ON THE DERIVATION OF THE COMPUTATIONAL FORMULAE OF THE ORTHGONALIZATION METHOD OF INVERSION INCLUDING EXPERIMENTS ON NEAR-SINGULAR MATRICES AND ON THE TRANSFORMATION OF 2-D ARRAY TO 1-D ARRAY

BY SYAMAL KUMAR SEN

(Central Instruments and Services Laboratory, Indian Institute of Science, Bangalore-12, India)

[Received: August 29, 1967]

ABSTRACT

Formulae for an automatic computation of the inversion of a matrix by an orthogonalization (4, 8) method, which is a more powerful form of Gram's orthonormalization process, have been derived. The main features of these formulae are the minimum storage demand with maximum automation. Numerical experiments have been carried out mainly on near-singular matrices with this computational procedure. A simple method for the transformation of two dimensional array to one dimentional array has also been included.

INTRODUCTION

The inversion poses a serious problem when the matrix is near-singular[9]. The methods both direct and iterative encounter rounding errors due to a good deal of divisions by small determinants[3] or some functions of determinants and sometimes due to extration of square-roots[9]. The present method consists in constructing an orthogonal set of column (or row) vectors from a set of linearly independent column (or row) vectors of the matrix; and if a particular column vector, say, the fifth column vector of a matrix (6×6) is linearly dependent on the preceding vectors, the first four orthogonal vectors will be satisfactorily accurate, only the fifth and sixth one will encounter some rounding errors (due to divisions only), that are, however, tolerable when compared to those of other methods[9]. The Grams orthonormalization method[9] will involve, in addition, a very conspicuous source of rounding errors due to extraction of square-roots.

These derived formulae will require storage for the original matrix A $(n \times n)$ and for the triangular matrix $P(n \times n)$, in addition to that of the program, but no storage for the orthogonal matrix $X(n \times n)$ and for P_i's $(n \times n)$, $i = 1, 2, \cdots, n$ of the method that finds description in the subsequent pages.

The automation resting on the use of these formulae is more pronounced than when using the orthogonalization method directly, since instructions, arithmetic operations and storage, in the former case, are comparatively less.

The detailed programming aspects including a brief description of the method as well as a few typical numerical examples given here, would reveal the extreme usefulness of such formulation in solving a system of linear and non-linear equations[11].

A simple method of transformation of two dimensional array of different forms to linear array has been mentioned in this connection, assuming a relation of the subscripts with the memory locations.

A brief description of the method[4, 8, 9]:

Let $A = [a_1 a_2 a_3 \dots a_n]$ be a matrix of order n where $a_j = \{a_{ij}\}'$ $i = 1, 2, 3, \cdots, n$ denotes the coloum vectors of the matrix A, while $(')$ indicates a transpose.

Let

$$x_1 = a_1 \qquad [1\cdot1]$$

Now we form x_2 by a linear combination of x_1 and a_2 so that x_1 and x_2 are orthogonal, then

$$x_2 = \beta_{11} x_1 + a_2 \qquad [1.2]$$

where

$$\beta_{11} = -(a_2 x_1)/|x_1|^2 \qquad [1\cdot3]$$

In general, the sth orthogonal vector can be formed from x_1, x_2, x_3, $\cdots$, x_{s-1} and a_s in a similar way,

$$x_s = \beta_{1,s-1} x_1 + \beta_{2,s-1} x_2 + \cdots + \beta_{s-1,s-1} x_{s-1} + a_s$$

where

$$\beta_{i,s-1} = -(a_s x_i)/|x_i|^2 \qquad [1.4]$$

Again,

$$X = (x_1, x_2, \cdots, x_n) = AP_1 P_2 \cdots P_n = AP \qquad [1.5]$$

where $P(= P_1 P_2 \cdots P_n)$ is an upper triangular matrix of order n.

P_1 is simply an identity matrix and P_s is an upper triangular matrix of the same order n, whose sth column vector is

$$\{\beta_{1,s-1}, \beta_{2,s-1}, \cdots, \beta_{s-1,s-1}, 1, 0, \cdots\}'$$

and all other vectors are unit vectors. or more clearly, all the diagonal elements are unity.

Let $X'X\} = \text{diag}\ (|x_1|^2,\ |x_2|^2,\ \cdots,\ |x_n|^2)\}$ be Δ therefore, $X^{-1} = \Delta^{-1}X'$ where

$$\Delta^{-1} = \text{diag}\,(1/|x_1|^2,\ 1/|x_2|^2,\ \cdots,\ 1/|x_n|^2)$$

From [1.5],

$$A^{-1} = PX^{-1} = P\Delta^{-1}X' = \Delta^{-1}PX' \qquad [1.6]$$

and the magnitude of the determinant of A is equal to

$$|\Delta|^{1/2} = |x_1|\ |x_2|\ |x_3|\ \cdots |x_n|$$

Formulae for Automatic Computation

The number of locations reserved for β-matrix and A-matrix are $n(n+1)/2$ and n^2 respectively. It is to be noted that the symbol '$=$' means 'is replaced by' inside this caption. For example, $a_{ij} = x_{ij}$ means the contents of a_{ij} are replaced by the contents of x_{ij}.

$$\begin{cases} \beta_{p,\,s+1} = -\dfrac{\sum_{l=1}^{n}(a_{l,\,s+1}\,a_{lp})}{\sum_{l=1}^{n}|a_{lp}|^2} \\ p = 1,\ 2,\ 3,\ \cdots,\ s \end{cases} \qquad [2.1]$$

$$\begin{cases} a_{i,\,s+1} = a_{i,\,s+1} + \sum_{p=1}^{s}\beta_{p,\,s+1}\,a_{ip} \\ i = 1,\ 2,\ 3,\ \cdots,\ n \\ s = 1,\ 2,\ 3,\ \cdots,\ n-1 \end{cases} \qquad [2.2]$$

$$\begin{cases} \beta_{pp} = 1 \\ p = 1,\ 2,\ 3,\ \cdots,\ n \end{cases} \qquad [2.3]$$

$$\begin{cases} x_s = \sum_{l=1}^{n}|a_{ls}|^2, \\ s = 1,\ 2,\ 3,\ \cdots,\ n \end{cases} \qquad [2.4]$$

$$\begin{cases} a_{i,\,s} = a_{i,\,s}/x_s \\ i = 1,\ 2,\ 3,\ \cdots,\ n \\ s = 1,\ 2,\ 3,\ \cdots,\ n \end{cases} \qquad [2.5]$$

$$\begin{cases} \beta_{i,s} = \beta_{i,s} + \sum_{p=i}^{s-2} \beta_{i,p+1}\, \beta_{p+1,s} \\ s = i+2,\ i+3,\ \cdots,\ n \\ i = 1,\ 2,\ \cdots,\ n-2 \end{cases} \qquad [2.6]$$

$$\begin{cases} a_{i,s} = \sum_{p=i}^{n} \beta_{i,p}\, a_{s,p} \\ s = 1,\ 2,\ 3,\ \cdots,\ n \\ i = 1,\ 2,\ 3,\ \cdots,\ n \end{cases} \qquad [2.7]$$

Procedure: It is very important to note that the subscripts inside each left parenthesis are to be varied sequentially; while varying one subsript, other subscripts are kept fixed. For example, in [2·1] we take $s = 1$, then $p = 1$ (which is equal to s) and find β_{12}; since we can not increase p beyond s in [2·1], we jump to the second [2·2] formula; in [2·2] we start taking $i = 1$ (s is already 1) and find a_{12}, then taking $i = 2$ we find a_{22} and similarly a_{32}, a_{42}, $\cdots$, a_{n2} by taking $i = 3, 4, \cdots, n$, respectively. Now s is increased by 1 (s is now 2), and we go back to [2·1] to find β_{13}, β_{23} ($\because p = 1, 2$), then to [2·2], to find a_{13}, a_{23}, a_{33}, $\cdots$, a_{n3} ($\because i = 1, 2, \cdots, n$).

Then s is again increased (s is now 3) and we go to [2·1] to determine β_{14}, β_{24}, β_{34}, ($p = 1, 2, 3$), then to [2·2] to determine a_{14}, a_{24}, a_{34}, $\cdots$, a_{n4} ($i = 1, 2, \cdots, n$).

We proceed in this way up to $s = n - 1$.

Now passing on to [2·3], all diagonal terms of β-matrix are replaced by 1, or, more explicitly. $p = 1$, $\beta_{11} = 1$; $p = 2$, $\beta_{22} = 1$; $p = 3$, $\beta_{33} = 1$; $\cdots$; $p = n$, $\beta_{nn} = 1$.

Then we go to [2·5] and find $x_1, x_2, \cdots, x_n$ and so on.

Explanation of the derived Formulae:

Formula [2.1]: β_{12}; β_{13}, β_{23}; $\beta_{14}, \beta_{24}, \beta_{34}$; $\cdots$; $\beta_{1n}, \beta_{2n}, \beta_{3n}, \cdots, \beta_{n-1,n}$ are nothing but β_{11}; β_{12}, β_{22}; $\beta_{13}, \beta_{23}, \beta_{33}$; $\cdots$; $\beta_{1,n-1}, \beta_{2,n-1}, \cdots, \beta_{n-1,n-1}$ of the method. In the actual formulation, one right shift of all the β-elements has been given only to make the β-matrix a perfect upper triangular matrix of the same order (i.e., n). The actual advantage lies in the fact that the matrix operation (such as multiplication, addition, subtraction which necessarily, demand the same order of both the matrix operands) will be more automatic.

Formula [2·2]: Here the left hand side elements are nothing but x_{ij}, $i = 1, 2, \cdots, n$; $j = 1, 2, \cdots, n$. In this connection, it is to be noted that no seperate storage has been supplied for X-matrix. Moreover, the

computations involving the successive replacement instructions are perfectly automatic and no separate working locations, even for storing a column, are necessary.

Formula [2.3]: The left hand side elements (which are all unity) are nothing but the diagonal elements of the P-matrix; P-matrix is the product matrix $P_1 P_2 P_3 \cdots P_n$.

Formula [2.4]: The L.H.S. members are $|x_i|^2$, $i = 1, 2, \cdots, n$. Though it demands an extra n locations, it enhances the automation and consequently simplifies the program.

Formula [2.5]: The L.H. terms are the elements of $(\Delta^{-1}X')'$. It is to be noted that no separate matrix multiplication subroutine (for multiplying Δ^{-1} and X') has been used, thus saving appreciable computing time.

Formula [2.6]: The left hand members are the elements corresponding to the elements of the P-matrix. It is important to note that the same locations for β-elements have been used for P-matrix (*i.e.*, the product matrix $P_1 P_2 \cdots P_n$); this not only improves the automation, but also saves the storage.

Formula [2.7]: The L.H. members represent $(A^{-1})'$ of the method.

Transformation of a 2-*D* Array to Linear Array

Very often this transformation is extremely useful and necessary not only from the storage point of view, but also from the point of view of Machine, SPS or sometimes Autocode language. This is particularly so for a triangular matrix. The DIMENSION statement, for example, in FORTRAN languages (FORTRAN II, III, IV, etc.) reserves a memory storage for a two dimensional array in a rectangular or square fashion and not in the two dimensional triangular fashion. In such case, the DIMENSION statement, if two dimensional array is set for a triangular form (*i.e.* triangular matrix of order $n \times n$), helps a storage loss of $n(n-1)/2$ locations. This fact, of course, depends on the FORTRAN compilers of different computers. The machine language, Autocode or SPS of most computers cannot handle a double-subscripted variable and so in such cases, it is essential to convert a double-subscripted variable to single-subscripted variable.

Suppose the elements of a square matrix of order n have been stored columnwise, starting from the location p_1 onwards, then

$$\text{location of } a_{ij} = p_1 + (j-1)n + (i-1) \qquad [3.1]$$

A lower triangular matrix L of order n with all elements above the diagonal being zeros, when stored rowise, starting from the location p_2 onwards, can have a single dimensional representation as

$$\text{location of } l_{ij} = p_2 + i(i-1)/2 + (j-1) \qquad [3.2]$$

For an upper triangular matrix R of order n with all the elements below diagonal being zeros and being stored columnwise, starting from a location p_3 onwards,

$$\text{location of } r_{ij} = p_3 + j(j-1)/2 + (i-1) \qquad [3.3]$$

If there is an array of the form

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & \cdots & p_{2,\,n-1} \\ p_{31} & p_{32} & \cdots & p_{3,\,n-2} & \\ \vdots & & \iddots & & \\ p_{n1} & & & & \end{bmatrix} \qquad [3.4]$$

then starting from the location q_0 onwards the location of p_{ij}, when stored columnwise, will be equal to

$$\left.\begin{aligned} &[q_0 + n(j-1) + (i-1)] \text{ for } j \leqslant 2 \\ \text{and} \quad &[q_0 + n(j-1) + (i-1) - j(j-1)/2 + (j-1)] \text{ for } j \geqslant 3 \end{aligned}\right\} \qquad [3.5]$$

All the above transformations have been deduced by assuming the location of an element as a polynomical function of i, j and n of some degree, say, 2 or 3. In [3·4], for example, location of p_{ij}, say, $P(i, j, n)$ is given by

$$P(i, j, n) = q_0 + d_1 i^2 + d_2 j^2 + d_3 n^2 + d_4 ij + d_5 jn + d_6 in + \cdots$$

where constants d_1, d_2, $\cdots$, d_6, $\cdots$ will be determined from the given values of i, j, n and corresponding value of P; since [3·4] has been stored columnwise from location q_0 onwards, we know i, j and corresponding P.

The form in [3·4], though having practically little use in matrix theory, may find its place in some particular type of statistical problems and so has been mentioned here.

Numerical Experiments

The tests were carried out on Hilbert's matrices of different orders and some other matrices including near-singular ones. Here only an example of Hilbert's matrix of order 4 along with an experiment on a mathematically singular matrix of different orders, has been mentioned. Hilbert's matrices are typical examples of matrices ill-conditioned with respect to inverse; its

(i, j)th element is $1/(i+j-1)$. These matrices are theoretically non-singular, but appear as singular during numerical computations. The singularity is more pronounced with higher order matrices.

All the calculations, unless otherwise stated, were carried out with 8-dit floating point arithmetic and the results were expressed in rounded 6 significant digits. Zeros on the least significant side have been suppressed. All the elements of the original matrices were retained correct up to 8 decimal places in all the cases.

Example: Hilbert's matrix of order 4

Det $A = 0{\cdot}165336 \times 10^{-6}$

β's of the method:

$\beta_{11} = -0{\cdot}561951$, $\beta_{12} = -0{\cdot}398049$. $\beta_{13} = -0{\cdot}309965$

$\beta_{22} = -0{\cdot}106008 \times 10$, $\beta_{23} = -0{\cdot}991637$

$\beta_{33} = -0{\cdot}15375$

$$X = \begin{bmatrix} 1 & -{\cdot}619512 \times 10^{-1} & {\cdot}957983 \times 10^{-3} & -{\cdot}49239 \times 10^{-5} \\ {\cdot}5 & {\cdot}523577 \times 10^{-1} & -{\cdot}452792 \times 10^{-2} & {\cdot}592247 \times 10^{-4} \\ {\cdot}333333 & {\cdot}626829 \times 10^{-1} & {\cdot}867967 \times 10^{-3} & -{\cdot}14827 \times 10^{-3} \\ {\cdot}25 & {\cdot}595122 \times 10^{-1} & {\cdot}406661 \times 10^{-2} & {\cdot}9894 \times 10^{-4} \end{bmatrix}$$

A^{-1} being symmetric, is given by (suppressing elements below diagonal)

$$A^{-1} = \begin{bmatrix} {\cdot}159693 \times 10^{2} & -{\cdot}119657 \times 10^{3} & {\cdot}239179 \times 10^{3} & -{\cdot}139468 \times 10^{3} \\ & {\cdot}119838 \times 10^{4} & -{\cdot}269615 \times 10^{4} & {\cdot}167751 \times 10^{4} \\ & & {\cdot}647945 \times 10^{4} & -{\cdot}41997 \times 10^{4} \\ & & & {\cdot}280244 \times 10^{4} \end{bmatrix}$$

Experiment on a mathematically singular matrix:

A matrix whose (i, j)th element is $(i-1)+j$, is always mathematically singular. And always the third column is linearly dependent on the preceding two columns. An experiment was conducted over this matrix of different orders with 8-dit floating point arithmetic.

Order 4: $|x_1|^2 = .276 \times 10^3$, $|x_2|^2 = .115942 \times 10$, $|x_3|^2 = .34861 \times 10^{-13}$

Order 6: $|x_1|^2 = .2166 \times 10^4$, $|x_2|^2 = .174515 \times 10$, $|x_3|^2 = .469891 \times 10^{-10}$

Order 7: $|x_1|^2 = .476 \times 10^4$, $|x_2|^2 = .210176 \times 10$, $|x_3|^2 = .291638 \times 10^{-10}$

Order 8: $|x_1|^2 = .9416 \times 10^4$, $|x_2|^2 = .228377 \times 10$, $|x_3|^2 = .364335 \times 10^{-9}$

If the calculations were carried out with larger significant digits (floating point arithmetic), $|x_3|^2$ in all the orders will be still smaller. On 8 significant digit work, $|x_3|^2$, in all the above mentioned cases, can be regarded good numerical zeros[9].

ACKNOWLEDGMENT

The author wishes to express his sincere thanks and gratitude to Prof. P L. Bhatnagar of Indian Institute of Science, Bangalore for his constant encouragement and for fruitful suggestions and to Dr. S. Dhawan, for having kindly permitted to publish this paper.

REFERENCES

1. Martin, R. S., Peters, G. and Wilkinson, J. H. *Num Maths.*, 1965, **7**, 362.
2. Nat. Bureau of Standards, Appl. Math. Series, 1954, 39.
3. Walsh, J. H. .. .. Numerical Analysis (Academic Press), 1966.
4. Rice, J. R. .. .. Maths. Comput (Am. Math. Soc.), 1966, **20**, 325.
5. Davis, P. J. and Rabinowitz, P. .. Advances in Computers, (F.L. Alt, Ed.), 1961, (Academic Press), **2**, 55.
6. Shamuel Kaniel .. .. Maths. Comput (Am. Math. Soc.), 1966, **20**, 369.
7. Randall. E. C .. .. *J. Soc. ind. appl. Math.*, 1964, **12**, 588.
8. Mitra, S. K. .. .. Proc. of the Second Congress on Th. and Appl. Mechanics, New Delhi, Oct. 15-16, 1956, 261.
9. Sen, S. K. .. .. *J. Indian Inst. of Sci.*, 1957, **49**, 37.
10. Fox, L. .. .. An Introduction to Numerical Linear Algebra, (Clarendon Press, Oxford), 1964, 130.
11. Kizner, W. .. .. *J. soc. ind. appl. Maths.*, 1964, **12**, 424.