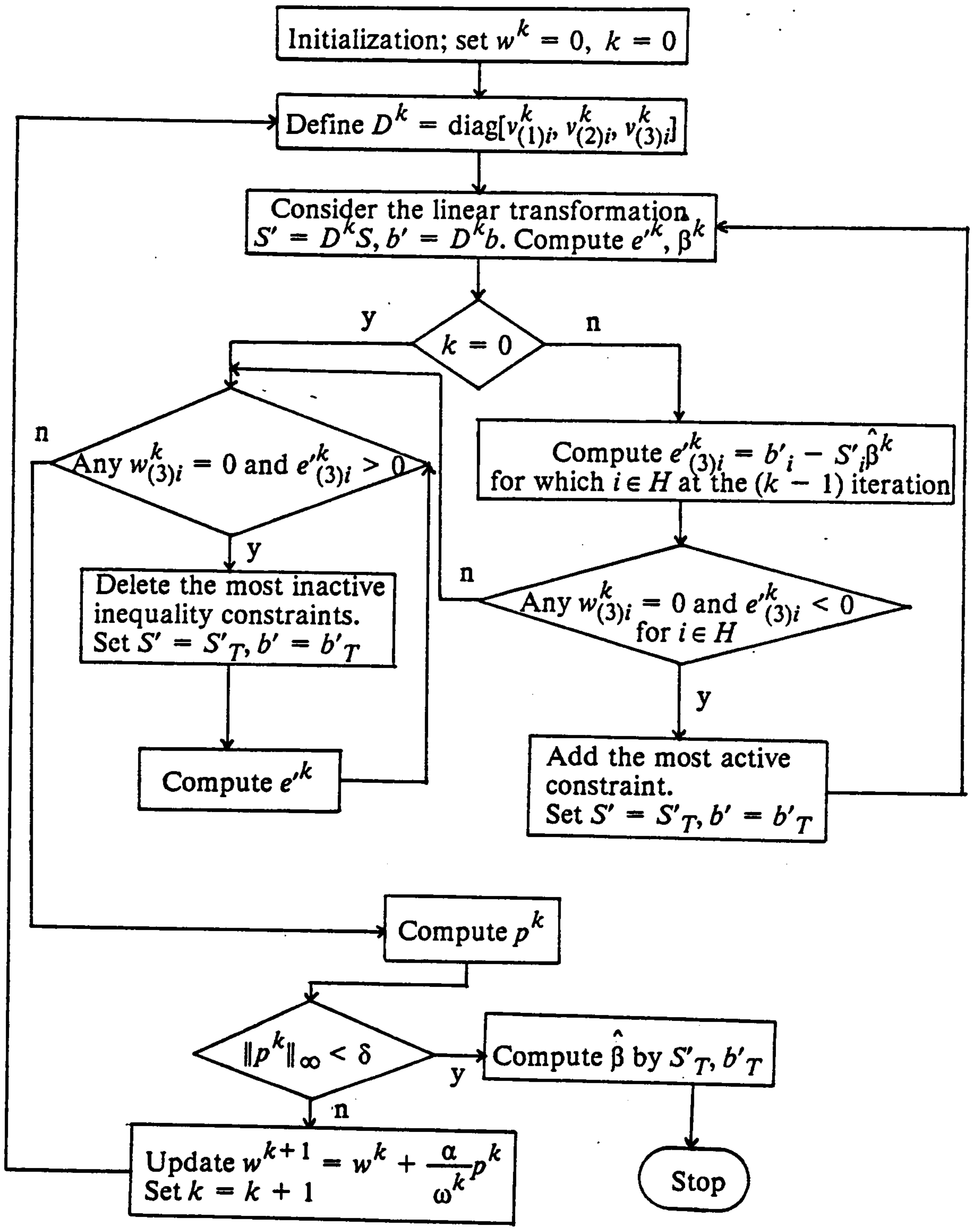
```
  80  CONTINUE
  90 CONTINUE
 100 CONTINUE
C    -------------------------------------------
C    SET UP TO COMPUTE B, RSD, OR XB
C    -------------------------------------------
     IF (CB) CALL DCOPY(K,QTY,1,B,1)
     KP1 = K + 1
     IF (CXB) CALL DCOPY(K,QTY,1,XB,1)
     IF (CR .AND. K .LT. N) CALL DCOPY(N-K,QTY(KP1),1,RSD(KP1),1)
     IF (.NOT.CXB .OR. KP1 .GT. N) GO TO 120
     DO 110 I = KP1, N
       XB(I) = 0.0D0
 110 CONTINUE
 120 CONTINUE
     IF (.NOT.CR) GO TO 140
     DO 130 I = 1, K
       RSD(I) = 0.0D0
 130 CONTINUE
 140 CONTINUE
     IF (.NOT.CB) GO TO 190
C    ---------------
C    COMPUTE B
C    ---------------
     DO 170 JJ = 1, K
       J = K - JJ + 1
       IF (X(J,J) .NE. 0.0D0) GO TO 150
       INFO = J
C      -------
C      EXIT
C      -------
       GO TO 180
 150   CONTINUE
       B(J) = B(J)/X(J,J)
       IF (J .EQ. 1) GO TO 160
       T = -B(J)
       CALL DAXPY(J-1,T,X(1,J),1,B,1)
 160   CONTINUE
 170 CONTINUE
 180 CONTINUE
 190 CONTINUE
     IF (.NOT.CR .AND. .NOT.CXB) GO TO 240
C    -------------------------------------------
C    COMPUTE RSD OR  XB  AS REQUIRED
C    -------------------------------------------
     DO 230 JJ = 1, JU
       J = JU - JJ + 1
       IF (QRAUX(J) .EQ. 0.0D0) GO TO 220
       TEMP = X(J,J)
       X(J,J) = QRAUX(J)
       IF (.NOT.CR) GO TO 200
       T = -DDOT(N-J+1,X(J,J),1,RSD(J),1)/X(J,J)
       CALL DAXPY(N-J+1,T,X(J,J),1,RSD(J),1)
 200   CONTINUE
       IF (.NOT.CXB) GO TO 210
       T = -DDOT(N-J+1,X(J,J),1,XB(J),1)/X(J,J)
       CALL DAXPY(N-J+1,T,X(J,J),1,XB(J),1)
 210   CONTINUE
       X(J,J) = TEMP
 220   CONTINUE
 230 CONTINUE
 240 CONTINUE
 250 CONTINUE
     RETURN
     END
C    **********************
C    SUBROUTINE DQRDC
C    **********************
     SUBROUTINE DQRDC(X,LDX,N,P,QRAUX,WORK,JOB)
C    -------------------------------------------------
C    DQRDC  USES HOUSEHOLDER TRANSFORMATIONS TO COMPUTE THE  QR
C    FACTORIZATION OF AN NXP MATRIX X. COLUMN PIVOTING BASED ON THE L2-
```

```
C     NORMS OF THE REDUCED COLUMNS MAY BE PERFORMED AT THE USERS OPTION.
C
C     +++++ ON ENTRY +++++
C       X ---- DOUBLE PRECISION(LDX,P), WHERE LDX.GE.N, CONTAINS THE
C              MATRIX WHOSE DECOMPOSITION IS TO BE COMPUTED.
C       LDX -- INTEGER, IS THE LEADING DIMENSION OF THE ARRAY X.
C       N ---- INTEGER, IS THE NUMBER OF ROWS OF THE MATRIX X.
C       P ---- INTEGER, IS THE NUMBER OF COLUMNS OF THE MATRIX X.
C       JPVT - INTEGER(P), CONTAINS INTEGERS THAT CONTROL THE SELECTION
C              OF THE PIVOT COLUMNS. THE K-TH COLUMN X(K) OF X IS
C              PLACED IN ONE OF THREE CLASSES ACCORDING TO THE VALUE
C              OF JPVT(K).
C                IF (JPVT(K) .GT. 0), THEN X(K) IS AN INITIAL COLUMN.
C                IF (JPVT(K) .EQ. 0), THEN X(K) IS A FREE COLUMN.
C                IF (JPVT(K) .LT. 0), THEN X(K) IS A FINAL COLUMN.
C              BEFORE THE DECOMPOSITION IS COMPUTED, INITIAL COLUMNS
C              ARE MOVED TO THE BEGINNING OF THE ARRAY X AND FINAL
C              COLUMNS TO THE END. BOTH INITIAL AND FINAL COLUMNS ARE
C              FROZEN IN PLACE DURING THE COMPUTATION AND ONLY FREE
C              COLUMNS ARE MOVED. AT THE K-TH STAGE OF THE REDUCTION,
C              IF X(K) IS OCCUPIED BY A FREE COLUMN IT IS INTERCHANGED
C              WITH THE FREE COLUMN OF LARGEST REDUCED NORM. JPVT IS
C              NOT REFERENCED IF (JOB .EQ. 0).
C       WORK - DOUBLE PRECISION(P), IS A WORK ARRAY. WORK IS NOT
C              REFERENCED IF (JOB .EQ. 0).
C       JOB -- INTEGER, IS AN INTEGER THAT INITIATES COLUMN PIVOTING.
C              IF (JOB .EQ. 0), NO PIVOTING IS DONE.
C              IF (JOB .NE. 0), PIVOTING IS DONE.
C
C     +++++ ON RETURN +++++
C       X ---- CONTAINS IN ITS UPPER TRIANGLE THE UPPER TRIANGULAR
C              MATRIX R OF THE QR FACTORIZATION. BELOW ITS DIAGONAL
C              X CONTAINS INFORMATION FROM WHICH THE ORTHOGONAL PART
C              OF THE DECOMPOSITION CAN BE RECOVERED. NOTE THAT IF
C              PIVOTING HAS BEEN REQUESTED, THE DECOMPOSITION IS NOT
C              THAT OF THE ORIGINAL MATRIX X BUT THAT OF X WITH ITS
C              COLUMNS PERMUTED AS DESCRIBED BY JPVT.
C       QRAUX - DOUBLE PRECISION(P), CONTAINS FURTHER INFORMATION REQUI-
C              RED TO RECOVER THE ORTHOGONAL PART OF THE DECOMPOSITION.
C       JPVT - JPVT(K) CONTAINS THE INDEX OF THE COLUMN OF THE
C              ORIGINAL MATRIX THAT HAS BEEN INTERCHANGED INTO
C              THE K-TH COLUMN, IF PIVOTING WAS REQUESTED.
C
C     DQRDC USES THE FOLLOWING FUNCTIONS AND SUBPROGRAMS.
C
C     BLAS DAXPY,DDOT,DSCAL,DSWAP,DNRM2
C     FORTRAN DABS,DMAX1,MIN0,DSQRT
C     ----------------------------------------------
      INTEGER LDX,N,P,JOB
      INTEGER JPVT(1)
      DOUBLE PRECISION X(LDX,1),QRAUX(1),WORK(1)
      INTEGER J,JP,L,LP1,LUP,MAXJ,PL,PU
      DOUBLE PRECISION MAXNRM,DNRM2,TT
      DOUBLE PRECISION DDOT,NRMXL,T
      LOGICAL NEGJ,SWAPJ
      PL = 1
      PU = 0
      IF (JOB .EQ. 0) GO TO 60
C     ----------------------------------------------
C     PIVOTING HAS BEEN REQUESTED.  REARRANGE
C     THE COLUMNS ACCORDING TO JPVT.
C     ----------------------------------------------
      DO 20 J = 1, P
        SWAPJ = JPVT(J) .GT. 0
        NEGJ = JPVT(J) .LT. 0
        JPVT(J) = J
        IF (NEGJ) JPVT(J) = -J
        IF (.NOT.SWAPJ) GO TO 10
        IF (J .NE. PL) CALL DSWAP(N,X(1,PL),1,X(1,J),1)
        JPVT(J) = JPVT(PL)
        JPVT(PL) = J
        PL = PL + 1
```

```
   10  CONTINUE
   20 CONTINUE
      PU = P
      DO 50 JJ = 1, P
       J = P - JJ + 1
       IF (JPVT(J) .GE. 0) GO TO 40
       JPVT(J) = -JPVT(J)
       IF (J .EQ. PU) GO TO 30
       CALL DSWAP(N,X(1,PU),1,X(1,J),1)
       JP = JPVT(PU)
       JPVT(PU) = JPVT(J)
       JPVT(J) = JP
   30  CONTINUE
       PU = PU - 1
   40  CONTINUE
   50 CONTINUE
   60 CONTINUE
C   ---------------------------------------------------
C    COMPUTE THE NORMS OF THE FREE COLUMNS
C   ---------------------------------------------------
      IF (PU .LT. PL) GO TO 80
      DO 70 J = PL, PU
        QRAUX(J) = DNRM2(N,X(1,J),1)
        WORK(J) = QRAUX(J)
   70 CONTINUE
   80 CONTINUE
C   ---------------------------------------------------
C    PERFORM THE HOUSEHOLDER REDUCTION OF X
C   ---------------------------------------------------
      LUP = MIN0(N,P)
      DO 200 L = 1, LUP
        IF (L .LT. PL .OR. L .GE. PU) GO TO 120
C   ---------------------------------------------------
C    LOCATE THE COLUMN OF LARGEST NORM AND
C    BRING IT INTO THE PIVOT POSITION.
C   ---------------------------------------------------
      MAXNRM = 0.0D0
      MAXJ = L
      DO 100 J = L, PU
        IF (QRAUX(J) .LE. MAXNRM) GO TO 90
        MAXNRM = QRAUX(J)
        MAXJ = J
   90  CONTINUE
  100 CONTINUE
      IF (MAXJ .EQ. L) GO TO 110
      CALL DSWAP(N,X(1,L),1,X(1,MAXJ),1)
      QRAUX(MAXJ) = QRAUX(L)
      WORK(MAXJ) = WORK(L)
      JP = JPVT(MAXJ)
      JPVT(MAXJ) = JPVT(L)
      JPVT(L) = JP
  110 CONTINUE
  120 CONTINUE
      QRAUX(L) = 0.0D0
      IF (L .EQ. N) GO TO 190
C   ---------------------------------------------------
C    COMPUTE THE HOUSEHOLDER TRANSFORMATION FOR COLUMN L
C   ---------------------------------------------------
      NRMXL = DNRM2(N-L+1,X(L,L),1)
      IF (NRMXL .EQ. 0.0D0) GO TO 180
      IF (X(L,L) .NE. 0.0D0) NRMXL = DSIGN(NRMXL,X(L,L))
      CALL DSCAL(N-L+1,1.0D0/NRMXL,X(L,L),1)
      X(L,L) = 1.0D0 + X(L,L)
C   ---------------------------------------------------
C    APPLY THE TRANSFORMATION TO THE
C    REMAINING COLUMNS, UPDATING THE NORMS.
C   ---------------------------------------------------
      LP1 = L + 1
      IF (P .LT. LP1) GO TO 170
      DO 160 J = LP1, P
        T = -DDOT(N-L+1,X(L,L),1,X(L,J),1)/X(L,L)
        CALL DAXPY(N-L+1,T,X(L,L),1,X(L,J),1)
```

```
         IF (J .LT. PL .OR. J .GT. PU) GO TO 150
         IF (QRAUX(J) .EQ. 0.0D0) GO TO 150
         TT = 1.0D0 - (DABS(X(L,J))/QRAUX(J))**2
         TT = DMAX1(TT,0.0D0)
         T = TT
         TT = 1.0D0 + 0.05D0*TT*(QRAUX(J)/WORK(J))**2
         IF (TT .EQ. 1.0D0) GO TO 130
         QRAUX(J) = QRAUX(J)*DSQRT(T)
         GO TO 140
  130    CONTINUE
         QRAUX(J) = DNRM2(N-L,X(L+1,J),1)
         WORK(J) = QRAUX(J)
  140    CONTINUE
  150    CONTINUE
  160 CONTINUE
  170 CONTINUE
C     ------------------------------------
C     SAVE THE TRANSFORMATION.
C     ------------------------------------
      QRAUX(L) = X(L,L)
      X(L,L) = -NRMXL
  180 CONTINUE
  190 CONTINUE
  200 CONTINUE
      RETURN
      END
```

Appendix D. FORTRAN Program for the Proposed Constrained L1 Estimation


Initialization; set $w^k = 0,\ k = 0$
Define $D^k = \mathrm{diag}[v^k_{(1)i}, v^k_{(2)i}, v^k_{(3)i}]$
Consider the linear transformation $S' = D^kS, b' = D^kb$. Compute e'^k, β^k
$k = 0$
y
n
Any $w^k_{(3)i} = 0$ and $e'^k_{(3)i} > 0$
n
y
Delete the most inactive inequality constraints. Set $S' = S'_T, b' = b'_T$
Compute e'^k
Compute $e'^k_{(3)i} = b'_i - S'_i\hat{\beta}^k$ for which $i \in H$ at the $(k-1)$ iteration
Any $w^k_{(3)i} = 0$ and $e'^k_{(3)i} < 0$ for $i \in H$
n
y
Add the most active constraint. Set $S' = S'_T, b' = b'_T$
Compute p^k
$\|p^k\|_\infty < \delta$
y
n
Compute $\hat{\beta}$ by S'_T, b'_T
Update $w^{k+1} = w^k + \frac{\alpha}{\omega^k}p^k$ Set $k = k + 1$
Stop

```
C     ************************
C     SUBROUTINE PROJEC
C     ************************
      SUBROUTINE PROJEC(XY,X,LDX,N,NE,NI,NEI,M,M1,K,QRAUX,WORK,JOB,
     *     Y,DUM,B,INFO,IR,MIR,RNORM,EPS,BIGM,W,W1,W2,RSD,PK,AR,MW)
C     ------------------------------------------------
C     A PROGRAM FOR THE L1 ESTIMATION WITH EQUALITY AND INEQUALITY
C     CONSTRAINTS IN A LINEAR REGRESSION MODEL
C     USING PROJECTIVE TRANSFORMATION METHOD VIA ORTHOGONAL TRIANGULAR
C     DECOMPOSITION.
C
C     + + + + + ON ENTRY + + + + +
C     XY — N X M1 INPUT MATRIX (X  MATRIX AUGMENTED BY  Y  VECTOR)
C     X —– N X M MATRIX OF  X  WHICH WILL BE DESTROYED BY SUBROUTINE.
C     Y —– M VECTOR OF  Y  WHICH WILL BE DESTROYED BY SUBROUTINE.
C     N —– NUMBER OF OBSERVATIONS
C     NE — NUMBER OF EQUALITY CONSTRAINTS
C     NI — NUMBER OF INEQUALITY CONSTRAINTS
C     M —– NUMBER OF PARAMETERS
C     M1 — M + 1
C     LDX – THE LEADING DEMENSION OF THE MATRIX  X.
C     K —– INTEGER WHICH IS LESS THAN OR EQUAL TO  MIN(N,M).
C     EPS – THE TOLERANCE LEVEL SMALL ENOUGH.
C     NEI — N + NE + NI
C     DUM – DUMMY VARIABLE IN THE CALLING PROGRAM.
C     MIR – MAXIMUM NUMBER OF ITERATIONS ALLOWED.
C
C     + + + + + ON RETURN + + + + +
C     B —– THE ESTIMATES OF THE LINEAR REGRESSION MODEL.
C     RNORM - L1 NORM OF THE LINEAR REGRESSION MODEL.
C     IR — NUMBER OF ITERATIONS
C     INFO – INDICATOR
C     RSD — RESIDUAL VECTOR, SAME NAME IS USED TO SAVE STORAGE.
C
C     PROJEC  USES SUBROUTINE  DQRDC  AND  DQRSL.
C     ------------------------------------------------
      DIMENSION XY(NEI,M1)
      DOUBLE PRECISION X(NEI,M),QRAUX(NEI),Y(NEI),DUM(NEI),B(M),W1(NEI),
     *        W2(NEI),W(NEI),WORK(M),PK(NEI),RSD(NEI),MW(NEI),AR(NEI)
      INTEGER LDX,K,JOB,INFO
      DOUBLE PRECISION DZERO,OMEGA1,OMEGA2,RNORM,MXW,BIGM
      DATA DZERO /0.D0/
      N1 = N + 1
      NL = N + NE
      NL1 = NL + 1
C     ----------------
C     INITIALIZATION
C     ----------------
      DO 10 I=NL1,NEI
        MW(I)=1.0
   10 CONTINUE
C     ------------------------------------------
C     COMPUTE RSD(I) FOR ALL CONSTRAINTS
C     ------------------------------------------
      CALL DQRDC(X,LDX,NEI,M,QRAUX,WORK,JOB)
      CALL DQRSL(X,LDX,NEI,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,10,INFO)
   11 CONTINUE
C     ------------------------------------------
C     FIND THE MOST INACTIVE INEQUALITY CONSTRAINT
C     ------------------------------------------
      IMAX=NL1
      DO 12 I=NL1,NEI
        IF (RSD(I) .GT. RSD(IMAX)) IMAX=I
   12 CONTINUE
      IF (RSD(IMAX) .LE. 0.) GO TO 25
C     ------------------------------------------
C     DELETE THE MOST INACTIVE INEQUALITY CONSTRAINT
C     ------------------------------------------
      MW(IMAX)=DZERO
C     ------------------------------
C     GET LINEAR TRANSFORMATION
C     ------------------------------
```

```
      DO 14 I=1,NL
        DO 13 J=1,M
   13     X(I,J)=XY(I,J)
        Y(I)=XY(I,M1)
   14 CONTINUE
      DO 17 I=NL1,NEI
        DO 16 J=1,M
   16     X(I,J)=MW(I)*XY(I,J)
        Y(I)=MW(I)*XY(I,M1)
   17 CONTINUE
C     ------------------------------------------------
C     COMPUTE RSD(I) FOR REMAINING CONSTRAINTS
C     ------------------------------------------------
      CALL DQRDC(X,LDX,NEI,M,QRAUX,WORK,JOB)
      CALL DQRSL(X,LDX,NEI,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,10,INFO)
      GO TO 11
   25 CONTINUE
C     ---------------
C     FIND OMEGA
C     ---------------
      DO 26 I=1,NEI
        AR(I)=DABS(RSD(I))
   26 CONTINUE
      DO 27 I=N1,NEI
        AR(I)=AR(I)/BIGM
   27 CONTINUE
      OMEGA1=DZERO
      DO 29 I=1,NEI
        IF (AR(I) .GE. OMEGA1) OMEGA1=AR(I)
   29 CONTINUE
C     -------------
C     UPDATE W
C     -------------
      OMEGA1=0.97/OMEGA1
      DO 31 I=1,NEI
        W(I)=OMEGA1*RSD(I)
   31 CONTINUE
C     ---------------------------
C     START THE 1ST ITERATION
C     ---------------------------
      IR=1.
   35 CONTINUE
C     ---------------------------
C     OBTAIN ELEMENTS OF D
C     ---------------------------
      DO 36 I=1,N
        W2(I)=1-W(I)
        W1(I)=1+W(I)
        MW(I)=W1(I)
        IF(MW(I) .GE. W2(I)) MW(I)=W2(I)
   36 CONTINUE
      DO 37 I=N1,NL
        W2(I)=BIGM-W(I)
        W1(I)=BIGM+W(I)
        MW(I)=W1(I)
        IF (MW(I) .GE. W2(I)) MW(I)=W2(I)
        MW(I)=MW(I)/BIGM
   37 CONTINUE
      DO 38 I=NL1,NEI
        W1(I)=BIGM+W(I)
        IF (IR .EQ. 1) W2(I)=DZERO
        MW(I)=W1(I)/BIGM
        IF (W(I) .EQ. DZERO) MW(I)=DZERO
   38 CONTINUE
C     ----------------------------------
C     GET LINEAR TRANSFORMATION
C     ----------------------------------
      DO 40 I=1,NEI
        DO 39 J=1,M
   39     X(I,J)=MW(I)*XY(I,J)
        Y(I)=MW(I)*XY(I,M1)
   40 CONTINUE
```

```
      DO 44 I=NL1,NEI
       IF (W2(I) .EQ. BIGM) GO TO 41
       GO TO 44
 41    DO 43 J=1,M
 43      X(I,J)=DZERO
       Y(I)=DZERO
 44   CONTINUE
C     --------------------------------------------
C     COMPUTE CURRENT ESTIMATE
C     AND RSD(I) FOR REMAINING CONSTRAINTS
C     --------------------------------------------
      CALL DQRDC(X,LDX,NEI,M,QRAUX,WORK,JOB)
      CALL DQRSL(X,LDX,NEI,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,110,INFO)
C     --------------------------------------------
C     COMPUTE RSD(I) FOR INEQUALITY CONSTRAINTS
C     DELETED AT THE PREVIOUS ITERATION
C     --------------------------------------------
      DO 46 I=NL1,NEI
       Y(I)=XY(I,M1)
       DO 45 J=1,M
 45      Y(I)=Y(I)-XY(I,J)*B(J)
 46   CONTINUE
C     --------------------------------------------
C     ADD THE ACTIVE INEQUALITY CONSTRAINTS
C     --------------------------------------------
      INC=0
      DO 48 I=NL1,NEI
       IF(MW(I) .EQ. DZERO .AND. Y(I) .LT. DZERO) GO TO 47
       GO TO 48
 47   MW(I)=1.0
      INC=INC+1
 48   CONTINUE
      IF (INC .EQ. 0) GO TO 51
C     ------------------------------
C     GET LINEAR TRANSFORMATION
C     ------------------------------
      DO 50 I=1,NEI
       DO 49 J=1,M
 49      X(I,J)=MW(I)*XY(I,J)
       Y(I)=MW(I)*XY(I,M1)
 50   CONTINUE
C     --------------------------------------------
C     COMPUTE CURRENT ESTIMATES AND RSD(I)
C     --------------------------------------------
      CALL DQRDC(X,LDX,NEI,M,QRAUX,WORK,JOB)
      CALL DQRSL(X,LDX,NEI,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,110,INFO)
 51   CONTINUE
C     --------------------------------------------
C     FIND THE MOST INACTIVE INEQUALITY CONSTRAINT
C     --------------------------------------------
      DO 52 I=NL1,NEI
       Y(I)=RSD(I)
 52   CONTINUE
 53   CONTINUE
      IMAX=NL1
      DO 54 I=NL1,NEI
       IF (Y(I) .GT. Y(IMAX)) IMAX=I
 54   CONTINUE
      IF (Y(IMAX) .LE. 0.) GO TO 61
C     --------------------------------------------
C     DELETE THE MOST INACTIVE INEQUALITY CONSTRAINT
C     --------------------------------------------
      IF ((W(IMAX) .EQ. DZERO) .OR. (W2(IMAX) .EQ. BIGM)) GO TO 55
      Y(IMAX)=0.
      GO TO 53
 55   MW(IMAX)=DZERO
C     ------------------------------
C     GET LINEAR TRANSFORMATION
C     ------------------------------
      DO 57 I=1,NEI
       DO 56 J=1,M
 56      X(I,J)=MW(I)*XY(I,J)
```

```
        Y(I)=MW(I)*XY(I,M1)
   57 CONTINUE
      DO 60 I=NL1,NEI
        IF (W2(I) .EQ. BIGM) GO TO 58
        GO TO 60
   58   DO 59 J=1,M
   59     X(I,J)=DZERO
        Y(I)=DZERO
   60 CONTINUE
C     ------------------------------------------
C      COMPUTE CURRENT ESTIMATES AND RSD(I)
C     ------------------------------------------
      CALL DQRDC(X,LDX,NEI,M,QRAUX,WORK,JOB)
      CALL DQRSL(X,LDX,NEI,K,QRAUX,Y,DUM,RSD,B,RSD,DUM,110,INFO)
      GO TO 51
   61 CONTINUE
      OMEGA1=DZERO
      OMEGA2=DZERO
C     ----------------
C      COMPUTE PK(I)
C     ----------------
      DO 62 I=1,NEI
        PK(I)=RSD(I)*MW(I)
   62 CONTINUE
C     --------------
C      FIND OMEGA
C     --------------
      DO 63 I=1,NL
        W2(I)=PK(I)/W2(I)
        MXW=-PK(I)/W1(I)
        IF (MXW .LE. W2(I)) MXW=W2(I)
        IF (MXW .GE. OMEGA1) OMEGA1=MXW
   63 CONTINUE
      DO 66 I=NL1,NEI
        IF (RSD(I)) 64,64,65
   64   W2(I)=-PK(I)/W1(I)
        MXW=W2(I)
        IF (MXW .GE. OMEGA1) OMEGA1=MXW
        GO TO 66
   65   IF (W(I).EQ.DZERO) GO TO 66
        W2(I)=-PK(I)/W(I)
        MXW=W2(I)
        IF (MXW .GE. OMEGA2) OMEGA2=MXW
   66 CONTINUE
      IF (OMEGA1 .LT. OMEGA2) OMEGA1=OMEGA2
C     -------------
C      UPDATE  W
C     -------------
      OMEGA1=0.97/OMEGA1
      DO 67 I=1,NEI
        W(I)=W(I) + OMEGA1*PK(I)
   67 CONTINUE
      OMEGA1=0.97/OMEGA1
      DO 68 I=NL1,NEI
        IF(RSD(I) .LE. DZERO) GO TO 68
        IF (W2(I) .EQ. OMEGA1) W2(I)=BIGM
   68 CONTINUE
C     ------------------------------------------
C      CHECK THE CONVERGENCE OF ALGORITHM
C     ------------------------------------------
      INC=0
      DO 70 I=1,NEI
        PK(I)=DABS(PK(I))
        IF (PK(I) .GE. EPS) INC=INC+1
   70 CONTINUE
      IF (INC .EQ. 0) GO TO 90
      IR=IR+1
      IF (IR .GT. MIR) GO TO 90
      GO TO 35
   90 CONTINUE
C     ----------------------
```

```
C     COMPUTE L1 NORM
C     ---------------------------
      RNORM = DZERO
      DO 91 I = 1,N
        AR(I) = DABS(RSD(I)/MW(I))
        RNORM = RNORM + AR(I)
   91 CONTINUE
      RETURN
      END
```

BIBLIOGRAPHY

Abdelmalek, N. N. (1980), "L_1 Solution of Overdetermined Systems of Linear Equations", *ACM Transactions on Mathematical Software*, 6, 220-227.

Armstrong, R. D., Frome, E. L. and Kung, D. S. (1979), "A Revised Simplex Algorithm for the Absolute Deviation Curve Fitting Problem", *Commun. Statist.- Simula. Computa.*, B(8),175-190.

Armstrong, R. D. and Hultz, J. W. (1977), "An Algorithm for a Restricted Discrete Approximation Problem in the L_1 Norm", *SIAM J. Numer. Anal.*, 14, 555-565.

Ashar, V. G. and Wallace, T. D. (1963), "A Sampling Study of Minimum Absolute Deviations Estimators", *Operations Research*, 11, 747-758.

Barrodale, I. and Roberts, F. D. K. (1973), "An Improved Algorithm for Discrete L_1 Linear Approximation" *SIAM J. Numer. Anal.*, 10, 839-848.

Barrodale, I. and Roberts, F. D. K. (1977), "Algorithms for Restricted Least Absolute Value Estimation", *Commun. Statist. - Simula. Computa.*, B(6), 353-363.

Barrodale, I. and Roberts, F. D. K. (1978), "An Efficient Algorithm for Discrete L_1 Linear Approximation with Linear Constraints", *SIAM. J. Numer. Anal.*, 15, 603-611.

Bartels, R. H. and Conn, A. R. (1980), "Linearly Constrained Discrete L_1 Problems", *ACM Transactions on Mathematical Software*, 6, 594-608.

Bartels, R. H., Conn, A. R. and Sinclair, J. W. (1978), "Minimization Techniques for Piecewise Differentiable Functions; The L_1 Solution to an Overdetermined Linear System", *SIAM. J. Numer. Anal.*, 15, 224-241.

Basset, Jr. G. and Koenker, R. (1978), "Asymptotic Theory of Least Absolute Error Regression", *JASA*, 73, 618-621.

Blattberg, R. and Sargent, T. (1971), "Regression with Non-Gaussian Stable Disturbances", *Econometrica*, 39, 501-510.

Brecht, H. D. (1976), "Regression Methodology with Gross Observation Errors in the Explanatory Variables", *Decision Sciences*, 7, 57-65.

Cavalier, T. M. and Soyster, A. L. (1985), "Some Computational Experience and a Modification of the Karmarkar Algorithm", Working Paper, Dept. of Industrial and Management System Engineering, Penn. State Univ., University Park, PA.

Cox, D. R. and Hinkley, D. V. (1974), *Theoretical Statistics*, Chapman and Hall, London.

Cramer, H. (1946), *Mathematical Methods of Statistics*, Princeton, NJ.

Dantzig, G. B. (1963), *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ.

Dielman, T. and Pfaffenberger, R. (1982), "LAV(Least Absolute Value) Estimation in Linear Regression; a Review", *TIMS/Studies in the Management Sciences*, 19, 31-52.

Fisher, W. D. (1961), "A Note on Curve Fitting with Minimum Deviations by Linear Programming", *JASA*, 56, 359-362.

Forsythe, A. B. (1972), "Robust Estimation of Straight Line Regression Coefficients by Minimizing p-th Power Deviations", *Technometrics*, 14, 159-166.

Gilsinn, J., Hoffman, K., Jackson, R. H. F., Leyendecker, E., Saunders, P. and Shier, D. (1977), "Methodology and Analysis for Comparing Discrete Linear L_1 Approximation Codes", *Commun. Statist. - Simula. Computa.*, B(6), 399-413.

Glahe, F. R. and Hunt, J. G. (1970), "The Small Sample Property of Simultaneous Equation Least Absolute Estimators vis-a-vis Least Squares Estimators", *Econometrica*, 38, 742-753.

Harter, H. L. (1977), "Nonuniqueness of Least Absolute Values Regression", *Commun. Statist. - Theor. Meth.*, A(6), 829-838.

Harvey, A. C. (1978), "On the Unbiasedness of Robust Regression Estimators", *Commun. Statist. - Theor. Meth.*, A(7), 779-783.

Hoffman, K. L. and Shier, D. R. (1980), "A Test Problem Generator for Discrete Linear L_1 Approximation Problems", *ACM Transactions on Mathematical Software*, 6, 587-593.

Hunt, J. G., Dowling, J. M. and Glahe, F. R. (1974), "L_1 Estimation in Small Samples with Laplace Error Distributions", *Decision Sciences*, 5, 22-29.

Karmarkar, N. (1984), "A New Polynomial-Time Algorithm for Linear Programming", *Combinatorica*, 4, 373-395.

Kennedy, W. J., Gentle, J. E. and Sposito, V. A. (1977), "A Computer Oriented Method for Generating Test Problems for L_1 Regression", *Commun. Statist. - Simula. Computa.*, B(6), 21-27.

Kiountouzis, E. A. (1973), "Linear Programming Techniques in Regression Analysis", *Appl. Statist.*, 22, 69-73.

Kortanek, K. O. and Shi, M. (1985), "Convergence Results and Numerical Experiments on a Linear Programming Hybrid Algorithm", Working Paper, Dept. of Mathematics, Carnegie-Mellon Univ..

McCormick, G. F. and Sposito, V. A. (1976), "Using the L_2 Estimator in L_1 Estimation", *SIAM J. Numer. Anal.*, 13, 337-343.

Meketon, M. S. (1985), "Least Absolute Value Regression", Working Paper, Operations Research Dept., AT&T Bell Laboratories, Hurray Hill, NJ.

Money, A. H., Affleck-Graves, J. F., Hart, M. L. and Barr, G. D. I. (1982), "The Linear Regression Model; L_p Norm Estimation and the Choice of p", *Commun. Statist. - Simula. Computa.*, 11, 89-109.

Paige, C. C. and Saunders, M. A. (1982), "LSQR; An Algorithm for Sparse Linear Equations and Least Squares Problems", *ACM Transactions on Mathematical Software*, 8, 43-71.

Pfaffenberger, R. C. and Dinkel, J. J. (1978), "Absolute Deviations Curve Fitting;An Alternative to Least Squares", *Contributions to Survey Sampling and Applied Statistics*, edited by H. A. David, Academic Press, NY, 279-294.

Rice, J. R. (1964), *The Approximation of Functions*, Vol. I. Reading, Mass. Addison-Wesley.

Rice, J. R. and White, J. S. (1964), "Norms for Smoothing and Estimation", *SIAM Review*, 6, 243-256.

Robers, P. D. and Ben-Israel, A. (1970), "A Suboptimization Method for Interval Linear Programming; a New Method for Linear Programming", *Linear Algebra and its Applications*, 3, 383-405.

Robers, P. D. and Robers, S. S. (1973), "Discrete Linear L_1 Approximation by Interval Linear Programming", *Communications of the ACM*, 16, 629-631.

Rosenberg, B. and Carson, D. (1977), "A Simple Approximation of the Sampling Distribution of Least Absolute Residuals Regression Estimates", *Commun. Statist. - Simula. Computa.*, B(6), 421-437.

Schlossmacher, E. J. (1973), "An Iterative Technique for Absolute Deviations Curve Fitting", *JASA*, 68, 857-859.

Searle, S. (1971), *Linear Models*, John Wiley and Sons, Inc., NY.

Sherali, H. D. (1986), "Algorithmic Insights and a Convergence Analysis for a Karmarkar-type of Algorithm for Linear Programming Problems", *Naval Research Logistics Quarterly*, to appear.

Siddiqui, M. M. (1962), "Approximations to the Moments of the Sample Median", *AMS*, 33, 157-168.

Sielken, Jr. R. L. and Hartley, H. O. (1973), "Two Linear Programming Algorithms for Unbiased Estimation of Linear Models", *JASA*, 68, 639-641.

Sposito, V. A. (1975), *Linear and Nonlinear Programming*, Iowa State Univ. Press, Ames.

Sposito, V. A. (1982), "On the Unbiased L_p Regression Estimators", *JASA*, 77, 652-653.

Sposito, V. A., Hand, M. L. and Skarpness, B. (1983), "The Efficiency of Using the Sample Kurtosis in Selecting Optimal L_p Estimators", *Commun. Statist.-Simula. Computa.*, 12, 265-272.

Usow, K. H. (1967), "On L_1 Approximation II, Computation for Discrete Functions and Discretization effects", *SIAM J. Numer. Anal.*, 4, 233-244.

Vanderbei, R. J., Meketon, M. S. and Freedman, B. A. (1985), "A Modification of Karmarkar's Linear Programming Algorithm", Working Paper, AT&T Bell Laboratories, Holmdel, NJ.

Wagner, H. (1959), "Linear Programming Techniques for Regression Analysis", *JASA*, 54, 206-212.

Wesolowsky, G. O. (1981), "A New Descent Algorithm for the Least Absolute Value Regression Problem", *Comm. in Statist.*, B(10), 479-491.

Wilson, H. G. (1978), "Least Squares versus Minimum Absolute Deviations Estimation in Linear Models", *Decision Sciences*, 9, 322-335.