JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematical Sciences
Vol. 73B, No. 2, April–June 1969

An Evaluation of Linear Least Squares Computer Programs

Roy H. Wampler

Institute for Basic Standards, National Bureau of Standards, Washington, D.C. 20234

(January 8, 1969)

Two linear least squares test problems, both fifth degree polynomials, have been run on more than twenty different computer programs in order to assess their numerical accuracy. Among the programs tested were representatives from various statistical packages as well as some from the SHARE library. Essentially five different algorithms were used in the various programs to obtain the coefficients of the least squares fits. The tests were run on several different computers, in double precision as well as single precision. By comparing the coefficients reported, it was found that those programs using orthogonal Householder transformations or Gram-Schmidt orthonormalization were much more accurate than those using elimination algorithms. Programs using orthogonal polynomials (suitable only for polynomial fits) also proved to be superior to those using elimination algorithms. One program, using congruential methods and integer arithmetic, obtained exact solutions. In a number of programs, the coefficients reported in one test problem were sometimes completely erroneous, containing not even one correct significant digit.

Key words: Computer programs; Gram-Schmidt orthogonalization; Householder transformations; least squares; linear equations; orthogonalization; orthogonal polynomials; regression; rounding error; stepwise regression.

1. Introduction

Since the time when the electronic computer began to supplant the desk calculator as the chief tool for solving linear least squares problems, numerous least squares computer programs have been written. These programs have utilized a variety of computational algorithms. Because least squares problems are by their very nature frequently ill-conditioned, the numerical accuracy achieved by a least squares program strongly depends upon the choice of the algorithm. Many programs have been written which use methods appropriate for desk calculators but inappropriate for computers. Anscombe [1][1] has aptly remarked: "Textbooks of statistical method display a wonderful unanimity in recommending computational procedures that are suited to desk calculators but are perilous for computers. Only with some determination can the statistician break himself of bad habits and become adequately informed about round-off error."

The present study was undertaken to assess the numerical accuracy of representative least squares programs from a variety of sources. Two test problems, both fifth degree polynomials, have been run on more than twenty different programs. Included in the study were programs from the BMD Biomedical Computer Programs collection, the C-E-I-R Multi-Access Computing Services library, the IBM SHARE library, the IBM System/360 Scientific Subroutine Package, the Univac MATH-PACK and STAT-PACK collections, and the Project MAC 7094 disk files. A detailed listing of the sources of the programs is given in appendix A, together with a brief description of each program.

For a number of programs, the test problems were run in double precision as well as in single precision. This, of course, necessitated certain changes in the original programs.

The programs included in this study used essentially five different algorithms: orthogonal

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

Householder transformations, Gram-Schmidt orthonormalization, orthogonal polynomials, Gaussian or Jordan elimination, and a congruential method with computations in integer arithmetic.

Previous studies appraising linear least squares programs and comparing the results of different algorithms have been made by Cameron [9], Freund [20], Bright and Dawkins [7], Zellner and Thornber [46], Longley [29], and Jordan [27]. The present study differs from the earlier ones mainly by including a larger selection of widely used and easily accessible programs.

The linear least squares problem may be briefly stated as follows: One has n observations or measurements of a "dependent" variable y which are statistically independent with common variance σ^2 whose expected values are given by a linear function of the corresponding values of k "independent" variables, $x_1, x_2, \ldots, x_k$, $k \leq n$. In matrix notation we say that the n observations have expected values $E(Y) = X\beta$, where Y is an $n \times 1$ vector, X is an $n \times k$ matrix, and β is a $k \times 1$ vector of unknown coefficients. Assuming that X is of rank k, the least squares estimates of the coefficients are given by $\hat{\beta} = (X'X)^{-1}X'Y$. Other quantities of interest are $\hat{Y} = X\hat{\beta}$, the vector of predicted values; $\delta = Y - \hat{Y}$, the vector of residuals; and $s^2 = \frac{1}{n-k}(Y-\hat{Y})'(Y-\hat{Y})$, an estimate of the variance σ^2.

In running certain programs, modifications were occasionally made to input and output formats. Other changes were made in five of the programs using elimination algorithms because the original versions of these programs failed to give solutions to the fifth degree polynomial problems. In particular, features that may have been intended to prevent execution of computations subject to excessive rounding error were sometimes bypassed. Details of these changes, and some remarks on the effectiveness of the features which were bypassed, will be given in section 8.

Four computers were used: the GE 235, the IBM 7094, and the Univac 1107 and 1108. The 1108 which was used is located at the National Bureau of Standards, and the 7094 which was chiefly used is located at Harry Diamond Laboratories, Washington, D. C. The programs run on the 235, the 1107 and the Project MAC 7094 utilized consoles at the National Bureau of Standards connected to computers at other locations.

2. The Test Problems

The two main test problems which were used throughout this investigation are identified as $Y1$ and $Y2$. Both were fifth degree polynomials, with the values of x being the integers 0, 1, 2, . . ., 20. The "observations," $Y1$ and $Y2$, were calculated from the following equations:

$$Y1\colon\ y = 1 + x + x^2 + x^3 + x^4 + x^5,\ x = 0(1)20,$$

$$Y2\colon\ y = 1 + 0.1x + 0.01x^2 + 0.001x^3 + 0.0001x^4 + 0.00001x^5,\ x = 0(1)20.$$

Thus the values of $Y1$ were integers having from one to seven digits, and those of $Y2$ were five-decimal numbers ranging from 1.00000 to 63.00000.

If the least squares solutions were computed with no rounding error, one would obtain

$$\hat{\beta}(Y1) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \hat{\beta}(Y2) = \begin{bmatrix} 1. \\ 0.1 \\ .01 \\ .001 \\ .0001 \\ .00001 \end{bmatrix},$$

and for both problems the residual standard deviation would be zero.

For some programs the input required was the 21 values of x and y. Some programs required, in addition, the powers x^2, x^3, x^4, and x^5 to be entered as input. Other programs required as input the 6 by 6 matrix $X'X$ and the 6 by 1 vector $X'Y$. It should be noted that the elements of $X'X$ are integers having from 2 to 14 digits, the elements of $X'Y$ for $Y1$ are 8- to 14-digit integers, and the elements of $X'Y$ for $Y2$ are 5-decimal numbers having up to 13 significant digits. The input is listed in table 9.

The two test problems, $Y1$ and $Y2$, were chosen because they are so highly ill-conditioned that some programs fail to obtain correct solutions while other programs succeed in obtaining reasonably accurate solutions. Polynomial problems were chosen because polynomial fitting is an important type of linear least squares problem which occurs frequently in practice.

The ill-conditioning of the two test problems can be described more explicitly. One measure of the condition of a matrix A is the P-condition, defined as

$$P(A)=\left|\frac{\lambda}{\mu}\right|$$

where λ is the numerically largest eigenvalue of A and μ is the numerically smallest eigenvalue of A. (See Newman [34, p. 240]).

For $A=X'X$, the 6×6 matrix associated with $Y1$ and $Y2$, the P-condition is 4.095×10^{13}. In this respect, it is similar to the Hilbert matrix of order 10, whose P-condition is 1.603×10^{13} (see Fettis and Caslin [17]). The P-condition of the Hilbert matrix of order 11 is 5.231×10^{14}. The relation between the Hilbert matrix and the matrix $X'X$ which arises in a polynomial fit is discussed in Forsythe [18].

Most of the programs which were tested obtained more accurate solutions for $Y2$ than for $Y1$. If we let A denote the 7×7 matrix

$$A=\begin{bmatrix} X'X & X'Y \\ Y'X & 0 \end{bmatrix}$$

we find that for $Y2$, $P(A)=4.095\times 10^{13}$, whereas for $Y1$, $P(A)=6.829\times 10^{13}$, indicating that the system involving $Y1$ is more ill-conditioned than that involving $Y2$.

The test problem used by Longley [29] was also highly ill-conditioned. For the 7×7 matrix $X'X$ of his problem, the P-condition is 2.361×10^{19}.

3. Summary of the Results

Tables 1 to 6 present a brief summary of the main results. A count, C_j, of the number of correct significant digits in each computed coefficient was obtained as follows:

Let β_j $(j=1, 2, \ldots, 6)$ denote the "true" value of the coefficient—that is, the value computed with no rounding error. Let $\hat{\beta}_j$ denote the value calculated by the computer. Then

$$C_j=\begin{cases} -\log_{10}\left|\dfrac{\beta_j-\hat{\beta}_j}{\beta_j}\right|, \text{ if } |\beta_j-\hat{\beta}_j|\neq 0 \text{ and } \beta_j\neq 0 \\ -\log_{10}|\beta_j-\hat{\beta}_j|, \text{ if } |\beta_j-\hat{\beta}_j|\neq 0 \text{ and } \beta_j=0 \\ D, \text{ the approximate number of decimal digits with which the machine computes, if } \beta_j-\hat{\beta}_j=0. \end{cases}$$

The above approach to counting the number of correct digits in a computed value has been used by Jordan [27] and others.

Tables 1 to 6, in the columns headed "Average Number of Correct Digits" report

$$C=\frac{1}{6}\sum_{j=1}^{6} C_j.$$

From the above definition, a negative count can occur. For example, if $\beta_j=1.0$, and $\hat{\beta}_j=136.0$, we get $C_j=-2.130$. This indicates that $\hat{\beta}_j$ is wrong by roughly two orders of magnitude.

TABLE 1. *Summary of programs run in single precision—8 digits*

Program	Computer	Algorithm [a]	Average number of correct digits		Rank		Date of run	
			Y1	Y2	Y1	Y2	Y1	Y2
ALSQ	1108	HT	4.098	5.368	3	5	10–18–67	10–18–67
BMD02R	1108	E	−0.106	1.981	12	14	12–13–67	11–17–67
BMD03R	7094	E	0.742	1.721	8	16	12–30–66	1– 3–67
BMD03R	1108	E	−0.123	2.287	13	12	12–18–67	12–18–67
DAM	7094	E	1.389	2.312	7	11	4–12–67	4–12–67
DAM	1108	E	−0.264	2.622	16	9	3– 5–68	3– 5–68
LINFIT (Miller)	7094	?	−2.756	−0.301	21	21	5–17–68	8–15–68
LSTSQ	1108	HT	4.528	5.840	1	3	5– 1–68	5– 6–68
MATH-PACK, ORTHLS	1108	OP	2.118	4.363	6	6	4–12–68	4–12–68
MPR3	7094	E	−0.140	1.856	14	15	5–16–67	5–18–67
OMNITAB (Invert)	7094	E	−0.607	1.460	17	18	12– 9–66	12– 9–66
OMNITAB (Invert)	1108	E	−0.907	1.224	19	19	2–29–68	2–29–68
OMNITAB (Ortho)	7094	GS	3.954	5.968	4	2	12– 5–66	12– 5–66
OMNITAB (Ortho)	1108	GS	4.137	5.464	2	4	10–18–67	10–18–67
ORTHO (no iteration)	1108	GS	−1.976	0.419	20	20	3– 7–68	3– 7–68
ORTHOL	1108	GS	3.593	6.197	5	1	9–10–68	9–10–68
POLRG	1108	E	−0.191	2.280	15	13	10– 7–68	10– 7–68
SPVMTX	1108	E	−0.658	1.527	18	17	11–14–67	11–14–67
STAT-PACK, GLH	1108	E	0.066	2.767	10½	7½	11– 7–67	11– 7–67
STAT-PACK, REBSOM	1108	E	0.066	2.767	10½	7½	11– 8–67	11– 9–67
STAT-PACK, RESTEM	1108	E	0.651	2.407	9	10	7– 2–68	7– 2–68
WRAP	7094	E	−5.300	−2.871	22	22	6–28–67	6–28–67

[a] E = Elimination method; GS = Gram-Schmidt orthonormalization; HT = Orthogonal Householder transformations; OP = Orthogonal polynomials.

TABLE 2. *Summary of programs run in single precision—9 digits*

Program	Computer	Algorithm [a]	Average number of correct digits		Rank		Date of run	
			Y1	Y2	Y1	Y2	Y1	Y2
LINFIT***	235	E	0.905	2.894	5	6	12– 1–67	12– 1–67
LSCF--***	235	E	0.308	2.483	7	7	12–28–66	12–28–66
LSFITW***	235	GS	4.102	6.354	1	1	1–25–67	1–25–67
POLFIT	235	OP	3.349	5.922	2	2	2–19–68	2–19–68
SIMEX-***	235	E	1.402	3.213	3	3	12–30–66	1– 5–67
STAT20***	235	E	0.612	2.920	6	5	11–30–67	11–30–67
STAT21***	235	E	1.169	3.183	4	4	1– 3–67	1– 3–67

[a] E = Elimination method; GS = Gram-Schmidt orthonormalization; OP = Orthogonal polynomials.

TABLE 3. *Summary of programs run in double precision—16 digits*

Program	Computer	Algorithm [a]	Average number of correct digits		Rank		Date of run	
			Y1	Y2	Y1	Y2	Y1	Y2
BMD05R	7094	E	6.953	6.230	2	2	1– 5–67	1– 5–67
DPVMTX	1107	E	7.882	9.959	1	1	1–23–67	1–23–67

[a] E = Elimination method.

TABLE 4. *Summary of programs run in double precision—18 digits*

Program	Computer	Algorithm [a]	Average number of correct digits		Rank		Date of run	
			Y1	Y2	Y1	Y2	Y1	Y2
ALSQ	1108	HT	12.667	15.322	4	4	10–19–67	1–29–68
BMD02R	1108	E	9.645	12.865	7	7	4–17–68	4–17–68
BMD05R	1108	E	9.368	11.791	9	10	9–10–68	9–10–68
DPVMTX	1108	E	9.744	13.484	6	6	2–27–68	2–27–68
LSTSQ	1108	HT	14.643	16.293	1	1	7–22–68	7–22–68
MATH-PACK, ORTHLS	1108	OP	12.098	14.461	5	5	10–16–68	10–16–68
ORTHO	1108	GS	13.188	15.514	3	3	1–29–68	1–29–68
ORTHO (no iteration)	1108	GS	7.963	10.354	11	11	3– 7–68	3– 7–68
ORTHOL	1108	GS	13.212	15.604	2	2	9–25–68	9–25–68
POLRG	1108	E	9.290	11.806	10	9	10– 7–68	10– 7–68
STAT-PACK, RESTEM	1108	E	9.494	12.019	8	8	7– 1–68	7– 1–68

[a] E = Elimination method; GS = Gram-Schmidt orthonormalization; HT = Orthogonal Householder transformations; OP = Orthogonal polynomials.

TABLE 5. *Summary of programs run in single precision (8 digits) with inner products accumulated in double precision (18 digits)*

Program	Computer	Algorithm [a]	Average number of correct digits		Rank		Date of run	
			Y1	Y2	Y1	Y2	Y1	Y2
ALSQ	1108	HT	3.506	6.530	3	1	10–18–68	10–18–68
LSTSQ	1108	HT	8.000	6.279	1	3	4–30–68	5– 6–68
ORTHO	1108	GS	3.904	6.459	2	2	10–21–68	10–21–68

[a] GS = Gram-Schmidt orthonormalization; HT = Orthogonal Householder transformations.

TABLE 6. *Summary of program run in multiple precision integer arithmetic*

Program	Computer	Algorithm [a]		Average number of correct digits		Date of run	
				Y1	Y2	Y1	Y2
SOLVER	1108	C	Rational form	∞	∞	7–16–68	7–16–68
			Floated form	18.000	17.347		

[a] C = Congruential method.

For two programs reported in table 1, BMD03R run on the 7094 and DAM run on the 7094, the count for several coefficients was made in a different manner. The BMD03R program printed the coefficients in a fixed-decimal format, with five decimals. The DAM program used a floating-point format with only three decimals printed. A coefficient printed as 0.00010, when the true coefficient was 0.0001, was given a count of 2, and 0.100E01, when the true coefficient was 1., was given a count of 3. In such cases the assigned count may have been too small, since the coefficients may have been calculated accurately to more digits than were printed. In running these two programs on the 1108, the output format was changed so that eight significant digits were printed.

Each of the tables (1 through 6) summarizes a set of results for a particular machine precision—8, 9, 16, 18, etc. digits. Within each table the various programs are ranked for each of the two test problems, with rank 1 denoting the best performance according to the count C.

Table 1 includes single precision (8-digit) programs run on two different computers, the 7094 and the 1108. It was felt that combining the results from two computers was justified in view of the similar performance of the four programs which were run on both computers. These four programs were BMD03R, DAM, OMNITAB (using INVERT), and OMNITAB (using ORTHO). The average number of correct digits for the two test problems from the 1108 agrees with the corresponding average from the 7094 to within 0.9 digits except in the case of $Y1$ run on DAM, where the difference is 1.653. This larger difference may possibly be attributable to modifications in the program. Furthermore, other test problems have been run on OMNITAB (using ORTHO) on both computers, and again the results were quite similar.

The symbols in the Algorithm column of the tables denote the following:

C Congruential method, integer arithmetic
E Elimination method
GS Gram-Schmidt orthonormalization
HT Orthogonal Householder transformations
OP Orthogonal polynomials.

From time to time at a given computer installation changes are made in hardware and software with the result that a particular job run on two different days may not produce identical numerical output. For this reason the date of the computer runs is included in the tables.

Details (individual coefficients and counts) supporting tables 1 through 6 are given in appendix B.

4. Programs Using Orthogonal Householder Transformations

LSTSQ is a program written by Peter A. Businger using orthogonal Householder transformations. This algorithm is described by Golub [21] and Businger and Golub [8]. The program applies a sequence of orthogonal transformations to the n by k least squares matrix X to obtain a decomposition $X=QR$, where R is upper triangular and $Q'Q=I_k$. A pivoting strategy is used so that at each step the column with the largest sum of squares is reduced next. Once an initial solution is obtained, the program iterates to obtain a (possibly) improved solution.

Of all the programs using floating-point arithmetic included in this study, LSTSQ appears to have given the best performance. In table 4, which reports the performance of eleven double precision programs, we see that LSTSQ ranked first for both $Y1$ and $Y2$. In table 1, which reports the performance of 22 single precision programs, we see that LSTSQ obtained rank 1 for $Y1$ and rank 3 for $Y2$. Ranks 1 and 2 for the $Y2$ problem were obtained by ORTHOL and OMNITAB (using ORTHO), two programs using Gram-Schmidt orthonormalization which will be discussed more fully in the next section. Table 5 reports the performance of three programs which used single precision arithmetic except for the accumulation of inner products, where double precision arithmetic was used. Here we see that LSTSQ ranked first for $Y1$ (having a perfect score of 8.000) and ranked third for $Y2$. In the two instances just mentioned where LSTSQ ranked third for $Y2$, we see that the difference separating it from the top-ranking program is small, being 0.357 in table 1 and 0.251 in table 5.

Golub and Businger recommend that all inner products be accumulated in double precision. By comparing tables 5 and 1 we see that when LSTSQ included this feature, the average counts increased from 4.528 to 8.000 for $Y1$ and from 5.840 to 6.279 for $Y2$. With *all* operations performed in double precision (see table 4), the counts increased to 14.643 and 16.293, respectively.

The other program using Householder transformations was ALSQ, written by G. W. Stewart, III. This program contains no pivoting and no iteration. In tables 1, 4, and 5 we see that ALSQ performed not quite as well as LSTSQ which included these features, except in one instance. In this one instance, $Y2$ in table 5, we note that its performance was slightly better than that of LSTSQ.

By examining tables 1 and 5, one may see the effect of accumulating inner products in double precision versus accumulating them in single precision. As one would expect, ALSQ did better in computing the coefficients of $Y2$ when the double precision accumulation was included. For $Y1$, surprisingly, we see that ALSQ lost accuracy with this feature included. A look at the details of the program revealed how this phenomenon occurred. After the matrix X has been decomposed to obtain QR (as described earlier), the coefficients are computed by back-substitution. The first coefficient to be computed is $\hat{\beta}_6$ (the coefficient for the fifth-degree term), and this is obtained from one arithmetic operation, a division. Correctly calculated to ten digits, this division is $\frac{21011.77901}{21011.77901}=1$. In the single precision version, the coefficient was calculated as

$$\hat{\beta}_6=\frac{21011.714}{21011.713}=1.0000000 \text{ (to 8 significant digits)} \tag{1}$$

whereas in the version with inner products accumulated in double precision the calculation was

$$\hat{\beta}_6=\frac{21011.761}{21011.753}=1.0000004 \text{ (to 8 significant digits).} \tag{2}$$

We note that in (1), both the numerator and denominator in question are farther from their true values than in (2), but they are closer to each other, so that $\hat{\beta}_6$ in (1) happens to be closer to the true value of $\hat{\beta}_6$. Subsequently, $\hat{\beta}_6$ enters into the calculations of the five other coefficients with the result that all the coefficients for $Y1$ from the single precision version are slightly more accurate than those from the version using double precision inner products.

5. Programs Using Gram-Schmidt Orthonormalization

ORTHO is a program written by Philip J. Walsh using a Gram-Schmidt orthonormalization process. This algorithm is described by Davis and Rabinowitz [13], [14], Davis [12], and Walsh [42]. ORTHO exists as a FORTRAN program, an ALGOL procedure, a BASIC program, and as a routine of the OMNITAB program (see Hilsenrath, et al., [23]), where it is called by the commands FIT and POLYFIT.

Starting with the $n\times k$ matrix X, the Gram-Schmidt process of ORTHO obtains $\varphi=XT'^{-1}$ and $\hat{\beta}=T'^{-1}\varphi'Y$, where T'^{-1} is upper triangular and $\varphi'\varphi=I_k$. This algorithm includes a feature of reorthonormalizing the vectors of φ, proceeding from a first approximation $\bar{\varphi}_j$ to a (usually) better approximation φ_j. From table 1 it is clear that this reorthonormalizing is vital to the algorithm, for ORTHO's good performance in handling $Y1$ and $Y2$ deteriorated when this iteration was omitted. For $Y1$, the count of correct digits dropped from 4.137 to -1.976, and for $Y2$ the drop was from 5.464 to 0.419. In table 4, also, we see that in double precision the omission of the iteration resulted in a loss of about five correct digits for both problems.

The LSFITW*** program, written in BASIC, listed in table 2 also uses the ORTHO algorithm. The computer used for running the programs of table 2 works with about one more decimal digit than the computers covered in table 1, so one would expect more accuracy from LSFITW*** than from OMNITAB (ORTHO). We find slight improvement for $Y2$, but no improvement for $Y1$. Of the seven programs reported in table 2, LSFITW*** ranked 1 on both problems. Note that there are no Householder transformation programs included in table 2.

The ORTHO program was also run in a version using single precision except for the accumulation of inner products, where double precision was used. In table 5 we see that there were three programs in this category, and for both problems ORTHO ranked second. Its performance on $Y2$ improved by about one digit compared to the performance of the ORTHO version using strictly single precision. On the $Y1$ problem, however, there was a slight loss in accuracy. Actually, three coefficients gained accuracy and three lost accuracy, with a net loss in the average count. (A similar loss which occurred with ALSQ was discussed in the previous section.) In ORTHO, the final calculation to obtain the coefficients $\hat{\beta}$ is the matrix multiplication $\hat{\beta}=(T')^{-1}\alpha$, where $(T')^{-1}$ is an upper triangular matrix such that $(T')^{-1}T^{-1}=(X'X)^{-1}$, and $\alpha=T^{-1}X'Y$. Nearly all of the nonzero elements of $(T')^{-1}$ and α were more accurately computed when inner products were accumulated in double precision than when this feature was omitted. In the three coefficients of $Y1$ which lost accuracy, an examination of the details showed that in the individual multiplications involved in the matrix multiplication, the version using double precision for inner products was always more accurate than the strictly single precision version. But in the final addition of the various products, where the terms have alternating signs, there was heavy cancellation and the errors combined in such a way that the $\hat{\beta}_j$'s from the single precision version happened to be closer to the true values of the coefficients than those computed with double precision accumulation of inner products.

ORTHOL is a program using a modification of the Davis-Rabinowitz algorithm written by James W. Longley and Roger A. Blau [30]. It differs from Walsh's ORTHO in two respects: (1) the iteration procedure includes the dependent variable as well as the independent variables, and (2) before any other operations are applied to the matrix X, from each element of each vector of X, the truncated mean of that vector is subtracted. (The "truncated mean" denotes the largest integer less than or equal to the mean if the mean is nonnegative, and the smallest integer greater than or equal to the mean if the mean is negative.) ORTHOL obtained the top rank for $Y2$ in single precision, but ranked fifth for $Y1$ (table 1). In double precision (table 4), it ranked second on both problems.

6. Programs Using Orthogonal Polynomials

Since the two test problems are both polynomial fits, we were able to test programs in which the algorithm used orthogonal polynomials. This method, described by Forsythe [18], is attractive because it generally requires many fewer operations than other methods.

Two such programs were included in this study. One was the UNIVAC 1108 MATH-PACK routine, ORTHLS (see Programmers Reference [40]). The other was POLFIT, an anonymous program written in BASIC.

In tables 1, 2, and 4 we see that the performance of the orthogonal polynomial programs is not as good as that of the Householder transformation and the Gram-Schmidt programs (with iteration), but the performance is better than that of any of the programs using elimination algorithms. This finding is in agreement with the results of Bright and Dawkins [7] who ran a number of polynomial test problems via two methods: matrix inversion using a Gauss-Jordan reduction, and orthogonal polynomials. In all cases they found the orthogonal polynomial method superior.

7. A Multiple Precision Integer Arithmetic Program Using Congruential Methods

Morris Newman, in his paper "Solving Equations Exactly" [35] described a congruential method for finding the exact solution of a system of linear equations $Ax=b$ where the elements of A and b are all integers. His FORTRAN program SOLVER will solve systems in which A is a square matrix at most 100 by 100 and the elements of A and b are numerically less than 10^{20}. This method is not at all sensitive to the condition of A, but it can be time-consuming for large systems.[2] The solution is printed in two versions: (1) $x=\left(\frac{1}{\det A}\right)z$, where z is a vector of integers and the determinant $\det A$ is an integer, and (2) x in floated double-precision format, accurate to about 17 digits on the 1108.

The two test problems, $Y1$ and $Y2$, were run on this program, as indicated in table 6. The input required was the matrices $X'X$ and $X'Y$. Since the elements of $X'Y$ for $Y2$ are not integers, it was necessary to multiply these numbers by 100,000 before obtaining the solution.

[2] The running time on the Univac 1108 for the solution of $Y1$ and $Y2$ was 11 seconds, including 5 seconds for compilation of the program. The six problems described in the latter part of this section, all having 6×6 systems, required 14 seconds, including 4 seconds for compilation. To solve a 20×20 system, the worst possible case requires about 30 seconds, and an "average" case takes less time. For a 40×40 system, an average case requires about one minute, and the worst possible case requires about six minutes. A "bad" 100×100 case might require 40 minutes or more running time.

Having a program which produces exact solutions, we can determine what will happen to the solution when we round the input, the elements of $X'X$ and $X'Y$. These elements for $Y1$ are integers having no more than 14 digits. Six additional problems were run in which the input was successively rounded to 13, 12, 11, 10, 9, and 8 significant digits. The effect of this rounding was to change the solution dramatically. In table 7, which gives the solutions to these six problems (rounded to 10 decimals), we see that "small" changes to the elements of $X'X$ and $X'Y$ produce "large" changes to the solution, $\hat{\beta}$.

At first glance, the fact that the coefficients calculated for the problem having input rounded to 13 significant digits agree with the coefficients obtained from unrounded input to only 4, 5, 6, or 8 digits seems quite surprising. But with a few simple calculations we can see why the agreement is no better than it is. First, referring to table 9, we note that there is only one 14-digit number in $X'X$, and since this ends with a zero, rounding to 13 significant digits leaves $X'X$ unaltered. In $X'Y$ only the last element has 14 digits. Here, 25,537,373,767,266 was rounded to 25,537,373,767,270. Let $B=(b_{ij})$, $i, j=1, \ldots, 6$, denote the inverse of $X'X$ and let $(q_j)=X'Y, j=1, \ldots, 6$. Consider $\hat{\beta}_1$, the first coefficient. We have

$$\hat{\beta}_1 = b_{11}q_1 + b_{12}q_2 + b_{13}q_3 + b_{14}q_4 + b_{15}q_5 + b_{16}q_6 = \sum_{j=1}^{5} b_{1j}q_j + b_{16}q_6.$$

The only quantity which is affected by changing from unrounded data to rounded data with 13 significant digits is q_6. Now

$b_{16} = -28{,}046{,}715{,}376{,}452{,}025{,}326{,}796{,}800/(\text{Det } X'X)$, where

Det $X'X = 1{,}677{,}193{,}579{,}511{,}831{,}114{,}542{,}448{,}640{,}000$. Since $q_6 = 25{,}537{,}373{,}767{,}266$ we have

$$b_{16}q_6 = -716{,}239{,}453{,}512{,}581{,}907{,}404{,}696{,}441{,}196{,}473{,}548{,}800/(\text{Det } X'X).$$

Also,

$$\sum_{j=1}^{5} b_{1j}q_j = 716{,}239{,}455{,}189{,}775{,}486{,}916{,}527{,}555{,}738{,}922{,}188{,}800/(\text{Det } X'X).$$

In combining the last two numbers, one positive and the other negative, we see that the first eight digits of the two numerators are identical, so that the numerator of $\hat{\beta}_1$ has eight fewer digits than has $\sum_{j=1}^{5} b_{1j}q_j$ or $b_{16}q_6$.

In solving for $\hat{\beta}_1'$ from input rounded to 13 significant digits, we have $q_6' = 25{,}537{,}373{,}767{,}270$ so that

$$b_{16}q_6' = -716{,}239{,}453{,}512{,}694{,}094{,}266{,}202{,}249{,}297{,}780{,}736{,}000/(\text{Det } X'X).$$

The numerator here differs from the numerator of $b_{16}q_6$ in the thirteenth significant digit, but after combining this with $\sum_{j=1}^{5} b_{1j}q_j$ and losing eight significant digits, we obtain

$$\begin{aligned}\hat{\beta}_1' &= \sum_{j=1}^{5} b_{1j}q_j + b_{16}q_6' \\ &= \frac{1{,}677{,}081{,}392{,}650{,}325{,}306{,}441{,}141{,}452{,}800}{1{,}677{,}193{,}579{,}511{,}831{,}114{,}542{,}448{,}640{,}000} \\ &= 0.99993\ 31104 \text{ (to 10 decimals).}\end{aligned}$$

There are similar losses of significant digits in calculating the other coefficients.

A rigorous presentation of the sensitivity of the solution of a system of equations $Ax=b$ with respect to variations in A and b is given in Wilkinson [43, p. 91] and Wilkinson [44, p. 189].

TABLE 7. *Exact solutions to approximate problems*

The column A(N) gives a count of the number of digits in the solution of $(X'X)\beta = X'Y$ for input rounded to N digits which are in agreement with the solution for unrounded input. The unrounded input (elements of $X'X$ and $X'Y$) consisted on integers having no more than 14 digits.

Solution for unrounded input	Solution for input rounded to 13 sig. digits	A(13)	Solution for input rounded to 12 sig. digits	A(12)
1.	0.99993 31104	4.175	0.99672 54481	2.485
1.	1.00015 06443	3.822	1.00718 28792	2.144
1.	0.99994 21662	4.238	0.99727 93936	2.565
1.	1.00000 79679	5.099	1.00037 12813	3.430
1.	0.99999 95470	6.344	0.99997 90427	4.679
1.	1.00000 00091	8.043	1.00000 04168	6.380
		Avg. = 5.287		Avg. = 3.614
	Solution for input rounded to 11 sig. digits	**A(11)**	**Solution for input rounded to 10 sig. digits**	**A(10)**
	1.06051 32874	1.218	0.91988 69708	1.096
	0.86927 95602	0.884	1.19460 48731	0.711
	1.04911 27057	1.309	0.92297 46106	1.113
	0.99333 63623	2.176	1.01080 85953	1.966
	1.00037 44589	3.427	0.99937 78148	3.206
	0.99999 25798	5.130	1.00001 25555	4.901
		Avg. = 2.357		Avg. = 2.166
	Solution for input rounded to 9 sig. digits	**A(9)**	**Solution for input rounded to 8 sig. digits**	**A(8)**
	3.70810 09327	−0.433	−24.56199 35653	−1.408
	−4.34362 06781	−0.728	52.60541 27451	−1.713
	2.92257 14823	−0.284	−17.89853 20445	−1.276
	0.74656 29295	0.596	3.52648 11096	−0.403
	1.01394 46989	1.856	0.85941 47174	0.852
	0.99972 81121	3.566	1.00276 62377	2.558
		Avg. = 0.762		Avg. = −0.232

8. Programs Using Elimination Algorithms

The majority of the programs tested in this investigation used some form of an elimination algorithm. Although this was the most popular method, it was the least successful. None of these programs performed as well as those using Householder's transformations, Gram-Schmidt orthonormalization (with iteration), or orthogonal polynomials.

Within this class of programs, there were several variations in the method of obtaining the least-squares coefficients. In some cases, the matrix $X'X$ was inverted, after which the inverse was postmultiplied by $X'Y$ to obtain $\hat{\beta} = (X'X)^{-1}X'Y$. In one program the matrix

$A = \begin{bmatrix} X'X & X'Y \\ 0 & 1 \end{bmatrix}$ was inverted. Here, the inverse is

$A^{-1} = \begin{bmatrix} (X'X)^{-1} & -\hat{\beta} \\ 0 & 1 \end{bmatrix}$. Another program inverted the matrix $Z'Z$

where the vectors of Z were obtained from the vectors of X by subtracting the mean of each vector from every element of that vector. A number of programs obtained the solution by inverting a matrix of correlation coefficients. The five stepwise regression programs made use of matrix partitioning in connection with inverting a matrix of correlation coefficients.

8.1. Stepwise Regression Programs

The five stepwise regression programs were BMD02R, MPR3, the STAT-PACK program RESTEM, WRAP, and STAT20***. They all, to a greater or lesser extent, follow Efroymson's algorithm [16]. Tables 1, 2 and 4 give the results of these five programs.

The BMD02R program is described in BMD Biomedical Computer Programs [15]. The UNIVAC 1108 STAT-PACK Program RESTEM is described in the Programmers Reference [41]. The two programs, MPR3 and WRAP, are both from the SHARE library [26]. The former was written by M. A. Efroymson and the latter was written by M. D. Fimple. The program STAT20*** is included in the C-E-I-R Multi-Access Computer Service library, and a brief writeup on how to use the program is given in C-E-I-R's "Library Programs Documentation" [10].

In running the two test problems on three of the stepwise programs, namely BMD02R, RESTEM and STAT20***, calculations stopped before the solutions were obtained. In all three programs, computations stopped in the $Y2$ problem after x, x^2, x^4, x^5 and a constant term had entered the regression equation. In the $Y1$ problem, BMD02R and STAT20*** stopped after x^4, x^5 and a constant term had entered, and RESTEM stopped after x^5 and a constant term had entered. In order to obtain the coefficients for the fifth degree equations, certain features of these three programs had to be bypassed.

In the printout of RESTEM and STAT20*** obtained from the original (unaltered) programs, there were no clues to indicate that there was a rounding error problem. The initial runs of the BMD02R program, however, printed the messages "ERROR TERMINATION IN SQRT ROUTINE" and "SQRT CALLED AT SEQUENCE NUMBER 01032 OF MAIN PROGRAM." These messages were produced by the computer system, not by the BMD02R program. They indicated the nature of some of the trouble—that the argument of a certain square root function was negative. The initial BMD02R runs furnished another clue of computational difficulties. The output of this program includes various calculated F- values which are needed for entering and removing variables from the regression. In both $Y1$ and $Y2$, there were one or more F-values (labeled "F TO ENTER") which were negative.

It was found that the RESTEM and STAT20*** programs had also calculated negative F-values, and checks involving F-values had to be bypassed in order to obtain the fifth degree solutions. Moreover, in the RESTEM program it was necessary to change the value of "minus infinity" from -10^{38} to -10^{34} before satisfactory results could be obtained for any least squares problem.

WRAP, the program with the lowest rankings in table 1, computed coefficients which were exceptionally far from the true values. These coefficients are listed below.

$Y1$		$Y2$	
True $\hat{\beta}$	Computed $\hat{\beta}$	True $\hat{\beta}$	Computed $\hat{\beta}$
1.	2991622.	1.	−33.84546
1.	−6065892.	0.1	71.54880
1.	2218821.	.01	−26.16913
1.	−296194.5	.001	3.493256
1.	16462.20	.0001	−0.1936966
1.	−322.5731	.00001	.003812985

Since WRAP performed so poorly on the two test problems, $Y1$ and $Y2$, some other test problems were run in order to verify that the program could handle problems which were not so badly conditioned. Let $U1(k)$ and $U2(k)$ be defined as follows:

$$U1(k)\colon\ y=1+x+x^2+\ .\ .\ .+x^k,$$

$$U2(k)\colon\ y=1+0.1\ x+0.01\ x^2+\ .\ .\ .+10^{-k}x^k.$$

Taking $x=0(1)20$, $k=1, 2, 3, 4$, the y-values were calculated for $U1(k)$ and $U2(k)$. Using these calculated y's as input, it was found that the coefficients for degrees 1, 2, and 3 computed by the WRAP program had some accuracy, but those for degree 4 were computed inaccurately. The results for degrees 3 and 4 are given below.

U1(3)		U2(3)	
True $\hat{\beta}$	Computed $\hat{\beta}$	True $\hat{\beta}$	Computed $\hat{\beta}$
1.	1.223297	1.	1.000416
1.	0.8286224	0.1	0.09971083
1.	1.022150	.01	.01003707
1.	0.9992682	.001	.0009987662

U1(4)		U2(4)	
True $\hat{\beta}$	Computed $\hat{\beta}$	True $\hat{\beta}$	Computed $\hat{\beta}$
1.	−731.1589	1.	0.8673919
1.	852.1714	0.1	.2555575
1.	−193.1382	.01	−.02548887
1.	15.94353	.001	.003731290
1.	0.6330207	.0001	.00003291620

8.2. Other Programs Using Elimination Algorithms

Two other BMD programs were tested. The BMD03R program, Multiple Regression with Case Combinations, inverts a matrix of correlation coefficients. BMD05R, Polynomial Regression, inverts the matrix $Z'Z$ where the vectors of Z are formed from the vectors of X by subtracting the mean of each vector from every element of that vector. All the crucial operations of BMD05R, such as the forming of inner products and matrix inversion, are carried out in double precision. The performance of BMD03R and BMD05R is shown in tables 1, 3, and 4.

DAM is a general-purpose computer program for data processing and multiple regression written by Rudolf R. Rhomberg, Lorette Boissonneault, and Leonard Harris, International Monetary Fund [36]. In running the two test problems on DAM on the 1108, computations stopped after a fourth degree polynomial was fitted, and the message "INSUFFICIENT NUMBER OF OBSERVATIONS OR DATA ARE ALL ZEROS, PROGRAM CANNOT COMPUTE EQUATION 5" was printed. It was found that a computed variance was zero and that this condition causes the computations to stop. By bypassing the checks on this computed variance, results for fifth degree fits were obtained. On the 7094, however, the fifth degree results were reached without any such difficulties. DAM's performance on the two computers is given in table 1.

Two lines of table 1 report the results from OMNITAB on the 7094 and the 1108 where the matrix commands INVERT, MMULT, and MTRANS were used. Here the 21 pairs of (x, y) values were read into the computer, the powers of x were generated, the matrices $X'X$ and $X'Y$ were obtained via MTRANS and MMULT, the inverse of $X'X$ was obtained via INVERT, and $\hat{\beta}$ was then obtained via MMULT. The solutions were far less accurate than those obtained from OMNITAB by using the command POLYFIT which calls on the ORTHO routine.

The program POLRG is the polynomial regression program of the IBM System/360 Scientific Subroutine Package [24], [25]. This program calls four subroutines, GDATA, ORDER, MINV, and MULTR, in the course of obtaining the least squares coefficients and other quantities of interest. These subroutines perform the following operations:

(1) GDATA generates the powers of the independent variable, finds means and standard deviations, and sets up a correlation matrix.

(2) ORDER chooses a dependent variable and a subset of independent variables from a larger set of variables.

(3) MINV inverts the correlation matrix using the "standard Gauss-Jordan method."

(4) MULTR computes the regression coefficients and related quantities, such as the sum of squares attributable to the regression and the sum of squares of deviations from the regression.

We see from table 1 that the single precision version of POLRG obtained rather low scores on both test problems. A double precision version of POLRG was also run, and the performance here as reported in table 4 was comparable to other programs using similar elimination algorithms.

The user of POLRG specifies m, the highest degree polynomial to be fitted, and the program automatically reports the results of fitting polynomials of successively increasing degrees, starting with the first degree. If there is no reduction in the residual sum of squares between two successive degrees of polynomials, the program stops the problem before completing the analysis for the highest degree specified. In running both test problems, $Y1$ and $Y2$, in single precision the analysis stopped after degree four, and in lieu of a fifth degree polynomial fit, the message "NO IMPROVEMENT" was printed. In order to complete the calculations for the fifth degree, the checks on "improvement" were bypassed. In the double precision version, fifth degree results were obtained without any such alterations.

The Programmer's Manual for the IBM System/360 Scientific Subroutine Package [25] contains some warnings regarding the accuracy of computations. The reader is informed that the accuracy of the computations in many of the routines is highly dependent upon the number of significant digits available for arithmetic operations. It is pointed out that matrix inversion and many of the statistical subroutines fall into this category, and that the user may, therefore, wish to use double precision versions of these routines. (The programs are so constructed that conversion to double precision is an easy matter.) An appendix of the manual classifies the subroutines of this package into three categories. These are: (1) subroutines having little or no effect on accuracy, (2) subroutines whose accuracy is dependent on the characteristics of the input data, and (3) subroutines in which definite statements on accuracy can be made. Only one of the four subroutines called by the POLRG program, namely ORDER, is in the first category. The other three subroutines, GDATA, MINV and MULTR, fall in the second category. In connection with this second category we read that "it cannot be assumed that the results are accurate simply because subroutine execution is completed."

A more explicit statement is given in connection with the subroutine GDATA. Here there is a comment in the program stating that if m, the highest degree polynomial to be fitted, is equal to 5 or greater, single precision may not be sufficient to give satisfactory results. Since the manual's test problem for POLRG specifies $m=4$ and has 15 data points, one might infer that satisfactory results would be obtained for this problem. This is not the case, however. In the solution to this problem given on page 410 of the manual, the intercept term for the polynomial regression of degree 4 is reported to be -5.26735. An accurate calculation shows that this term is actually -6.04262, so that the reported term had no correct significant digits. The four reported regression coefficients were correctly computed to only one or two digits. Furthermore, the sum of squares of deviations from the regression is reported to be 128.85156, whereas it is actually 17.67310. This error is also propagated into the calculation of the mean square, the F value, and the improvement in terms of sum of squares. The calculated values of $\hat{Y}$ were found to be correct to one, two or three significant digits, with the residuals correct to one digit or less.

In concluding this digression concerning the accuracy of the test problem accompanying a particular program of a particular package, we note a remark given in the Programmer's Manual under "Purposes and Objectives of the Package": "While this package may provide many of the tools necessary to solve the more commonly encountered problems in engineering and science, there is no intent to imply that these subroutines represent the current state of the art in statistics or numerical analysis."

The programs SPVMTX and DPVMTX appearing in tables 1, 3, and 4 use single and double precision versions, respectively, of the same algorithm. These two programs were adapted by

Sally T. Peavy, National Bureau of Standards, from two subroutines in the SHARE library: A. R. Sadaka's 7090–F1 3180INV1 Single Precision Matrix Inversion with Selective Pivoting and A. R. Sadaka's 7090–F1 3181INV2 Double Precision Matrix Inversion with Selective Pivot. Mrs. Peavy's adaptations of these programs included accuracy checks on the computed inverse. A brief description of these accaracy checks is in order. Let A be the input matrix to be inverted and Z the result of the inversion. Let $E=I-AZ$. Let $B=(b_{ij})$ be an $n\times n$ matrix, and let $N(B)$ be a norm defined in any of the following ways:

$$N_1(B)=\left(\sum_{i,j}\left|b_{ij}\right|^2\right)^{1/2} \quad \text{(the Euclidean or Frobenius norm)}$$

$$N_2(B)=n\max_{i,j}|b_{ij}|$$

$$N_3(B)=\max_i\sum_{j=1}^{n}|b_{ij}|\ .$$

In order to guarantee that Z be a good approximation to A^{-1}, it is only necessary to have $\dfrac{N(Z)\,N(E)}{1-N(E)}$ small. This quantity, $\dfrac{N(Z)\,N(E)}{1-N(E)}$, is computed for each of the three norms N_1, N_2, and N_3, and provides upper bounds on the error in the elements of the computed inverse. See Newman [34, pp. 227–230] and Taussky [38, pp. 284–286] for a fuller discussion of the norms described above.

The matrix which was inverted by SPVMTX and DPVMTX in solving the two test problems was

$$\begin{bmatrix} X'X & X'Y \\ 0 & 1 \end{bmatrix} \text{ whose inverse is } \begin{bmatrix} (X'X)^{-1} & -\hat{\beta} \\ 0 & 1 \end{bmatrix}.$$

In the single precision solution, the three bounds for $Y1$ were, respectively, -160, -940, and -140, and those for $Y2$ were -2.2, -7.0 and -2.7. If an accurate inverse had been obtained, the error bounds would have been small positive numbers. That the bounds were *negative* is a clear warning that the computed inverses, including $\hat{\beta}$, are not accurate. The double precision 1108 solution obtained the bounds 0.00048, 0.0075, and 0.00046 for $Y1$, and 0.000000039, 0.00000064, and 0.000000087 for $Y2$. In both problems these bounds are quite conservative. In the solution of $Y1$, the largest error in the elements of $(X'X)^{-1}$ is 5.5×10^{-13}, and the largest error in the $\hat{\beta}_j$'s is 5.5×10^{-7}. In the solution of $Y2$, the largest error in the elements of $(X'X)^{-1}$ is again 5.5×10^{-13}, and in the $\hat{\beta}_j$'s is 3.3×10^{-13}.

The 1108 version of OMNITAB now uses the SPVMTX routine for matrix inversion and prints out the smallest error bound. OMNITAB reported the smallest error bound for the inverse of $X'X$ to be -6, a negative number. This was in agreement with the results from inverting $X'X$ via the FORTRAN program SPVMTX where the three error bounds were given as -1.7, -6.0 and -2.1.

Each of the two STAT-PACK programs, GLH, General Linear Hypotheses, and REBSOM, Back Solution Multiple Regression, has its individual features, but for the two test problems the solutions were carried out in the same manner, so that the coefficients obtained from the two programs were identical, as is indicated in table 1. Both programs invert $X'X$ by calling a matrix inversion subroutine called JIM which uses a Gauss-Jordan elimination scheme with maximal column pivoting and row scaling. The GLH program has an option whereby the user can enter restraints in the case $X'X$ is not of full rank. The REBSOM program has the feature that the user can enter an F-value to be used as a criterion for removing variables from the regression after an initial solution has been computed.

An error was found in the REBSOM program in the calculation of the variance of Y. After estimating k coefficients (including possibly a constant term) from n observations, the formula used for the variance of Y is var $Y=\dfrac{\Sigma(y_i-\hat{y}_i)^2}{n-k-1}$. The denominator of this formula should read $n-k$ rather than $n-k-1$.

The BASIC program LINFIT***, available in the C-E-I-R Multi-Access Computer Service, in order to obtain $\hat{\beta}$, inverts the matrix $A=\begin{bmatrix} X'X & X'Y \\ Y'X & \Sigma y_i^2 \end{bmatrix}$ whose inverse, if it exists, is

$$\begin{bmatrix} (X'X)^{-1}+\dfrac{\hat{\beta}\hat{\beta}'}{\Sigma y_i^2-Y'\hat{Y}} & \dfrac{-\hat{\beta}}{\Sigma y_i^2-Y'\hat{Y}} \\ \\ \dfrac{-\hat{\beta}'}{\Sigma y_i^2-Y'\hat{Y}} & \dfrac{1}{\Sigma y_i^2-Y'\hat{Y}} \end{bmatrix}$$

When $\hat{Y}=Y$, the matrix A is singular. In the two test problems $\hat{Y}=Y$, and the matrix A, if it were formed in the computer without any rounding error, would be singular. But A, for $Y1$ and $Y2$, contains 14-digit numbers, whereas the GE 235 computer works with approximately nine-digit numbers, so that rounding of the elements of A is inevitable, and the version of A contained in the computer is not singular. An "inverse" was obtained, and from this $\hat{\beta}$ was immediately computed. Table 2 gives the results.

A third problem was used to test the program LINFIT***. Here we had

$$X=\begin{bmatrix} 1 & 0 & 5 \\ 1 & 0 & 4 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \end{bmatrix},\ Y=\begin{bmatrix} 6 \\ 5 \\ 4 \\ 5 \end{bmatrix},\ \text{and } A=\begin{bmatrix} 4 & 4 & 12 & 20 \\ 4 & 8 & 6 & 18 \\ 12 & 6 & 46 & 64 \\ 20 & 18 & 64 & 102 \end{bmatrix}$$

The last column (row) of A is the sum of the three preceding columns (rows), so that A is clearly singular. Unlike $Y1$ and $Y2$, this problem is such that the elements of A have fewer than nine digits. An "inverse" was obtained by LINFIT***, however, and was printed as

$$\begin{bmatrix} a & a & a & -a \\ a & a & a & -a \\ a & a & a & -a \\ -a & -a & -a & a \end{bmatrix} \quad \text{where } a=3.67091\times 10^7.$$

To obtain the inverse of A in this program, the matrix command MAT Q=INV(A) is used. This command inverts the matrix A by using an elimination method with row pivoting (Kurtz [28]). The method is described in section 1.2 of Stiefel [37].

LSCF-- *** is another BASIC least squares program available in the C-E-I-R Multi-Access Computer Service. Here the solution, $\hat{\beta}$, is obtained by inverting $X'X$ and then post-multiplying the inverse by $X'Y$. Table 2 shows that LSCF-- *** ranked below the other programs of this table. The inverse of $X'X$ is obtained by using the matrix command INV, the same command as was used in LINFIT***.

The SIMEX-*** program originated at the Naval Ordnance Laboratory. The input required for this program was $X'X$ and $X'Y$, since SIMEX-*** solves n equations in n unknowns. An elimination algorithm is used to obtain the solution. The input for this program was limited to nine significant digits. Recalling the results of table 7 which gave the exact solution for the $Y1$ problem when input data was rounded to nine digits, it is not surprising that the average number of correct digits for $Y1$, reported in table 2, is only 1.402. This lies between the "accuracy" achieved by the nine-digit and ten-digit problems of table 7.

The BASIC program STAT21*** obtains $(X'X)^{-1}$ and $\hat{\beta}$ by applying Jordan elimination to $X'X$ and $X'Y$; the results appear in table 2.

The LINFIT program included in table 1 is one of eighteen statistical routines described in *On-line Analysis for Social Scientists* by James R. Miller [32]. This library of routines exists in the Project MAC 7094 disk files. The two test problems were run on the LINFIT program on a time-shared computer via a remote console communicating with Project MAC. A description of Project

MAC may be found in Crisman [11]. Miller states that "these routines may be used without extensive prior training in mathematics, statistics, or computer operations," but in view of LINFIT's poor performance on these two problems, it appears that there may be pitfalls in using this program. The method used by the LINFIT program is unknown. By conjecture, it has been included in this section among programs using elimination algorithms.

9. Results from a Problem Having a Nonzero Standard Deviation

In the two test problems, *Y*1 and *Y*2, treated thus far, the residual standard deviation was zero. A third test problem, *Y*1*, is one where the standard deviation is nonzero. This problem was run on five programs in both single and double precision to see whether the fact that a least squares fit has a standard deviation of zero might be a factor influencing the accuracy of computations.

The values of *Y*1* were derived from the values of *Y*1 by adding 2.0 to the *Y*1 value when x is even and subtracting 2.0 from the *Y*1 value when x is odd. The input for *Y*1* is listed in table 9. A fifth degree polynomial fit for the *Y*1* problem has the solution (to 16 decimals)

$$\hat{\beta}(Y1^*)=\begin{bmatrix} 2.0459627329192547 \\ 0.1815856777493606 \\ 1.1701440301521066 \\ 0.9870776685960425 \\ 1.0003230582850989 \\ 1.0000000000000000 \end{bmatrix}$$

with the residual standard deviation equal to 2.3251684.

TABLE 8. *Comparison of results from two problems: one with nonzero standard deviation* (Y1*) *and one with zero standard deviation* (Y1)

All problems were run on the 1108 computer

Single Precision (8 Digits)

Program	Algorithm [a]	Average number of correct digits		Rank		Date of run	
		*Y*1*	*Y*1	*Y*1*	*Y*1	*Y*1*	*Y*1
BMDO2R	E	0.464	−0.106	4	4	12–19–67	12–13–67
LSTSQ	HT	3.485	4.528	2	1	10–15–68	5– 1–68
MATH-PACK, ORTHLS	OP	2.053	2.118	3	3	10–15–68	4–12–68
OMNITAB (Ortho)	GS	3.711	4.137	1	2	10–15–68	10–18–67
POLRG	E	−0.074	−0.191	5	5	10–15–68	10– 7–68

Double Precision (18 Digits)

Program	Algorithm [a]	Average number of correct digits		Rank		Date of run	
		*Y*1*	*Y*1	*Y*1*	*Y*1	*Y*1*	*Y*1
BMDO2R	E	9.657	9.645	4	4	4–17–68	4–17–68
LSTSQ	HT	13.913	14.643	1	1	10–16–68	7–22–68
MATH-PACK, ORTHLS	OP	12.079	12.098	3	3	10–16–68	10–16–68
ORTHO	GS	13.136	13.188	2	2	3–27–68	1–29–68
POLRG	E	9.270	9.290	5	5	10–17–68	10– 7–68

[a] E = Elimination method; GS = Gram-Schmidt orthonormalization; HT = Orthogonal Householder transformations; OP = Orthogonal polynomials.

TABLE 9. *Input for fifth degree polynomials*

X	$Y1$	$Y2$	$Y1^*$
0.	1.	1.00000	3.
1.	6.	1.11111	4.
2.	63.	1.24992	65.
3.	364.	1.42753	362.
4.	1365.	1.65984	1367.
5.	3906.	1.96875	3904.
6.	9331.	2.38336	9333.
7.	19608.	2.94117	19606.
8.	37449.	3.68928	37451.
9.	66430.	4.68559	66428.
10.	111111.	6.00000	111113.
11.	177156.	7.71561	177154.
12.	271453.	9.92992	271455.
13.	402234.	12.75603	402232.
14.	579195.	16.32384	579197.
15.	813616.	20.78125	813614.
16.	1118481.	26.29536	1118483.
17.	1508598.	33.05367	1508596.
18.	2000719.	41.26528	2000721.
19.	2613660.	51.16209	2613658.
20.	3368421.	63.00000	3368423.

Matrix $X'X$

21.	210.	2870.	44100.	722666.	12333300.
210.	2870.	44100.	722666.	12333300.	216455810.
2870.	44100.	722666.	12333300.	216455810.	3877286700.
44100.	722666.	12333300.	216455810.	3877286700.	70540730666.
722666.	12333300.	216455810.	3877286700.	70540730666.	1299155279940.
12333300.	216455810.	3877286700.	70540730666.	1299155279940.	24163571680850.

Matrix $X'Y$ for $Y1$	Matrix $X'Y$ for $Y2$	Matrix $X'Y$ for $Y1^*$
13103167.	310.39960	13103169.
229558956.	5058.55410	229558976.
4106845446.	87258.40800	4106845866.
74647573242.	1549291.38666	74647581842.
1373802809082.	28043466.66600	1373802985062.
25537373767266.	514843723.46850	25537377366266.

Table 8 summarizes the results, comparing the accuracy of the coefficients for $Y1^*$ with the corresponding accuracy for $Y1$. We see that the results for the two problems are quite similar, in both single and double precision. The largest differences occurred with the program LSTSQ in single precision, where the average number of correct digits was 4.528 for $Y1$, and the average for $Y1^*$ was 3.485, a decrease of 1.043.

On the basis of this comparison it appears that the fact that the standard deviation was zero in the test problems did not appreciably affect the accuracy of computations.

10. Concluding Remarks

(1) Computational procedures appropriate for desk calculators may be perilous for computers.

(2) Of the four procedures using floating-point arithmetic which were included in this study, orthogonal Householder transformations and Gram-Schmidt orthonormalization proved to be the best. Orthogonal polynomials ranked next. Elimination methods were the least successful but the most popular. The multiple precision integer arithmetic procedure using congruential methods was unique in obtaining exact solutions.

338-397 O-69—2

(3) Some other algorithms apparently of high quality which have been published in the last few years were not included in this study. These include:

(a) Bauer [2],
(b) Björck and Golub [6],
(c) Björck [5].

Bauer [2] gives an ALGOL procedure using iterative refinement for finding the least squares solution of $X\beta = Y$, where X is $n \times k$ $(k \leq n)$ of rank k and Y is $n \times p$. The procedure is based on the decomposition of X into UDR where U is $n \times k$ with orthogonal columns, $D = (U'U)^{-1}$, and R is upper triangular. This decomposition yields a triangular system $R\beta = U'Y$ which is solved by back substitution. The reduction to $R\beta = U'Y$ is carried out by a Gaussian elimination scheme, but with a suitably weighted combination of rows used for elimination instead of a single row.

Björck and Golub [6] and Björck [5] (see also Björck [3], [4]) give two least squares algorithms with certain common features. Both take advantage of the fact that $X'\delta = 0$, where δ is the vector of residuals, to obtain the solution β in $X\beta = Y$ from the augmented system of $n+k$ equations:

$$\begin{bmatrix} I & X \\ X' & 0 \end{bmatrix} \begin{bmatrix} \delta \\ \beta \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix}.$$

Both algorithms include δ as well as β in the iterative refinement procedure.

The two algorithms are based on (i) orthogonal Householder transformations, and (ii) a modified Gram-Schmidt orthogonalization process.

Both the classical Gram-Schmidt orthogonalization process and the modified Gram-Schmidt orthogonalization process, as described by Björck [3], decompose the matrix X into QR where $Q'Q$ is diagonal and R is upper triangular. In the classical procedure, at the ith stage, the ith column vector is made orthogonal to each of the $i-1$ previously orthogonalized column vectors; this is done for column indices $i = 2, 3, \ldots, k$. In the modified procedure which Björck uses, at the ith stage, the $(k-i+1)$ column vectors indexed $i, i+1, \ldots, k$ are made orthogonal to the $(i-1)$-th column vector; this is done for column indices $i = 2, 3, \ldots, k$. Jordan [27] shows why the modified procedure is superior to the classical procedure. Björck [3] states that his modified Gram-Schmidt procedure is equivalent to Bauer's method using weighted row combinations mentioned above. The algorithms for both the orthogonal Householder transformation method and the modified Gram-Schmidt method are generalized to handle the case where X is of less than full rank. In this case linear constraints are entered.

The papers of Björck [3, 5] and Björck and Golub [6] discuss the number of operations and the storage requirements of their least squares algorithms.

(4) Programmers who have been writing least squares programs, especially for statistical packages, have often not been taking advantage of the advances in this area made by numerical analysts in recent years.

(5) The importance of accumulating inner products in double precision cannot be overstressed. A number of recent papers on least squares computations have emphasized this point. These include Businger and Golub [8], Bauer [2], Golub and Wilkinson [22], Björck and Golub [6], and Björck [5]. On many third-generation computers which have double precision built into the hardware, double precision arithmetic is quite efficient.

(6) Iterative refinement is another valuable feature of recent algorithms. The three algorithms described in remark (3) above all include this feature. Four programs included in the present study (LSFITW***, LSTSQ, ORTHO, and ORTHOL) made effective use of iterative refinement. Golub and Wilkinson [22], who discuss this topic, also mention that the condition number of $X'X$ is approximately the square of the condition number of X, so that it is advantageous to work with X rather than $X'X$ whenever possible. Moler [33] and Forsythe [19] discuss the details of iterative refinement in connection with solving $n \times n$ systems of linear equations.

(7) The users of least squares programs can take certain precautionary steps to gain an awareness of whether or not a rounding error problem exists. Various suggestions were made in the previous studies of Cameron, Freund, Zellner and Thornber, and Longley. These suggestions included the following:

(a) Run test problems where the coefficients are known.
(b) Transform the data (e.g., by subtracting means).
(c) Do the calculations several times, scaled differently each time.
(d) Shuffle the columns of X and run the problem more than once.
(e) Check whether $X'\delta = 0$.
(f) Use double precision arithmetic.

(8) Another check on the accuracy of least squares coefficients, suggested by Joseph M. Cameron, is the following. After carrying out the usual fit of Y to k independent variables, do a second fit, taking $\hat{Y}$ (the predicted values) and refitting to the k original independent variables. If there were no rounding error at all, one would obtain exactly the same coefficients from the refit as from the original fit, and the standard deviation of the refit would be zero. The extent to which the second set of coefficients agrees with the original set can give one some information about the severity of rounding error.

A number of test problems were run on the 7094 and the 1108 in order to investigate the relationships among the coefficients of the original fit, those of the refit, and those one would obtain if there were no rounding error. The test problems consisted of 55 polynomials with various ranges of x, various degrees from 1 to 8, and various coefficients. All 55 were run in both single and double precision on the 7094, and twelve of them were run in single and double precision on the 1108.

In these test problems, the following result was obtained: If the coefficients from the refit (denoted by $\hat{b}$) agreed with the coefficients from the original fit ($\hat{\beta}$) to an average of more than three digits, and if the elements of $X'X$ and $X'Y$ can be represented in the computer without rounding, then the number of digits in $\hat{\beta}_j$ in agreement with $\hat{b}_j (j=1, \ldots, k)$ was approximately the same as the number of correct digits in $\hat{\beta}_j$. More precisely, whenever the two conditions just stated were met, it was found (with one exception) that the number of digits in $\hat{\beta}_j$ in agreement with those of $\hat{b}_j$ (i) in double precision was within 1.0 of the number of correct digits in $\hat{\beta}_j$, and (ii) in single precision was within 2.0 of the number of correct digits in $\hat{\beta}_j$, for all j. The one exception occurred in a sixth degree polynomial with $x = -10(1)10$. In the double precision run the two sets of coefficients agreed to an average of about 12.5 digits, and the elements of $X'X$ and $X'Y$ had at most 13 significant digits; here $\hat{\beta}_2$ agreed with $\hat{b}_2$ to 16 digits but was correct only to about 13 digits.

(9) Efforts to provide comparative data on the amount of computer time required by the various programs included in this investigation, as well as comparisons of storage requirements, were unsuccessful. The programs which were included in this study originated from many sources, and they exhibited considerable variation with respect to what quantities were calculated as well as with respect to the methods of calculation. The program ALSQ, for example, at one end of the spectrum, calculated simply the coefficients, the residuals, the predicted values, and the residual sum of squares for the requested fifth degree polynomial. The single precision version of ALSQ required eight seconds to process both test problems on the 1108; the storage requirements were 709 memory cells for the code and 323 for the data. The double precision version of ALSQ processed both problems in seven seconds, requiring 715 memory cells for the code and 618 for the data. Nearer the other end of the spectrum was the Biomed program BMD05R. The output here consisted of the coefficients and their standard deviations for polynomial fits of degrees 1, 2, 3, 4, 5, an analysis of variance for degrees 1, 2, 3, 4, 5, predicted values and residuals for degree 5, a plot of observed and predicted values for degree 5, and means and correlation coefficients of the input data. This program (computing some operations in double precision) required 20 seconds on the 1108 to process the two test problems; the storage requirements were 3,119 memory cells for the code and 15,168 for the data. It becomes evident that an intercomparison of running time among the different programs is not meaningful.

Moreover, in repeated runs of a particular program there is fluctuation from run to run in the amount of time required. For example, on the same day three separate jobs were submitted to be run on OMNITAB (using the command POLYFIT) on the 1108, with the following results:

(a) *Y*1 alone: 8 seconds.

(b) *Y*2 alone: 12 seconds.

(c) *Y*1 and *Y*2 together: 8 seconds.

The 1108 version of OMNITAB requires about 50,000 memory cells for storage. The run times given here include unknown components of time for operation of the computer system.

Although one would expect a double precision version of a particular program to require more time than a single precision version, there were several instances on the 1108 where double precision required less time than single precision.

It was outside the scope of this investigation to make a detailed comparison of algorithms with respect to efficiency of computation time and storage requirements. Similarly, no comparative examination of the outputs provided by the programs was made. Rather, this study focused attention on the performance of existing programs.

(10) In any mathematical calculation carried out on a computer, it is desirable to know whether an accurate solution has been obtained or whether the result of a calculation is contaminated by rounding error to such an extent that it is worthless. This goal has been achieved in some areas. Martin, Peters, and Wilkinson [31], in their paper giving an algorithm for solving $Ax=b$, where A is an $n\times n$ positive definite matrix and b is an $n\times p$ matrix, state that their procedure "either produces the correctly rounded solutions of the equation $Ax=b$ or indicates that A is too ill-conditioned for this to be achieved without working to higher precision (or is possibly singular)." Similarly, Wilkinson's program [45] for the solution of an ill-conditioned system of equations $Ax=b$, where A is $n\times n$, "gives either a solution of the system which is correct to working accuracy or alternatively indicates that the system is too ill-conditioned to be solved without working to higher precision and may even be singular."

It appears that the goal set out above has now been achieved in the linear least squares program of Björck and Golub [6]. The authors state that their procedure may be used to compute accurate solutions and residuals to linear least squares problems, but that the procedure will fail when X modified by rounding errors has less than full rank, and that it will also fail if X is so ill-conditioned that there is no perceptible improvement in the iterative refinement. The user is easily informed of these situations.

I would like to express my appreciation to Joseph M. Cameron who suggested this investigation and made many valuable contributions, to Joan R. Rosenblatt for helpful discussions, and to Joseph Hilsenrath, Russell A. Kirsch, and Thomas Hoover for use of their time-shared computer facilities to run several problems. Thanks are due to Gene H. Golub, Stanford University, for his constructive remarks. The assistance of James Doyle, Univac Division, Sperry Rand Corporation, in debugging one program is also appreciated.

11. Appendix A. Sources of the Programs, With Brief Descriptions

ALSQ. A FORTRAN IV subroutine to solve the linear least squares problem, written by G. W. Stewart III, Union Carbide Corp., Oak Ridge, Tennessee (present address: The University of Texas, Austin, Texas). This program uses a modification of the Businger-Golub algorithm [8].

BMD02R, Stepwise Regression. One of the Biomedical Computer Programs, written in FORTRAN [15].

BMD03R, Multiple Regression with Case Combinations. One of the Biomedical Computer Programs, written in FORTRAN [15].

BMD05R, Polynomial Regression. One of the Biomedical Computer Programs, written in FORTRAN [15].

DAM. A general purpose computer program for data processing and multiple regression, written in FORTRAN by Rudolf R. Rhomberg, Lorette Boissonneault, and Leonard Harris, International Monetary Fund [36].

DPVMTX. A double precision FORTRAN IV program for inverting a matrix or solving a set of linear equations. To a program from the SHARE library (7090–F1 3181INV2 Double Precision Matrix Inversion with Selective Pivot, written by A. R. Sadaka [26]), Sally T. Peavy, National Bureau of Standards, incorporated accuracy checks.

LINFIT. A program which fits a linear function to collected data via least squares. Optional constraints may be applied to the fitting coefficients to make them nonnegative, add to a constant, etc. One of eighteen statistical routines written by James R. Miller [32]. This library of routines exists in the Project MAC 7094 in the disk files of user number T169 2750.

LINFIT*.** A program written in BASIC for linear least squares curve fitting and computing correlations. Origin: Dartmouth College, Hanover, N.H. Available in the C-E-I-R Multi-Access Computer Services library [10].

LSCF--*.** A least squares polynomial curve fitting subroutine written in BASIC. Origin: Dartmouth College, Hanover, N.H. Available in the C-E-I-R Multi-Access Computer Services library [10].

LSFITW*.** A least squares curve fitting program written in BASIC. Adapted by John B. Shumaker, National Bureau of Standards, from Philip J. Walsh's ORTHO algorithm [42]. Available in the C-E-I-R Multi-Access Computer Services library [10].

LSTSQ. A FORTRAN IV subroutine which solves for X the overdetermined system $AX=B$ of m linear equations in n unknowns for p right-hand sides. Written by Peter Businger, Computation Center, University of Texas (present address: Bell Telephone Laboratories, Murray Hill, N.J.), using the Businger-Golub algorithm [8].

MATH-PACK, ORTHLS, Orthogonal Polynomial Least-Squares Curve Fitting. One of the Univac 1108 MATH-PACK programs, written in FORTRAN V [40].

MPR3, Stepwise Multiple Regression with Variable Transformations. A FORTRAN II program written by M. A. Efroymson, Esso Research and Engineering Co., Madison, N.J., using the Efroymson algorithm [16]. Available in the SHARE library: 7090–G2 3145MPR3 [26].

OMNITAB, a general-purpose computer program for statistical and numerical analysis. Developed at the National Bureau of Standards by Joseph Hilsenrath et al. [23]. Now available in an A. S. A. FORTRAN version, OMNITAB allows the user to communicate with a computer in an efficient manner by means of simple English sentences.

ORTHO. A program written by Philip J. Walsh, National Bureau of Standards (present address: University Computing Co., East Brunswick, N.J.), which uses a Gram-Schmidt orthonormalization process for least squares curve fitting. ORTHO exists as an ALGOL procedure [42], a FORTRAN program, a BASIC program (see LSFITW*** above), and as a routine of OMNITAB [23], where it is called by the commands FIT and POLYFIT.

ORTHOL. A modification of the Davis-Rabinowitz orthonormalization algorithm [12, 13, 14], written in FORTRAN II by James W. Longley, Bureau of Labor Statistics, Washington, D.C., and Roger A. Blau, Bureau of Labor Statistics and Carnegie-Mellon University, Pittsburgh, Pa. [30].

POLFIT. An anonymous program written in BASIC for least squares polynomial curve fitting using orthogonal polynomials.

POLRG, Polynomial Regression. One of the programs of the IBM System/360 Scientific Subroutine Package written in FORTRAN IV [24, 25].

SIMEX-*.** A program written in BASIC for solving n simultaneous equations in n unknowns. Origin: Naval Ordnance Laboratory, Silver Spring, Md. Available in the C-E-I-R Multi-Access Computer Services library [10].

SOLVER. A FORTRAN program written by Morris Newman, National Bureau of Standards, for obtaining the exact solution of the system $AX=B$, or the inverse of a matrix A, by congruential methods [35]. The elements of A and B must be integers.

SPVMTX. A single precision FORTRAN IV program for inverting a matrix or solving a set of linear equations. To a program from the SHARE library (7090–F1 3180INV1 Single Precision Matrix Inversion with Selective Pivoting, written by A. R. Sadaka [26]), Sally T. Peavy, National Bureau of Standards, incorporated accuracy checks.

STAT-PACK, GLH, General Linear Hypotheses. One of the Univac 1108 STAT-PACK programs, written in FORTRAN V [41].

STAT-PACK, REBSOM, Back Solution Multiple Regression. One of the Univac 1108 STAT-PACK programs, written in FORTRAN V [41].

STAT-PACK, RESTEM, Stepwise Multiple Regression. One of the Univac 1108 STAT-PACK programs, written in FORTRAN V [41].

STAT20*.** A program written in BASIC for stepwise multiple linear regression. Written by Thomas E. Kurtz, Dartmouth College, Hanover, N.H. Available in the C-E-I-R Multi-Access Computer Services library [10].

STAT21*.** A program written in BASIC for multiple linear regression with detailed output. Written by Gerald Childs, Dartmouth College, Hanover, N.H. Available in the C-E-I-R Multi-Access Computer Services library [10].

WRAP, Weighted Regression Analysis Program. A FORTRAN II program written by M. D. Fimple, Sandia Corp., Albuquerque, New Mexico. Available in the SHARE library: 7090–G2 3231 WRAP [26].

APPENDIX B

DETAILS FOR TABLE 1 -- SINGLE PRECISION (8 DIGITS)

ALSQ 1108 EXAMPLE 1

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0228963	1.640	1.0000002	6.699
.99553592	2.350	.10000037	5.432
.99941321	3.232	.0099998263	4.760
1.0001108	3.955	.0010000220	4.658
.99999613	5.412	.000099998902	4.959
1.0000000	8.000	.000010000020	5.699
AVERAGE =	4.098	AVERAGE =	5.368

BMD02R 1108 EXAMPLE 2

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-17.13281	-1.258	.99954	3.337
39.34436	-1.584	.10098775	2.005
-13.26675	-1.154	.0096306379	1.433
2.92344	-.284	.0010499504	1.301
.89241	.968	.000097199970	1.553
1.00212	2.674	.000010055379	2.257
AVERAGE =	-.106	AVERAGE =	1.981

BMD03R 7094 EXAMPLE 3

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
394.23438	-1.898	1.04353	1.361
-16.00000	-1.230	.10083	2.081
28.00000	-1.431	.01013	1.886
-1.00000	-.301	.00101	2.000
1.00000	6.000	.00010	2.000
1.00049	3.310	.00001	1.000
AVERAGE =	.742	AVERAGE =	1.721

BMD03R 1108 EXAMPLE 4

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
5161.95310	-2.642	1.07353	1.134
40.000000	-1.591	.10131836	1.880
4.0000001	-.477	.010009766	3.010
.50000000	.301	.00099945068	3.260
.89062500	.961	.000097751617	1.648
.99804688	2.709	.0000099837780	2.790
AVERAGE =	-.123	AVERAGE =	2.287

DAM 7094 EXAMPLE 5

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
2.20	-.079	1.000	4.000
.460	.268	.101	2.000
.920	1.097	.00975	1.602
1.03	1.523	.00103	1.523
.997	2.523	.0000982	1.745
1.00	3.000	.0000100	3.000
AVERAGE =	1.389	AVERAGE =	2.312

DAM 1108 EXAMPLE 6

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
26.798895	-1.412	1.0000993	4.003
-53.926606	-1.740	.099779484	2.657
21.511053	-1.312	.010084331	2.074
-1.7723664	-.443	.00098840467	1.936
1.1553755	.809	.00010065825	2.182
.99692726	2.512	.0000099868524	2.881
AVERAGE =	-.264	AVERAGE =	2.622

DETAILS FOR TABLE 1 -- SINGLE PRECISION (8 DIGITS)

LINFIT (MILLER) 7094 EXAMPLE 7

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
7360.000	-3.867	1.074	1.131
-16598.000	-4.220	-.066	-.220
6379.500	-3.805	.074	-.806
-877.906	-2.944	-.008	-.954
50.989	-1.699	.001	-.954
.000	.000	.000	.000
AVERAGE =	-2.756	AVERAGE =	-.301

LSTSQ 1108 EXAMPLE 8

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.99973875	3.583	.99999997	7.523
1.0006891	3.162	.10000011	5.959
.99970413	3.529	.0099999484	5.287
1.0000452	4.345	.0010000083	5.081
.99999718	5.550	.000099999460	5.268
1.0000001	7.000	.000010000012	5.921
AVERAGE =	4.528	AVERAGE =	5.840

MATH-PACK 13.5, ORTHLS, ORTHOGONAL POLYNOMIAL CURVE FITTING 1108 EXAMPLE 9

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.94458008	1.256	.99999909	6.041
1.1799316	.745	.10000322	4.492
.91607666	1.076	.0099984696	3.815
1.0135651	1.868	.0010002495	3.603
.99912310	3.057	.000099983743	3.789
1.0000196	4.708	.000010000364	4.439
AVERAGE =	2.118	AVERAGE =	4.363

MPR3, STEPWISE MULTIPLE REGRESSION, SHARE LIBRARY 3145MPR3 7094 EXAMPLE 10

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-20.24219	-1.327	.99933	3.174
43.49164	-1.628	.1013436	1.872
-14.37052	-1.187	.009510736	1.310
3.03207	-.308	.001065033	1.187
.88800	.951	.00009639953	1.444
1.00219	2.660	.00001007054	2.152
AVERAGE =	-.140	AVERAGE =	1.856

OMNITAB, USING THE MATRIX COMMANDS MTRANS, MMULT, INVERT 7094 EXAMPLE 11

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-91.999999	-1.968	.99780273	2.658
136.00000	-2.130	.10278320	1.555
-48.000000	-1.690	.0086669922	.875
5.5000000	-.653	.0011596680	.797
.75000000	.602	.000092506409	1.125
1.0063477	2.197	.000010177493	1.751
AVERAGE =	-.607	AVERAGE =	1.460

OMNITAB, USING THE MATRIX COMMANDS MTRANS, MMULT, INVERT 1108 EXAMPLE 12

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-220.65660	-2.346	.99512554	2.312
316.45422	-2.499	.10715805	1.145
-96.578077	-1.989	.0077479167	.647
10.859373	-.994	.0012353163	.628
.52858572	.327	.000088502186	.939
1.0087148	2.060	.000010213975	1.670
AVERAGE =	-.907	AVERAGE =	1.224

DETAILS FOR TABLE 1 -- SINGLE PRECISION (8 DIGITS)

OMNITAB, USING ORTHO SUBROUTINE — 7094 — EXAMPLE 13

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0012817	2.892	.99999949	6.292
.99780273	2.658	.10000013	5.886
.99932861	3.173	.0099999756	5.613
1.0001755	3.756	.0010000018	5.745
.99998569	4.844	.000099999866	5.873
1.0000004	6.398	.000010000004	6.398
AVERAGE =	3.954	AVERAGE =	5.968

OMNITAB, USING ORTHO SUBROUTINE — 1108 — EXAMPLE 14

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0064697	2.189	.99999990	7.000
.99902344	3.010	.099999700	5.523
.99975586	3.612	.010000125	4.903
.99996948	4.515	.00099998200	4.745
1.0000100	5.000	.00010000109	4.963
.99999968	6.495	.0000099999778	5.654
AVERAGE =	4.137	AVERAGE =	5.464

ORTHO, WITH RE-ORTHOGONALIZATION OMITTED — 1108 — EXAMPLE 15

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-1216.5426	-3.085	.98419483	1.801
2752.0557	-3.439	.13523918	.453
-1057.0931	-3.025	-.0034660707	-.129
146.97336	-2.164	.0028495983	-.267
-7.3080225	-.919	-.0000049256487	-.021
1.1663037	.779	.000012094996	.679
AVERAGE =	-1.976	AVERAGE =	.419

ORTHOL — 1108 — EXAMPLE 16

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.99784447	2.666	1.0000000	8.000
.98687472	1.882	.099999778	5.654
1.0029743	2.527	.010000041	5.387
.99961372	3.413	.00099999654	5.461
1.0000213	4.672	.00010000013	5.886
.99999960	6.398	.0000099999984	6.796
AVERAGE =	3.593	AVERAGE =	6.197

POLRG, IBM SYSTEM/360 SCIENTIFIC SUBROUTINE PACKAGE — 1108 — EXAMPLE 17

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-1823.8047	-3.261	.98438931	1.807
28.622013	-1.441	.10009000	3.046
-3.6844511	-.671	.010146444	1.834
3.0450442	-.311	.0010054175	2.266
.93157484	1.165	.000099141363	2.066
1.0004238	3.373	.000010021910	2.659
AVERAGE =	-.191	AVERAGE =	2.280

SPVMTX — 1108 — EXAMPLE 18

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
64.191025	-1.801	1.0014403	2.842
-134.65426	-2.132	.097101737	1.538
51.859977	-1.706	.011047948	.980
-5.8942945	-.838	.00086162069	.859
1.3872223	.412	.00010761766	1.118
.99232943	2.115	.0000098514820	1.828
AVERAGE =	-.658	AVERAGE =	1.527

DETAILS FOR TABLE 1 -- SINGLE PRECISION (8 DIGITS)

STAT-PACK 8.13, GLH, GENERAL LINEAR HYPOTHESIS 1108 EXAMPLE 19

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-11.970093	-1.113	.99989064	3.961
27.245361	-1.419	.10018034	2.744
-8.5661011	-.981	.0099404901	2.225
2.2717514	-.104	.0010073970	2.131
.92961073	1.152	.000099610348	2.409
1.0013784	2.861	.000010007344	3.134
AVERAGE =	.066	AVERAGE =	2.767

STAT-PACK 9.2, REBSOM, BACK SOLUTION MULTIPLE REGRESSION 1108 EXAMPLE 20

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-11.97009	-1.113	.9998906	3.961
27.24536	-1.419	.1001803	2.744
-8.56610	-.981	.009940490	2.225
2.27175	-.104	.001007397	2.131
.92961	1.152	.00009961035	2.409
1.00138	2.860	.00001000734	3.134
AVERAGE =	.066	AVERAGE =	2.767

STAT-PACK 9.1, RESTEM, STEPWISE MULTIPLE REGRESSION 1108 EXAMPLE 21

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-1.5703125	-.410	1.0002102	3.677
7.5583214	-,817	.099611598	2.411
-1.5695286	-.410	.010136487	1.865
1.3551083	.450	.00098222795	1.750
.97985181	1.696	.00010097076	2.013
1.0004015	3.396	.0000099811599	2.725
AVERAGE =	.651	AVERAGE =	2.407

WRAP, WEIGHTED REGRESSION, SHARE LIBRARY 3231WRAP 7094 EXAMPLE 22

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
2991622.	-6.476	-33.84546	-1.542
-6065892.	-6.783	71.54880	-2.854
2218821.	-6.346	-26.16913	-3.418
-296194.5	-5.472	3.493256	-3.543
16462.20	-4.216	-.1936966	-3.287
-322.5731	-2.510	.003812985	-2.580
AVERAGE =	-5.300	AVERAGE =	-2.871

DETAILS FOR TABLE 2 -- SINGLE PRECISION (9 DIGITS)

LINFIT 235 EXAMPLE 1

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
2.76639	-.247	1.00006	4.222
-2.72473	-.571	.0998764	2.908
2.38633	-.142	.010045	2.347
.812925	.728	.000994014	2.223
1.01048	1.980	.000100332	2.479
.999793	3.684	.00000999350	3.187
AVERAGE =	.905	AVERAGE =	2.894

LSCF 235 EXAMPLE 2

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
-5.5	-.813	.999893	3.971
-12.	-1.114	.099762	2.623
-3.	-.602	.00991058	2.049
0.	.000	.000970840	1.535
.957031	1.367	.0000993013	2.156
.999023	3.010	.00000997260	2.562
AVERAGE =	.308	AVERAGE =	2.483

DETAILS FOR TABLE 2 -- SINGLE PRECISION (9 DIGITS)

LSFITW — 235 — EXAMPLE 3

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.999130249	3.061	.999999897555	6.990
.99761963	2.623	.0999999824213	6.755
1.00102997	2.987	.0100000116561	5.933
.999854088	3.836	.00099999815736	5.735
1.00000715256	5.146	.000100000104819	5.980
.999999890104	6.959	.00000999999813838	6.730
AVERAGE =	4.102	AVERAGE =	6.354

POLFIT — 235 — EXAMPLE 4

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.99387360	2.213	.9999999618158	7.418
1.00894165	2.049	.1000000573928	6.241
.99534607	2.332	.0099999622960	5.424
1.000703812	3.153	.00100000655382	5.184
.9999548197	4.345	.000099999526381	5.325
1.000000998378	6.001	.0000100000114824	5.940
AVERAGE =	3.349	AVERAGE =	5.922

SIMEX — 235 — EXAMPLE 5

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.74226	.129	.999966	4.469
-.313568	-.118	.100063	3.201
1.44267	.354	.00997838	2.665
.944463	1.255	.00100276	2.559
1.00294	2.532	.0000998520	2.830
.999945	4.260	.0000100028	3.553
AVERAGE =	1.402	AVERAGE =	3.213

STAT20 — 235 — EXAMPLE 6

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
4.70801	-.569	1.00006	4.222
-6.48121	-.874	.0998837	2.934
3.72065	-.435	.010042	2.377
.638874	.442	.000994436	2.255
1.01997	1.700	.000100307	2.513
.999609	3.408	.00000999400	3.222
AVERAGE =	.612	AVERAGE =	2.920

STAT21 — 235 — EXAMPLE 7

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
2.089	-.037	1.00003	4.523
-1.11166	-.325	.0999349	3.186
1.75217	.124	.0100234	2.631
.901511	1.007	.000996913	2.510
1.00539	2.268	.000100170	2.770
.999895	3.979	00000999668	3.479
AVERAGE =	1.169	AVERAGE =	3.183

DETAILS FOR TABLE 3 -- DOUBLE PRECISION (16 DIGITS)

BMD05R — 7094 — EXAMPLE 1

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0000003	6.523	.99999998	7.699
.99999920	6.097	.10000004	6.398
1.0000002	6.699	.0099999798	5.695
.99999996	7.398	.0010000031	5.509
.9999999	7.000	.000099999792	5.682
.99999999	8.000	.000010000004	6.398
AVERAGE =	6.953	AVERAGE =	6.230

DPVMTX — 1107 — EXAMPLE 2

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.000000206882520	6.684	1.000000000006017	11.221
.9999995965200948	6.394	.09999999998899705	9.958
1.000000145134836	6.838	.01000000000383598	9.416
.9999999808270597	7.717	.0009999999995039349	9.304
1.000000001057576	8.976	.0001000000000269390	9.570
.9999999999793303	10.685	.00000999999999479781	10.284
AVERAGE =	7.882	AVERAGE =	9.959

DETAILS FOR TABLE 4 -- DOUBLE PRECISION (18 DIGITS)

ALSQ — 1108 — EXAMPLE 1

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.000000000005630	11.249	1.00000000000000000	18.000
.9999999999933983	11.180	.099999999999999140	15.065
1.000000000002151	11.667	.010000000000000339	14.470
.9999999999997248	12.560	.00099999999999995720	14.368
1.000000000000015	13.824	.00010000000000000224	14.650
.9999999999999997	15.520	.0000099999999999999580	15.376
AVERAGE =	12.667	AVERAGE =	15.322

BMD02R — 1108 — EXAMPLE 2

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.9999999968749762	8.505	1.000000000000007	14.155
1.000000006749235	8.171	.09999999999998616	12.859
.9999999974683392	8.597	.01000000000000481	12.318
1.000000000342994	9.465	.0009999999999993756	12.205
.9999999999807494	10.716	.0001000000000000339	12.470
1.000000000000381	12.419	.00000999999999999342	13.182
AVERAGE =	9.645	AVERAGE =	12.865

BMD05R — 1108 — EXAMPLE 3

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.00000000634827302	8.197	1.00000000000007691	13.114
.999999987016701379	7.887	.0999999999998442950	11.808
1.00000000476533960	8.322	.0100000000000568478	11.245
.999999999362724245	9.196	.00099999999999992425020	11.121
1.00000000003545420	10.450	.00010000000000000420311	11.376
.999999999999302617	12.157	.0000099999999999999174930	12.084
AVERAGE =	9.368	AVERAGE =	11.791

DPVMTX — 1108 — EXAMPLE 4

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.9999999971390853	8.543	.9999999999999998	15.693
1.000000005557233	8.255	.1000000000000033	13.481
.9999999980049357	8.700	.0099999999999983[illegible]	12.790
1.000000000263132	9.580	.0010000000000002[illegible]	12.607
.9999999999855033	10.839	.0000999999999999[illegible] 7	12.826
1.000000000000283	12.548	.00001000000000000 1	13.509
AVERAGE =	9.744	AVERAGE =	13.484

DETAILS FOR TABLE 4 -- DOUBLE PRECISION (18 DIGITS)

LSTSQ 1108 EXAMPLE 5

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.999999999999999464	15.269	1.00000000000000000	18.000
.999999999999943084	13.245	.0999999999999999950	16.284
1.00000000000004282	13.368	.0100000000000000021	15.683
.999999999999991067	14.049	.00099999999999999710	15.538
1.00000000000000068	15.169	.000100000000000000017	15.771
.99999999999999985	16.761	.0000099999999999999970	16.480
AVERAGE =	14.643	AVERAGE =	16.293

MATH-PACK 13.5, ORTHLS, ORTHOGONAL POLYNOMIAL CURVE FITTING 1108 EXAMPLE 6

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.999999999992709829	11.137	.999999999999999901	15.997
1.0000000001946887	10.711	.100000000000000225	14.648
.999999999991466379	11.069	.0099999999999988980	13.958
1.00000000000133404	11.875	.0010000000000001872	13.728
.999999999999915178	13.071	.00009999999999999987360	13.898
1.00000000000000188	14.727	.0000100000000000000292	14.535
AVERAGE =	12.098	AVERAGE =	14.461

ORTHO 1108 EXAMPLE 7

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.999999999997051246	11.530	.999999999999999946	16.242
1.00000000000051159	12.291	.0999999999999999970	16.488
1.00000000000034817	12.458	.0100000000000000078	15.109
.999999999999897859	12.991	.00099999999999998330	14.777
1.00000000000000760	14.119	.000100000000000000121	14.918
.999999999999999820	15.740	.0000099999999999999720	15.550
AVERAGE =	13.188	AVERAGE =	15.514

ORTHO, WITH RE-ORTHOGONALIZATION OMITTED 1108 EXAMPLE 8

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.999999858679196051	6.850	.999999999998151750	11.733
1.00000031785242527	6.498	.100000000004101749	10.387
.999999878054865120	6.914	.0099999999843660520	9.806
1.00000001679285067	7.775	.0010000000021433129	9.669
.999999999045590670	9.020	.000099999999878591650	9.916
1.00000000001908297	10.719	.0000100000000002421209	10.616
AVERAGE =	7.963	AVERAGE =	10.354

ORTHOL 1108 EXAMPLE 9

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
.999999999999754768	12.610	1.00000000000000000	18.000
1.00000000000255572	11.592	.100000000000000036	15.444
.999999999999179468	12.086	.0099999999999998660	14.873
1.00000000000011276	12.948	.0010000000000000191	14.719
.999999999999993318	14.175	.00009999999999999998880	14.950
1.00000000000000014	15.863	.0000100000000000000023	15.640
AVERAGE =	13.212	AVERAGE =	15.604

POLRG, IBM SYSTEM/360 SCIENTIFIC SUBROUTINE PACKAGE 1108 EXAMPLE 10

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.00000011110114428	6.954	1.00000000000136645	11.864
.999999990457514712	8.020	.0999999999999411990	12.231
1.00000000321449412	8.493	.0100000000000305940	11.514
.999999999544922239	9.342	.00099999999999995547370	11.351
1.00000000002322160	10.634	.000100000000000227181	11.644
.999999999999491835	12.294	.0000099999999999999416480	12.234
AVERAGE =	9.290	AVERAGE =	11.806

DETAILS FOR TABLE 4 -- DOUBLE PRECISION (18 DIGITS)

STAT-PACK 9.1, RESTEM, STEPWISE MULTIPLE REGRESSION 1108 EXAMPLE 11

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.00000000453928806	8.343	1.00000000000004872	13.312
.999999990379828084	8.017	.0999999999999056810	12.025
1.00000000358299878	8.446	.0100000000000336338	11.473
.999999999516629162	9.316	.000999999999995591090	11.356
1.00000000002705048	10.568	.000100000000000241649	11.617
.999999999999465664	12.272	.0000099999999999530180	12.328
AVERAGE =	9.494	AVERAGE =	12.019

DETAILS FOR TABLE 5 -- SINGLE PRECISION (8 DIGITS), WITH INNER PRODUCTS ACCUMULATED IN DOUBLE PRECISION (18 DIGITS)

ALSQ 1108 EXAMPLE 1

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0366969	1.435	1.0000001	7.000
.99202651	2.098	.099999869	5.883
.99865498	2.871	.010000017	5.770
1.0003119	3.506	.00099999901	6.004
.99998119	4.726	.00010000003	6.523
1.0000004	6.398	.0000099999999	8.000
AVERAGE =	3.506	AVERAGE =	6.530

LSTSQ 1108 EXAMPLE 2

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0000000	8.000	.99999999	8.000
.99999999	8.000	.10000004	6.398
1.0000000	8.000	.0099999798	5.695
1.0000000	8.000	.0010000032	5.495
1.0000000	8.000	.000099999794	5.686
1.0000000	8.000	.000010000004	6.398
AVERAGE =	8.000	AVERAGE =	6.279

ORTHO 1108 EXAMPLE 3

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0027425	2.562	1.0000003	6.523
.99674778	2.488	.099999898	5.991
1.0013667	2.864	.010000011	5.959
.99983171	3.774	.00099999974	6.585
1.0000096	5.018	.000099999980	6.699
.99999981	6.721	.000010000001	7.000
AVERAGE =	3.904	AVERAGE =	6.459

DETAILS FOR TABLE 6 -- MULTIPLE PRECISION INTEGER ARITHMETIC

SOLVER 1108 EXAMPLE 1

BETA-HAT (Y1)	COUNT	BETA-HAT (Y2)	COUNT
1.0000000000000000	18.000	.999999999999999995	17.159
1.0000000000000000	18.000	.0999999999999999995	17.187
1.0000000000000000	18.000	.00999999999999999996	17.169
1.0000000000000000	18.000	.000999999999999999996	17.294
1.0000000000000000	18.000	.0000999999999999999996	17.276
1.0000000000000000	18.000	.0000100000000000000000	18.000
AVERAGE =	18.000	AVERAGE =	17.347

12. References

[1] Anscombe, F. J., Topics in the investigation of linear relations fitted by the method of least squares, Journal of the Royal Statistical Society, Series B, **29**, p. 1–52, 1967.

[2] Bauer, F. L., Elimination with weighted row combinations for solving linear equations and least squares problems, Numerische Mathematik **7**, p. 338–352, 1965.

[3] Björck, Ake, Solving linear least squares problems by Gram-Schmidt orthogonalization, BIT (Nordisk tidskrift for informations-behandling), **7**, p. 1–21, 1967.

[4] Björck, Ake, Iterative refinement of linear least squares solutions, I, BIT **7**, p. 257–278, 1967.

[5] Björck, Ake, Iterative refinement of linear least squares solutions, II, BIT **8**, p. 8–30, 1968.

[6] Björck, Ake, and Golub, Gene, ALGOL Programming, Contribution No. 22: Iterative refinement of linear least square solutions by Householder transformation, BIT **7**, p. 322–337, 1967.

[7] Bright, J. W., and Dawkins, G. S., Some aspects of curve fitting using orthogonal polynomials, Industrial and Engineering Chemistry Fundamentals **4**, p. 93–97, 1965.

[8] Businger, Peter, and Golub, Gene H., Linear least squares solutions by Householder transformations, Numerische Mathematik **7**, p. 269–276, 1965.

[9] Cameron, J. M., Some examples of the use of high speed computers in statistics, Proceedings of the First Conference on the Design of Experiments in Army Research, Development and Testing, Office of Ordnance Research, Report No. 57–1, p. 129–135, 1957.

[10] C-E-I-R Multi-Access Computer Services, Library programs documentation, MAC 71–7–1, Nov. 17, 1967. Addendum, MAC 71–7–1, A 12–368, Mar. 1, 1968.

[11] Crisman, P. A., (ed.), The compatible time-sharing system: A programmer's guide, second edition, M. I. T. Computation Center, M. I. T. Press, Massachusetts Institute of Technology, Cambridge, Mass., 1965.

[12] Davis, Philip J., Orthonormalizing codes in numerical analysis, ch. 10 in [39], 1962.

[13] Davis, Philip, and Rabinowitz, Philip, A multiple purpose orthonormalizing code and its uses, Journal of the Association for Computing Machinery **1**, p. 183–191, 1954.

[14] Davis, Philip J., and Rabinowitz, Philip, Advances in orthonormalizing computation, in Franz L. Alt (ed.), Advances in Computers, p. 55–133, Vol. **2**, (Academic Press, New York, 1961).

[15] Dixon, W. J., (ed.), BMD Biomedical Computer Programs, Health Sciences Computing Facility (University of California, Los Angeles, 1964. Revised 1965 and 1967).

[16] Efroymson, M. A., Multiple regression analysis, in Anthony Ralston and Herbert S. Wilf (editors), Mathematical Methods for Digital Computers, Ch. 17, Vol. **1**, pp. 191–203 (John Wiley & Sons, New York, N.Y. 1960).

[17] Fettis, Henry E., and Caslin, James C., Eigenvalues and eigenvectors of Hilbert matrices of order 3 through 10, Mathematics of Computation **21**, p. 431–441, 1967.

[18] Forsythe, George E., Generation and use of orthogonal polynomials for data-fitting with a digital computer, Journal of the Society for Industrial and Applied Mathematics **5**, p. 74–88, 1957.

[19] Forsythe, George E., Today's computational methods of linear algebra, SIAM Review **9**, p. 489–515, 1967.

[20] Freund, R. J., A warning of roundoff errors in regression, The American Statistician **17**, p. 13–15, 1963.

[21] Golub, G., Numerical methods for solving linear least squares problems, Numerische Mathematik **7**, p. 206–216, 1965.

[22] Golub, G. H., and Wilkinson, J. H., Note on the iterative refinement of least squares solution, Numerische Mathematik **9**, p. 139–148, 1966.

[23] Hilsenrath, Joseph, Ziegler, Guy G., Messina, Carla G., Walsh, Philip J., and Herbold, Robert, OMNITAB, A Computer Program for Statistical and Numerical Analysis, National Bureau of Standards Handbook 101, U. S. Government Printing Office, Washington, D.C., 1966. Reissued Jan. 1968, with corrections.

[24] IBM Application Program, System/360 Scientific Subroutine Package (360A–CM–03X) Version III, Application Description, H20–0166–5, International Business Machines Corp., 1968.

[25] IBM Application Program, System/360 Scientific Subroutine Package (360A–CM–03X) Version III, Programmer's Manual, H20–0205–3, International Business Machines Corp., 1968.

[26] IBM Systems Reference Library, Catalog of programs for IBM 704–709–7040–7044–7090 and 7094 Data Processing Systems, Form C20–1604–3, International Business Machines Corp., December 1965.

[27] Jordan, T. L., Experiments on error growth associated with some linear least-squares procedures, Mathematics of Computation **22**, p. 579–588, 1968.

[28] Kurtz, Thomas E., personal communication, 1968.

[29] Longley, James W., An appraisal of least squares programs for the electronic computer from the point of view of the user, Journal of the American Statistical Association **62**, p. 819–841, 1967.

[30] Longley, James W., personal communication, 1968.

[31] Martin, R. S., Peters, G., and Wilkinson, J. H., Iterative refinement of the solution of a positive definite system of equations, Numerische Mathematik **8**, p. 203–216, 1966.

[32] Miller, James R., On-line analysis for social scientists, MAC–TR–40, Project MAC, Massachusetts Institute of Technology, Cambridge, Mass., 1967.

[33] Moler, Cleve B., Iterative refinement in floating point, Journal of the Association for Computing Machinery **14,** p. 316–321, 1967.

[34] Newman, Morris, Matrix computations, ch. 6 in [39], 1962.

[35] Newman, Morris, Solving equations exactly, J. Res. NBS **71B** (Math. and Math. Phys.), No. 4, 171–179, 1967.

[36] Rhomberg, Rudolf R., Boissonneault, Lorette, and Harris, Leonard, A general purpose computer program for data processing and multiple regression (DAM), Revision 3, Research and Statistics Department, International Monetary Fund, Washington, D.C., Sept. 1965.

[37] Stiefel, Eduard L., An Introduction to Numerical Mathematics, (Academic Press, New York, 1963).

[38] Taussky, Olga, Some topics concerning bounds for eigenvalues of finite matrices, ch. 8 in [39], 1962.

[39] Todd, John (ed.), Survey of Numerical Analysis (McGraw-Hill Book Co., New York, 1962).

[40] Univac 1108 Multi-Processor System, MATH-PACK Programmers Reference, UP–7542, Univac Division of Sperry Rand Corporation, 1967.

[41] Univac 1108 Multi-Processor System, STAT-PACK Programmers Reference, UP–7502, Univac Division of Sperry Rand Corporation, 1967.

[42] Walsh, Philip J., Algorithm 127, ORTHO, Communications of the ACM **5,** p. 511–513, 1962.

[43] Wilkinson, J. H., Rounding Errors in Algebraic Processes (Prentice-Hall, Englewood Cliffs, N.J., 1963).

[44] Wilkinson, J. H., The Algebraic Eigenvalue Problem (Clarendon Press, Oxford, 1965).

[45] Wilkinson, J. H., The solution of ill-conditioned linear equations, in Anthony Ralston and Herbert S. Wilf (editors), Mathematical Methods for Digital Computers, ch. 3, p. 65–93, Vol. **2,** (John Wiley & Sons, New York, 1967).

[46] Zellner, A., and Thornber, H., Computational accuracy and estimation of simultaneous equation econometric models, Econometrica **34,** pp. 727–729, 1966.

(Paper 73B2–289)