JOURNAL OF RESEARCH of the National Bureau of Standards
Vol. 84, No. 6, November–December 1979

Generators for Discrete Polynomial L_1 Approximation Problems*

P. Domich, J. Lawrence, and D. Shier**

Center for Applied Mathematics, National Bureau of Standards, Washington, DC 20234

June 27, 1979

Polynomial approximation problems represent a class of specially structured problems which are frequently encountered in empirical curve-fitting. Two generators for creating such problems have been developed, implemented and used in the testing of discrete L_1 approximation codes. Both generators permit automatic generation of problems with specified characteristics and (for one generator) having known, unique and controllable solutions.

Key words: Algorithm testing; approximation; computational experiment; least absolute deviation; polynomial approximation; test problems

1. Introduction

Recent years have seen increased interest in least absolute deviation (L_1) data fitting, either as an alternative to or in conjunction with the usual least squares approach [11, 19].[1] Estimation in the L_1 norm has certain desirable statistical properties (such as "robustness" [14, 16] when the underlying error distribution is long-tailed) and it can be carried out using reasonably efficient mathematical programming procedures.

While a number of algorithms for discrete L_1 approximation have recently been advanced [1, 3, 5, 6, 21, 22], computational comparisons of specific implementations ("codes") for these approaches have not only been limited [3, 6, 22] but have often produced conflicting evidence. Resolution of these conflicts awaits the establishment of reliable and comprehensive methodology for testing such mathematical software.

As a first step in developing a sound evaluation methodology for comparing mathematical programming codes, four L_1 codes (representing a range of solution techniques and implementation strategies) were evaluated on particular classes of test problems [13]. The first group of test problems were "hand-picked" problems [20]—those created with a specified structure in mind or arising from actual applications. The second group consisted of pseudo-randomly generated problems, produced by means of a test problem generator [17] and representative of a range of general L_1 data fitting problems; such problems could be constructed with a number of controllable characteristics such as degeneracy, rank loss and optimal solution.

The two generators POLY1 and POLY2 described in this paper have subsequently been utilized to produce other types of problem classes for which L_1 approximation is appropriate. In particular, these are problems in which it is desired to obtain the best L_1 polynomial approximation to a specific function (POLY1) or to a set of discrete observations (POLY2). Polynomial approximation problems are frequently encountered in practice (e.g., among our hand-picked problems), and also enjoy an extensive theoretical basis [7, 18, 23]. Moreover, such problems are known to admit a range of "ill-conditioning" and numerical difficulties [10], notably evidenced in our previous testing efforts using 27 hand-picked problems [13] and in recent results obtained using the first of these two generators [8].

The use of generated test problems in code evaluation offers several advantages. Generators provide a virtually inexhaustible source of test problems, and a source that can be made relatively transportable from one computer to another. Since generators are able to provide test problems with controllable characteristics (such as size and structure), fairly well-defined classes of problems can be efficiently created. Moreover,

* AMS Subject Classification: 65D10, 65D15, 68A20

** This work was done while the author was a National Research Council-National Academy of Sciences Postdoctoral Research Associate at the National Bureau of Standards, Washington, D.C. 20234.

[1] Numbers in brackets indicate literature references at the end of this paper.

through the invocation of pseudo-random number generators, test problems can be "randomly" selected from such problem classes and valid statistical conclusions can be drawn about code performance with respect to these classes. Another desirable property of test problem generators, and one that is useful in judging the accuracy or correctness of a code, involves the ability to specify in advance the solution to a generated problem.

Two generators are presented here for obtaining polynomial approximation test problems for L_1 curve-fitting. Both generators permit automatic generation of test problems with stipulated characteristics, and one of them (POLY2) allows the specification of known and unique solutions to the generated problems. The theoretical design of both generators is discussed in section 2, and the following two sections describe computer implementations of the theoretical design. The Appendices contain FORTRAN listings for both generators, written to be machine-independent.[2]

2. Design of the Generators

The discrete linear L_1 approximation problem can be formulated as follows. Given a set of n observations on a single dependent variable y and each of $m + 1$ independent variables $z_0, \ldots, z_m$, find parameters $\beta_0, \ldots, \beta_m$ that minimize

$$\phi = \sum_{i=1}^{n} \left| y_i - \sum_{j=0}^{m} \beta_j z_{ij} \right| .$$

The L_1 *polynomial approximation problem*, with which we will be concerned, is a restricted case of the above formulation: namely, minimize by choice of $\beta = (\beta_0, \ldots, \beta_m)$

$$\phi = \phi(\beta) = \sum_{i=1}^{n} \left| y_i - \sum_{j=0}^{m} \beta_j x_i^j \right| .$$

In the above, y_i refers to the i-th observation on variable y and x_i refers to the i-th observation on (the single independent) variable x. Equivalently, it is required to find a polynomial of *degree m* which best fits the given data, in the sense of minimizing the sum of absolute values of the *residuals*

$$e_i = y_i - \sum_{j=0}^{m} \beta_j x_i^j .$$

A vector β^* which yields the minimum value ϕ^* of $\phi(\beta)$ is termed a *solution vector*, with optimum *objective function value* ϕ^*. Unlike the case of L_1 polynomial approximation over a continuous interval [23, p. 38], the optimum solution vector β^* to a discrete L_1 polynomial approximation problem need not be unique. It is known [2, 4, 12] that such a solution vector can always be found which *interpolates* at least $m + 1$ of the data points; that is, at least $m + 1$ residuals e_i are identically zero at the solution. A given problem exhibits *degeneracy* if more than $m + 1$ residuals equal zero at the optimum.

A number of algorithms for solving discrete L_1 approximation problems make essential use of the above "interpolation" or "extreme point" result. Namely, these algorithms examine only *basic solutions*, where precisely $m + 1$ residuals are zero, and move in a systematic way from one basic solution to another. Any such basic solution is defined by a set of $m + 1$ indices (or rows) $i \in \{1, 2, \ldots, n\}$ where $e_i = 0$. For convenience, these indices are said to correspond to *active constraints*, while the remaining indices correspond to *inactive constraints*. Thus, in the absence of degeneracy the set $N = \{1, 2, \ldots, n\}$ can for any basic solution be partitioned as $N = N^0 \cup N^+ \cup N^-$, where N^0 indicates the active constraints, N^+ the inactive constraints with positive residuals, and N^- the inactive constraints with negative residuals.

[2] All FORTRAN programs have been checked for portability using the PFORT verifier (B. G. Ryder, Software-Practice and Experience, Vol. 4, 1974, 359–377) and conform to a subset of ANSI FORTRAN (X3.9-1966, Amer. Nat. Stand. Inst., New York, 1966).

The two generators for polynomial approximation problems discussed here will, for specified n and m, produce the following test problem data:

$$X = (x_i^j), \quad i = 1, \ldots, n; \quad j = 0, \ldots, m,^3 \tag{1}$$

and

$$y = (y_i), \quad i = 1, \ldots, n. \tag{2}$$

The basic difference between the generators resides in the type of problem structure they simulate.

The first generator, POLY1, is intended to model the situation where one wishes to approximate, over a discrete set, a continuous function $f(x)$ by a polynomial of specified degree. Accordingly, the values y_i in (2) are given by $y_i = f(x_i)$. Controllable features of this generator include: the function $f(x)$, the real interval I over which $f(x)$ is to be approximated, the number and distribution of observation points $x_i \in I$, and the degree m of the approximating polynomial. A more detailed description of this generator is provided in section 3. It should be noted that POLY1 does not produce a known solution to the test problem created; therefore, other means are necessary in order to verify whether or not a given L_1 code has actually "solved" a generated problem. For this reason, we have also devised a computer routine to check optimality of code-produced solutions by examination of the Kuhn-Tucker conditions for an associated linear program [4, 17].

The second generator, POLY2, models the situation where an L_1 fit is required to a given discrete set of data (y_i, x_i), $i = 1, \ldots, n$. Unlike the case for POLY1, the y_i are not assumed to be generated via some known functional relation. Controllable features of this second generator include: the real interval of approximation I, the number and distribution of observation points $x_i \in I$, the degree of the approximating polynomial, and the statistical distribution of the residuals e_i. Moreover, one is also able to specify *in advance* a solution vector β^* which will be guaranteed to be the unique L_1 solution to the generated problem.

The approach used for generating a data set (X, y) with known optimal solution β^* will now be outlined. The matrix X in (1) is partitioned as

$$X = \begin{bmatrix} X^0 \\ X^+ \\ X^- \end{bmatrix},$$

corresponding to indices $i \in N^0$, N^+, and N^- respectively. Note that X^0 is a square $(m + 1) \times (m + 1)$ nonsingular matrix, with the proviso that the x_i's for $i \in N^0$ are distinct. By a result proved in [17], a sufficient condition for the optimality and uniqueness of β^* in (X, y) is that

$$\sum_{i \in N^+} x_i^j = \sum_{i \in N^-} x_i^j, \quad j = 0, \ldots, m, \tag{3}$$

and that

$$y = X\beta^* + e, \tag{4a}$$

where

$$\begin{aligned} e_i &= 0 \quad \text{if} \quad i \in N^0, \\ e_i &> 0 \quad \text{if} \quad i \in N^+, \\ e_i &< 0 \quad \text{if} \quad i \in N^-. \end{aligned} \tag{4b}$$

It will now be shown how values x_i can be found that satisfy (3).

[3] For the first generator POLY1, one may also specify whether the polynomial has a constant term ($j = 0, \ldots, m$) or not ($j = 1, \ldots, m$).

First, we note that for $j = 0$, equation (3) requires $|N^+| = |N^-|$. For notational convenience, let $r = |N^+| = |N^-|$ and redefine the x_i's ($i \in N^+$) as $v_1, \ldots, v_r$; similarly, let the x_i's ($i \in N^-$) be denoted as $w_1, \ldots, w_r$. Also, write

$$[v_1, \ldots, v_r]_q = [w_1, \ldots, w_r]_q \tag{5}$$

if

$$\sum_{p=1}^{r} v_p^h = \sum_{p=1}^{r} w_p^h, \quad \text{for} \quad h = 0, 1, \ldots, q.$$

Then, following the development in [15], it is straightforward to verify that for any d

$$\begin{aligned} [v_1, \ldots, v_r]_q = [w_1, \ldots, w_r]_q &\Rightarrow [v_1, \ldots, v_r, w_1 + d, \ldots, w_r + d]_{q+1} \\ &= [w_1, \ldots, w_r, v_1 + d, \ldots, v_r + d]_{q+1}. \end{aligned} \tag{6}$$

Relation (6) can be used recursively to produce values x_i that satisfy (3). For example, since

$$[0]_0 = [1]_0,$$

equation (6) gives

$$[0, 3]_1 = [1, 2]_1, \quad \text{using} \quad d = 2,$$

$$[0, 3, 5, 6]_2 = [1, 2, 4, 7]_2, \quad \text{using} \quad d = 4,$$

and so forth. Accordingly, the consecutive integers $0, 1, 2, \ldots, 2^{q+1} - 1$ can be divided into two disjoint subsets $\{v_1, \ldots, v_r\}$ and $\{w_1, \ldots, w_r\}$, with $r = 2^q$, such that $[v_1, \ldots, v_r]_q = [w_1, \ldots, w_r]_q$. The first subset consists of all integers l, $0 \le l \le 2^{q+1} - 1$ with an even number of ones in their binary expansion, and the second subset consists of all such integers with an odd number of ones. Dividing each v_p and w_p by $2^{q+1} - 1$ produces a set of numbers equally spaced on [0, 1], which in turn can be scaled and translated to yield a set of numbers $\{v'_p, w'_p: p = 1, \ldots, r\}$ equally spaced on an arbitrary interval $I = [a, b]$. Moreover, relation (5) still holds for the scaled and translated v'_p, w'_p. Finally, these values can be used in (3) as the required x_i's for $i \in N^+$ and $i \in N^-$, respectively.

More generally the recursive process based on (6) can be initiated, using any positive integer d_0, by means of

$$[0]_0 = [d_0]_0$$

and propagated using any $d_1, d_2, \ldots, d_q$ to produce 2^{q+1} numbers satisfying (5), and therefore (3). We have chosen (see sect. 4) to generate the d_j's from a uniform probability distribution. The resulting (scaled and translated) values for x_i are no longer equally-spaced on I, but tend to be approximately normally distributed within the (finite) interval I. With probability one, all such x_i values generated are distinct.

The above procedures form the theoretical basis for implementation of the second generator. Either equidistant x_i or nonequidistant x_i (corresponding to inactive constraints) can thus be generated. To ensure that (3) holds for $j = 0, \ldots, m$, the number of observations n is required to be of the form

$$n = m + 1 + 2^{m+k+1}, \tag{7}$$

where k is an integer, $k \ge 0$. Further details of POLY2 are described in section 4.

3. Computer Implementation of POLY1

This section describes a computerized version, in FORTRAN, for the first generator (POLY1) discussed in section 2. We will indicate those problem design characteristics available as user-specified input options to POLY1, and then discuss how these inputs are utilized in creating test problem data.

A variety of characteristics are available to the user as controllable options and are specified through input parameters to POLY1. These options include:

1. The number of observations n.
2. The degree m of the approximating polynomial.
3. The specific function $f(x)$ to be approximated. Ten such functions,[4] representing a variety of shapes likely to be encountered in practice, have been incorporated within the program (see table 1 and fig. 1).
4. The interval of approximation $I = [a, b]$.
5. A switch ICOL to indicate whether the approximating polynomial includes a constant term or not. (The latter case is useful in forcing the approximating polynomial to pass through the origin.)
6. The location of the n observation points x_i within I: either drawn from a uniform probability distribution or equally-spaced over I.
7. A perturbation factor EPS, possibly zero, used in perturbing equally-spaced observation points x_i.
8. Two initial starting seeds, utilized internally by a pseudo-random number generator.

TABLE 1. *Functions and Their Intervals of Approximation.*

$f(x)$	$I = [a, b]$
1. $e^{-x}\sin x$	[0, 4]
2. $e^{x}\sin x$	[0, 4]
3. $e^{x\sin x}$	[0, 7]
4. $e^{2x}/2x$	[.05, 1]
5. $75x/[1 + (7.5x)^2]$	[0, 2]
6. $10xe^{-.5x}$	[0, 4]
7. $1/[1 + (x - 2.5)^4]$	[0, 5]
8. $x^{1/3}$	[−1, 1]
9. $x^{1/2}$	[0, 1]
10. $1/[1 + x^4]$	[−2.5, 2.5]

Given these input parameters, generation of the data sets can begin. The first step involves selecting the n observation points x_i from the desired interval, according to the distributional form specified. The interval $I = [a, b]$ is determined by the user (if CSWTCH $\neq 0$) through input parameters ENDL $= a$ and ENDR $= b$, or (if CSWTCH $= 0$) by means of the default interval settings shown in table 1. The observation points are chosen according to a uniform probability distribution over I (if MSWTCH $\neq 0$) or equally-spaced over I (if MSWTCH $= 0$). In the latter case, the user has the ability of perturbing the equally-spaced observation points x_i: namely, if $I = [a, b]$ then

$$x_i \rightarrow x_i + \text{EPS}\cdot\Delta\cdot\alpha_i, \qquad i = 2, \ldots, n-1,$$

where $\Delta = \dfrac{b-a}{n-1}$ is the common subinterval length for equal spacing, the α_i are independent uniform random variates over $[-1, 1]$, and EPS is a user-supplied input representing the (maximum) fraction of the subinterval length Δ used for perturbation. If EPS $= 0.0$, no perturbation is performed; otherwise, a reasonable range for EPS is $0 < \text{EPS} < 1/2$.

Next, the values $y_i = f(x_i)$ are calculated, as well as the successive powers x_i^j. If the user has specified a constant term in the approximating polynomial (ICOL $= 1$), then the above index j has range $j = 0, 1, \ldots, m$. Otherwise (ICOL $\neq 1$), a constant term is not included and $j = 1, \ldots, m$. This stage of calculating successive powers x_i^j is most sensitive to finite-precision arithmetic and to accumulated round-off error. Accordingly, such quantities are calculated in double-precision, using the recursion $x_i^{j+1} = x_i(x_i^j)$.

[4] It will be noted that function 10 is simply a translation of function 7. It has been included as a separate entity to illustrate the effect of "scaling": namely, the successive powers x_i^j produce more ill-conditioning when $|x_i|$ is larger (function 7) than smaller (function 10).

Finally, the problem data are returned in a single matrix PRBMAT. The first column of PRBMAT contains the vector **y**, and the remaining columns contain the design matrix X. In addition, the number of rows (NOBS) and the number of columns (NPAR) in matrix PRBMAT are returned to the calling routine, as well as a single-precision vector containing the observation points x_i. The final seeds available from the random number generator are also returned in ISEED and JSEED, thus allowing the creation of different test problems (having the same input specifications) in successive calls of POLY1 from a main routine.

All error messages are written out using unit IOUT = 6. Changing this specific value of IOUT once at the beginning of the program permits other print unit numbers to be accommodated. Appendix A gives a complete listing of POLY1, together with the subroutine UNIRAN [9], a well-tested uniform random number generator which it calls.

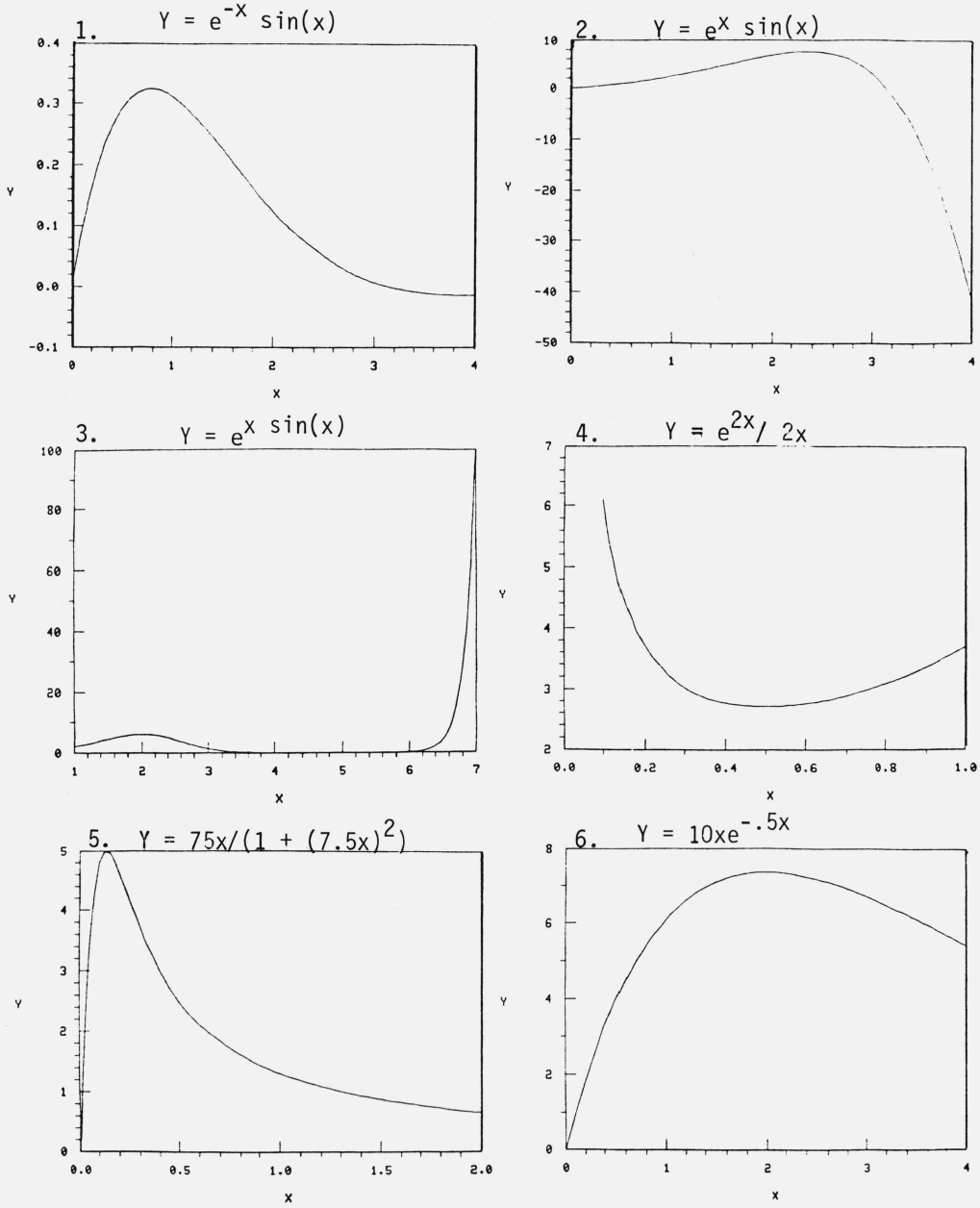

FIGURE 1. *Graphs of functions.*

Figure 2 illustrates the type of test problem that can be created by POLY1. The $n = 50$ data points (y_i, x_i) are derived from function 7 of table 1 over $I = [0, 5]$ and are shown here by the symbol "x"; in this case, the x_i's were chosen to be equally-spaced (without perturbation) over I. By way of reference, the solid curve indicates the best L_1 fitting polynomial of degree $m = 5$ to these generated data.

4. Computer Implementation of POLY2

This section describes a computer implementation of the second generator (POLY2) for polynomial approximation problems. As indicated in section 2, an important feature of this generator is its ability to create data sets whose optimal L_1 solution vector can be specified in advance. Consequently, the correctness and accuracy of solutions produced by various L_1 codes can be readily assessed using the known solutions to such problems. The fact that these solutions are guaranteed to be unique is also advantageous, since in the case of nonunique solutions it becomes difficult to compare the efficiency and accuracy of codes that reach correct, but different, solutions.

A number of problem characteristics can be controlled by the user through appropriate input specifications. Options available in POLY2 include:

1. The degree m of the approximating polynomial.
2. The value k in (7), which together with m determines the number of observations n.
3. The (unique) solution vector BETA to the generated problem.
4. The interval of approximation $I = [a, b]$.
5. The location of the observation points x_i corresponding to active constraints: either drawn from a uniform probability distribution or equally-spaced over I.
6. The location of the observation points x_i corresponding to inactive constraints: either equally-spaced or unequally-spaced over I.

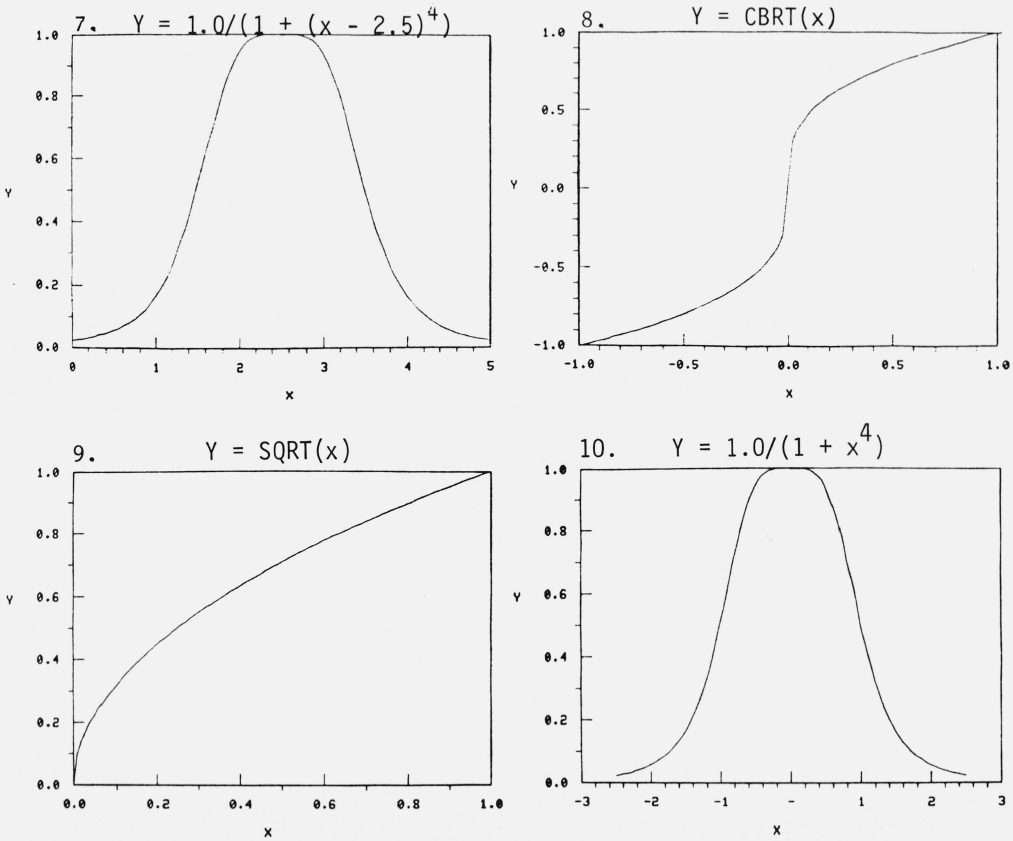

FIGURE 1. *(cont.)*

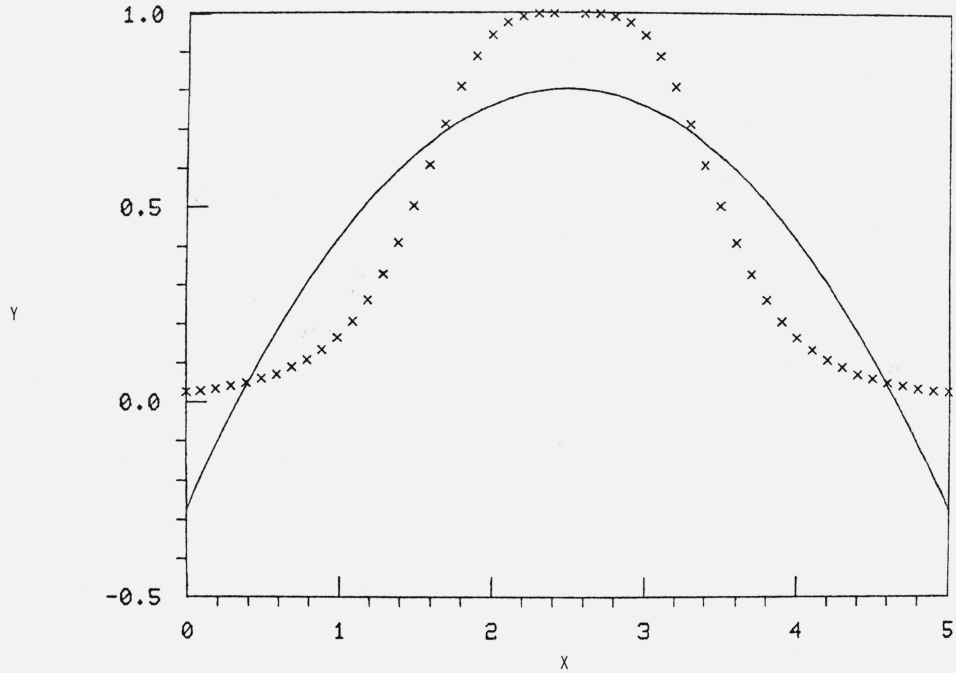

FIGURE 2. *Sample test problem created by POLY1 for function* 7, $m = 5$, *and* $n = 50$.

7. A perturbation factor EPS, possibly zero, used in perturbing equally-spaced x_i associated with active constraints.
8. The distribution of the residuals e_i associated with inactive constraints.
9. Two initial starting seeds, utilized internally by pseudo-random number generators.

The first step in developing a test problem is to select the solution vector BETA, if the user has not chosen to input a vector to POLY2. The solution vector is randomly selected, in this case, from a uniform probability distribution over an interval specified by the user (BSW = 1) or defined within the program (BSW $\neq$ 1).

Next, the observation points x_i are generated from the interval I according to the desired distribution. Interval I is either specified on input (CSW = 1) or assigned a default setting within the program (CSW $\neq$ 1). The x_i's corresponding to active constraints are derived from a uniform probability distribution over I (MSW = 1) or are equally-spaced over I (MSW $\neq$ 1). In this latter case, the x_i can be perturbed using the factor EPS, exactly as described for POLY1. The observation points x_i corresponding to inactive constraints are generated so that equation (3) is satisfied. Such x_i are either equally-spaced (ISW = 1) or unequally-spaced (ISW $\neq$1) over the interval I; see section 2 for details of how these two methods of generation are achieved.

Also, the entries x_i^j of the design matrix are successively calculated for $j = 0, \ldots, m$ using appropriate care to preserve numerical accuracy. At the same time, the residuals e_i, $i \in N^+$ or $i \in N^-$, are generated from either a uniform (RSW = 1) or a normal distribution (RSW $\neq$1), are given the appropriate sign, and are then used to calculate y_i from (4a) − (4b). The optimum objective function value SUMRES is calculated in double-precision, based on these residuals. The input parameter RESID indirectly controls the magnitude of this objective function value. Namely, RESID acts as a scale factor with respect to the size of the residuals, which are either selected from a uniform distribution on [−RESID, RESID], whose standard deviation is thus RESID/$\sqrt{3}$, or from a normal distribution having mean 0 and standard deviation RESID.

Finally, the problem data are returned as before in a single matrix PRBMAT. The first column of PRBMAT contains the **y** vector, and the remaining columns contain the design matrix X. The number of rows (KKM1) and the number of columns (M2) are also returned, as well as BETA, SUMRES and LOC (a vector containing the $m + 1$ row locations of the active constraints). The final seeds available from the last call to the random number generator are returned in ISEED and JSEED.

All error messages are written out using unit IOUT, which can be easily modified to accommodate differing print unit assignments. Appendix B gives a complete listing of POLY2, together with the subroutines it calls: SORTM, SET1, SET2, EQUAL, NORRAN, SORTP and UNIRAN. The call structure of these subroutines is shown in figure 3; the latter three subroutines are fairly well-tested and are described further in [9].

Figures 4 and 5 illustrate the types of data sets that can be produced using POLY2. Both show as dots 134 data points (y_i, x_i) together with the best L_1 fitting polynomial of degree 5 (solid curve). In figure 4, the x_i are generated via the equally-spaced option (ISW = 1), while in figure 5 they are generated via the unequally-spaced option (ISW $\neq$ 1).

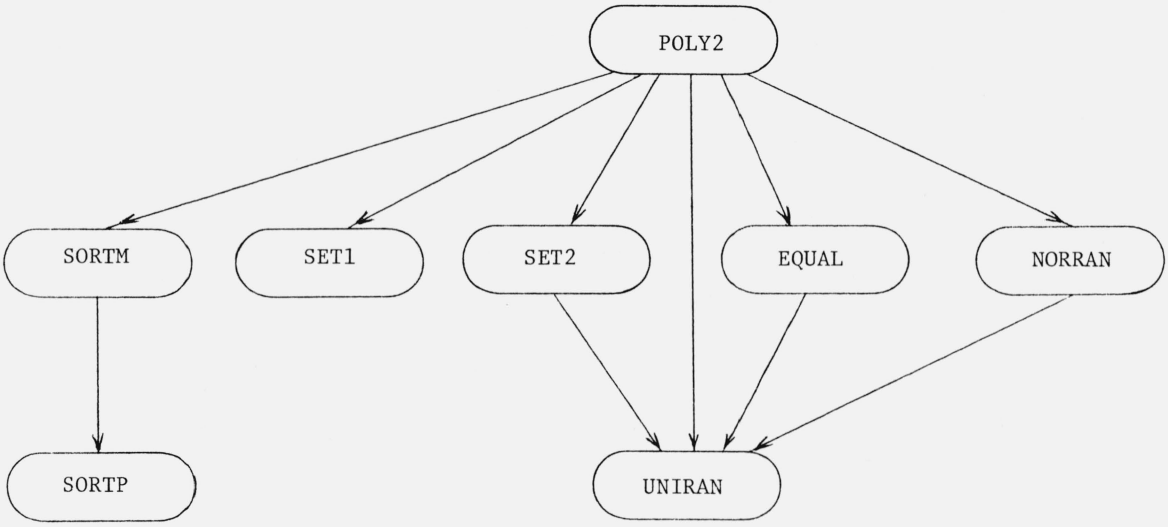

FIGURE 3. *Subroutine call structure for POLY2.*

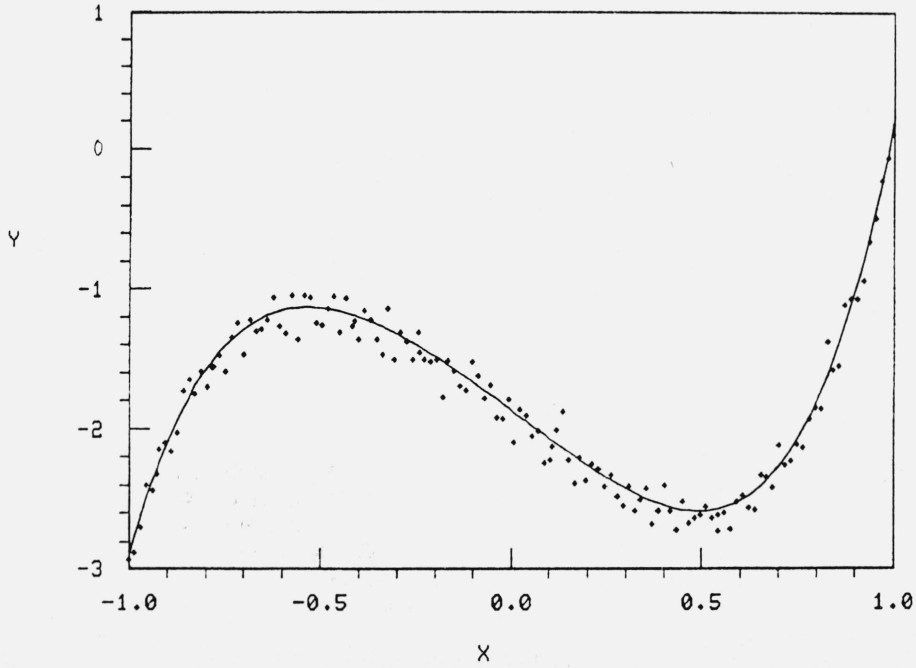

FIGURE 4. *Sample test problem created by POLY2:* $n = 134$, $m = 5$, *equally-spaced* x_i.

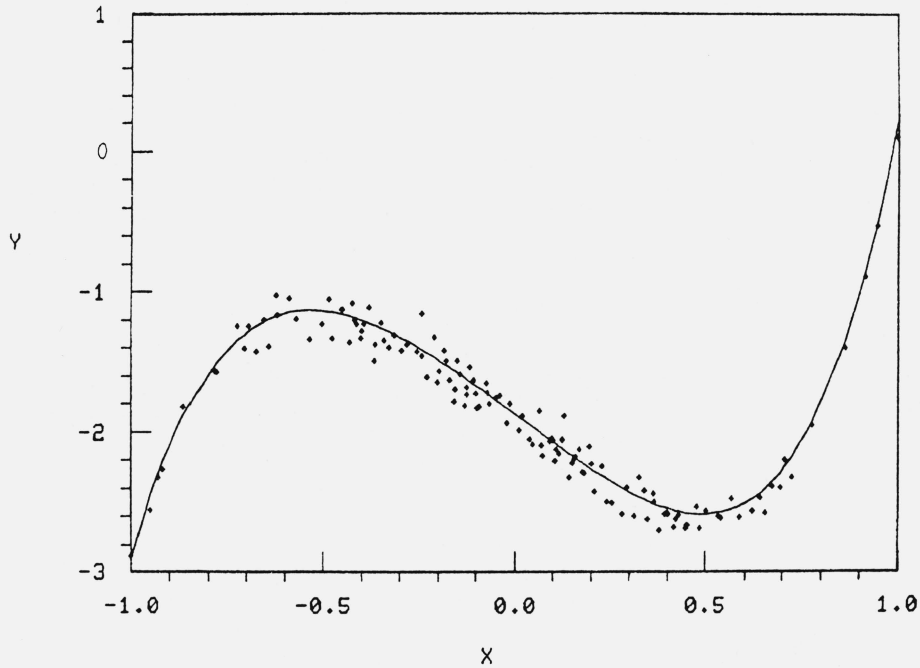

FIGURE 5. *Sample test problem created by POLY2:* $n = 134$, $m = 5$, *unequally-spaced* x_i.

5. References

[1] Abdelmalek, N. N., An efficient method for the discrete linear L_1 approximation problem, Mathematics of Computation, Vol. **29**, (1975), 844–850.

[2] Appa, G. and Smith, C., On L_1 and Chebyshev estimation, Mathematical Programming, Vol. **5** (1973), 73–87.

[3] Armstrong, R. D. and Frome, E. L., A comparison of two algorithms for absolute deviation curve fitting, Journal of the American Statistical Association, Vol. **71**, No. 354 (1976), 328–330.

[4] Barrodale, I. and Roberts, F. D. K., An improved algorithm for discrete l_1 linear approximation, SIAM Journal on Numerical Analysis, Vol. **10**, No. 5 (1973), 839–848.

[5] Barrodale, I. and Roberts, F. D. K., Solution of an overdetermined system of equations in the l_1 norm, Communications of the Association for Computing Machinery, Vol. **17** (1974), 319–320.

[6] Bartels, R. H., Conn, A. R. and Sinclair, J. W., Minimization techniques for piecewise differentiable functions: The l_1 solution to an overdetermined linear system, SIAM Journal on Numerical Analysis, Vol. **15**, No. 2 (1978), 224–241.

[7] Davis, P. J., *Interpolation and Approximation*, (Blaisdell, New York, 1965).

[8] Domich, P. D., Hoffman, K. L., Jackson, R. H. F., Saunders, P. B. and Shier, D. R., Evaluation of L_1 codes using polynomial approximation problems, unpublished working paper, National Bureau of Standards.

[9] Filliben, J. J., DATAPAC: A data analysis package, Proceedings of the Ninth Interface Symposium on Computer Science and Statistics (Boston, MA, April 1–2, 1976), 212–216.

[10] Forsythe, G. E., Generation and use of orthogonal polynomials for data-fitting with a digital computer, SIAM Journal, Vol. **5**, No. 2 (1957), 74–88.

[11] Gentle, J. E., Least absolute values estimation: An introduction, Communications in Statistics, Vol. **B6**, No. 4 (1977), 313–328.

[12] Gentle, J. E., Kennedy, W. J. and Sposito, V. A., On least absolute values estimation, Communications in Statistics, Vol. **A6**, No. 9 (1977), 839–845.

[13] Gilsinn, J., Hoffman, K., Jackson, R. H. F., Leyendecker, E., Saunders, P. and Shier, D., Methodology and analysis for comparing discrete linear L_1 approximation codes, Communications in Statistics, Vol. **B6**, No. 4 (1977), 399–413.

[14] Hampel, F. R., A general qualitative definition of robustness, Annals of Mathematical Statistics, Vol. **42** (1971), 1887–1896.

[15] Hardy, G. H. and Wright, E. M., *An Introduction to the Theory of Numbers*, (Clarendon Press, Oxford, 1954).

[16] Harter, H. L., Nonuniqueness of least absolute values regression, Communications in Statistics, Vol. **A6**, No. 9 (1977), 829–838.

[17] Hoffman, K. and Shier, D., A test problem generator for discrete linear L_1 approximation problems, to appear in Transactions on Mathematical Software.

[18] Rice, J. R., *The Approximation of Functions*, Vol. I, (Addison-Wesley, Reading, MA, 1964).

[19] Rice, J. R. and White, J. S., Norms for smoothing and estimation, SIAM Review, Vol. **6**, No. 3 (1964), 243–256.

[20] Saunders, P. and Shier, D., A collection of test problems for discrete linear L_1 data fitting, unpublished working paper, National Bureau of Standards.

[21] Schlossmacher, E. J., An iterative technique for absolute deviations curve fitting, Journal of the American Statistical Association, Vol. **68,** No. 344 (1973), 857–859.

[22] Spyropoulos, K., Kiountouzis, E. and Young, A., Discrete approximation in the L_1 norm, Computer Journal, Vol. **16,** No. 2 (1973), 180–186.

[23] Timan, A. F., *Theory of Approximation of Functions of a Real Variable*, trans. by J. Berry, (MacMillan Co., New York, 1963).

6. Appendix A.

Listing of FORTRAN Subroutines for First Generator:

POLY1

UNIRAN

```
      SUBROUTINE POLY1(NFUNC,M,N,PRBMAT,CSWTCH,MSWTCH,ENDL,
     +      ENDR,ICOL,NOBS,NPAR,ISEED,JSEED,X,NROW,EPS)
C
C*****************************************************************
C
C     DESCRIPTION:
C
C     THIS SUBROUTINE GENERATES DISCRETE POLYNOMIAL APPROXIMATION
C     PROBLEMS WHICH CAN BE USED IN THE TESTING AND EVALUATION OF
C     ALGORITHMS FOR L1 (LEAST ABSOLUTE DEVIATION) CURVE FITTING.
C     THE USER CAN SPECIFY THE FOLLOWING CHARACTERISTICS OF THE
C     GENERATED DATA SETS:
C
C
C          * PROBLEM SIZE
C          * THE SPECIFIC FUNCTION TO BE APPROXIMATED
C          * THE REAL INTERVAL OVER WHICH THE FUNCTION IS APPROXIMATED
C          * THE DISTRIBUTION OF THE OBSERVATION POINTS IN THAT INTERVAL
C
C
C     THE POLYNOMIAL APPROXIMATION PROBLEM CONSIDERED HERE IS TO
C     APPROXIMATE (IN THE L1 NORM) A GIVEN FUNCTION BY A POLYNOMIAL
C     OF SPECIFIED DEGREE, OVER A DISCRETE SET.  FOR I=1,N  X(I) IS
C     A SELECTED POINT FROM SOME INTERVAL AND Y(I) IS THE CORRES-
C     PONDING FUNCTIONAL VALUE AT X(I).  IN THE PROBLEMS GENERATED
C     BY THIS SUBROUTINE, THE CONSTANT TERM OF THE POLYNOMIAL CAN BE
C     INCLUDED OR EXCLUDED (IN THE LATTER CASE FORCING THE APPROX-
C     IMATING POLYNOMIAL TO PASS THROUGH THE ORIGIN).
C
C     FOR CONVENIENCE THE Y VECTOR AND THE DESIGN MATRIX, WHICH
C     CONTAINS SUCCESSIVE POWERS OF THE OBSERVATION POINTS X(I),
C     ARE BOTH RETURNED IN A SINGLE MATRIX 'PRBMAT'.  THE FIRST
C     COLUMN OF PRBMAT CONTAINS THE Y VALUES AND THE REMAINING
C     COLUMNS CONTAIN THE DESIGN MATRIX.
C
C     NOTE:
C
C          * THE X(I) VALUES ARE ARRANGED TO BE IN INCREASING ORDER
C            FOR I=1,N. IN PARTICULAR, X(1) IS ALWAYS THE LEFT-HAND
C            ENDPOINT OF THE INTERVAL, AND X(N) IS ALWAYS THE RIGHT-
C            HAND ENDPOINT.
C
C     NOTE:
C
C          * SINCE AN OPTIMAL SET OF COEFFICIENTS FOR THE APPROXIMAT-
C            ING POLYNOMIAL IS NOT PRODUCED BY THIS GENERATOR, VERIFI-
C            CATION OF AN OPTIMAL SOLUTION SHOULD BE MADE BY SOME MEANS
C            (E.G., EXAMINING THE KUHN-TUCKER CONDITIONS).
C
C*****************************************************************
C
C     INPUT ARGUMENTS:
C
C          NFUNC - DETERMINES THE SPECIFIC FUNCTION BEING APPROXIMATED
C          M ----- DEGREE OF THE APPROXIMATING POLYNOMIAL
C          N ----- NUMBER OF OBSERVATION POINTS X(I)
```

```
C           CSWTCH- INPUT SWITCH USED FOR INTERVAL SELECTION, IF
C                    = 1 USES ENDPOINTS SUPPLIED BY THE USER TO DEFINE
C                        THE INTERVAL
C                    = 0 SELECTS THE INTERVAL USING DEFAULT SETTINGS
C           MSWTCH- INPUT SWITCH FOR THE DISTRIBUTION OF THE OBSERVATION
C                   POINTS, IF
C                    = 1 PRODUCES POINTS FROM A UNIFORM PROBABILITY DIS-
C                        TRIBUTION OVER THE INTERVAL
C                    = 0 PRODUCES POINTS EQUALLY SPACED OVER THE INTERVAL
C                        ( THOUGH POSSIBLY PERTURBED BY A RANDOM AMOUNT
C                        IF EPS > 0 )
C           ENDL -- USER SPECIFIED LEFT ENDPOINT OF THE INTERVAL
C           ENDR -- USER SPECIFIED RIGHT ENDPOINT OF THE INTERVAL
C           ICOL -- INPUT SWITCH, IF
C                    = 1 PRODUCES AN L1 PROBLEM WITH A CONSTANT COLUMN
C                    = 0 PRODUCES AN L1 PROBLEM W/O A CONSTANT COLUMN
C           ISEED - FIRST OF TWO INITIAL SEEDS FOR THE RANDOM NUMBER
C                   GENERATOR
C           JSEED - SECOND OF TWO INITIAL SEEDS FOR THE RANDOM NUMBER
C                   GENERATOR
C           NROW -- MAXIMUM NUMBER OF OBSERVATION POINTS
C           EPS --- FRACTION OF THE SUBINTERVAL LENGTH USED IN PER-
C                   TURBING EQUALLY-SPACED OBSERVATION POINTS, IF
C                   MSWTCH=0
C
C     NOTE:
C
C           * WHEN EPS = 0, NO PERTURBATION IS PERFORMED. OTHERWISE, A
C             REASONABLE RANGE IS 0<EPS<0.5 .
C
C           * THE DEFAULT VALUE FOR EACH OF THE SWITCHES ICOL, CSWTCH,
C             MSWTCH IS ZERO.
C
C     OUTPUT ARGUMENTS:
C
C           PRBMAT- THE GENERATED PROBLEM MATRIX, WITH NOBS ROW AND NPAR
C                   COLUMNS
C           NOBS -- NUMBER OF ROWS IN PRBMAT
C           NPAR -- NUMBER OF COLUMNS IN PRBMAT
C           ISEED - FIRST OF TWO FINAL SEEDS AVAILABLE FROM THE RANDOM
C                   NUMBER GENERATOR
C           JSEED - SECOND OF TWO FINAL SEEDS AVAILABLE FROM THE RANDOM
C                   NUMBER GENERATOR
C           X ----- VECTOR OF OBSERVATION POINTS, OF DIMENSION NOBS
C
C
C     RESTRICTIONS:
C
C           N MUST BE GT 1
C           N MUST BE LE NROW
C           M MUST BE GT 0
C           N CANNOT BE LT M
C           NFUNC MUST BE BETWEEN 1 AND 10, INCLUSIVE
C
C*********************************************************************
C
C     A VARIETY OF FUNCTIONAL FORMS AND SHAPES ARE CURRENTLY AVAILABLE:
C
```

```
C
C         NFUNC                  FUNCTION              DEFAULT INTERVAL
C         -----                  --------              ----------------
C          1                  EXP(-X)*SIN(X)              [0.0,4.0]
C          2                  EXP(X)*SIN(X)               [0.0,4.0]
C          3                  EXP(X*SIN(X))               [0.0,7.0]
C          4                  .5*EXP(2.*X)/X              [.05,1.0]
C          5                  75.*X/(1.+(7.5*X)**2)       [0.0,2.0]
C          6                  10.*X*EXP(-.5*X)            [0.0,4.0]
C          7                  1.0/(1.0+(X-2.5)**4)        [0.0,5.0]
C          8                  CBRT(X)                     [-1.0,1.0]
C          9                  SQRT(X)                     [0.0,1.0]
C         10                  1.0/(1.0+(X**4))            [-2.5,2.5]
C
C     SUBPROGRAMS USED:
C
C          SIN --- FORTRAN LIBRARY FUNCTION
C          EXP --- FORTRAN LIBRARY FUNCTION
C          CBRT -- FORTRAN LIBRARY OR USER-SUPPLIED FUNCTION
C                  TO COMPUTE CUBE ROOTS
C          SQRT -- FORTRAN LIBRARY FUNCTION
C          FLOAT - FORTRAN LIBRARY FUNCTION
C          SORT -- SUBROUTINE USED TO SORT THE ELEMENTS OF A VECTOR OF
C                  DIMENSION N INTO ASCENDING ORDER
C        UNIRAN -- SUBROUTINE TO GENERATE PSEUDO-RANDOM NUMBERS FROM
C                  A UNIFORM DISTRIBUTION OVER [0,1]
C
C     LANGUAGE: ANSI FORTRAN (1966)
C
C     REFERENCES:
C
C        K.L.HOFFMAN & D.R.SHIER,'A TEST PROBLEM GENERATOR FOR
C        DISCRETE LINEAR L1 APPROXIMATION PROBLEMS', TO APPEAR IN
C        TRANS. ON MATH. SOFTWARE
C        P.D.DOMICH & D.R.SHIER,'GENERATORS FOR DISCRETE POLYNOMIAL
C        L1 APPROXIMATION PROBLEMS'
C
C     WRITTEN BY:
C
C        PAUL D. DOMICH
C        CENTER FOR APPLIED MATHEMATICS
C        NATIONAL BUREAU OF STANDARDS
C        WASHINGTON D.C. 20234
C
C        JANUARY, 1979
C
C*********************************************************************
C
      DOUBLE PRECISION UNIINC,EPSA,DA,DPROD
      REAL PRBMAT(NROW,1),ENDS(10,2),X(NROW)
      INTEGER CSWTCH
      LOGICAL CONST
C
C***** DEFAULT SETTINGS FOR THE INTERVAL SPECIFICATIONS
C
      DATA ENDS(1,1),ENDS(2,1),ENDS(3,1),ENDS(4,1),ENDS(5,1),
     +     ENDS(6,1),ENDS(7,1),ENDS(8,1),ENDS(9,1),ENDS(10,1),
     +     ENDS(1,2),ENDS(2,2),ENDS(3,2),ENDS(4,2),ENDS(5,2),
```

```
     +      ENDS(6,2),ENDS(7,2),ENDS(8,2),ENDS(9,2),ENDS(10,2)/
     L    0.0, 0.0, 0.0, .05, 0.0, 0.0, 0.0, -1.0, 0.0, -2.5,
     R    4.0, 4.0, 7.0, 1.0, 2.0, 4.0, 5.0,  1.0, 1.0,  2.5/
C          1    2    3    4    5    6    7     8    9    10  = NFUNC
C
C***** CHECK INPUT DATA FOR CONSISTENCY
C
      IOUT=6
      IF(N .LE. 1 .OR. N .LT. M) GO TO 2001
      IF(N .EQ. M .AND. ICOL .EQ. 1) GOTO 2001
      IF(M .LT. 1) GOTO 2001
      IF(N .GT. NROW) GOTO 2002
      IF(NFUNC .LT. 1 .OR. NFUNC .GT. 10) GOTO 2003
      IF(NFUNC .EQ. 4 .AND. ENDL .LE. 0.0 .AND. ENDR .GE. 0.0) GOTO 2004
      IF(NFUNC .EQ. 9 .AND.(ENDR .LT. 0.0 .OR. ENDL .LT. 0.0)) GOTO 2005
C
C***** CALCULATION OF THE INTERVAL BY SPECIFICATION OR DEFAULT
C
      IF(CSWTCH .EQ. 1) A=ENDL
      IF(CSWTCH .EQ. 1) B=ENDR
      IF(CSWTCH .NE. 1) A=ENDS(NFUNC,1)
      IF(CSWTCH .NE. 1) B=ENDS(NFUNC,2)
      IF(A .LE. B) GOTO 4
      HOLD=A
      A=B
      B=HOLD
    4 XINT=B-A
C
C***** CALCULATE THE REQUIRED NUMBER OF COLUMNS IN THE DESIGN MATRIX
C
      IF (ICOL .EQ. 1) NVAR=M+1
      IF (ICOL .EQ. 1) CONST=.TRUE.
      IF (ICOL .NE. 1) NVAR=M
      IF (ICOL .NE. 1) CONST=.FALSE.
C
C***** CALCULATE THE OBSERVATION POINTS WITHIN THE INTERVAL [A,B]
C
      CALL UNIRAN(N,1,X,ISEED,JSEED)
      X(1)=A
      X(N)=B
      K=N-1
      IF(K.EQ.1) GOTO 8
      IF (MSWTCH .EQ. 1) GOTO 6
C
C***** CALCULATE THE SUBINTERVAL LENGTH FOR EQUALLY-SPACED POINTS,
C       POSSIBLY MODIFIED BY A PERTURBATION FACTOR (EPSA).
C
      UNIINC=XINT/FLOAT(N-1)
      EPSA=EPS*UNIINC
      DO 5 I=2,K
      X(I)= A+FLOAT(I-1)*UNIINC+EPSA*(2.0*X(I)-1.0)
    5 CONTINUE
      GOTO 8
C
C***** CALCULATE THE OBSERVATION POINTS USING A UNIFORM DISTRIBUTION
C       AND SORT THE POINTS INTO ASCENDING ORDER
C
    6 DO 7 I=2,K
      X(I)=XINT*X(I)+A
    7 CONTINUE
      CALL SORT(X,N,X)
C
```

```
C***** CALCULATION OF THE Y VECTOR
C
    8 DO 1030 I=1,N
      A=X(I)
      GOTO(10,20,30,40,50,60,70,80,90,100),NFUNC
   10 PRBMAT(I,1)=EXP(-A)*SIN(A)
      GOTO 1003
   20 PRBMAT(I,1)=EXP(A)*SIN(A)
      GOTO 1003
   30 PRBMAT(I,1)=EXP(A*SIN(A))
      GOTO 1003
   40 PRBMAT(I,1)=.5*EXP(2.0*A)/A
      GOTO 1003
   50 PRBMAT(I,1)=75.0*A/(1.0+(7.5*A)**2)
      GOTO 1003
   60 PRBMAT(I,1)=10.0*A*EXP(-.5*A)
      GOTO 1003
   70 PRBMAT(I,1)=1.0/(1.0+(A-2.5)**4)
      GOTO 1003
   80 PRBMAT(I,1)=CBRT(A)
      GOTO 1003
   90 PRBMAT(I,1)=SQRT(A)
      GOTO 1003
  100 PRBMAT(I,1)=1.0/(1.0+(A**4))
 1003 CONTINUE
C
C***** CALCULATION OF THE ENTRIES IN THE DESIGN MATRIX
C
      DA=A
      IF( CONST ) DPROD=1.0
      IF( .NOT. CONST ) DPROD=A
      PRBMAT(I,2)=DPROD
      IF(NVAR.EQ.1) GOTO 1030
      DO 1004 J=2,NVAR
      DPROD=DPROD*DA
      PRBMAT(I,J+1)=DPROD
 1004 CONTINUE
 1030 CONTINUE
C
C***** DEFINE ROW AND COLUMN DIMENSIONS OF THE PROBLEM MATRIX (PRBMAT)
C
      NOBS=N
      NPAR=NVAR+1
      GOTO 3000
 2001 WRITE(IOUT,901) N,M
  901 FORMAT(1H0,21HTHE INPUT VALUES N  =,I5,10H  AND M  =,I5,
     +        18H  ARE NOT FEASIBLE)
      GOTO 3000
 2002 WRITE(IOUT,902) N,NROW
  902 FORMAT(1H0,17HTHE VALUE OF N  =,I5,21H  EXCEEDS THE MAXIMUM,
     +        21H DIMENSION OF NROW  =,I5)
      GOTO 3000
 2003 WRITE(IOUT,903) NFUNC
  903 FORMAT(1H0,22HTHE VALUE OF NFUNC   =,I5,18H   IS NOT IN RANGE)
      GOTO 3000
 2004 WRITE(IOUT,904)NFUNC
  904 FORMAT(1H0,30H THE INTERVAL SET FOR FUNCTION,I3,11H IS INVALID)
      GOTO 3000
 2005 WRITE(IOUT,904)NFUNC
 3000 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE UNIRAN(N,ISTART,X,ISEED,JSEED)
C
C     PURPOSE--THIS SUBROUTINE GENERATES A RANDOM SAMPLE OF SIZE N
C              FROM THE UNIFORM (RECTANGULAR)
C              DISTRIBUTION ON THE UNIT INTERVAL (0,1).
C              THIS DISTRIBUTION HAS MEAN = 0.5
C              AND STANDARD DEVIATION = SQRT(1/12) = 0.28867513.
C              THIS DISTRIBUTION HAS THE PROBABILITY
C              DENSITY FUNCTION F(X) = 1.
C     INPUT  ARGUMENTS--N       = THE DESIRED INTEGER NUMBER
C                                 OF RANDOM NUMBERS TO BE
C                                 GENERATED.
C                     --ISTART = AN INTEGER FLAG CODE WHICH
C                                 (IF SET TO 0) WILL START THE
C                                 GENERATOR OVER AND HENCE
C                                 PRODUCE THE SAME RANDOM SAMPLE
C                                 OVER AND OVER AGAIN
C                                 UPON SUCCESSIVE CALLS TO
C                                 THIS SUBROUTINE WITHIN A RUN; OR
C                                 (IF SET TO SOME INTEGER
C                                 VALUE NOT EQUAL TO 0,
C                                 LIKE, SAY, 1) WILL ALLOW
C                                 THE GENERATOR TO CONTINUE
C                                 FROM WHERE IT STOPPED
C                                 AND HENCE PRODUCE DIFFERENT
C                                 RANDOM SAMPLES UPON
C                                 SUCCESSIVE CALLS TO
C                                 THIS SUBROUTINE WITHIN A RUN.
C     OUTPUT ARGUMENTS--X       = A SINGLE PRECISION VECTOR
C                                 (OF DIMENSION AT LEAST N)
C                                 INTO WHICH THE GENERATED
C                                 RANDOM SAMPLE WILL BE PLACED.
C     OUTPUT--A RANDOM SAMPLE OF SIZE N
C             FROM THE RECTANGULAR DISTRIBUTION ON (0,1).
C     PRINTING--NONE UNLESS AN INPUT ARGUMENT ERROR CONDITION EXISTS.
C     RESTRICTIONS--THERE IS NO RESTRICTION ON THE MAXIMUM VALUE
C                   OF N FOR THIS SUBROUTINE.
C     OTHER DATAPAC   SUBROUTINES NEEDED--NONE.
C     FORTRAN LIBRARY SUBROUTINES NEEDED--NONE.
C     MODE OF INTERNAL OPERATIONS--SINGLE PRECISION.
C     LANGUAGE--ANSI FORTRAN.
C     COMMENT--********************
C              THE PARAMETERS OF THE CONGRUENTIAL GENERATOR
C              USED HEREIN ARE OF NECESSITY STRONGLY MACHINE-
C              DEPENDENT IN TERMS OF THE QUALITY OF THE
C              RANDOM NUMBERS PRODUCED.  KNOWN GOOD RESULTS
C              HAVE BEEN OBTAINED ON THE UNIVAC 1108 (36 BIT)
C              COMPUTER.  KNOWN POOR RESULTS HAVE BEEN OBTAINED
C              ON THE CDC 3800 (48 BIT) COMPUTER.
C              IT IS STRONGLY RECOMMENDED THAT UPON
C              IMPLEMENTATION OF UNIRAN, THE OUTPUT
C              FROM THIS SUBROUTINE SHOULD BE CHECKED FOR INDEPENDENCE
C              AND UNIFORMITY.  DATAPAC SUBROUTINES PLOTX,
C              PLOTXX, PLOTU, RUNS, TIMESE, HIST, TAIL, LOC, AND
C              SCALE MAY BE USEFULLY EMPLOYED TO DO SUCH
C              CHECKING.  SUCH CHECKING IS ESPECIALLY IMPORTANT
C              IN LIGHT OF THE FACT THAT OTHER DATAPAC RANDOM
C              NUMBER GENERATOR SUBROUTINES (NORRAN--NORMAL,
C              LOGRAN--LOGISTIC, ETC.) ALL MAKE USE OF INTERMEDIATE
C              OUTPUT FROM UNIRAN.
```

```
C                 ********************
C      REFERENCES--KRONMAL, 'EVALUATION OF THE PSEUDO-RANDOM
C                    NORMAL NUMBER GENERATOR', JOURNAL OF THE
C                    ASSOCIATION FOR COMPUTING MACHINERY, 1964,
C                    PAGES 357-363.
C                  --HAMMERSLEY AND HANDSCOMB, MONTE CARLO METHODS,
C                    1964, PAGE 36.
C                  --JOHNSON AND KOTZ, CONTINUOUS UNIVARIATE
C                    DISTRIBUTIONS--2, 1970, PAGES 57-74.
C      WRITTEN BY--JAMES J. FILLIBEN
C                    STATISTICAL ENGINEERING LABORATORY (205.03)
C                    NATIONAL BUREAU OF STANDARDS
C                    WASHINGTON, D. C. 20234
C                    PHONE:  301-921-2315
C      ORIGINAL VERSION--JUNE       1972.
C      UPDATED          --AUGUST    1974.
C      UPDATED          --SEPTEMBER 1975.
C      UPDATED          --NOVEMBER  1975.
C
C-----------------------------------------------------------------------
C
      DIMENSION X(1)
C
      DATA DIV,K1,K2,K3,I1,I2/34359738368.0,4097,129,22182922491,2086735
     10019,18575103187/
C
      IPR=6
C
C     CHECK THE INPUT ARGUMENTS FOR ERRORS
C
      IF(N.LT.1)GOTO50
      GOTO90
   50 WRITE(IPR, 5)
      WRITE(IPR,47)N
      RETURN
   90 CONTINUE
    5 FORMAT(1H , 91H***** FATAL ERROR--THE FIRST  INPUT ARGUMENT TO THE
     1 UNIRAN SUBROUTINE IS NON-POSITIVE *****)
   47 FORMAT(1H , 35H***** THE VALUE OF THE ARGUMENT IS ,I8    ,6H *****)
C
C-----START POINT-----------------------------------------------------
C
      I1=ISEED
      I2=JSEED
      IF(ISTART.NE.0)GOTO150
      I1=20867350019
      I2=18575103187
C
  150 DO100I=1,N,2
      I1=IABS(I1*K1+1)
      I2=IABS((I2*K2+K3)*K2+K3)
      U1=FLOAT(I1)/DIV
      U2=FLOAT(I2)/DIV
      X(I)=U1
      IF(I.EQ.N)GOTO100
      X(I+1)=U2
  100 CONTINUE
C
      ISEED=I1
      JSEED=I2
      RETURN
      END
```

7. Appendix B.

Listing of FORTRAN Subroutines for Second Generator:

POLY2
SET1
SET2
UNIRAN*
NORRAN
EQUAL
SORTM
SORTP

* For a listing of UNIRAN, refer to Appendix A.

```
      SUBROUTINE POLY2(M,K,NROW,JSW,BETA,BSW,BLEFT,BRIGHT,CSW,CLEFT,
     + CRIGHT,ISW,MSW,EPS,RSW,RESID,ISEED,JSEED,PRBMAT,SUMRES,KKM1,M2,
     + LOC)
C
C*********************************************************************
C
C     DESCRIPTION:
C
C     THIS SUBROUTINE GENERATES DISCRETE POLYNOMIAL APPROXIMATION
C     PROBLEMS WHICH CAN BE USED IN THE TESTING AND EVALUATION OF
C     ALGORITHMS FOR L1 (LEAST ABSOLUTE DEVIATION) CURVE FITTING.
C     THE USER CAN SPECIFY THE FOLLOWING CHARACTERISTICS OF THE
C     GENERATED DATA SETS:
C
C          * PROBLEM SIZE
C          * THE UNIQUE SOLUTION VECTOR TO THE GENERATED L1 PROBLEM
C          * THE REAL INTERVAL OVER WHICH THE APPROXIMATION IS DEFINED
C          * THE DISTRIBUTION OF THE OBSERVATION POINTS WITHIN THE
C            INTERVAL
C          * THE DISTRIBUTION OF THE RESIDUALS
C
C     THE POLYNOMIAL APPROXIMATION PROBLEM CONSIDERED HERE IS TO
C     APPROXIMATE (IN THE L1 NORM) A DISCRETE SET OF DATA VALUES Y
C     BY A POLYNOMIAL OF SPECIFIED DEGREE OVER SOME INTERVAL. FOR
C     I=1,KKM1  X(I) IS AN OBSERVATION POINT IN THE INTERVAL AND
C     Y(I) IS THE DATA VALUE CORRESPONDING TO X(I). THE VALUES FOR
C     X(I) AND Y(I) ARE SELECTED IN SUCH A MANNER TO GUARANTEE OPT-
C     IMALITY OF THE SOLUTION VECTOR BETA IN THE GENERATED PROBLEM.
C
C     FOR CONVENIENCE, THE Y-VALUES AND THE DESIGN MATRIX CONTAINING
C     THE APPROPRIATE POWERS OF THE OBSERVATION POINTS X(I) ARE BOTH
C     RETURNED IN A SINGLE MATRIX 'PRBMAT'. THE FIRST COLUMN OF PRBMAT
C     CONTAINS THE Y-VALUES AND THE REMAINING COLUMNS CONTAIN THE DESIGN
C     MATRIX.
C
C     NOTE:
C
C          * A CONSTANT TERM IS INCLUDED IN THE APPROXIMATING POLYNOMIAL
C
C          * THE OBSERVATION POINTS ARE ARRANGED IN INCREASING ORDER, BY
C            ROW, WITHIN THE GENERATED PROBLEM MATRIX
C
C          * THE PROCEDURE GIVEN HERE GENERATES AN L1 APPROXIMATION
C            PROBLEM HAVING A KNOWN AND UNIQUE SOLUTION VECTOR BETA.
C            TWO METHODS OF GUARANTEEING THESE PROPERTIES ARE PROVIDED
C            IN SUBROUTINES SET1 AND SET2 WHICH GENERATE THE OBSERVA-
C            TION POINTS FOR THE INACTIVE CONSTRAINTS. THE METHOD USED
C            IN SUBROUTINE SET2 UTILIZES A RANDOM NUMBER GENERATOR AND
C            THUS CAN CREATE DIFFERENT L1 PROBLEMS IN SUCCESSIVE RUNS
C            HAVING THE SAME INPUT SETTINGS, WHEREAS THE SAME PROBLEM
C            WILL BE REPRODUCED WHEN USING SUBROUTINE SET1.
```

```
C
C*********************************************************************
C
C     INPUT ARGUMENTS:
C
C          M ----- DEGREE OF THE APPROXIMATING POLYNOMIAL
C          K ----- INTEGER USED IN DETERMINING THE NUMBER OF INACTIVE
C                  CONSTRAINTS TO THE PROBLEM
C          NROW -- MAXIMUM NUMBER OF OBSERVATION POINTS
C          JSW --- INPUT SWITCH USED IN THE SELECTION OF THE SOLUTION
C                  VECTOR BETA, IF
C                   = 1 PRODUCES A SOLUTION VECTOR WITH ELEMENTS DRAWN
C                       FROM A UNIFORM DISTRIBUTION
C                   = 0 USES THE SOLUTION VECTOR SPECIFIED EXPLICITLY BY
C                       THE USER
C          BETA -- USER SPECIFIED SOLUTION VECTOR TO THE GENERATED PROB-
C                  LEM (IF JSW=0) OF DIMENSION M+1
C          BSW --- INPUT SWITCH USED TO DEFINE THE INTERVAL FROM WHICH
C                  ELEMENTS OF THE BETA VECTOR ARE SELECTED (WHEN JSW=1)
C                  IF
C                   = 1 USES ENDPOINTS SUPPLIED BY USER TO DEFINE THE
C                       INTERVAL
C                   = 0 SELECTS ENDPOINTS BY DEFAULT SETTINGS
C          BLEFT - LEFT ENDPOINT OF THE INTERVAL USED IN SELECTING
C                  ELEMENTS OF THE SOLUTION VECTOR BETA (IF JSW=BSW=1)
C          BRIGHT- RIGHT ENDPOINT OF THE INTERVAL USED IN SELECTING
C                  ELEMENTS OF THE SOLUTION VECTOR BETA (IF JSW=BSW=1)
C          CSW --- INPUT SWITCH USED TO DEFINE THE INTERVAL FROM WHICH
C                  OBSERVATION POINTS ARE SELECTED, IF
C                   = 1 USES ENDPOINTS SUPPLIED BY USER TO DEFINE THE
C                       INTERVAL
C                   = 0 SELECTS ENDPOINTS BY DEFAULT SETTINGS
C          CLEFT - LEFT ENDPOINT OF THE INTERVAL USED IN SELECTING OB-
C                  SERVATION POINTS (IF CSW=1)
C          CRIGHT- RIGHT ENDPOINT OF THE INTERVAL USED IN SELECTING OB-
C                  SERVATION POINTS (IF CSW=1)
C          ISW --- INPUT SWITCH WHICH DEFINES THE METHOD OF GENERATING
C                  OBSERVATION POINTS FOR THE INACTIVE CONSTRAINTS, IF
C                   = 1 PRODUCES EQUIDISTANT OBSERVATION POINTS BY
C                       SET1
C                   = 0 PRODUCES NON-EQUIDISTANT OBSERVATION POINTS BY
C                       SET2
C          MSW --- INPUT SWITCH WHICH SPECIFIES DISTRIBUTION OF
C                  OBSERVATION POINTS FOR THE ACTIVE CONSTRAINTS, IF
C                   = 1 PRODUCES OBSERVATION POINTS DRAWN FROM A
C                       UNIFORM DISTRIBUTION
C                   = 0 PRODUCES OBSERVATION POINTS EQUALLY SPACED
C                       OVER THE INTERVAL (THOUGH POSSIBLY PERTURBED
C                       BY A RANDOM AMOUNT IF EPS > 0)
C          EPS --- FRACTION OF THE SUBINTERVAL LENGTH USED IN PER-
C                  TURBING EQUALLY SPACED OBSERVATION POINTS FOR
C                  THE ACTIVE CONSTRAINTS (IF MSW=0)
C          RSW --- INPUT SWITCH WHICH SPECIFIES THE DISTRIBUTION
C                  OF THE (NONZERO) RESIDUALS ASSOCIATED WITH THE IN-
C                  ACTIVE CONSTRAINTS, IF
C                   = 1 PRODUCES RESIDUALS DRAWN FROM A UNIFORM DIS-
C                       TRIBUTION
C                   = 0 PRODUCES RESIDUALS DRAWN FROM A NORMAL DIS-
```

```
C                        TRIBUTION
C          RESID - STANDARD DEVIATION OF THE RESIDUAL DISTRIBUTION
C                  (WHEN RSW=0), OR ABSOLUTE VALUE OF THE UPPER
C                  AND LOWER LIMITS OF THE UNIFORM RESIDUAL DIS-
C                  TRIBUTION (WHEN RSW=1)
C          ISEED - FIRST OF TWO INITIAL SEEDS FOR THE RANDOM
C                  NUMBER GENERATORS
C          JSEED - SECOND OF TWO INITIAL SEEDS FOR THE RANDOM
C                  NUMBER GENERATORS
C
C     NOTE:
C
C          * THE NUMBER OF ACTIVE CONSTRAINTS IS EQUAL TO M+1; THE
C            NUMBER OF INACTIVE CONSTRAINTS IS EQUAL TO 2**(M+K+1)
C            FOR M .GE. 1 AND K .GE. 0
C
C          * THE DEFAULT VALUE FOR EACH OF THE SWITCHES BSW,CSW,ISW,
C            JSW,MSW,RSW IS ZERO
C
C          * WHEN EPS=0, NO PERTURBATION OF EQUALLY SPACED OBSERVA-
C            TION POINTS IS PERFORMED (MSW=0). OTHERWISE, A REASONABLE
C            RANGE IS  0 < EPS < 0.5
C
C      OUTPUT ARGUMENTS:
C
C          BETA -- SOLUTION VECTOR TO THE GENERATED PROBLEM OF
C                  DIMENSION M1
C          PRBMAT- THE GENERATED PROBLEM MATRIX WITH KKM1 ROWS
C                  AND M2 COLUMNS
C          SUMRES- SUM OF THE ABSOLUTE VALUES OF THE RESIDUALS IN THE
C                  OPTIMAL L1 SOLUTION
C          KKM1 -- NUMBER OF ROWS IN PRBMAT
C          M2 ---- NUMBER OF COLUMNS IN PRBMAT
C          LOC --- VECTOR OF DIMENSION M1 CONTAINING THE ROW LOCATIONS
C                  OF THE ACTIVE CONSTRAINTS IN PROBLEM MATRIX PRBMAT
C          ISEED - FIRST OF TWO RANDOM SEEDS AVAILABLE UPON RETURN
C          JSEED - SECOND OF TWO RANDOM SEEDS AVAILABLE UPON RETURN
C
C       RESTRICTIONS:
C
C          M MUST BE .GE. 1
C          K MUST BE .GE. 0
C          M+K MUST BE .LE. 8
C          NROW MUST BE .GE. M+1+2**(M+K+1)
C
C*********************************************************************
C
C      SUBPROGRAMS USED:
C
C          ABS ----- FORTRAN LIBRARY FUNCTION
C          UNIRAN -- SUBROUTINE TO GENERATE PSEUDO-RANDOM NUMBERS FROM
C                    A UNIFORM DISTRIBUTION OVER [0,1]
C          NORRAN -- SUBROUTINE TO GENERATE PSEUDO-RANDOM NUMBERS FROM
C                    A NORMAL DISTRIBUTION WITH MEAN ZERO AND STANDARD
C                    DEVIATION EQUAL TO ONE
C          SORTM --- SUBROUTINE USED TO SORT THE ELEMENTS OF A VECTOR
C                    INTO INCREASING ORDER. ALSO PROVIDES A MAPPING
C                    (NPOS) FROM OLD POSITION TO NEW (SORTED) POSITION
```

```
C           SET1 ---- SUBROUTINE USED TO CALCULATE EQUIDISTANT OBSER-
C                     VATION POINTS FOR THE INACTIVE CONSTRAINTS
C           SET2 ---- SUBROUTINE USED TO CALCULATE NON-EQUIDISTANT
C                     OBSERVATION POINTS FOR THE INACTIVE CONSTRAINTS
C           EQUAL --- SUBROUTINE TO PRODUCE OBSERVATION POINTS EQUALLY
C                     SPACED ON [0,1], THOUGH POSSIBLY PERTURBED BY A
C                     RANDOM AMOUNT IF EPS > 0
C
C       LANGUAGE: ANSI FORTRAN (1966)
C
C       REFERENCES:
C
C       K.L.HOFFMAN & D.R.SHIER,'A TEST PROBLEM GENERATOR FOR
C       DISCRETE LINEAR L1 APPROXIMATION PROBLEMS', TO APPEAR IN
C       TRANS. ON MATH. SOFTWARE
C       P.D.DOMICH & D.R.SHIER,'GENERATORS FOR DISCRETE POLYNOMIAL
C       L1 APPROXIMATION PROBLEMS'
C
C       WRITTEN BY:
C
C       PAUL DOMICH & DOUGLAS SHIER
C       CENTER FOR APPLIED MATHEMATICS
C       NATIONAL BUREAU OF STANDARDS
C       WASHINGTON D.C. 20234
C
C       JANUARY, 1979
C
C*********************************************************************
C                              * START OF SUBROUTINE POLY2 *
C
      INTEGER BSW,CSW,RSW,LOC(1)
      REAL PRBMAT(NROW,1),BETA(1)
      DIMENSION X(521),A(256,2),UU(521,10),NPOS(521)
      DOUBLE PRECISION HOLD,HOLD1,XPT,XPT1,R1,R2,DX,DX1,DSUM
      DATA BLFT,BRIT/-10.0,10.0/
      DATA CLFT,CRIT/-1.0,1.0/
C
C     INITIALIZATION: THE GENERATED PROBLEM WILL HAVE M1 ACTIVE
C     CONSTRAINTS,  KK INACTIVE CONSTRAINTS WITH POSITIVE RESIDUALS,
C     AND KK INACTIVE CONSTRAINTS WITH NEGATIVE RESIDUALS
C
      IOUT=6
      DSUM=0.0D0
      M1=M+1
      M2=M+2
      MK=M+K
      KK=2**MK
      KK2=KK+KK
      KM1=KK+M1
      KKM1=KK2+M1
C
C     CHECK THE CONSISTENCY OF THE INPUT PARAMETERS
C
      IF(M .LT. 1 .OR. K .LT. 0) GOTO 9990
      IF(MK .GT. 9) GOTO 9991
      IF(KKM1 .GT. NROW) GOTO 9991
      IF(BLEFT .GT. BRIGHT) GOTO 1001
1000  IF(CLEFT .GT. CRIGHT) GOTO 1002
```

```
      GOTO 1003
1001  XHOLD=BLEFT
      BLEFT=BRIGHT
      BRIGHT=XHOLD
      GOTO 1000
1002  XHOLD=CLEFT
      CLEFT=CRIGHT
      CRIGHT=XHOLD
1003  CONTINUE
C
C     DETERMINE INTERVALS USED IN PROBLEM BY SPECIFICATION OR DEFAULT
C     VALUES
C
      IF(BSW .NE. 1)BLEFT=BLFT
      IF(BSW .NE. 1)BRIGHT=BRIT
      IF(CSW .NE. 1)CLEFT=CLFT
      IF(CSW .NE. 1)CRIGHT=CRIT
      BINTER=BRIGHT-BLEFT
      CINTER=CRIGHT-CLEFT
C
C     DETERMINE THE SOLUTION VECTOR BY USER SPECIFICATION OR BY
C     RANDOM GENERATION
C
      IF(JSW .NE. 1) GOTO 2001
      CALL UNIRAN(M1,1,X,ISEED,JSEED)
      DO 2000 I=1,M1
      BETA(I)=BLEFT+X(I)*BINTER
2000  CONTINUE
2001  CONTINUE
C
C     CALCULATE THE M1 OBSERVATION POINTS CORRESPONDING TO THE
C     ACTIVE CONSTRAINTS. NOTE: ACTIVE CONSTRAINTS ARE DEFINED
C     TO HAVE IDENTICALLY ZERO RESIDUALS
C
      IF(MSW .EQ. 1)CALL UNIRAN(M1,1,X,ISEED,JSEED)
      IF(MSW .NE. 1)CALL EQUAL(M1,X,EPS,ISEED,JSEED)
      DO 3001 I=1,M1
      HOLD=BETA(M1)
      UU(I,2)=1.0
      DX=1.0D0
      XPT=CLEFT+X(I)*CINTER
      DO 3000 J=1,M
      N=M1-J
      HOLD=BETA(N)+XPT*HOLD
      DX=XPT*DX
      UU(I,J+2)=DX
3000  CONTINUE
      UU(I,1)=HOLD
3001  CONTINUE
C
C     CALCULATE THE OBSERVATION POINTS CORRESPONDING TO THE INACTIVE
C     CONSTRAINTS BY 'SET1' (EQUIDISTANT) OR 'SET2' (NON-EQUIDISTANT)
C
      IF(ISW .EQ. 1)CALL SET1(KK,A)
      IF(ISW .NE. 1)CALL SET2(KK,MK,A,ISEED,JSEED)
C
C     CALL RANDOM NUMBER GENERATOR FOR CALCULATION OF RESIDUALS
C     AND GENERATE THE DESIGN MATRIX FOR THE INACTIVE CONSTRAINTS
```

```
C
      IF(RSW .EQ. 1)CALL UNIRAN(KK2,1,X,ISEED,JSEED)
      IF(RSW .NE. 1)CALL NORRAN(KK2,1,X,ISEED,JSEED)
      DO 4001 I=M2,KM1
      N=I-M1
      IKK=I+KK
      NKK=N+KK
      HOLD=BETA(M1)
      HOLD1=HOLD
      UU(I,2)=1.0
      UU(IKK,2)=1.0
      DX=1.0D0
      DX1=1.0D0
      XPT=CLEFT+A(N,1)*CINTER
      XPT1=CLEFT+A(N,2)*CINTER
      DO 4000 J=1,M
      L=M1-J
      HOLD=BETA(L)+XPT*HOLD
      HOLD1=BETA(L)+XPT1*HOLD1
      DX=XPT*DX
      DX1=XPT1*DX1
      UU(I,J+2)=DX
      UU(IKK,J+2)=DX1
4000  CONTINUE
C
C     CALCULATE THE RESIDUALS AND ADD TO Y-VALUES
C
      R1=ABS(RESID*X(N))
      R2=ABS(RESID*X(NKK))
      HOLD=HOLD+R1
      HOLD1=HOLD1-R2
      DSUM=DSUM+R1+R2
      UU(I,1)=HOLD
      UU(IKK,1)=HOLD1
4001  CONTINUE
      SUMRES=DSUM
C
C     SORT THE OBSERVATION POINTS, REARRANGE UU AND STORE THE RESULT
C     IN PRBMAT
C
      DO 5000 I=1,KKM1
      X(I)=UU(I,3)
5000  CONTINUE
      CALL SORTM(X,KKM1,NPOS)
      DO 5002 I=1,KKM1
      KLOC=NPOS(I)
      DO 5001 J=1,M2
      PRBMAT(KLOC,J)=UU(I,J)
5001  CONTINUE
5002  CONTINUE
      DO 5003 I=1,M1
      LOC(I)=NPOS(I)
5003  CONTINUE
      GOTO 9999
9990  WRITE(IOUT,1) M,K
      GOTO 9999
9991  WRITE(IOUT,2) M,K
1     FORMAT(10X,19HINPUT VALUE OF M  =,I5,2X,7HOR K  =,I5,2X,3HIS ,
     +       7HILLEGAL)
2     FORMAT(10X,20HINPUT VALUES OF M  =,I5,2X,8HAND K  =,I5,2X,6HEXCEED
     +       ,18H AVAILABLE STORAGE)
9999  RETURN
      END
```

```
      SUBROUTINE SET1(KK,A)
C
C*********************************************************************
C
C     THIS SUBROUTINE CREATES A SET OF 2*KK REAL NUMBERS EQUALLY
C     SPACED ON [0,1].  THIS SET IS DIVIDED INTO TWO SUBSETS OF KK
C     ELEMENTS WHICH HAVE THE FOLLOWING PROPERTIES:  THE SUM OF THE
C     P-TH POWERS OF ELEMENTS IN THE FIRST SUBSET EQUALS THE SUM OF
C     THE P-TH POWERS OF ELEMENTS IN THE SECOND SUBSET, FOR VALUES
C     P = 0, 1, ..., LOG2(KK).
C
C     THE VALUES OF THE ELEMENTS RANGE BETWEEN 0 AND 1, WITH THE
C     SMALLEST VALUE IDENTICALLY ZERO AND THE LARGEST VALUE IDEN-
C     TICALLY ONE.
C
C     THE FIRST SUBSET IS RETURNED IN THE FIRST COLUMN OF ARRAY A
C     AND THE SECOND SUBSET IS RETURNED IN THE SECOND COLUMN.
C
C*********************************************************************
C
      DIMENSION A(256,2)
      DOUBLE PRECISION XD
C
C***** INITIALIZE THE FIRST TWO ELEMENTS OF EACH SUBSET
C
      A(1,1)=0.0
      A(1,2)=1.0
      A(2,1)=3.0
      A(2,2)=2.0
      N=2
C
C***** CALCULATE THE REMAINING ELEMENTS OF THE TWO SUBSETS
C
      IF(KK.EQ.2) GO TO 120
      DO 100 I=3,KK
      M=I-(2**(N-1))
      A(I,1)=A(M,2)+FLOAT(2**N)
      A(I,2)=A(M,1)+FLOAT(2**N)
      IF ((2**(N-1)) .EQ. M) N=N+1
100   CONTINUE
C
C***** STANDARDIZE THE VALUES TO LIE IN [0,1]
C
      XD=2*KK-1
      DO 110 I=1,KK
      A(I,1)=A(I,1)/XD
      A(I,2)=A(I,2)/XD
110   CONTINUE
120   RETURN
      END
```

```
      SUBROUTINE SET2(KK,MK,A,ISEED,JSEED)
C
C*********************************************************************
C
C     THIS SUBROUTINE CREATES A SET OF 2*KK REAL NUMBERS UNEQUALLY
C     SPACED ON [0,1].  THIS SET IS DIVIDED INTO TWO SUBSETS OF KK
C     ELEMENTS WHICH HAVE THE FOLLOWING PROPERTIES:  THE SUM OF THE
C     P-TH POWERS OF ELEMENTS IN THE FIRST SUBSET EQUALS THE SUM OF
C     THE P-TH POWERS OF ELEMENTS IN THE SECOND SUBSET, FOR VALUES
C     P = 0, 1, ...., LOG2(KK).
C
C     THE VALUES OF THE ELEMENTS RANGE BETWEEN 0 AND 1, WITH THE
C     SMALLEST VALUE IDENTICALLY ZERO AND THE LARGEST VALUE IDEN-
C     TICALLY ONE.
C
C     THE FIRST SUBSET IS RETURNED IN THE FIRST COLUMN OF ARRAY A
C     AND THE SECOND SUBSET IS RETURNED IN THE SECOND COLUMN.
C
C*********************************************************************
C
      DIMENSION A(256,2),X(9)
      DOUBLE PRECISION XHIGH
      A(1,1)=0.0
      MK1=MK+1
      M=1
      I=1
C
C***** SELECT ELEMENT A(1,2) RANDOMLY
C
      CALL UNIRAN(MK1,1,X,ISEED,JSEED)
      A(1,2)= X(I)
      XHIGH=X(I)
      I=I+1
C
C***** CALCULATE THE REMAINING ELEMENTS OF THE TWO SUBSETS
C
      DO 100 J=2,KK
      A(J,1)=A(M,2)+X(I)
      A(J,2)=A(M,1)+X(I)
      IF (XHIGH .LT. A(J,2)) XHIGH=A(J,2)
      IF (XHIGH .LT. A(J,1)) XHIGH=A(J,1)
      IF (FLOAT(J)/2.0-FLOAT(M) .GE. 0.1) GOTO 90
      M=0
      I=I+1
90    CONTINUE
      M=M+1
100   CONTINUE
C
C***** STANDARDIZE THE VALUES TO LIE IN [0,1]
C
      DO 110 I=1,KK
      A(I,1)=A(I,1)/XHIGH
      A(I,2)=A(I,2)/XHIGH
110   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NORRAN(N,ISTART,X,ISEED,JSEED)
C
C     PURPOSE--THIS SUBROUTINE GENERATES A RANDOM SAMPLE OF SIZE N
C              FROM THE THE NORMAL (GAUSSIAN)
C              DISTRIBUTION WITH MEAN = 0 AND STANDARD DEVIATION = 1.
C              THIS DISTRIBUTION IS DEFINED FOR ALL X AND HAS
C              THE PROBABILITY DENSITY FUNCTION
C              F(X) = (1/SQRT(2*PI))*EXP(-X*X/2).
C     INPUT  ARGUMENTS--N       = THE DESIRED INTEGER NUMBER
C                                 OF RANDOM NUMBERS TO BE
C                                 GENERATED.
C                     --ISTART = AN INTEGER FLAG CODE WHICH
C                                 (IF SET TO 0) WILL START THE
C                                 GENERATOR OVER AND HENCE
C                                 PRODUCE THE SAME RANDOM SAMPLE
C                                 OVER AND OVER AGAIN
C                                 UPON SUCCESSIVE CALLS TO
C                                 THIS SUBROUTINE WITHIN A RUN; OR
C                                 (IF SET TO SOME INTEGER
C                                 VALUE NOT EQUAL TO 0,
C                                 LIKE, SAY, 1) WILL ALLOW
C                                 THE GENERATOR TO CONTINUE
C                                 FROM WHERE IT STOPPED
C                                 AND HENCE PRODUCE DIFFERENT
C                                 RANDOM SAMPLES UPON
C                                 SUCCESSIVE CALLS TO
C                                 THIS SUBROUTINE WITHIN A RUN.
C     OUTPUT ARGUMENTS--X       = A SINGLE PRECISION VECTOR
C                                 (OF DIMENSION AT LEAST N)
C                                 INTO WHICH THE GENERATED
C                                 RANDOM SAMPLE WILL BE PLACED.
C     OUTPUT--A RANDOM SAMPLE OF SIZE N
C             FROM THE NORMAL DISTRIBUTION
C             WITH MEAN = 0 AND STANDARD DEVIATION = 1.
C     PRINTING--NONE UNLESS AN INPUT ARGUMENT ERROR CONDITION EXISTS.
C     RESTRICTIONS--THERE IS NO RESTRICTION ON THE MAXIMUM VALUE
C                   OF N FOR THIS SUBROUTINE.
C     OTHER DATAPAC   SUBROUTINES NEEDED--UNIRAN.
C     FORTRAN LIBRARY SUBROUTINES NEEDED--ALOG, SQRT, SIN, COS.
C     MODE OF INTERNAL OPERATIONS--SINGLE PRECISION.
C     LANGUAGE--ANSI FORTRAN.
C     METHOD--BOX-MULLER ALGORITHM.
C     REFERENCES--BOX AND MULLER  'A NOTE ON THE GENERATION
C                 OF RANDOM NORMAL DEVIATES', JOURNAL OF THE
C                 ASSOCIATION FOR COMPUTING MACHINERY, 1958,
C                 PAGES 610-611.
C               --TOCHER, THE ART OF SIMULATION,
C                 1963, PAGES 33-34.
C               --HAMMERSLEY AND HANDSCOMB, MONTE CARLO METHODS,
C                 1964, PAGE 39.
C               --JOHNSON AND KOTZ, CONTINUOUS UNIVARIATE
C                 DISTRIBUTIONS--1, 1970, PAGES 40-111.
C     WRITTEN BY--JAMES J. FILLIBEN
C                 STATISTICAL ENGINEERING LABORATORY (205.03)
C                 NATIONAL BUREAU OF STANDARDS
C                 WASHINGTON, D. C. 20234
C                 PHONE:  301-921-2315
```

```
C     ORIGINAL VERSION--JUNE       1972.
C     UPDATED          --SEPTEMBER 1975.
C     UPDATED          --NOVEMBER  1975.
C     UPDATED          --JULY      1976.
C
C-----------------------------------------------------------------------
C
      DIMENSION X(1)
      DIMENSION Y(2)
      DATA PI/3.14159265358979/
C
      IPR=6
C
C     CHECK THE INPUT ARGUMENTS FOR ERRORS
C
      IF(N.LT.1)GOTO50
      GOTO90
   50 WRITE(IPR, 5)
      WRITE(IPR,47)N
      RETURN
   90 CONTINUE
    5 FORMAT(1H , 91H***** FATAL ERROR--THE FIRST  INPUT ARGUMENT TO THE
     1 NORRAN SUBROUTINE IS NON-POSITIVE *****)
   47 FORMAT(1H , 35H***** THE VALUE OF THE ARGUMENT IS ,I8   ,6H *****)
C
C-----START POINT-----------------------------------------------------
C
C     GENERATE N UNIFORM (0,1) RANDOM NUMBERS;
C     THEN GENERATE 2 ADDITIONAL UNIFORM (0,1) RANDOM NUMBERS
C     (TO BE USED BELOW IN FORMING THE N-TH NORMAL
C     RANDOM NUMBER WHEN THE DESIRED SAMPLE SIZE N
C     HAPPENS TO BE ODD).
C
      CALL UNIRAN(N,ISTART,X,ISEED,JSEED)
      CALL UNIRAN(2,1,Y,ISEED,JSEED)
C
C     GENERATE N NORMAL RANDOM NUMBERS
C     USING THE BOX-MULLER METHOD.
C
      DO200I=1,N,2
      IP1=I+1
      U1=X(I)
      IF(I.EQ.N)GOTO210
      U2=X(IP1)
      GOTO220
  210 U2=Y(2)
  220 ARG1=-2.0*ALOG(U1)
      ARG2=2.0*PI*U2
      SQRT1=SQRT(ARG1)
      Z1=SQRT1*COS(ARG2)
      Z2=SQRT1*SIN(ARG2)
      X(I)=Z1
      IF(I.EQ.N)GOTO200
      X(IP1)=Z2
  200 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE EQUAL(M1,X,EPS,ISEED,JSEED)
C
C*********************************************************************
C
C     THIS SUBROUTINE GENERATES EQUALLY SPACED OBSERVATION POINTS
C     THOUGH POSSIBLY PERTURBED BY A RANDOM AMOUNT (IF EPS .NE. 0.0).
C     THE VALUES OF THESE OBSERVATION POINTS RANGE BETWEEN 0 AND
C     1, WITH THE SMALLEST VALUE IDENTICALLY ZERO AND THE LARGEST
C     VALUE IDENTICALLY ONE.  THE PERTURBATION APPLIED (WHEN EPS
C     .NE. 0.0) IS GENERATED FROM A UNIFORM PROBABILITY DISTRIBUTION.
C
C*********************************************************************
C
      REAL X(1)
      DOUBLE PRECISION H,EPSA
      M=M1-1
      CALL UNIRAN(M1,1,X,ISEED,JSEED)
      X(1)=0.0
      X(M1)=1.0
      IF(M .LT. 2) GOTO 25
      H=1.0/FLOAT(M)
      EPSA=H*EPS
      DO 1000 I=2,M
      X(I)=H*FLOAT(I-1)+EPSA*(2.0*X(I)-1.0)
1000  CONTINUE
25    RETURN
      END




      SUBROUTINE SORTM(X,N,NPOS)
C
C*********************************************************************
C
C     THIS SUBROUTINE SORTS THE INPUT SINGLE-PRECISON VECTOR X WITH
C     N ELEMENTS, AND RETURNS IN X THE ELEMENTS SORTED INTO ASCENDING
C     ORDER. THE VECTOR NPOS GIVES THE MAPPING FROM THE ORIGINAL
C     POSITION IN X TO THE NEW SORTED POSITION.  THAT IS, X(I) WILL
C     NOW CORRESPOND TO THE NPOS(I) - TH  SMALLEST ELEMENT OF THE
C     VECTOR X.
C
C*********************************************************************
C
      DIMENSION X(1),NPOS(1),XPOS(600)
      CALL SORTP(X,N,X,XPOS)
      DO 10 I=1,N
      NX=XPOS(I)+0.00005
      NPOS(NX)=I
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SORTP(X,N,Y,XPOS)
C
C     PURPOSE--THIS SUBROUTINE SORTS (IN ASCENDING ORDER)
C              THE N ELEMENTS OF THE SINGLE PRECISION VECTOR X,
C              PUTS THE RESULTING N SORTED VALUES INTO THE
C              SINGLE PRECISION VECTOR Y,
C              AND PUTS THE POSITION (IN THE ORIGINAL VECTOR X)
C              OF EACH OF THE SORTED VALUES.
C              INTO THE SINGLE PRECISION VECTOR XPOS.
C              THIS SUBROUTINE GIVES THE DATA ANALYST
C              NOT ONLY THE ABILITY TO DETERMINE
C              WHAT THE MIN AND MAX (FOR EXAMPLE)
C              OF THE DATA SET ARE, BUT ALSO
C              WHERE IN THE ORIGINAL DATA SET
C              THE MIN AND MAX OCCUR.
C              THIS IS ESPECIALLY USEFUL FOR
C              LARGE DATA SETS.
C     INPUT  ARGUMENTS--X       = THE SINGLE PRECISION VECTOR OF
C                                 OBSERVATIONS TO BE SORTED.
C                      --N       = THE INTEGER NUMBER OF OBSERVATIONS
C                                 IN THE VECTOR X.
C     OUTPUT ARGUMENTS--Y       = THE SINGLE PRECISION VECTOR
C                                 INTO WHICH THE SORTED DATA VALUES
C                                 FROM X WILL BE PLACED.
C                      --XPOS    = THE SINGLE PRECISION VECTOR
C                                 INTO WHICH THE POSITIONS
C                                 (IN THE ORIGINAL VECTOR X)
C                                 OF THE SORTED VALUES
C                                 WILL BE PLACED.
C     OUTPUT--THE SINGLE PRECISION VECTOR XS
C             CONTAINING THE SORTED
C             (IN ASCENDING ORDER) VALUES
C             OF THE SINGLE PRECISION VECTOR X, AND
C             THE SINGLE PRECISION VECTOR XPOS
C             CONTAINING THE POSITIONS
C             (IN THE ORIGINAL VECTOR X)
C             OF THE SORTED VALUES.
C     PRINTING--NONE UNLESS AN INPUT ARGUMENT ERROR CONDITION EXISTS.
C     RESTRICTIONS--THE DIMENSIONS OF THE VECTORS IL AND IU
C                   (DEFINED AND USED INTERNALLY WITHIN
C                   THIS SUBROUTINE) DICTATE THE MAXIMUM
C                   ALLOWABLE VALUE OF N FOR THIS SUBROUTINE.
C                   IF IL AND IU EACH HAVE DIMENSION K,
C                   THEN N MAY NOT EXCEED 2**(K+1) - 1.
C                   FOR THIS SUBROUTINE AS WRITTEN, THE DIMENSIONS
C                   OF IL AND IU HAVE BEEN SET TO 36,
C                   THUS THE MAXIMUM ALLOWABLE VALUE OF N IS
C                   APPROXIMATELY 137 BILLION.
C                   SINCE THIS EXCEEDS THE MAXIMUM ALLOWABLE
C                   VALUE FOR AN INTEGER VARIABLE IN MANY COMPUTERS,
C                   AND SINCE A SORT OF 137 BILLION ELEMENTS
C                   IS PRESENTLY IMPRACTICAL AND UNLIKELY,
C                   THEN THERE IS NO PRACTICAL RESTRICTION
C                   ON THE MAXIMUM VALUE OF N FOR THIS SUBROUTINE.
C                   (IN LIGHT OF THE ABOVE, NO CHECK OF THE
C                   UPPER LIMIT OF N HAS BEEN INCORPORATED
C                   INTO THIS SUBROUTINE.)
```

```
C     OTHER DATAPAC   SUBROUTINES NEEDED--NONE.
C     FORTRAN LIBRARY SUBROUTINES NEEDED--NONE.
C     MODE OF INTERNAL OPERATIONS--SINGLE PRECISION.
C     LANGUAGE--ANSI FORTRAN.
C     COMMENT--THE SMALLEST ELEMENT OF THE VECTOR X
C              WILL BE PLACED IN THE FIRST POSITION
C              OF THE VECTOR Y.
C              THE SECOND SMALLEST ELEMENT IN THE VECTOR X
C              WILL BE PLACED IN THE SECOND POSITION
C              OF THE VECTOR Y.
C              ETC.
C     COMMENT--THE POSITION (1 THROUGH N) IN X
C              OF THE SMALLEST ELEMENT IN X
C              WILL BE PLACED IN THE FIRST POSITION
C              OF THE VECTOR XPOS.
C              THE POSITION (1 THROUGH N) IN X
C              OF THE SECOND SMALLEST ELEMENT IN X
C              WILL BE PLACED IN THE SECOND POSITION
C              OF THE VECTOR XPOS.
C              ETC.
C              ALTHOUGH THESE POSITIONS ARE NECESSARILY
C              INTEGRAL VALUES FROM 1 TO N. IT IS TO BE
C              NOTED THAT THEY ARE OUTPUTED AS SINGLE
C              PRECISION INTEGERS IN THE SINGLE PRECISION
C              VECTOR XPOS.
C              XPOS IS SINGLE PRECISION SO AS TO BE
C              CONSISTENT WITH THE FACT THAT ALL
C              VECTOR ARGUMENTS IN ALL OTHER
C              DATAPAC SUBROUTINES ARE SINGLE PRECISION.
C     COMMENT--THE INPUT VECTOR X REMAINS UNALTERED.
C     COMMENT--IF THE ANALYST DESIRES A SORT 'IN PLACE',
C              THIS IS DONE BY HAVING THE SAME
C              OUTPUT VECTOR AS INPUT VECTOR IN THE CALLING SEQUENCE.
C              THUS, FOR EXAMPLE, THE CALLING SEQUENCE
C              CALL SORTP(X,N,X,XPOS)
C              IS ALLOWABLE AND WILL RESULT IN
C              THE DESIRED 'IN-PLACE' SORT.
C     COMMENT--THE SORTING ALGORTHM USED HEREIN
C              IS THE BINARY SORT.
C              THIS ALGORTHIM IS EXTREMELY FAST AS THE
C              FOLLOWING TIME TRIALS INDICATE.
C              THESE TIME TRIALS WERE CARRIED OUT ON THE
C              UNIVAC 1108 EXEC 8 SYSTEM AT NBS
C              IN AUGUST OF 1974.
C              BY WAY OF COMPARISON, THE TIME TRIAL VALUES
C              FOR THE EASY-TO-PROGRAM BUT EXTREMELY
C              INEFFICIENT BUBBLE SORT ALGORITHM HAVE
C              ALSO BEEN INCLUDED--
C              NUMBER OF RANDOM        BINARY SORT       BUBBLE SORT
C               NUMBERS SORTED
C                N = 10                  .002 SEC          .002 SEC
C                N = 100                 .011 SEC          .045 SEC
C                N = 1000                .141 SEC         4.332 SEC
C                N = 3000                .476 SEC        37.683 SEC
C                N = 10000              1.887 SEC      NOT COMPUTED
C     REFERENCES--CACM MARCH 1969, PAGE 186 (BINARY SORT ALGORITHM
C                 BY RICHARD C. SINGLETON).
C               --CACM JANUARY 1970, PAGE 54.
```

```
C                --CACM OCTOBER 1970, PAGE 624.
C                --JACM JANUARY 1961, PAGE 41.
C      WRITTEN BY--JAMES J. FILLIBEN
C                  STATISTICAL ENGINEERING LABORATORY (205.03)
C                  NATIONAL BUREAU OF STANDARDS
C                  WASHINGTON, D. C. 20234
C                  PHONE--301-921-2315
C      ORIGINAL VERSION--JUNE       1972.
C      UPDATED          --NOVEMBER  1975.
C
C-----------------------------------------------------------------------
C
      DIMENSION X(1),Y(1),XPOS(1)
      DIMENSION IU(36),IL(36)
C
C     CHECK THE INPUT ARGUMENTS FOR ERRORS
C
      IPR=6
      IF(N.LT.1)GOTO50
      IF(N.EQ.1)GOTO55
      HOLD=X(1)
      DO60I=2,N
      IF(X(I).NE.HOLD)GOTO90
   60 CONTINUE
      WRITE(IPR, 9)HOLD
      DO61I=1,N
      Y(I)=X(I)
      XPOS(I)=I
   61 CONTINUE
      RETURN
   50 WRITE(IPR,15)
      WRITE(IPR,47)N
      RETURN
   55 WRITE(IPR,18)
      Y(1)=X(1)
      XPOS(1)=1.0
      RETURN
   90 CONTINUE
    9 FORMAT(1H ,108H***** NON-FATAL DIAGNOSTIC--THE FIRST  INPUT ARGUME
     1NT (A VECTOR) TO THE SORTP  SUBROUTINE HAS ALL ELEMENTS = ,E15.8,6
     1H *****)
   15 FORMAT(1H , 91H***** FATAL ERROR--THE SECOND INPUT ARGUMENT TO THE
     1 SORTP  SUBROUTINE IS NON-POSITIVE *****)
   18 FORMAT(1H ,100H***** NON-FATAL DIAGNOSTIC--THE SECOND INPUT ARGUME
     1NT TO THE SORTP  SUBROUTINE HAS THE VALUE 1 *****)
   47 FORMAT(1H , 35H***** THE VALUE OF THE ARGUMENT IS ,I8   ,6H *****)
C
C-----START POINT-----------------------------------------------------
C
C     COPY THE VECTOR X INTO THE VECTOR Y
      DO100I=1,N
      Y(I)=X(I)
  100 CONTINUE
C
C     DEFINE THE XPOS (POSITION) VECTOR.  BEFORE SORTING, THIS WILL
C     BE A VECTOR WHOSE I-TH ELEMENT IS EQUAL TO I.
C
      DO150I=1,N
```

```
      XPOS(I)=I
  150 CONTINUE
C
C     CHECK TO SEE IF THE INPUT VECTOR IS ALREADY SORTED
C
      NM1=N-1
      DO200I=1,NM1
      IP1=I+1
      IF(Y(I).LE.Y(IP1))GOTO200
      GOTO250
  200 CONTINUE
      RETURN
  250 M=1
      I=1
      J=N
  305 IF(I.GE.J)GOTO370
  310 K=I
      MID=(I+J)/2
      AMED=Y(MID)
      BMED=XPOS(MID)
      IF(Y(I).LE.AMED)GOTO320
      Y(MID)=Y(I)
      XPOS(MID)=XPOS(I)
      Y(I)=AMED
      XPOS(I)=BMED
      AMED=Y(MID)
      BMED=XPOS(MID)
  320 L=J
      IF(Y(J).GE.AMED)GOTO340
      Y(MID)=Y(J)
      XPOS(MID)=XPOS(J)
      Y(J)=AMED
      XPOS(J)=BMED
      AMED=Y(MID)
      BMED=XPOS(MID)
      IF(Y(I).LE.AMED)GOTO340
      Y(MID)=Y(I)
      XPOS(MID)=XPOS(I)
      Y(I)=AMED
      XPOS(I)=BMED
      AMED=Y(MID)
      BMED=XPOS(MID)
      GOTO340
  330 Y(L)=Y(K)
      XPOS(L)=XPOS(K)
      Y(K)=TT
      XPOS(K)=ITT
  340 L=L-1
      IF(Y(L).GT.AMED)GOTO340
      TT=Y(L)
      ITT=XPOS(L)
  350 K=K+1
      IF(Y(K).LT.AMED)GOTO350
      IF(K.LE.L)GOTO330
      LMI=L-I
      JMK=J-K
      IF(LMI.LE.JMK)GOTO360
      IL(M)=I
```

```
      IU(M)=L
      I=K
      M=M+1
      GOTO380
360   IL(M)=K
      IU(M)=J
      J=L
      M=M+1
      GOTO380
370   M=M-1
      IF(M.EQ.0)RETURN
      I=IL(M)
      J=IU(M)
380   JMI=J-I
      IF(JMI.GE.11)GOTO310
      IF(I.EQ.1)GOTO305
      I=I-1
390   I=I+1
      IF(I.EQ.J)GOTO370
      AMED=Y(I+1)
      BMED=XPOS(I+1)
      IF(Y(I).LE.AMED)GOTO390
      K=I
395   Y(K+1)=Y(K)
      XPOS(K+1)=XPOS(K)
      K=K-1
      IF(AMED.LT.Y(K))GOTO395
      Y(K+1)=AMED
      XPOS(K+1)=BMED
      GOTO390
      END
```