Kyungpook Math. J.
Volume 20, Number 2
December, 1980.

THE COMPUTATION OF THE GENERALIZED INVERSE

By Kern O. Kymn,* J. R. Norsworthy, and Tatsuo Okamoto

The purpose of this paper is to report our computer experience of computing the generalized ($g-$) inverse on a UNIVAC 1108 utilizing two formulas for the g-inverse given by Graybill[1]. The two formulas for the g-inverse that we have selected are the ones given 1 in Theorems 6.5.1 [pp. 108—10, 1] and 6.5.8 [pp. 117—8, 1]. In this paper we briefly describe the algorithms and programming of the two selected formulas and evaluate their relative performances in computing g-inverses. The evaluation was accomplished in terms of CPU time elapsed and accuracy evidenced in computing the g-inverse.

We begin our discussion with the definition of a generalized inverse.

DEFINITION. Let A be an $m\times n$ matrix. If a matrix A^g exists that satisfied the following four conditions, A^g is a *generalized* ($g-$) *inverse* of A. (i) AA^g is symmetric (ii) A^gA is symmetric (iii) $AA^gA=A$ (iv) $A^gAA^g=A^g$.

We applied this definition to check the accuracy of the calculation each time after the g-inverse was computed.

In the following, we briefly describe the two selected algorithms to relate them to their programming[2].

THEOREM 1. *If b is a nonzero vector, then* $b^g=(b'b)^{-1}\ b'$.

PROOF. (1) $bb^g=b(b'b)^{-1}\ b'$, symmetric

(2) $b^gb=(b'b)^{-1}\ b'b=I$, symmetric

(3) $bb^gb=b(b'b)^{-1}\ b'b=b$

(4) $b^gbb^g=(b'b)^{-1}b'b(b'b)^{-1}\ b'=(b'b)^{-1}b'=b^g$

The definition of a g-inverse is satisfied. b^g is the g-inverse of b.

We utilized Theorem 1 whenever a g-inverse of a nonzero vector was needed.

THEOREM 2. *For any matrix A, we get*

$$(AA')^g=(A^g)'\ A^g.$$

PROOF. $(A')^g=(A^g)'$. $(AA')^g=(A')^gA^g=(A^g)'A^g$.

We utilized Theorem 2 whenever A^g was known and the computation of $(AA')^g$ was required.

ALGORITHM 1. Algorithm 1 describes the iterative scheme given by Graybill [Theorem 6.5.1, pp. 108—9, 1] to find the g-inverse of a matrix A.

Let A be an $m\times t$ matrix; let B_t be an $m\times k$ matrix that consists of the fir k columns of A; let A_{k-1} be a matrix that consists of the first $(k-1)$ colum of B_k, and let a_k be the k^{th} column of B_k. Partition B_2 by

$$B_2=(A_1,\ a_2)=(a_1,\ a_2)$$

The g-inverse of B_2 is given by

$$B_2^g=\begin{pmatrix} A_1^g-A_1^g a_2\, b_2^g \\ b_2^g \end{pmatrix}$$

where

$$b_2=\begin{cases} (I-A_1A_1^g)a_2 \text{ if } a_2\neq A_1A_1^g a_2 \\ \dfrac{[1+a_2'(A_1A_1')^g\ a_2](A_1A_1')^g a_2}{a_2'(A_1A_1')^g(A_1A_1')^g a_2} \text{ if } a_2=A_1A_1^g\ a_2. \end{cases}$$

Next we partition B_3 by

$$B_3=(A_2,\ a_3)=(B_2,\ a_3)$$

The g-inverse of B_3 is given by

$$B_3^g=\begin{pmatrix} B_2^g-B_2^g a_3\, b_3^g \\ b_3^g \end{pmatrix}$$

where

$$b_3=\begin{cases} (I-B_2B_2^g)a_3 \text{ if } a_3\neq B_2B_2^g a_3 \\ \dfrac{[1+a_3'(B_2B_2')^g\ a_3]\ (B_2B_2')^g a_3}{a_3'(B_2B_2')^g\ (B_2B_2')^g a_3} \text{ if } a_3=B_2B_2^g a_3. \end{cases}$$

We continue until we partition $B_t=A$ by

$$B_t=(A_{t-1},\ a_t)=(B_{t-1},\ a_t)$$

The g-inverse of B_t is given by

$$B_t^g=\begin{pmatrix} B_{t-1}^g-B_{t-1}^g a_t b_t^g \\ b_t^g \end{pmatrix}$$

where

$$b_t=\begin{cases} [I-B_{t-1}B_{t-1}^g)\ a_t \text{ if } a_t\neq B_{t-1}B_{t-1}^g\ a_t \\ \dfrac{[1+a_t'(B_{t-1}B_{t-1}')^g a_t](B_{t-1}B_{t-1}')^g a_t}{a_t'(B_{t-1}B_{t-1}')^g(B_{t-1}B_{t-1}')^g a_t} \text{ if } a_t=B_{t-1}B_{t-1}^g a_t. \end{cases}$$

Since $B_t=A$, we have the desired result $A^g=B_t^g$.

We utilized Theorem 1 to compute b_j^g, $j=2, 3, \cdots, t$.

We utilized Theorem 2 to compute $(B_jB_j')^g$ in b_{j+1}, $j=2, 3, \cdots, t-1$.

Flow Chart I translates Algorithm 1.

FLOW CHART I

SUBROUTINE GINV (A, B, NR, NC, CC)

WHERE:

A is a matrix to be inverted.

B is the G-inverse of A.

NR is the row dimension of A.

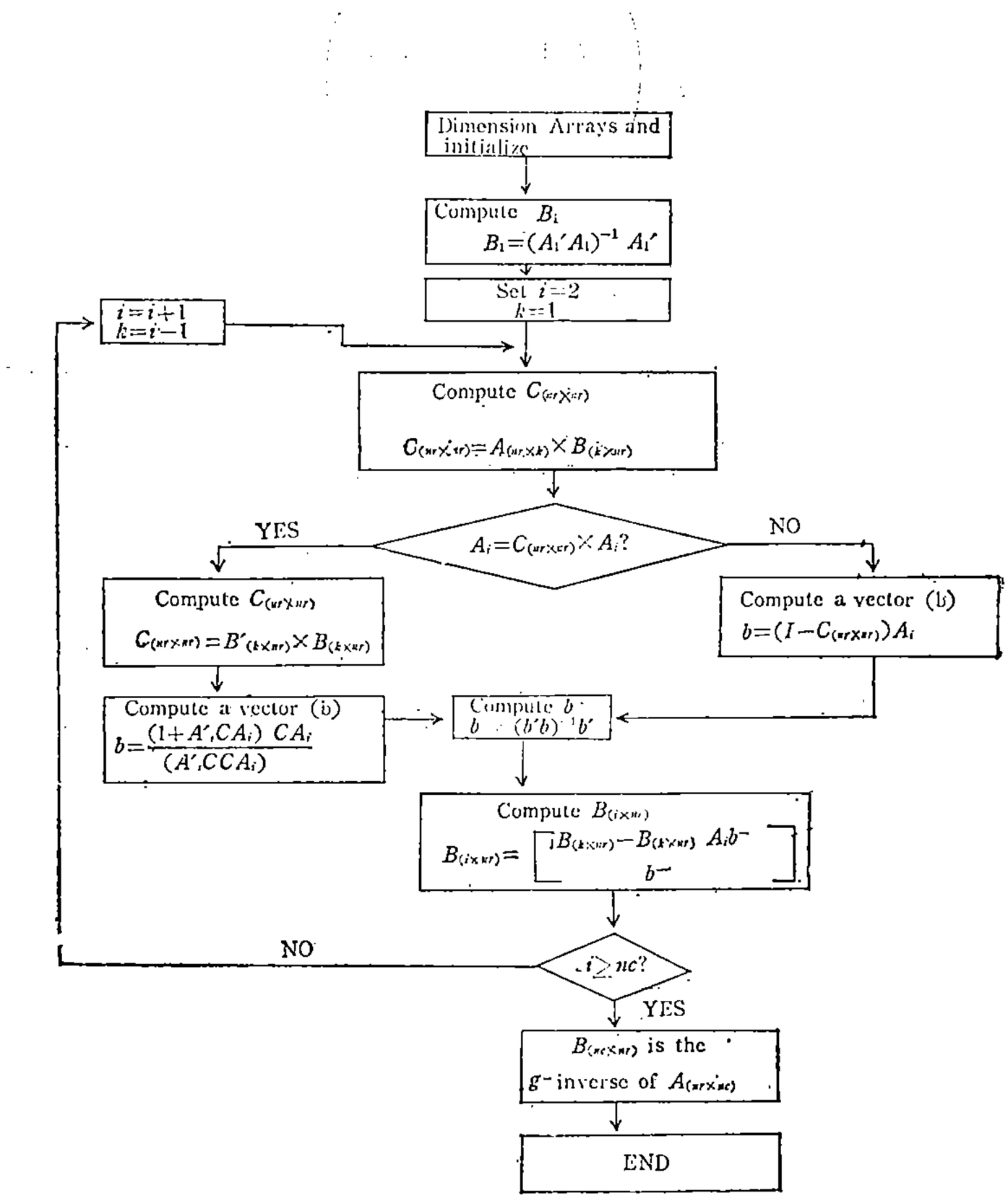
NC is the column dimension of A.

$A_{(nr\times i)}$ specifies the submatrix of the first i columns of A.

$B_{(i\times nr)}$ specifies the submatrix of the first i rows of B.

A_i and B_i specify the i^{th} column and row of A and B respectively.

CC must be dimensional at least $(nr\times nr)+(3\times nc)$.



ALGORITHM 2. Algorithm 2 describes another computing formula given by Graybill [Theorem 6. 5. 8, pp. 117—8, 1] to find the g-inverse of a matrix A.

Let A be an $m\times n$ matrix of rank r. The g-inverse of A is found by the following steps:

$$B=A'A$$
$$C_1=I$$
$$C_2=I\ (1/1)\ \mathrm{tr}\ C_1B-C_1B$$
$$C_3=I\ (1/2)\ \mathrm{tr}\ C_2B-C_2B$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

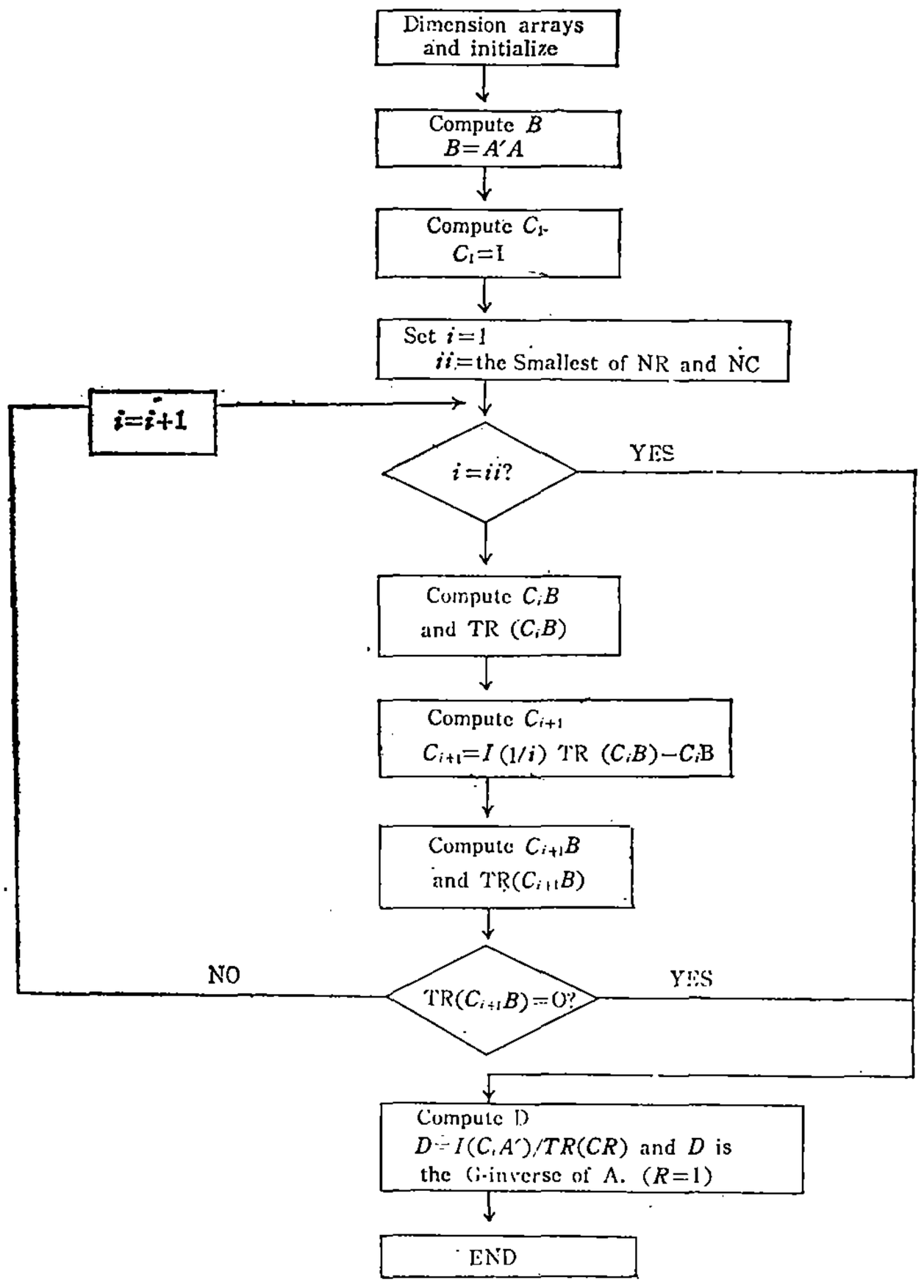
$$C_r = I\ (1/_{r-1})\ \text{tr}\ C_{r-1}B - C_{r-1}B$$

Terminate the interation if $C_{r+1}B=0$.

We have

$$A^g = rC_rA'/\text{tr}\ (C_rB),\ \text{tr}\ (C_rB) \neq 0.$$

Flow Chart Ⅱ translates Algorithm 2.



FLOW CHART Ⅱ

SUBROUTINE CINVX (*A*, *D*, *NR*, *NC*, *CC*)

WHERE:

A is a matrix to be inverted.

D is the *G*-inverse of *A*.

NR is the row dimension of *A*.

NC is the column dimension of *A*.

R specifies the rank of *A*.

TR(*X*) specifies the trace of a matrix *X*.

CC must be dimensioned at least $3\times(NC\times NC)$.

Table 1 summarizes our computer experience[2] of computing *g*-inverses utilizing Algorithms 1 [Theorem 6. 5. 1, pp. 108, 10, 1] and 2 [Theorem 6. 5. 8, pp. 117—8, i].

It was found that Algorithm 1 gave accurate *g*-inverses in all the cases.[3] Algorithm 2 produced errors in most of the cases excepting small-size matrices that are perhaps calculable utilizing a desk calculator.

In Flow Chart Ⅱ, two separate computing steps have been each marked by an asterisk. It was found that the summing operation in each starred step was the major contributor to errors. When the size of matrices grew, the summing operation in forming each element of $B=A'A$ and the summing of the diagonal elements of C_iB in computing the trace of C_iB each became the main source of errors.

Also it was found that Algorithm 1 required far shorter CPU time than Algorithm 2 excepting small-size matrices. Among varying sizes of matrices, however, computing *g*-inverses by Algorithm 1 required sharply increased CPU time as the size of a matrix grew. In Flow Chart Ⅰ, two separate computing steps have been each marked by an asterisk while a third one has been marked by double asterisks. As k became large, it appeared that progressively longer CPU time was required in the computing steps that are marked by a single asterisk. As the row dimension of the matrix became larger, increasingly longer CPU time was required in the computing step that is marked by double asterisks.

Table 1. Performances of two G-Inverse Computing Formulas

Order of Matrix[a]	CPU Time (Seconds. Milliseconds) and Accuracy[b]	
	Algorithm 1	Algorithm 2
1. Singular		
5×5	4.264 (*)	3.563 (*)
10×10	4.386 (*)	4.161 (**)
20×20	6.981 (*)	10.918 (**)
50×50	1 : 02.220[c] (*)	(N)
2. $m\times n$ $(m\neq n)$		
5×15	4.104 (*)	4.819 (*)
10×20	5.127 (*)	6.313 (**)
20×30	10.306 (*)	20.437 (**)
50×100	5 : 11.250[c] (*)	(N)

a. Random numbers generated by RANDU, a UNIVAC systems software. Each element consists of 6 digits decimal fractions.

b. Includes check and random number generating time. The symbols represent no error (*), error (**), no trial (N). No trial is recorded if an immediately preceding matrix produced too large an error warranting no trial on a larger matrix.

c. 1 minute 2 seconds 220 milliseconds, and 5 minutes 11 seconds 250 milliseconds.

It was concluded that Algorithm 2 should not be applied to compute g-inverses of martrices excepting small-size matrices that are perhaps calculable by a desk calculator. Evaluated in terms of CPU time and accuracy, Algorithm 1 was concluded to be a recommendable formula to compute g-inverses of the size of matrices that would reruire a computer processing.

Footnotes

* Kern O. Kymn's research for this project was supported by Contract DACA 31-73-C-0058, U.S. Army Corps of Engineers. The conclusions and opinions expressed in this paper do not necessarily reflect those of the General Services Administration, the U.S. Army Corps of Engineers or the Bureau of Labor Statistics.

1. Graybill recommends the formula given in Theorem 6. 5. 1 as perhaps the most useful for digital computers if a matrix is large.
2. Each element of the matrices in this paper was generated by utilizing RANDU, a UNIVAC 1108 random number generating systems software. Singular matrices were formed by equating the components in the first column to

those in the second column.

3. We applied the definition of the g-inverse to check the accuracy of computation. The numbers were printed out to the fifth decimal place utilizing floating point format.

West Virginia University
Bureau of Lavor Statistics
and General Service
Administration

REFERENCE

[1] Graybill, F. A., *Introduction to Matrices With Applications In Statistics*, Belmont, Wadsworth Publishing Co., 1969.