CS13

COMPUTATION OF THE PSEUDOINVERSE OF A MATRIX OF UNKNOWN RANK

BY

VICTOR PEREYRA and J. B. ROSEN

TECHNICAL REPORT CS13
SEPTEMBER 1, 1964

COMPUTER SCIENCE DIVISION
School of Humanities and Sciences
STANFORD UNIVERSITY

COMPUTATION OF THE PSEUDOINVERSE OF A MATRIX OF UNKNOWN RANK [**]

by

Victor Pereyra [*] and J. B. Rosen

Abstract

A program is described which computes the pseudoinverse, and other related quantities, of an m X n matrix A of unknown rank. The program obtains least square solutions to singular and/or inconsistent linear systems Ax = B, where $m \leq n$ or $m > n$ and the rank of A may be less than min(m,n). A complete description of the program and its use is given, including computational experience on a variety of problems.

---

[*] On leave from Dept. of Mathematics and Computation Center, Buenos Aires University, Argentina.

 Prepared under NASA Grant Ns G 565 at Stanford University. Stanford California

I. Introduction.

A method for the computation of the pseudoinverse, and other related quantities, corresponding to an $m \times n$ matrix A of unknown rank r, has recently been described [5]. The method determines the pseudoinverse A^+ of A and a related matrix $A^\#$. The pseudoinverse has the property that given the linear system $Ax = b$, the solution $x_m = A^+b$ satisfies $\|Ax_m - b\| < \|Ax - b\|$ for all x, and $\|x_m\| \leq \|x\|$ for all x such that $\|Ax_m - b\| = \|Ax - b\|$. The minimum basic solution $x_b = A^\#b$ has the property that $\|Ax_b - b\| \leq \|Ax - b\|$ for all x, and x_b has at most r non-zero elements.

The computational difficulty for this problem arises primarily because the rank r is not known. In particular, it may be difficult to assign the correct rank if one or more of the singular values of A are small but non-zero [3]. Several other recent papers [1], [2], [4], on the computation of the pseudoinverse have not considered this important practical question.

The approach used here to handle this difficulty can be summarized as follows. The desired matrices $A^\#$ and A^+ are formed from a matrix B, which consists of linearly independent columns selected from A'.

We would like to determine B so that it spans the same space as A, in which case B will contain r columns. Suppose we have a matrix B_q with q linearly independent columns selected from A, (where $q < r$) and the corresponding approximation A_q^+ to A+. Adding another linearly independent column of A to $B_{4'}$ giving B_{q+1}, should give an improved approximation A_{q+1}^+ to the pseudoinverse. However, due to roundoff error in the calculation it may turn out for an ill-conditioned system that the new approximation is actually worse in the sense that $\|AA_{q+1}^+ - I\| > \|AA_q^+ - I\|$. Such a test is made in the pseudoinverse determination with the result that the effective-rank of A (the number of columns in B) is the maximum possible consistent with minimizing the error $\|AA^+ - I\|$.

A closely related aspect of the method used here to compute the pseudoinverse is what might be called its "smoothing" property. In many practical situations one would like to obtain a solution to a linear system which is stable in the sense that small changes in the matrix elements do not cause large changes in the solution vector. In general, the solution $x = A^+b$, where A^+ is the true pseudoinverse, will not behave smoothly. In fact, the norm of x will increase without bound as a singular value of A approaches zero. This difficulty can be eliminated by imposing a predetermined upper bound on the norm of $(B'B)^{-1}$. This is accomplished by estimating the effect of adding a new column of A to B_q and only adding this new column to B_q if it does not cause any element of A_{q+1}^+ to exceed the bound. Details of this selection procedure and the manner in which it depends on the choice of the bound SUPER is discussed in the next section.

In Section 3 the use of the Algol program, written to perform this algorithm, is described and suggested values for the input parameters are given. The program use is illustrated by means of a sample problem. A large number of problems have been solved using this program. Several different kinds of tests have been performed:

a) Very ill-conditioned matrices like the segments of the Hilbert matrix [6] have given a clear example of the smoothing property of the method.

b) Random rectangular matrices of random sizes have been generated and the pseudoinverse have been computed. The sizes were allowed to vary between 1 and 25. In all the cases the results were satisfactory.

c) Same as in b) but with random ranks. In every case the rank was correctly determined by the program.

d) Random matrices of specified size covering a range of values of m and n were run in order to obtain time estimates for different size problems.

e) A number of least-square problems, i.e., with $m \gg n$ and only one right-hand side.

f) A variety of matrices for which an independent check on the accuracy of the solution was available.

Tests b) through f) showed that in reasonable problems in which the rank is well determined the program will work very well, while a) has shown that in very ill-conditioned cases the smoothing property of the method is effective.

These test results are discussed more fully in Section 4. The notation used in [5] will also be followed here.

Details of storage requirements are given in Section 5.

A copy of the program appears in the Appendix.

II. Program Description.

The method used to compute $A^{\#}$ and A^{+} from B is essentially that given in Section 2 of [5]. For convenience we will repeat the key relations here. The pseudoinverse of the m x r matrix B of rank r is given by

$$B^{+} = (B'B)^{-1} B' \tag{2.1}$$

The non-zero rows of the n x m matrix $A^{\#}$ then consists of the corresponding rows of B^{+}. An r x n matrix of rank r is also obtained from B^{+} according to

$$C = B+A \tag{2.2}$$

Note that, if B contains all the independent columns of A, then A = BC. Finally, A^{+} is obtained from C and B^{+} by

$$A^{+} = C'(CC')^{-1} B^{+} \tag{2.3}$$

The determination of B is based on the algorithm of Section 3 in [5], using the more sophisticated selection procedure described below.

The program consists essentially of two parts. One part has all the input-output and the other is a PROCEDURE called PSEUDOINVER which may also be used separately as a part of other programs'. The program solves the matricial problem,

$$AX=RHS \tag{2.4}$$

where RHS is a matrix containing several right hand sides.

a) The first part of PSEUDOINVER normalizes the matrix A by scaling each column so that its Euclidean norm is equal to one. The normalization constants are saved in order to get back to the original problem.

The search for independent columns of A is then made to determine the matrix B, according to the formulas described in section 3 of [5]. At this stage, the condition for a vector to be accepted as independent of the ones already included in the basis is that α, the square of the norm of the projection on the orthogonal subspace to that basis, be less than a quantity ORTP, which is an input parameter. Later we will discuss the appropriate choice of ORTP and the other parameters appearing in the program.

As the columns chosen in this fashion might not necessarily be the first columns of A, a record is kept of the column number of the accepted vectors,

After all the columns have been inspected two situations can arise; either all the n columns of A have been accepted or some have been rejected. In the first case we have finished and the computations indicated at the beginning of this section are performed to get A+, $A^{\#}$, X_b and X_m. Other computed quantities are the residuals, NXM = $\|AX_m - RHS\|$ NXB = $\|AX_b - RHS\|$ corresponding to X_m and X_b, and EST = $\|BC - A\|$.

EST would be zero if the computation were performed exactly; in general EST will be very small for well-conditioned matrices and will increase with the ill-conditioning or if an almost dependent column is added to B.

If only $q < n$ columns of A are selected for inclusion in B then the basis thus constructed is called B_q and the second part of PSEUDOINVER is called.

b) The projections on the subspace orthogonal to B_q are computed for all the rejected columns. The Euclidean norm of each projection is computed,

(2.5) $$\alpha_j = \|(I - B_q B_q^+) a_j\|^2$$

and the column corresponding to the maximum α_j (the most independent one) is stored in SAV.

c) A test is now performed which is based upon an estimation of the norm that $(B'_{q+1} B_{q+1})^{-1}$ would have if we were to include SAV in the basis,, The norm used is $\|A\| = \max_i \sum_{j=1}^{n} |a_{ij}|$ and the estimate is derived from the formula,

(2.6) $$(B'_{q+1} B_{q+1})^{-1} = \left(\begin{array}{c|c} (B'_q B_q)^{-1} & 0 \\ \hline 0 & 0 \end{array}\right) + \alpha_{q+1}^{-1} \begin{pmatrix} u_q \\ \hline -1 \end{pmatrix} (u'_q \mid -1)$$

where $u_q = B_q^+$ SAV and α_{q+1} is the square of the norm of the projection of SAV on the orthogonal subspace to that spanned by B_q. Then

$$\|(B'_{q+1} B_{q+1})\| \leq \text{ESTIM} = \|(B'_q B_q)^{-1}\| + \alpha_{q+1}^{-1} (1 + \sqrt{q})$$

If ESTIM is larger than SUPER (an input parameter) then SAV is rejected and B_q is taken as the final B .

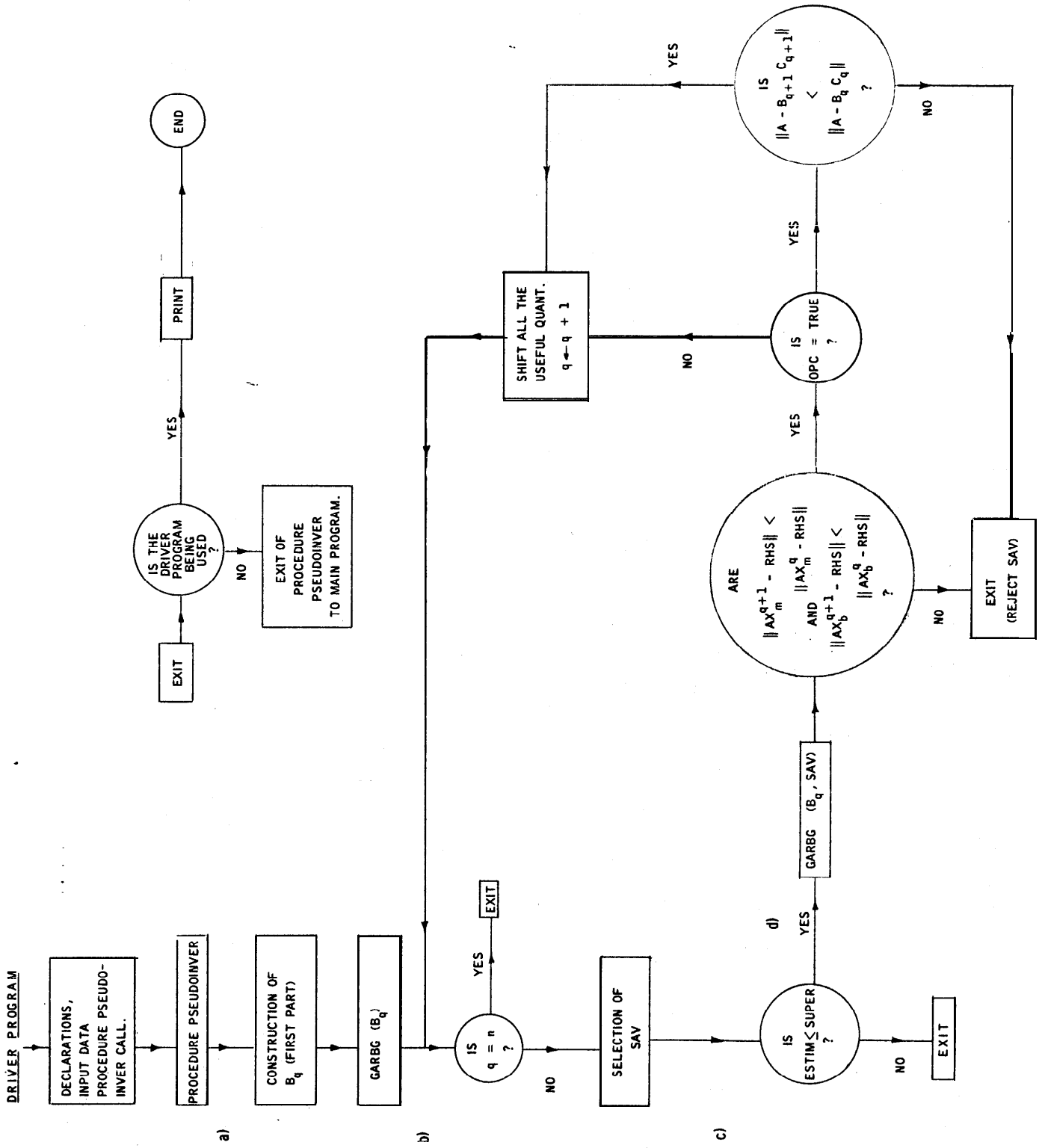
This test avoids large elements in the pseudoinverse and gives the smoothing property discussed in the introduction.

d) If the test in c) is passed then the PROCEDURE GARBG, which computes all the matrices and quantities mentioned at the beginning of this section, is called and a second test is made. GARBG is used again, now with the basis B_q plus the column SAV . The test consists in comparing the values of $\|AX_m - RHS\|$, $\|AX_b - RHS\|$ and $\|BC - A\|$ obtained with one basis, with the corresponding ones obtained with the incremented basis. If all these values for B_{q+1} are smaller than for B_q then SAV is definitely

accepted. After shifting all the useful quantities, part (b) is repeated for the new basis B_{q+1} and so on, until either an exit is provided for one of the tests or the columns of A are exhausted. All the scalar products are performed in double precision. The block diagram in Fig. I shows the most essential parts of the program.

It is worth noting that this strategy has been dictated by the problem itself and achieves the best numerical pseudoinverse possible using the method of [5] and taking into account the numerical roundoff error of the computer being used. This strategy takes advantage of the step by step algorithm for determining B, and constructs an independent basis, the degree of independence being determined by the parameter ORTP. By picking the most independent vector among the remaining ones, and checking to see if this decreases the residuals (by taking this vector in the basis) we are answering in a direct manner the two questions: how many columns of A do we need to minimize the residual? and, among all the possible sets of independent columns,which set gives the best representation of the pseudoinverse?

Fig. 1 - BLOCK DIAGRAM



III. Program Use.

As described in the previous section, several parameters are needed besides the matrix A . Now-we will explain the use and possibilities of these parameters.

Input.

M (integer) number of rows in A .

N (integer) -number of columns in A .

T (integer) number of right-hand sides.

OPC (Boolean) If OPC is equal to 1 then the program will compute the matrices A^+ and $A^\#$, and the right-hand side RHS = I(m x m) will be automatically provided.

Moreover, OPC decides if in the test described in Section 2,d) the quality of the representation (A = BC) is controlled. That test is done only if OPC = TRUE . If OPC is equal to FALSE then RHS an M x T matrix has to be provided and the program will compute $X_B = A^\#.RHS$, and $X_M = A^+.RHS$, matrices that will be printed out instead of A^+ and $A^\#$.

SUPER (real) It is the SUPER of Section 2,c). If an upper bound for the elements of A^+ is known then SUPER can be set to this bound to take advantage of the smoothing property of this method; otherwise it is suggested that 10^{14} be used. It should be noted that, in general, a larger value than 10^{14}

will increase computing time by throwing unnecessary decisions into the test of Section 2,d). On the other hand, much smaller values may completely eliminate from further consideration some columns which could be used to decrease the error,

ORTP (real) This parameter was described in Section 2, a). Small values for ORTP (around 10^{-4}) in general will accelerate the process because the first part (construction of a basis of strongly independent column) is the fastest and as many columns as possible should be accepted there. Nevertheless, there are at least two cases in which a more careful choice of ORTP may be important. If higher precision in the answers is desired (at the cost of increased computing time), then a larger value of ORTP should be used, say 0.05. This will allow the second part of the program to choose "better" columns.

The other delicate case occurs when the matrix is very ill-conditioned and the rank is therefore not well defined. Here the use of a relatively'large ORTP is important. Again values around 0.05 are recommended.

Summarizing, in a reasonable, well behaved problem a recommended set of parameters is:

$$\text{SUPER} = 10^{14} \quad , \quad \text{ORTP} = 10^{-4} \quad .$$

If the representation becomes very bad ($\|A - BC\|$ too large), the problem is not well behaved and more burden should be passed on to the second and safer test by increasing SUPER and decreasing ORTP .

If the user has information that certain variables are more significant than others, this information can be used by ordering the matrix A so that the columns of A corresponding to these variables appear first. This will insure that these columns are considered first for inclusion in the basis B .

If the complete program is used, then only the numerical data have to be punched,, This is done in the following way.

As all the read statements are in the FREE FIELD form, available in the EXTENDED ALGOL for the B5000 at Stanford, no special format is necessary. Numbers can be punched in any format, needing just one space in between to separate them.

1st. card:	M	N	T	O P C	SUPER	O R T P
for instance	10	10	10	1	@14	0.001

Next cards will contain the matrix A punched by rows. As each READ asks for a whole row, care must be taken not to mix different rows in the same card.

Finally, if 0 P C = 0 the right-hand sides (RHS) have to be provided and are read by columns. Each new column must be started on a new card, so that there will be at least T cards required for the RHS .

If the PROCEDURE is used separately, then all these quantities are input parameters (with the same names as above).

A complete sample input is given by,

```
        { 3      3      1     0      @14     0.001
        { 1.3    2            -5"
A       { 4      1            0
        { -1     -1.3         @3

RHS     { 1      2            2.1
```

output

All the matrices printed out by the program will have the following format:

Eight columns per line, each number in floating point with 6 significant digits. If the matrix is more than eight columns wide, then successive blocks will be printed in new pages. All the rows are printed together.

The output is described now in the order in which it will occur.

First the matrix A is printed out.

Then, if 0 P C = 0 , the right-hand sides are printed out.

Norm of (BC - A) .

The residuals $\|Ax_m^{(i)} - RHS^{(i)}\|$, $\|Ax_b^{(i)} - RHS^{(i)}\|$.

If 0 P C = 1 , then the matrix pseudoinverse is printed out with the format explained above; also in this case the non-zero rows of the matrix $A^{\#}$ are printed, each of them with a heading: ROW NUMBER.... .

If O P C = O , then instead of these two last matrices, the minimum and basic solutions are printed out. As an example, we give the output for the problem:

$$\begin{pmatrix} 9 & 21 \\ 21 & 49 \end{pmatrix} X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

PSEUDOINVERSION OF THE MATRIX A . A IS 2 x 2 .

g. 00000 @ + 00 2.10000 @ + 01

2.10000 @ + 01 4.90000 @ + 01

[PAGE]

NORM OF (BC - A) 2.37582 @ - 12

RESIDUAL FOR XM

9.19145 @ - 01 3.93919 @ - 01

RESIDUAL FOR XB

9.19145 @ - 01 3.93919 @ - 01

[PAGE]

MATRIX APSEUDOINVERSE

2.67532 @ -03 6.24314 @ - 03

6.24381 @ - 03 1.45794 @ - 02

[PAGE]

MATRIX ADAGGER

ROW NUMBER 1

1.72461 @ - 02 4.02332 @ - 02

END OF THE RUN

* * * * * * * * * * * * * * * * * * * *

The PROCEDURE PSEUDOINVER.

The call for this PROCEDURE is,

PSEUDOINVER (M, N, TI, OPC, SUPER, ORTP, A, RHS, EST, NXM, NXB,

APSEUDO, ADAGER, COF, XM, XB);

The first 8 parameters are input parameters and they have been described before. The only detail needed is: A(double real array [0:M, 0:N]), RHS [0:M, 0:TI];

OUTPUT PARAMETERS:

EST (real) Contains $\|BC - A\|$.

NXM, NXB (single real arrays [0:TI]) . They contain the residuals $\|AX_m - RHS\|$ and $\|AX_b - RHS\|$ respectively.

APSEUDO, ADAGER (Double real array [0:N, 0:M])

They contain the pseudoinverse of A and the matrix $A^{\#}$.

COF (Single integer array [0:N]) .

If COF[I] = 0 then both, the I^{th} row of $A^{\#}$ and X_b are different from zero, otherwise they are zero and that means the program has decided that the corresponding columns in A were linearly dependent with respect to-the current basis.

XM, XB (double real arrays, [0:N, 0:TI])

They contain the minimum and basic solutions.

IV. Test Problems.

a) Square segments of the Hilbert matrix have been tried, sizes varying between 3 and 10.

For $5 \leq n \leq 10$ the rank found in each case was 4 . The norm of the pseudoinverses remained below 10^3 while for the true pseudoinverse (the inverse in these cases) the norms ranged between 10^5 for $n=5$ and 10^{13} for $n=10$. The norm, $\|A - BC\|$ was around 10^{-5} for all cases.

As is well known, the ill-conditioning of the Hilbert matrix segments increase with their dimension. However, because of the smoothing property of the method a bounded and reasonably accurate representation for the pseudoinverse was always obtained.

b) Eighteen random matrices with random dimensions varying between 1 and 25 were generated and pseudoinverted. The norm $\|A - BC\|$ was always below 10^{-9} and the ranks were always found to be equal to min(m,n).

c) Given three random integers m, n and r in the interval [1, 25] a routine generated two random matrices, L (mxr) and R (r x n) . Multiplying them we obtained a matrix A (m x n) with rank at most equal to r [1] . With 20 matrices generated in this way, the results were similar to b) . In every case the rank r was correctly determined. For most of these cases the rank r was less than min(m,n), and of course was unknown for the program.

(1) This test was suggested by Professor Gene H. Golub.

d) For each pair of values (m,n) several random matrices were generated and pseudoinverted. Average values of $\|A-BC\|$ for these matrices with m=10,20,30 and n=10,20,30 are shown in Table I. For the same problems, average computation time on the Burroughs B5000 at Stanford Computation Center are shown in Table II.

In all these matrices the rank was the maximum possible, i.e., rank = min (m,n) and it was properly determined by the program.

TABLE I

m \ n	10	20	30
10	5×10^{-9}	9×10^{-9}	4.5×10^{-8}
20	3×10^{-10}	4×10^{-8}	2.9×10^{-7}
30	5.8×10^{-10}	6.9×10^{-9}	2.6×10^{-7}

$\|A - BC\|$

TABLE I I

m \ n	10	20	30
10	7.6	13.9	20.7
20	16.5	49	91
30	27.3	92.5	180.6

Comp. time in seconds

For a 40 x 40 matrix the answers were:

t = 413 sec. | $\|A - BC\| = 2.7 \times 10^{-7}$

rank = 40 .

e) A common problem in many branches of applied sciences is the least squares fit, and is therefore one of the most important applications for this program. A related feature of the program is that, by ordering the variables, the user will be able to test their independence and eventually to decide if his model is appropriate to the phenomena being investigated. This is done by ordering

the matrix A so that the first coefficients correspond to the more important variables. The program will attempt, to use these columns first to form the basis B . The necessity for such an ordering is clear from the fact that if we have n columns in A and the subspace spanned by these columns has dimension (n-p) then we can construct with these columns as many as $\binom{n}{n-p}$ linearly independent sets.

In Figures II and III are shown the results obtained by running the program with least squares type problems. Again the elements of the matrices were generated randomly. Fig. II shows computation times on the B5000 for different values of m and n=5, 10. Fig. III shows the norm $\|A - BC\|$ for the same problems.

f) Matrices with exact known inverses were tried obtaining good results and accuracy. Of course this program should not be used to invert a matrix which is known to be nonsingular and well-conditioned, because it will be around four times slower than an efficient matrix inverter. The program has also been used to obtain the pseudoinverse of singular and almost singular matrices steming from the discretization of integral equations of the first kind, and problems in pattern recognition.

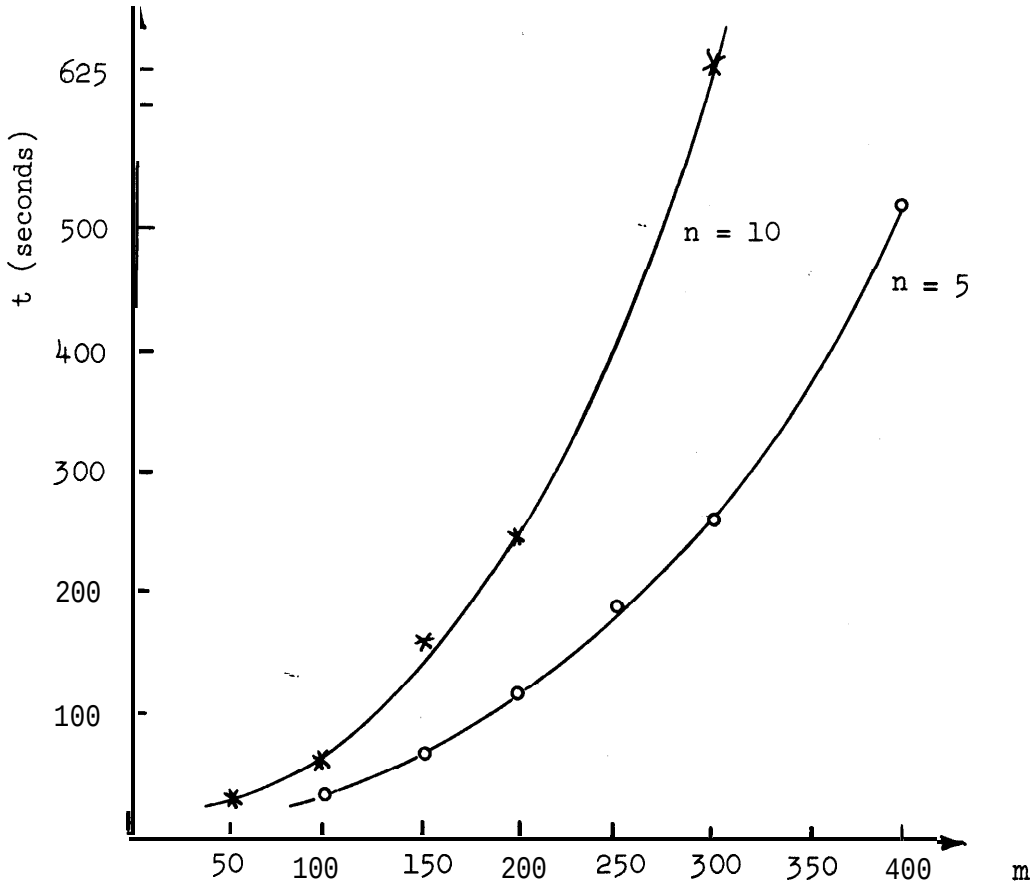


FIGURE II

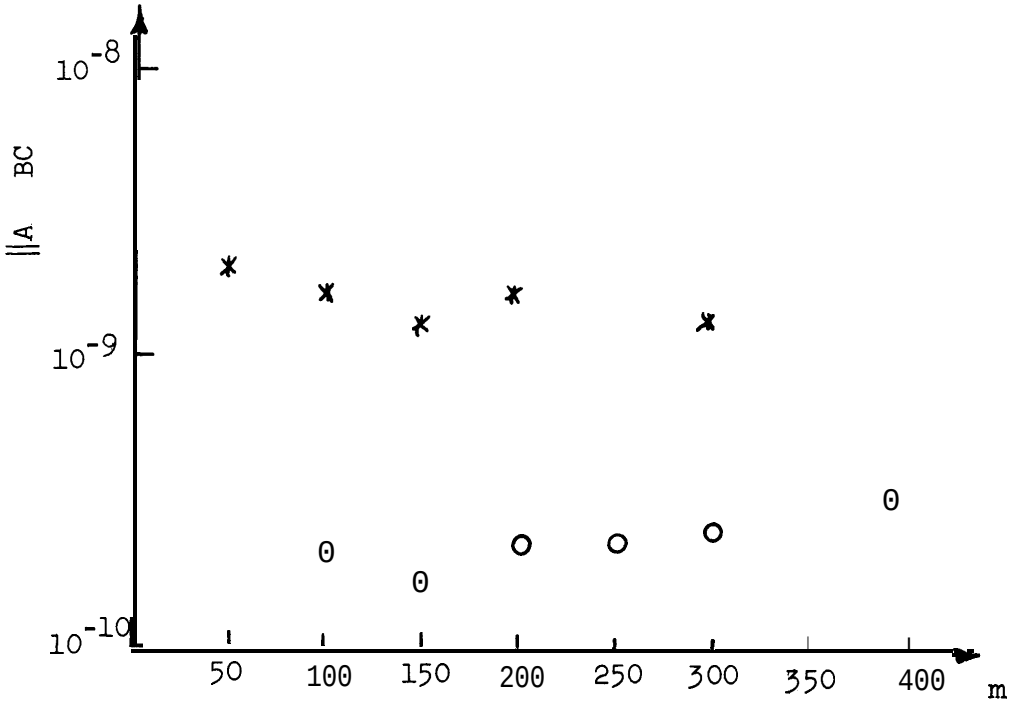


FIGURE III

V. Storage Requirements.

As all the array declarations are dynamical, the amount of storage depends on several parameters. If M, N, T are as before, and R is the final rank (number of accepted columns) then an estimate for the storage used in the PROCEDURE PSEUDOINVER is,

$$\text{Storage} \cong N^2 + 5MN + MR + 2NR + 2NT + \max\,(R^2, MN, MT)$$

the last term is present because in the PROCEDURE GARBG we have several independent blocks, and the storage corresponding to certain arrays is not simultaneously used.

If, as usual, R is not known, then it can be replaced by min(n,m) . If the complete program is used then additional storage is needed,

$$\text{Addit. storage} \cong 3MN + MT + 2NT$$

Computer time rapidly increases when abusive use of the drum is made. From the experience obtained with the test problems, it is suggested that the total storage be less than 40,000 words.

REFERENCES

[1] J. W. Blattner, Bordered Matrices, J. SIAM, 10, 528-536 (1962).

[2] J. C. G. Boot , The Computation of the Generalized Inverse of Singular or Rectangular Matrices, Amer. Math. Monthly, 70, 302-303 (1963).

[3] G. Golub and W. Kahan, Calculating the Singular Values and Pseudo-inverse of a Matrix, Stanford Univ. Technical Report CS8, (1964).

[4] A. Ben-Israel and S. J. Wersan, An Elimination Method for Computing the Generalized Inverse of an Arbitrary Complex Matrix, J. ACM, 10, 532-537 (1963).

[5] J. B. Rosen, Minimum and Basic Solutions to Singular Linear Systems. J. SIAM, 12, 156-162 (1964).

[6] J. Todd, The Condition of the Finite Segments of the Hilbert Matrix, NBS App. Math. Series, 39, 109-116 (1954).

APPENDIX

AN EXTENDED ALGOL PROGRAM TO COMPUTE THE PSEUDOINVERSE OF AN M x N REAL MATRIX AND OTHER RELATED QUANTITIES.

```
BEGIN   COMMENT   PSEUDOINVERSE OF A M×N MATRIX OF UNKNOWN RANK;
      INTEGER  M,N,TI,I,J ; REAL SUPER,EST,TPD,ORTP ;
      BOOLEAN  OPC ; LABEL OVER,FIU;
   COMMENT   DRIVER PROGRAM. FIRST PARAMETERS ARE READ IN,
             AND USED FOR FURTHER DYNAMICAL ARRAY DECLARATIONS;
   OVER:  READ (M,N,TI,OPC,SUPER,ORTP) [FIU];
   BEGIN   INTEGER ARRAY COF[0:N];
      ARRAY  A[0:M,0:N],RHS[0:M,0:TI],XM,XB[0:N,0:TI],NXM,NXB[0:TI],
             APSEUDO,ADAGER[0:N,0:M] ; LABEL  NOPR,NOPR1 ;
      FORMAT   PRMAT(( 8E14.5)///),
         TIT1("PSEUDOINVERSION OF THE MATRIX A"///"A IS",I3,"x",I3/),
         TIT7("NORM OF (BC-A)"//X3,E15.5/"RESIDUAL FOR XM"/(8E15.5/)),
         TIT8("MATRIX A PSEUDOINVERSE"/),TIT9("MATRIX A DAGGER"/),
         TIT10(X6,"XM"/),TIT20(//X6,"XB"/),TITL38(/"ROW NUMBER",I3/),
         TIT77(/"RESIDUAL FOR XB "/(8E15.5/)),SOL(8E15.6/),
         ENDE(//"END OF THE RUN"//X15,"* * * * * * * * * *" ////);
      PROCEDURE  PRT(A,M,N) ;
         INTEGER  M,N; ARRAY A[0,0];
   COMMENT   PRT PRINT OUT THE M×N MATRIX A;
      BEGIN   FORMAT TITL46(/ 8E14.5); INTEGER SE,K,R;
            SE ← N DIV 8   ; K ← N MOD 8 ;
            FOR R←0 STEP 1 UNTIL SE-I DO
         BEGIN FOR I←1 STEP 1 UNTIL M DO
            WRITE (TITL46,FOR J←1 STEP 1 UNTIL 8 DO A[I,8 ×R+J]);
            WRITE ([PAGE])
          END ; IF K≠0 THEN FOR I←1 STEP 1 UNTIL M DO
                   WRITE(TITL46,FOR J←1 STEP 1 UNTIL'K DO A[I,8 ×SE+J]);
                   WRITE ([PAGE])
      END PRT ;
      PROCEDURE    PSEUDOINVER (M,N,TI,SUPER,OPC,A,RHS,ORTP,EST,NXM,
                                NXB,APSEUDO,ADAGER,COF,XM,XB);
         INTEGER M,N,TI ; INTEGER ARRAY COF[0]; BOOLEAN OPC;
         REAL SUPER, EST,ORTP ;
         ARRAY  A,RHS,APSEUDO,ADAGER,XM,XB[0,0],NXM,NXB[0];
   COMMENT   PSEUDOINVER COMPUTES THE PSEUDOINVERSE OF A M×N MATRIX A,
         AND OTHER RELATED QUANTITIES. THE ESPECIAL WAY OF ROUNDING-OFF
             AFTER DOUBLE PRECISIDN OPERATIONS IS DUE TO
             MR. PETER RUSINGER AND PROF. GENE GOLUB;
      BEGIN   INTEGER J,CONT,Q,K,T,BUENO,R,I,PE,MA; BOOLEAN SUITCH;
              ARRAY BQ,AN[0:M,0:N],INVQ[0:N,0:N],NRHS,UPI,DOPI[0:TI],
                    G,UQ,SAV[0:N],X1,X2[0:N,0:TI],BPS,API,ADI[0:N,0:M];
              REAL  CLUF,ALFAQ,BEQ,AL,SUPALF,B,NXB1,NXM1,AAA,BBB,CCC,
                    UPI1,DOPI1,ESTI,ASO,SUM,ES,PERTUB,ESTIM,TPD,MINIR;
         MA t IF M<N THEN N ELSE M ;
         BEGIN     ARRAY  TRUC,TU,VQ,TEMP[0:MA]; LABEL  SECND,RFIN,
                   NONES,LOP,TRES,CUATRO,MAIS,CAS,FINI,OTRA,FORCED;
   COMMENT   TRMAVC,MATRIMUL,MULTIVEC,ESC,TRANSP AND VECSUM ARE
             PROCEDURES PERFORMING MATRIX AND VECTOR OPERATIONS,
             SOME OF THEM IN DOUBLE PRECISION;
         PROCEDURE  TRMAVC  (A,V,I,J,M,N);
                      ARRAY  A[0,0],V[0]; INTEGER I,J,M,N ;
            BEGIN      INTEGER K;
               IF I=0  THEN  FOR K←1 STEP 1 UNTIL M DO V[K]← A[K,J]
               ELSE   FOR K←1 STEP 1 UNTIL, N DO V[K]← A[I,K]
            END  TRMAVC ;
```

```
PROCEDURE   MATRIMUL (A,B,C,P,Q,R) ;
            ARRAY  A,B,C[0,0] ;INTEGER P,Q,R ;
   BEGIN    INTEGER  I,J,K ;REAL AC,BC;
      FOR I←1 STEP 1 UNTIL P DO FOR J←1 STEP 1 UNTIL R DO
      BEGIN  AC←BC←0;   FOR K←1 STEP 1 UNTIL Q DO
         DOUBLE(A[I,K],0,B[K,J],0,×,AC,BC,+,←,AC,BC);
         C[I,J] + AC←BC&AC[1:1:8]/5.4975581389l@11
      END
   END   MATRIMUL ;
PROCEDURE   MULTIVEC (A,V1,V2,P,Q);
            ARRAY  A[0,0],V1,V2[0] ;INTEGER P,Q;
   BEGIN    INTEGER  I,J; REAL AC,BC;
      FOR I←1 STEP 1 UNTIL P 00
      BEGIN    AC←BC←0; FOR J←1 STEP 1 UNTIL Q DO
         DOUBLE (A[I,J],0,V1[J],0,×,AC,BC,+,←,AC,BC);
         V2[I]← AC←BC&AC[1:1:8]/5.4975581389l@11
      END
   END   MULTIVEC ;
PROCEDURE  ESC(A,B,C,P);
            ARRAY  A,B[0] ;REAL C ; INTEGER  P;
   BEGIN    INTEGER I ;
      C←0; FOR I←1 STEP 1 UNTIL P 00 C←A[I]× B[I]+C
   END   ESC ;
PROCEDURE   TRANSP (A,B,P,Q) ;
            ARRAY  A,B [0,0] ; INTEGER  P,Q ;
   BEGIN    INTEGER I,J;
      FOR I←1 STEP 1 UNTIL P DO
      FOR J←1 STEP I UNTIL Q DO   B[J,I]   ← A[I,J] ,
   END   TRANSP ;
PROCEDURE   VECSUM (A,B,C,ALF,BET,N) ;
            ARRAY  A,B,C[0]; REAL  ALF,BET ;INTEGERN;
   BEGIN   INTEGER I;
      FOR I←1 STEP 1 UNTIL N DO C[I]←A[I]×ALF+B[I]× BET
   END   VECSUM ;
PROCEDURE   PSEUDO (NQ,ALFQ,UQ,Q) ;
            ARRAY  NQ[0,0],UQ[0]; REAL  ALFQ; INTEGER  Q ;
COMMENT   GIVEN (BQ"BQ) INVERSE, PSEUOO CONSTRUCT (B(Q+1)"B(Q+1))
          INVERSE;
   BEGIN    REAL  A; INTEGER  I,J ;
      ALFQ ← 1/ALFQ;
      FDR I←1 STEP 1 UNTIL Q DO FOR J←1 STEP 1 UNTIL Q DO
         DOUBLE(ALFQ,0,UQ[I],0,×,UQ[J],0,×,NQ[I,J],0,+,←,
                NQ[I,J],A);  Q←Q+1 ; UQ[Q]← -1 ;
      FOR J←1 STEP 1 UNTIL Q DO
         NQ[J,Q]← NQ[Q,J] ← -ALFQ × UQ[J]
   END   PSEUDO ;
PROCEDURE  GARBG(M,N,T,TI,COF,BQ,G,RHS,BPS,A,NRHS,AAA,
                 NXM,NXB,NXM1,NXB1,APSEUDO,ADAGER,XM,XB,EST);
            REAL  NXM1,NXB1,EST,AAA; INTEGER N,M,T,TI;
            INTEGER ARRAY  COF[0];
            ARRAY XM,XB,APSEUDO,ADAGER,BQ,RHS,BPS,A[0,0],
                  NXB,NXM,G,NRHS[0];
COMMENT   GIVEN THE BASIS BQ:A PSEUDOINVERSE, A DAGGER, XM, XB,
          NXM, NXB, AND NORM OF (A-BQC) ARE COMPUTED)
   BEGIN    INTEGER I,Q,J,R,MA ; REAL B,PER,ALFAQ ;
```

```
            ARRAY  C[0:T,0:N], BQ1[0:M,0:T],UQ[0:N] ;
         IF  M≤N  THEN  MA←N ELSE  MA ←M ;
         BEGIN     ARRAY  AN[0:T,0:N] ;
            BEGIN ARRAY  INVQ1[0:T,0:T],TU,TEMP[0:MA];
COMMENT  A DAGGER IS CALCULATED;
            FOR I←1 STEP I UNTIL N DO
            BEGIN.    Q ← COF[I] ;
               IF Q≠0 THEN FOR J←1 STEP 1 UNTIL M 00
               BEGIN   BQ1[J,Q]  ←  BQ[J,Q]/G[I]  ;
                  ADAGER[I,J]← BPS[Q,J]← BPS[Q,J]× G[I];
               END
            END;
COMMENT  C AND C-PSEUDOINVERSE ARE CALCULATED;
            MATRIMUL(BPS,A,C,T,M,N);
            TRMAVC(C,TU,1,1,T,N); ESC(TU,TU,B,N);
            INVQ1[1,1]← l/B;
            FOR Q← 2 STEP 1 UNTIL T DO
            BEGIN  R← Q-1; TRMAVC(C,TU,Q,Q,T,N);
               MULTIVEC(C,TU,TEMP,R,N);
               MULTIVEC(INVQ1,TEMP,UQ,R,R);
               FOR I←1 STEP 1 UNTIL N DO
               BEGIN  PER← 0;
                  FOR J←1 STEP 1 UNTIL R 00
                  PER+  PER+C[J,I] × UQ[J]; TEMP[I] ← PER
               END;
               VECSUM(TU,TEMP,TEMP,1,-1,N);
               ESC(TEMP,TEMP,ALFAQ,N);
               PSEUDO(INVQ1,ALFAQ,UQ,R)
            END ;    MATRIMUL(INVQ1,C,AN,T,T,N) ;
         END;
         BEGIN     ARRAY  BQC[0:MA,0:N] ;
COMMENT  NORM OF (A-BC) IS COMPUTED;
            TRANSP(AN,BQC,T,N) ;
            MATRIMUL(BQC,BPS,APSEUDO,N,T,M);
            MATRIMUL(BQ1,C,BQC,M,T,N); EST ← 0;
            FOR I←1 STEP 1 UNTIL M DO
            BEGIN   PER+ Of
               FOR J+l STEP 1 UNTIL N DO
               PER+ PER+ABS(BQC[I,J]-A[I,J]);
               IF PER≥ EST  THEN  EST ← PER
            END
         END
      END;
      EST ← EST/AAA;
COMMENT  THE MINIMUM AND BASIC SOLUTIONS ARE COMPUTED;
      MATRIMUL(APSEUDO,RHS,XM,N,M,TI);
      MATRIMUL(ADAGER,RHS,XB,N,M,TI);
      BEGIN   ARRAY  AN[0:M,0:TI];
COMMENT  THE RESIDUALS FOR THESE SOLUTIONS ARE COMPUTED!
         MATRIMUL (A,XM,AN,M,N,TI);
         FOR  J←1 STEP 1 UNTIL  TI DO
         BEGIN  NXM1← 0;
            FOR I←1 STEP 1 UNTIL M 00
            NXMI ←(AN[I,J]-RHS[I,J])*2+NXM1;
            NXM1← SQRT(NXM1); NXM[J] + NXM1/NRHS[J]
```

```
                END;
                MATRIMJL(A,XB,AN,M,N,TI);
                FOR J←1 STEP 1 UNTIL TI DO
                BEGIN NXB1←0;
                   FOR I←1 STEP 1 UNTIL M DO
                   NXB1←(AN[I,J]-RHS[I,J])*2 + NXB1 ;
                   NXB1← SQRT(NXB1); NXB[J]← NXB1/NRHS[J]
                END ; NXB1 ← NXM1 ← 0;
                FOR I←1 STEP 1 UNTIL TI DO
                BEGIN NXB1 ← NXB1+ NXB[I]*2 ;
                   NXM1 ← NXM1+NXM[I] *2
                END ; NXM1← SQRT(NXM1); NXB1← SQRT(NXB1);
             END;
          END GARBG ;
COMMENT    PROCEDURE PSEUDOINVER BODY ;
          HINIR ←@-20;   AAA←CCC←0 ;
COMMENT    NORM OF A AND NORM OF THE RHS;
          FOR J←1 STEP 1 UNTIL N DO FOR I←1 STEP 1 UNTIL M DO
          BEGIN BBB← ABS(A[I,J]); If BBB>AAA THEN AAA←BBB
          END ; FOR J←1 STEP 1 UNTIL TI Do
          BEGIN CCC←0; FOR I←1 STEP 1 UNTIL M DO
             CCC" CCC+RHS[I,J]* 2 ;  CCC← SQRT(CCC);
           --- NRHS[J]← IF CCC>1 THEN CCC ELSE 1 ;
          END;  IF AAA<1 THEN AAA←1 ;
          FOR I←1 STEP 1 UNTIL N DO
          BEGIN COF[I]← 0;
             FORJ←1 STEP 1 UNTIL M DO ADI[I,J]←ADAGER[I,J]← 0;
          END ;
       COMMENT  THE MATRIX A IS NORMALIZED AND STORED ON AN ,
                G CONTAINS THE NORMALIZING COEFFICIENTS ;
             FOR J←1 STEP 1 UNTIL N DO
             BEGIN  TRMAVC(A,TU,0,J,M,N);
                ESC(TU,TU,CLUF,M);
                G[J]←IF CLUF > 1.0 THEN 1.0/SQRT(CLUF) ELSE 1.0;
             END ; FOR I←1 STEP 1 UNTIL M DO
             BEGIN FOR J←1 STEP 1 UNTIL N DO
                   AN[I,J] ← A[I,J]× G[J]
             END ;
       COMMENT THE CONSTRUCTION OF A BASIS OF STRONGLY
                INDEPENDENT VECTORS IS STARTED;
             CONT←1; SUPALF← 0 ; Q← K← T← 1;
             COF[1]←1; INVQ[1,1]← 1; SUITCH← FALSE ;
             FOR I←1 STEP 1 UNTIL M DO BQ[I,1]← AN[I,1];
COMMENT      SEARCH FOR INDEPENDENT COLUMNS Of A, WHEN THE COLUMNS
             ARE EXHAUSTED AN EXIT IS PROVIDED To LABEL FINI, IN CONT
             A RECORD IS KEPT ON THE WAY IN WHICH COLUMS ARE ACCEPTED1
LOP :        If Q=N THEN Go TO FXNI ;
             T←K ; Q←Q+1; K←K+1;
COMMENT      PROJECTION OF A COLUMN OF AN ON THE ORTHOGONAL
             SUBSPACE OF BQ;
CAS a        TRMAVC(AN,VQ,0,Q,M,N);
             FOR I←1 STEP 1 UNTIL T DO
             BEGIN ALFAQ← 0;
                FOR J←1 STEP 1 UNTIL M DO
                ALFAQ← ALFAQ+BQ[J,I]×VQ[J]; TEMP[I]← ALFAQ
```

```
          END ;
          MULTIVEC(INVQ,TEMP,UQ,T,T);
          MULTIVEC(BQ,UQ,TEMP,M,T);
          VECSUM(VQ,TEMP,TEMP,1,-1,M);
          ESC(TEMP,TEMP,ALFAQ,M);
          If SUITCH THEN GO TO TRES;
COMMENT    FIRST TEST FOR ACCEPTANCE AS AN INDEPENDENT COLUMN;
          If ALFAQ ≤ ORTP THEN GO TO NONES;
          If CONT=2 THEN CONT← 3 ;
COMMENT    CONSTRUCTION OF B(Q+1);
          PSEUDO (INVQ,ALFAQ,UQ,T);
          FOR I←1 STEP 1 UNTIL M DO BQ[I,K]← VQ[I];
          COF [Q]← K ; GO TO LOP ;
NONES :   COF[Q] ← 0; K← K-1 ;
          If CONT ≠ 3 THEN
          BEGIN CONT←2 ; GO TO TRES ;
          END ; GO TO LOP ;
COMMENT    If CONT≠2 THEN THE REJECTED COLUMNS ARE REVISED IN ORDER
           TO TAKE THE MOST INDEPENDENT WITH RESPECT TO THE BASIS BQ.
           CONT=1 MEANS THAT ALL THE COLUMNS HAVE BEEN TAKEN IN THE
           FIRST SWEPT, CONT=3 YEANS THAT A COLUMN HAS BEEN REJECTED
           AND LATER ON,ANOTHER HAS BEEN ACCEPTED;
DTRA :    Q← 2 ; SUPALF← 0; BUENO← 0;
MAIS :    If Q=N+1 THEN GO TO CUATRD ;
          IF COF[Q]≠ 0 THEN
          BEGIN Q← Q+1 ; GO TO MAIS ;
          END ; GO TO CAS ;
TRES :    IF ALFAQ ≥ SUPALF THEN
          BEGIN SUPALF ← ALFAQ ;
             FOR I←1 STEP 1 UNTIL M DO TRUC[I]← VQ[I];
             FOR I←1 STEP 1 UNTIL T DO
             SAV[I]← UQ[I] ; BUENO ← Q
          END ; IF CONT=2 THEN GO TO LOP ;
          If Q=N THEN GO TO CUATRO ;
          Q← Q+1; GO TO MAIS;
COMMENT    If THE PROJECTION OF THE SELECTED COLUMN IS LESS THAN e=20
           THEN SAV IS REJECTED AND WE FINISH;
CUATRD : If SUPALF S MINIR THEN GO TO RFIN ;
          AL ←SUM ← 0;
COMMENT    THE NORM OF (B(Q+1)"B(Q+1)) INVERSE IS ESTIMATED,AND ITS
           VALUE IS CONTROLLED;
          FOR I←1 STEP 1 UNTIL T DO
          BEGIN FOR J←1 STEP 1 UNTIL T DO
             SUM ←SUM+ABS(INVQ[I,J]);
             IF SUM>AL THEN AL←SUM
          END ;
          ESTIM ← AL←(SQRT(T)+1.0)/SUPALF ;
          If ESTIM ≥ SUPER THEN GO TO RFIN ;
COMMENT    SAV HAS PASSED THE TESTS Of SECTION 2,C), NOW IS USED
           TENTATIVELY IN B(Q+1) TO SEE If THE RESIDUALS DIMINISH;
FORCED : PSEUDO(INVQ,SUPALF,SAV,T);
          COF[BUENO]← T ;
          FORI←1 STEP 1 UNTIL M DO BQ[I,T]← TRUC[I];
  COMMENT       WE CONSTRUCT NOW B-PSEUDO,C,A-PSEUDO AND ADAGGER;
FIN1 :    FOR I←1 STEP 1 UNTIL T DO FOR J←1 STEP 1 UNTIL M DO
```

```
          BEGIN  ES←0; FOR PE←1 STEP 1  UNTIL  T DO
             ES ←  ES  + INVQ[I,PE] × BQ[J,PE] ; BPS[I,J] ←ES
COMMENT    AS WE WANT TO COMPARE RESULTS FOR TWO DIFFERENT BASES,
           SUITCH PROVIDES A WAY TO DECIDE THE CALL OF GARBG;
          END ;  IF SUITCH THEN GO TO SECND ;
          GARBG (M,N,T,TI,COF,BQ,G,RHS,BPS,A,NRHS,AAA,
                 NXM,NXB,NXM1,NXB1,APSEUDO,ADAGER,XM,XB,EST) ;
          If  T=N  THEN  GO  TO  RFIN ;  SUITCH ← TRUE ;
          If  CONT = 2   THEN
          BEGIN    CONT ← 1;   GO TO CUATRO ;
          END ;    GO TO OTRA ;
SECND :   GARBG (M,N,T,TI,COF,BQ,G,RHS,BPS,A,NRHS,AAA,
                 UPI,DOPI,UPI1,DOPI1,API,ADI,X1,X2,ESTI);
COMMENT    NOW THE TEST OF SECTION 2,D) IS MADE;
          If  NXB1≥ DOPI1   AND   NXM1≥UPI1   THEN
          BEGIN    IF  NOT  OPC OR  EST≥ ESTI  THEN
COMMENT    If SAV IS ACCEPTED THEN ALL THE USEFUL QUANTITIES
           ARE SHIFTED;
            BEGIN    NXM1←UPI1; NXB1← DOPII;  EST←ESTI ;
               FOR I←1 STEP 1 UNTIL N DO
               BEGIN  FOR  J←1   STEP 1 UNTIL M DO
                  BEGIN  APSEUDO[I,J]← API[I,J] ;
                     ADAGER[I,J]←ADI[I,J]
                  END;   FOR  J←1  STEP  1 UNTIL TI DO
                  BEGIN  XM[I,J]←X1[I,J]; XB[I,J]← X2[I,J];
                       NXM[J]←UPI[J]; NXB[J]←DOPI[J];
                     END;
               END; IF T=N  THEN
               BEGIN    BUENO←0 ; GO  TO RFIN
               END  ;  GO  TO  OTRA
            END
          END ;
RFIN :    COF[BUENO]← 0;
       END
   END    PSEUDOINVER ;
COMMENT    BODY OF THE DRIVER PROGRAM, THE INPUT-OUTPUT AND THE CALL
           Of PSEUDOINVER ARE INCLUDED;
   FOR I←1 STEP 1 UNTIL M DO
   READ(FOR J←1 STEP 1 UNTIL N DO A[I,J]);
   WRITE( [PAGE]); WRITE(TIT1,M,N); PRT(A,M,N);
   IF NOT OPC THEN FOR I←1 STEP 1 UNTIL  TI DO
   READ( FOR J←1 STEP 1 UNTIL  M DO RHS[J,I])
   ELSE  FOR I←1 STEP 1 UNTIL  M DO FOR  J←1 STEP  1 UNTIL  M DO
         RHS[I,J]← If  I=J  THEN  1  ELSE  0 ;
   PSEUDOINVER(M,N,TI,SUPER,OPC,A,RHS,ORTP,EST,NXM,
               NXB,APSEUDO,ADAGER,COF,XM,XB);
   WRITE(TIT7,EST,FOR I←1  STEP  1  UNTIL TI DO NXM[I]);
   WRITE(TIT77, FOR I←1 STEP 1 UNTIL TX DO NXB[I]);
   If OPC THEN
   BEGIN    WRITE([PAGE]) ; WRITE(TIT8);
      PRT(APSEUDO,N,M); WRITE (TIT9);
      FOR I←1 STEP 1 UNTIL N DO
      BEGIN IF COF[I]=0  THEN  GO  TO  NOPR ;
         WRITE (TITL38,I);
         WRITE  (PRMAT,FOR J←1  STEP  1 UNTIL M DO ADAGER[I,J]) ;
```

```
    NOPR: END
       END        ELSE
                     BEGIN  WRITE([PAGE]); WRITE(TIT10) ;
                        FOR  I←1 STEP 1 UNTIL N DO
                        WRITE(SOL,FOR J←1 STEP 1 UNTIL TI DOXM[I,J]);
                        WRITE (TIT20)-; FOR I←1 STEP 1 UNTIL N DO
                        BEGIN  IF COF [I]=0 THEN GO TO NOPR1;
                           WRITE(TITL38,I );
                           WRITE(SOL,FOR J←1 STEP 1 UNTIL TI DO XB[I,J]);
    NOPR1 :                END;
    COMMENT     JUST BY AdD InG nEw SETS OF DATAS MORE PROBLEMS CAN BE RUN;
                     END ;  WRITE(ENDE); GO TO OVER ;
    END  ;
    FIU :
END .
```