NASA TECHNICAL MEMORANDUM

NASA TM X-3003

NASA TM X-3003

STAR ADAPTATION FOR TWO ALGORITHMS USED ON SERIAL COMPUTERS

by Lona M. Howser and Jules J. Lambiotte, Jr.

Langley Research Center
Hampton, Va. 23665

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JUNE 1974

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM X-3003		

4. Title and Subtitle	5. Report Date
STAR ADAPTATION FOR TWO ALGORITHMS USED ON SERIAL COMPUTERS	June 1974
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Lona M. Howser and Jules J. Lambiotte, Jr.	L-9380
	10. Work Unit No.
9. Performing Organization Name and Address	023-11-16-05
NASA Langley Research Center Hampton, Va. 23665	11. Contract or Grant No.
	13 Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Two representative algorithms used on a serial computer and presently executed on the Control Data Corporation 6000 computer were adapted to execute efficiently on the Control Data STAR-100 computer. Gaussian elimination for the solution of simultaneous linear equations and the Gauss-Legendre quadrature formula for the approximation of an integral are the two algorithms discussed. This paper describes how the programs were adapted for STAR and why these adaptations were necessary to obtain an efficient STAR program. Some points to consider when adapting an algorithm for STAR are discussed. Program listings of the 6000 version coded in 6000 FORTRAN, the adapted STAR version coded in 6000 FORTRAN, and the STAR version coded in STAR FORTRAN are presented in the appendices.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
STAR computer Gaussian elimination Numerical integration	Unclassified – Unlimited STAR Category 19

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	30	$3.25

*For sale by the National Technical Information Service, Springfield, Virginia 22151

STAR ADAPTATION FOR TWO ALGORITHMS USED ON SERIAL COMPUTERS

By Lona M. Howser and Jules J. Lambiotte, Jr.
Langley Research Center

SUMMARY

Two representative algorithms used on a serial computer and presently executed on the Control Data Corporation 6000 computer were adapted to execute efficiently on the Control Data STAR-100 computer. Gaussian elimination for the solution of simultaneous linear equations and the Gauss-Legendre quadrature formula for the approximation of an integral are the two algorithms discussed. This paper describes how the programs were adapted for STAR and why these adaptations were necessary to obtain an efficient STAR program. Some points to consider when adapting an algorithm for STAR are discussed. Program listings of the 6000 version coded in 6000 FORTRAN, the adapted STAR version coded in 6000 FORTRAN, and the STAR version coded in STAR FORTRAN are presented in the appendices.

INTRODUCTION

Many algorithms which are presently used on the Control Data Corporation 6000 computer and executed in a serial mode are suitable for the Control Data STAR-100 computer. However, if these algorithms were converted line-by-line to the STAR coding, it is not likely that they would make an efficient STAR program and might actually produce code which would run inefficiently on a vector computer such as STAR. The 6000 code will need to be adapted for STAR to produce an efficient STAR code. This paper discusses two algorithms of this nature. A comparison of the 6000 coding and STAR coding of the identical algorithm is made for two different algorithms: one for the solution of simultaneous linear equations using Gaussian elimination with partial pivoting and the other for numerical evaluation of an integral using the Gauss-Legendre quadrature formula. The paper discusses how the 6000 program was adapted for STAR, the reasons for the adaptations, and some factors which should be considered when adapting a program. FORTRAN codings of the algorithms are presented in the appendices. The STAR codings use the FORTRAN language defined in reference 1.

AIDS FOR ADAPTING AN ALGORITHM

When beginning to adapt an algorithm for STAR, it is not enough to look at just segments in the 6000 coding, but it is necessary to look at the entire algorithm to get the total picture. Some questions to pose are: What is the final result?, What is needed at various steps in the algorithm?, and What is computed independent of other steps and what is repeated?

It will be helpful to review a few definitions and terms which are important to remember when formulating a program for the STAR computer. (For a comprehensive discussion of STAR architecture and hardware instructions, see ref. 2.)

(1) A vector is a set of elements stored in contiguous locations in memory. (Array and vector are used interchangeably in this discussion.)

(2) Vector timing is the time required by the central processing unit to process a vector. It is obtained by the equation $T = S + l/c$, where

T	time in clocks (1 clock represents 40 nanoseconds)
S	startup time, different for each vector instruction or macro
l	length of vector (number of elements in the vector)
c	constant depending on the type of instruction and whether the vector elements are each stored in 32- or 64-bit words

(3) A page is a block of storage which contains data or instructions. A program is made up of one or more pages. (See ref. 3.)

(4) A program's working set is the smallest set of pages which must be in central memory for the program to operate efficiently.

(5) A page fault occurs when a program references a page which is not contained in central memory; that is, it is not in the program's present working set.

(6) Paging is the process of bringing a page into central memory or releasing a page.

When adapting the 6000 code to the STAR code, the following factors are important:

(1) Use vector instructions. This requirement may mean reordering steps in an algorithm so that elements in contiguous locations can be operated on or it may mean rearranging storage.

(2) Use long vectors. If a choice is available whether to use many short vectors or a few long vectors, use the long vectors unless the overhead to create the long vectors

is too great. This procedure reduces the effect of the startup time associated with each vector instruction.

(3) Avoid or use sparingly coding which will generate costly vector instructions and macros, that is, costly as compared with some of the faster vector instructions. Refer to the most current timings available. Examples of such instructions and macros are divide, transmit indexed list, and dot product.

(4) Avoid unnecessary paging problems by creating a reasonable working set for the program. When working with a particular array, perform all the operations possible with this array before working with another array. This factor will be more critical with long arrays, but it is a good habit to form and should help reduce page faults.

(5) Investigate the feasibility of creating more answers than are really needed. Because of the high result rate of vector instructions, it may be advantageous to use an approach which generates a larger number of results than are needed in order to avoid scalar computation. This idea, however, should be used cautiously. (See the discussion in ref. 4 on parallel algorithms for tri-diagonal equation solvers.)

ADAPTATION OF AN ALGORITHM FOR THE SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS

A Langley Research Center 6000 library subroutine GELIM (see listing in appendix A) uses Gaussian elimination with partial pivoting to obtain the solution of the set of simultaneous linear equations, $AX = B$, where A is the square matrix of coefficients of order n, X is a vector of unknowns of length n, and B is a constant matrix of order $n \times r$ where r is the number of right-hand sides. The matrix A is factored into a lower unit triangular matrix and an upper triangular matrix. (For numerical details of the algorithm, see any numerical analysis text, such as ref. 5.) The subroutine also contains an option for the evaluation of the determinant of matrix A.

The version of GELIM on the Langley Research Center 6000 library performs Gaussian elimination as generally defined by operating on one row of A at a time. At the kth step, column k is searched for the largest element; then row k and the row which contained the largest element are interchanged. The pivot element is then used to obtain zeroes in all positions in its column below the diagonal. This procedure requires multiplying a row of A by the appropriate scalar and subtracting this product from another row of A.

Since these row modifications are the usual way the steps in Gaussian elimination are thought of being performed, it would seem normal that for STAR the matrix would be

stored by rows. This method of storage means that elements in one row of a matrix would be stored in contiguous locations so a row of the matrix could be a vector and the steps would be performed by using vector instructions.

In the present version of STAR FORTRAN, two-dimensional arrays are stored columnwise and an optional storage arrangement by rows is not available. This column storage means that elements in one row of matrix A are not stored in contiguous locations and modifying the matrix a row at a time would mean that few vector instructions could be used. This fact makes it desirable to see whether Gaussian elimination can be performed by modifying matrix A by using columns so that vector instructions can be used. As will be shown below, Gaussian elimination can be performed by operating on columns of the matrix.

An option to store matrices by rows may be available in a later version of STAR FORTRAN, but when the 6000 code was modified to perform Gaussian elimination by columns, many advantages for the column storage over the row storage appeared; therefore, column storage is recommended. The following section will show how Gaussian elimination can be performed by using vector instructions when the matrix is stored by columns and will identify these advantages. The section entitled "Row Storage" shows the sequence of steps performed in the row storage which is identical to the present 6000 algorithm.

Column Storage

Gaussian elimination can be performed by modifying one column of the matrix at a time. This is done by a reordering of the operations from the usual row operations. Accomplishing the triangularization of matrix A by performing the work on columns makes efficient use of STAR and does the identical arithmetic normally done when performing the work on rows.

The kth step of the triangularization can be performed as shown below, where n is the number of equations and r is the number of right-hand sides. All references to the kth column refer to column entries below the diagonal.

(1) Divide the kth column of A by the a_{kk} element and store in the kth column. This is a vector divided by a scalar:

$$a_{ik} = \frac{a_{ik}}{a_{kk}} \qquad (i = k + 1, \ldots, n)$$

(2) Multiply the kth column by the a_{kj} element and subtract this result from the jth column. This is a vector multiplied by a scalar and then a vector subtract:

$$a_{ij} = a_{ij} - a_{ik}a_{kj} \qquad (i = k + 1, \ldots, n)$$

(3) Repeat the sequence of vector instructions in step (2) for the columns of A $(j = k + 1, \ldots, n)$.

(4) Multiply the kth column of A by the b_{kj} element of B and subtract this result from the jth column of B. This is a vector multiplied by a scalar and then a vector subtract:

$$b_{ij} = b_{ij} - a_{ik}b_{kj} \qquad (i = k + 1, \ldots, n)$$

(5) Repeat the vector instructions in step (4) for the remaining right-hand sides of B $(j = 1, 2, \ldots, r)$.

(6) Repeat steps (1) to (5) until the triangularization of matrix A is complete $(k = 1, 2, \ldots, n - 1)$.

Often a subroutine for the solution of simultaneous equations requires the user to append the right-hand sides to the original matrix. The column storage arrangement allows the right-hand sides to be done as separate vector instructions without having to append the right-hand sides to the original matrix. This arrangement is less cumbersome for the user since the right-hand side can be a separate array.

GELIM uses partial pivoting which means that rows will need to be interchanged at times. At the kth step, column k is searched for the largest element and the row containing the largest element and the kth row are interchanged. The column search to find the largest element can easily make use of vector instructions, but the row interchange presents a problem. Elements of rows will need to be interchanged and none of them will be stored in contiguous locations. Not only are they not contiguously stored, but for a very large matrix a column could use one or more pages. For a large matrix it is unlikely that a program will be allowed a working set large enough to contain the entire matrix. This situation would mean that the row elements to be interchanged would be on separate pages and would have to be brought into and then out of core only to reference two elements on the page. Then when the column modifications are performed, the same pages will need to be brought back into core again. This procedure would be extremely inefficient.

A form of indexing could be set up to achieve row interchange, but it would need one of the more time-consuming instructions (transmit indexed list) and the referencing across page boundaries would still be present. No vector instruction can help perform the interchange efficiently; thus, a scalar interchange will be just as efficient.

The possible paging problem can be alleviated by not interchanging an entire row at one time as it is performed on the present 6000 version. At the kth step when the largest element in column k is found, interchange the two elements only in column k. When working with the jth column, both column k and column j are needed. Both of the elements which need interchanging are in the jth column which will be in the program's working set at that time. Therefore, before beginning the operations on the jth column, interchange the two elements. In this way, the page or pages containing that column will need to be brought into core only once per step of the algorithm.

The subroutine also computes the value of the determinant of matrix A which is equal to the product of the elements of the diagonal of the triangular matrix. In the 6000 version, the product is computed after the triangularization is completed. For the STAR, this computation presents a similar situation as the rows interchange; the diagonal elements may be on separate pages and are not stored contiguously. Therefore, vector instructions cannot be utilized. The product will be scalar multiplication, but after the kth step has been completed, the partial product can be formed by using the diagonal element of column k while column k is still in the program's working set. This grouping of row interchange, triangularization of the matrix, and evaluation of the determinant should create an efficient working set for the program with a minimum of paging.

The remaining task of the subroutine is to perform the back substitution. Back substitution generally uses the dot product of a row and the solution vector. This method is used on the 6000 version, but presents problems for STAR column storage since it uses the costly dot product macro and references elements of one array by rows and references elements of the other array by columns. This problem can be eliminated by reordering the steps needed to perform back substitution and all the work can be performed on columns.

The steps to find the kth unknown by back substitution by using vector instructions but not using the dot product are as follows:

(1) A scalar divide is always necessary, $b_{kj} = \frac{b_{kj}}{a_{kk}}$. This step obtains the kth unknown for the jth right-hand side and stores the unknown in the right-hand side vector.

(2) Multiply column k of A by the kth unknown obtained in step (1) and subtract this result from the jth column of B. This is a vector multiplied by a scalar and then a vector subtract:

$$b_{ij} = b_{ij} - a_{ik}b_{kj} \qquad (i = 1, 2, \ldots, k - 1)$$

(3) Steps (1) and (2) are repeated for all the right-hand sides (columns of B) (j = 1, 2, . . ., r).

(4) Steps (1), (2), and (3) are repeated for all the unknowns ($k = n, \ldots, 2, 1$). When $k = 1$, step (2) is omitted.

Row Storage

The way of performing the steps in Gaussian elimination as commonly seen in texts is by operating with rows. The elements in one row would be stored contiguously and the triangularization of matrix A would make efficient use of vector instructions.

To make the triangularization most efficient, the right-hand sides must be appended to the matrix A for row storage. The length of the longest vector used would be $n + r - 1$ where n is the number of unknowns and r is the number of right-hand sides. If a separate array is used for the right-hand side, two identical vector instructions would be needed for each operation, one of length n and one of length r. This method would be inefficient because the startup times would be multiplied by a factor of 2 and if r is 1, it would mean using vector instructions of length 1. Appending the right-hand side is a disadvantage in that it is awkward for the user.

Step k of the triangularization when the matrix is stored by rows is as follows:

(1) Perform a scalar divide, a_{ik}/a_{kk} and store in a_{ik}.

(2) Multiply row k of A by a_{ik} and subtract from the ith row. This is a vector multiplied by a scalar and then a vector subtract:

$$a_{ij} = a_{ij} - a_{ik}a_{kj} \qquad (j = k + 1, \ldots, n + r)$$

(3) Repeat steps (1) and (2) for all rows ($i = k + 1, \ldots, n$).

(4) Repeat steps (1), (2), and (3) for all columns until the triangularization is complete ($k = 1, 2, \ldots, n - 1$).

The row interchanges necessary for partial pivoting are very easy when the matrix is stored by rows, but the column search will be scalar operations. In addition, there will be no way to avoid possible paging problems for a large matrix during the column search. The elements on each row of the column may be on separate pages, but the search has to be completed before any row operations can begin.

The determinant evaluation could be performed the same way as in the column storage, after a step in the Gaussian elimination is completed. The pages for a large matrix would not have to be brought in only to get one element for the evaluation of the determinant, but the information would be used while the page was still in memory.

When performing the back substitution, if the matrix is stored by rows, there is no way of using the scheme devised to eliminate the need for the dot product macro. The

back substitution references a row of the matrix and a column of the right-hand side. To be able to use vector instructions, a vector would need to be created which would contain the elements in the column. This method would be expensive and inefficient, and it is likely that the vector code would be no better than the scalar code. Either would be inefficient here, because of the referencing of a row and a column.

Table I summarizes the comparison of the two storage arrangements at the various steps in the algorithm. The column storage is more advantageous for STAR than the row storage arrangement.

TABLE I.- COMPARISON OF ROWWISE AND COLUMNWISE STORAGE OF MATRIX

Step	Rowwise	Columnwise
Column search	Scalar: no way to avoid possible paging problems	Vector
Row interchange	Vector: easy	Scalar: can avoid possible paging problems
Triangularization	Vector: usual way it is done	Vector: with steps reordered
Back substitution	Dot product macro: costly, referencing row and column, possibly not vectorizable	Vectors: columns only, no dot product, more efficient
Determinant evaluation	Scalar	Scalar
Treatment of right-hand sides	Must be appended to matrix for efficient vector use	Can be separate variable and still use vectors efficiently

Flow Chart

Subroutine GELIM was adapted for use on the STAR computer by using the same numerical method that was used on the 6000 computer. The matrix is stored by columns. By reordering computational steps, vector instructions can be used and a reasonable working set for the program established. Figure 1 shows a flow chart of the 6000 and STAR versions of the algorithm.

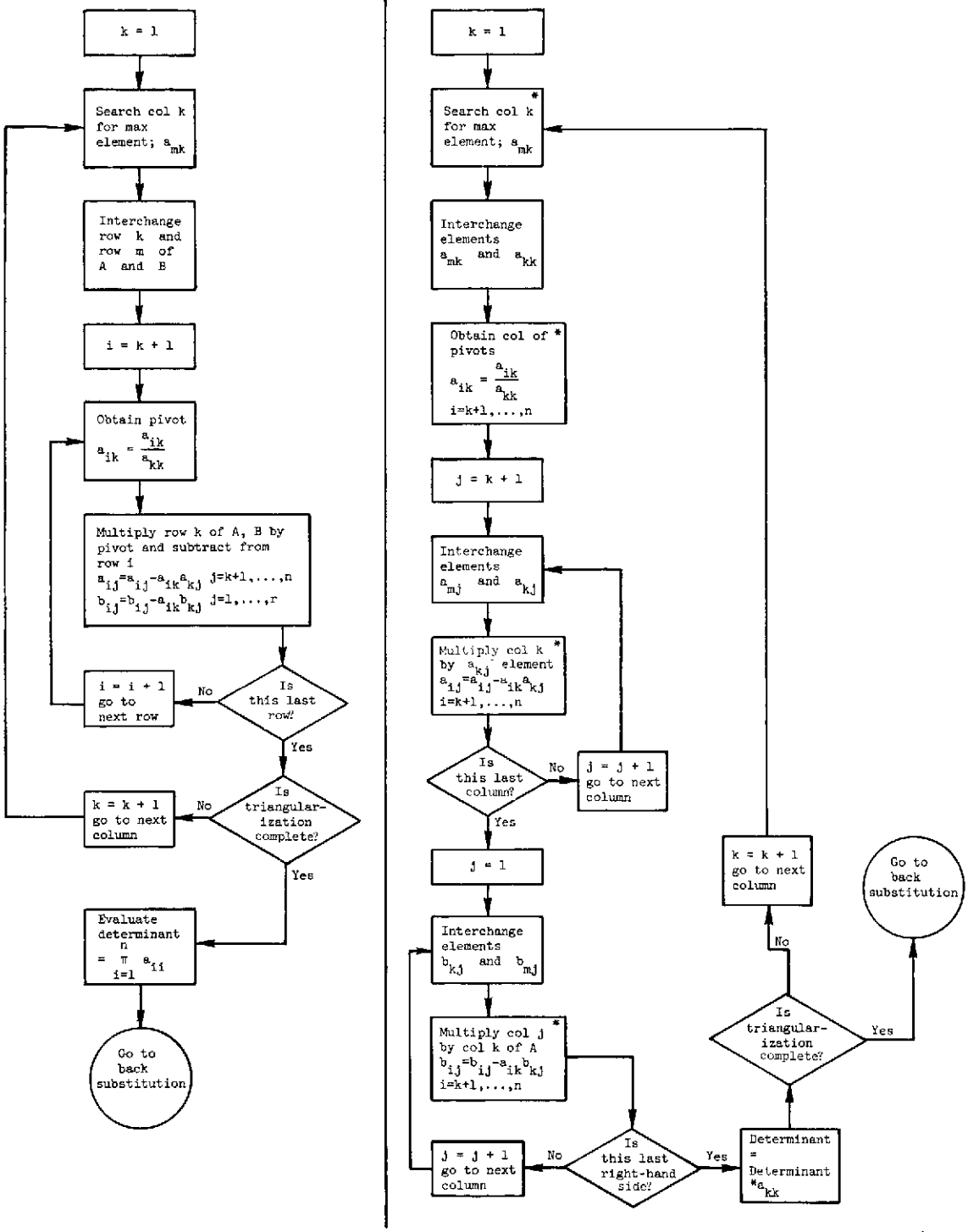

Figure 1.- Flow chart of 6000 and STAR version of segments of Gaussian elimination. An asterisk denotes use of vector instructions.

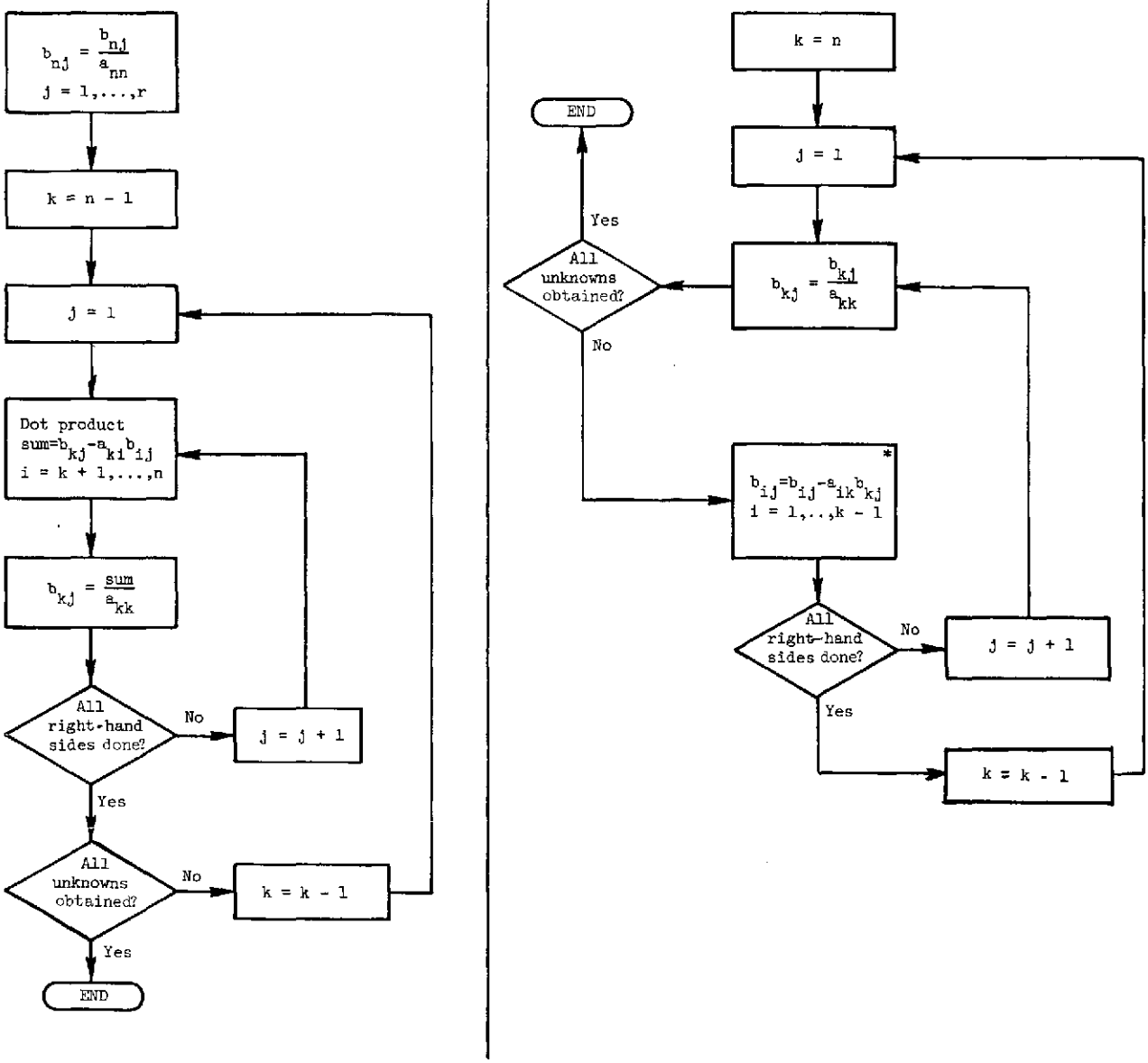

Figure 1.- Concluded.

Appendix A contains coding of the versions of the algorithm. The 6000 version and the STAR version coded in 6000 FORTRAN and in STAR FORTRAN are included. The calling sequence of the subroutine remains the same and no additional storage was necessary.

Additional Comment

An additional interesting fact was noted when the STAR version of subroutine GELIM was executed on the 6000 computer. The STAR version ran faster than the 6000 version; as a result, there was about a 10-percent decrease in execution time.

ADAPTATION OF A NUMERICAL INTEGRATION ALGORITHM

Subroutine GLEGEN (see listing in appendix B) uses the Gauss-Legendre quadrature formula to evaluate simultaneously an array of integrals

$$\int_a^b f_k(x)\,dx \qquad (k = 1, 2, \ldots, N)$$

where N is the number of functions. This subroutine is an example of the manner in which vectors rather than single variables can be used on STAR. As in the preceding example, the sequence of instructions is not the same as in the 6000 version. The subroutine subdivides the integration interval (a, b) into NQ panels and the Gauss-Legendre quadrature formula is applied to each panel by using a 3- or 10-point formula. The resulting integral I_k is

$$I_k = \Delta \sum_{j=1}^{NQ} \sum_{i=1}^{NP} r_i f_k(p_i) \qquad (k = 1, 2, \ldots, N)$$

where

f_k function

NQ number of quadratures or panels

NP number of points per quadrature

$p_i = l_j + \Delta x_i$ (where l_j is the lower limit of integration for quadrature j)

r_i weights for the point formula used

x_i abscissas for the point formula used

$$\Delta = \frac{b - a}{NQ}$$

(For a complete discussion of the Gauss-Legendre quadrature formula, see any numerical analysis text, such as ref. 5.)

The 6000 version of the subroutine obtains the first point in a quadrature, evaluates the function at that point, multiplies the function by the appropriate weight, and adds the product to a sum. It continues in the same manner until all the points have been evaluated and used in that quadrature; then the process is repeated for the remaining quadratures until the integral has been evaluated over the desired range. The sum is then multiplied by the appropriate delta to obtain the value of the integral.

Looking at this sequence of instructions shows that much of this work is actually computed independently. The STAR version takes advantage of this. All the information required to compute the points is known so all the points for all the quadratures can be computed to form a long array. This array can be passed to the subroutine to evaluate the functions and a savings can be obtained by using vector instructions to evaluate the functions. This routine will now return an array of function values. All the weights are known so an array can be formed which contains the weights for all the quadratures. The remaining task is to multiply the weights by the functions and sum the results. This computation can use the dot product macro; even though it is costly compared with most vector instructions, it is less expensive than a vector multiply followed by scalar adds. This formulation requires the use of the dot product only once. A scalar multiply of the sum by the appropriate delta completes the integral evaluation.

The weights and abscissas for each quadrature are identical; therefore, the length of these arrays in the 6000 version is equal to 10 for the 10-point formula. A version of the subroutine using short vectors could have been formed. The STAR version could have used vector instructions and computed results using one quadrature at a time with the same array of weights and array length as in the 6000 version. Computing the results one quadrature at a time means that the startup time associated with each vector instruction and the startup times used in the function evaluation also would be multiplied by the number of quadratures used. Depending upon the nature of the function, this amount of time could be significant.

Vector transmits could be used to create a long vector of weights which is really a repetition of the short vector. The long vector is

$$(r_1, r_2, \ldots, r_m, r_1, r_2, \ldots, r_m, \ldots, \overbrace{r_1, r_2, \ldots, r_m}^{q \text{ repetitions}})$$

where m depends upon the quadrature formula and q is the number of quadratures. The cost of setting up this vector seems to be minimal when compared with the multiplication of the startup times which would have prevailed in the short-vector version.

A problem of this type (excluding the function evaluation) would probably not produce a significant paging problem. All the arrays are one dimensional and will most likely not be extremely long. However, consideration should still be given to the working set which is created. Currently, there is a tendency to initialize variables at the beginning of the program. The vector of weights could be created at the beginning of the program, but they are not used until after the functions are evaluated; therefore, the vector is not created until just before it is used. Forming the vector of weights and then immediately using them results in less paging; therefore, a better working set is created than would have been created by initializing the vector.

Subroutine GLEGEN was adapted for the STAR computer by using the same numerical method as was used on the 6000. An efficient STAR routine was obtained by using long vectors and reordering the sequence of instructions. Figure 2 shows a flow chart of the 6000 and STAR versions.

Appendix B contains the coding of the two versions of the algorithm. The 6000 version and the STAR version in 6000 FORTRAN and STAR FORTRAN are included. Notice that the calling sequence of the STAR version reflects the use of long arrays in that more working space is needed.

CONCLUDING REMARKS

Two algorithms used on the Control Data Corporation 6000 computer were adapted for use on the STAR computer. This adaptation required a rethinking of the flow of the entire problem. Array variables are used where the 6000 code used a single-value variable and the steps in the algorithm are not computed in the same order for the two versions. This reordering of steps and changes in variable assignments allow vector instructions to be used and a reasonable working set can be established for the program; thereby efficient use of the STAR computer and some implied improvements for the Control Data Corporation 6000 computer are made.

Langley Research Center,
National Aeronautics and Space Administration,
Hampton, Va., February 20, 1974.

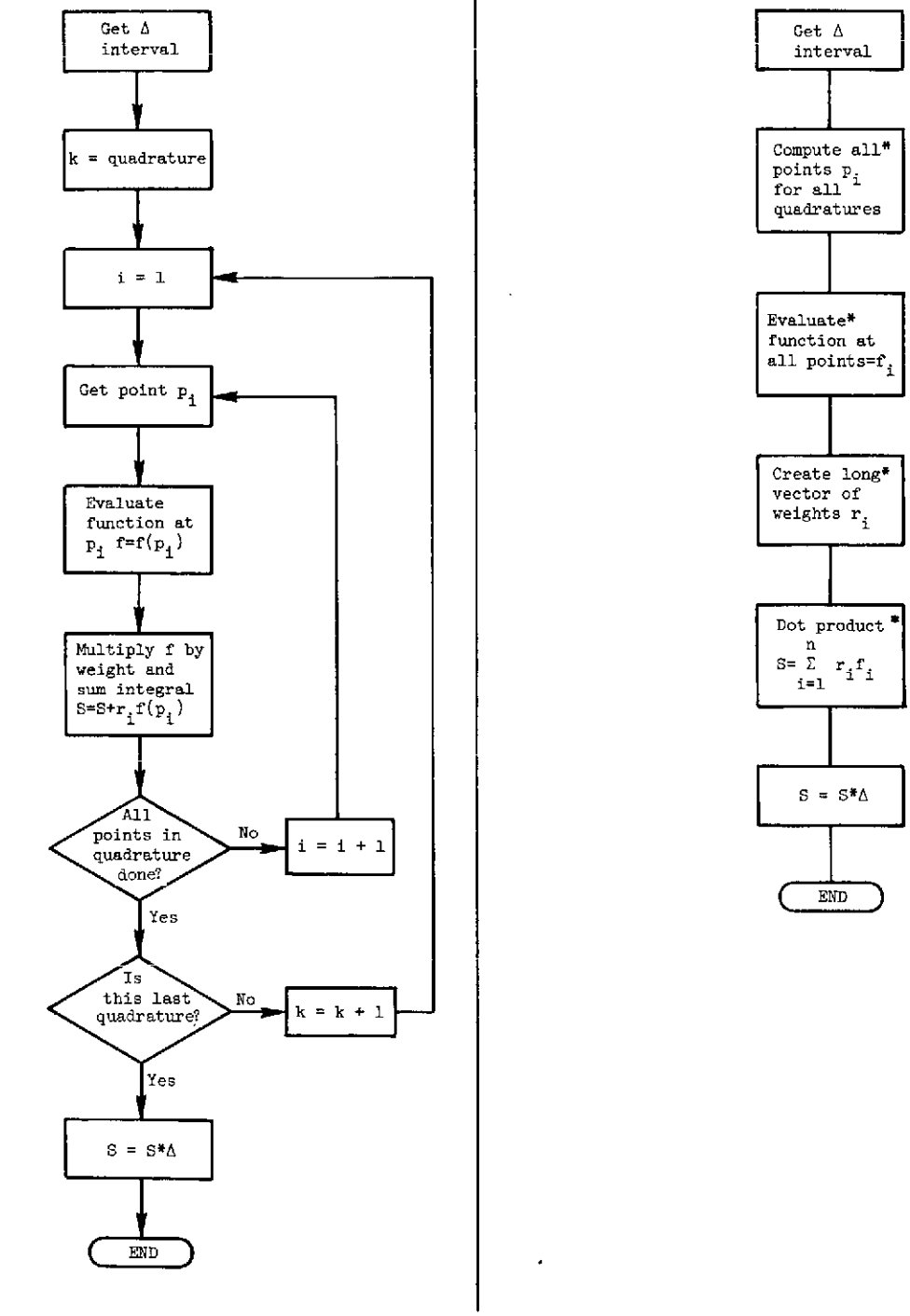

Figure 2.- Flow chart of 6000 and STAR versions of Gauss-Legendre quadrature formula for numerical integration. An asterisk denotes use of vector instructions.

APPENDIX A

FORTRAN CODING OF 6000 AND STAR VERSIONS OF ALGORITHM FOR THE SOLUTION OF A SET OF SIMULTANEOUS EQUATIONS

6000 Version in 6000 FORTRAN

```
      SUBROUTINE GELIM(A,N,B,NRHS,MAXN,IPIVOT,IOP,DETERM,ISCALE)          GELIM001
      DIMENSION A(MAXN,MAXN),B(MAXN,NRHS),IPIVOT(MAXN)                   GELIM002
C     A(MAXN,MAXN) = SQUARE MATRIX OF COEFFICIENTS(A IS DESTROYED)       GELIM003
C     N = NUMBER OF ROWS AND COLUMNS IN A                                GELIM004
C     B(MAXN,NRHS) = MATRIX OF CONSTANTS(REPLACED BY SOLUTION MATRIX)    GELIM005
C     NRHS = NUMBER OF COLUMNS IN B                                      GELIM006
C     MAXN = MAXIMUM NUMBER OF ROWS AND COLUMNS IN A                     GELIM007
C     IPIVOT(MAXN) = RECORD OF ROW INTERCHANGES                          GELIM008
C     IOP = IOP=1,EVALUATE DETERMINANT. IOP=0,SKIP DETERMINANT EVALUATION    009
C     DETERM = GIVES VALUE OF DETERMINAT(DETERM=(10**100)**ISCALE*DETERM)    010
C     ISCALE = SCALE FACTOR COMPUTED BY SUBROUTINE IN DETERM EVALUATION GELIM011
C              TO KEEP DETERM WITHIN THE FLOATING POINT WORD SIZE OF THE     012
C              COMPUTER                                                  GELIM013
      SIGN=1.0                                                           GELIM020
      DO 35 I=1,N                                                        GELIM030
35    IPIVOT(I)=I                                                        GELIM040
    5 ISCALE=0                                                           GELIM050
    6 R1=10.0**100                                                       GELIM060
    7 R2=1.0/R1                                                          GELIM070
   10 DETERM=1.0                                                         GELIM080
      NM1=N-1                                                            GELIM090
      IF(NM1.EQ.0) GO TO 1000                                            GELIM100
C*****FIND LARGEST REMAINING TERM IN I-TH COLUMN FOR PIVOT               GELIM110
      DO 200 I=1,NM1                                                     GELIM120
      BIG=0.                                                             GELIM130
      DO 50 K=I,N                                                        GELIM140
      TERM=ABS(A(K,I))                                                   GELIM150
      IF(TERM-BIG)50,50,30                                               GELIM160
30    BIG=TERM                                                           GELIM170
      L=K                                                                GELIM180
50    CONTINUE                                                           GELIM190
      IPIVOT(I)=L                                                        GELIM200
      IF(BIG)80,60,80                                                    GELIM210
60    DETERM=0.0                                                         GELIM220
      RETURN                                                             GELIM225
80    IF(I-L)90,120,90                                                   GELIM230
******SWAP ROWS OF A AND B. SET IPIVOT(J)=K.                             GELIM240
90    SIGN=-SIGN                                                         GELIM250
      DO 100 J=1,N                                                       
      TEMP1=A(I,J)                                                       GELIM270
      A(I,J)=A(L,J)                                                      GELIM280
100   A(L,J)=TEMP1                                                       GELIM290
      DO 101 J=1,NRHS                                                    GELIM300
      TEMP2=B(I,J)                                                       GELIM310
      B(I,J)=B(L,J)                                                      GELIM320
101   B(L,J)=TEMP2                                                       GELIM330
120   CONTINUE                                                           GELIM340
******STORE PIVOT IN A(I,J). MULTIPLY A AND B BY PIVOT.                  GELIM350
      IP1=I+1                                                            GELIM360
      DO 31 II=IP1,N                                                     GELIM370
      A(II,I)=A(II,I)/A(I,I)                                             GELIM380
      X3=A(II,I)                                                         GELIM390
      DO 32 K=IP1,N                                                      GELIM400
```

```
32      A(II,K)=A(II,K)-X3*A(I,K)                                       GELIM410
        DO 33 K=1,NRHS                                                  GELIM420
33      B(II,K)=B(II,K)-X3*B(I,K)                                       GELIM430
31      CONTINUE                                                        GELIM440
200     CONTINUE                                                        GELIM450
        IF(IOP.EQ.0)GO TO 221                                           GELIM460
C                                                                       GELIM470
******SCALE THE DETERMINANT                                            GELIM480
C                                                                       GELIM490
        DO 122 I=1,N                                                    GELIM500
        PIVOTI=A(I,I)                                                   GELIM510
 1005   IF(ABS(DETERM)-R1)1030,1010,1010                                GELIM520
 1010   DETERM=DETERM/R1                                                GELIM530
        ISCALE=ISCALE+1                                                 GELIM540
        IF(ABS(DETERM)-R1)1060,1020,1020                                GELIM550
 1020   DETERM=DETERM/R1                                                GELIM560
        ISCALE=ISCALE+1                                                 GELIM570
        GO TO 1060                                                      GELIM580
 1030   IF(ABS(DETERM)-R2)1040,1040,1060                                GELIM590
 1040   DETERM=DETERM*R1                                                GELIM600
        ISCALE=ISCALE-1                                                 GELIM610
        IF(ABS(DETERM)-R2)1050,1050,1060                                GELIM620
 1050   DETERM=DETERM*R1                                                GELIM630
        ISCALE=ISCALE-1                                                 GELIM640
 1060   IF(ABS(PIVOTI)-R1)1090,1070,1070                                GELIM650
 1070   PIVOTI=PIVOTI/R1                                                GELIM660
        ISCALE=ISCALE+1                                                 GELIM670
        IF(ABS(PIVOTI)-R1)320,1080,1080                                 GELIM680
 1080   PIVOTI=PIVOTI/R1                                                GELIM690
        ISCALE=ISCALE+1                                                 GELIM700
        GO TO 320                                                       GELIM710
 1090   IF(ABS(PIVOTI)-R2)2000,2000,320                                 GELIM720
 2000   PIVOTI=PIVOTI*R1                                                GELIM730
        ISCALE=ISCALE-1                                                 GELIM740
        IF(ABS(PIVOTI)-R2)2010,2010,320                                 GELIM750
 2010   PIVOTI=PIVOTI*R1                                                GELIM760
        ISCALE=ISCALE-1                                                 GELIM770
  320   DETERM=DETERM*PIVOTI                                            GELIM780
122     CONTINUE                                                        GELIM790
C                                                                       GELIM800
******PERFORM BACK SUBSTITUTION                                        GELIM810
221     CONTINUE                                                        GELIM820
        DO 57 IC=1,NRHS                                                 GELIM830
        B(N,IC)=B(N,IC)/A(N,N)                                          GELIM840
57      CONTINUE                                                        GELIM850
        DO 61 KK=1,NM1                                                  GELIM860
        I=N-KK                                                          GELIM870
        I1=I+1                                                          GELIM880
        DO 61 J=1,NRHS                                                  GELIM900
        SUM=B(I,J)                                                      GELIM910
        DO 62 K=I1,N                                                    GELIM920
   62   SUM=SUM-A(I,K)*B(K,J)                                           GELIM930
61      B(I,J)=SUM/A(I,I)                                               GELIM940
        DETERM=DETERM*SIGN                                              GELIM950
        RETURN                                                          GELIM960
1000    IF(A(1,1).EQ.0) GO TO 60                                        GELIM970
        DO 1500 J=1,NRHS                                                GELIM975
1500    B(1,J)=B(1,J)/A(1,1)                                            GELIM980
        DETERM=A(1,1)                                                   GELIM990
        RETURN                                                          GELIM995
        END                                                             GELM1000
```

STAR Version in 6000 FORTRAN

```
      SUBROUTINE GELIM(A,N,B,NRHS,MAXN,IPIVOT,IOP,DETERM,ISCALE)          GIL60010
      DIMENSION A(MAXN,MAXN),B(MAXN,NRHS),IPIVOT(MAXN)                   GIL60020
C     A(MAXN,MAXN) = SQUARE MATRIX OF COEFFICIENTS(A IS DESTROYED)       GIL60030
C     N = NUMBER OF ROWS AND COLUMNS IN A                                GIL60040
C     B(MAXN,NRHS) = MATRIX OF CONSTANTS(REPLACED BY SOLUTION MATRIX)    GIL60050
C     NRHS = NUMBER OF COLUMNS IN B                                      GIL60060
C     MAXN = MAXIMUM NUMBER OF ROWS AND COLUMNS IN A                     GIL60070
C     IPIVOT(MAXN) = RECORD OF ROW INTERCHANGES                          GIL60080
C     IOP - IOP=1,EVALUATE DETERMINANT. IOP=0,SKIP DETERMINANT EVALUATIOGIL60090
C     DETERM - GIVES VALUE OF DETERMINAT(DETERM=(10**100)**ISCALE*DETERMGIL60100
C     ISCALE = SCALE FACTOR COMPUTED BY SUBROUTINE IN DETERM EVALUATION GIL60110
C              TO KEEP DETERM WITHIN THE FLOATING POINT WORD SIZE OF THEGIL60120
C              COMPUTER                                                  GIL60130
      IO=0                                                               GIL60140
      SIGN=1.0                                                           GIL60150
      DO 3 I=1,N                                                         GIL60160
    3 IPIVOT(I)=I                                                        GIL60170
    5 ISCALE=0                                                           GIL60180
    6 R1=10.0**100                                                       GIL60190
    7 R2=1.0/R1                                                          GIL60200
   10 DETERM=1.0                                                         GIL60210
      NM1=N-1                                                            GIL60220
      IF (NM1.EQ.0) GO TO 600                                            GIL60230
      NP1=N+1                                                            GIL60240
C*****FIND LARGEST REMAINING TERM IN I-TH COLUMN FOR PIVOT               GLI60250
      DO 200 I=1,NM1                                                     GIL60260
      BIG=0.                                                             GIL60270
      DO 50 K=I,N                                                        GIL60280
      TERM=ABS(A(K,I))                                                   GIL60290
      IF(TERM-BIG)50,50,30                                               GIL60300
30    BIG=TERM                                                           GIL60310
      L=K                                                                GIL60320
50    CONTINUE                                                           GIL60330
      IPIVOT(I)=L                                                        GIL60340
      IF(BIG)80,60,80                                                    GIL60350
60    DETERM=0.0                                                         GIL60360
      RETURN                                                             GIL60370
80    IF(I-L)90,120,90                                                   GIL60380
C*****INTERCHANGE ELEMENTS IN COLUMN I ONLY
90    SIGN=-SIGN                                                         GIL60400
      TEMP1 = A(I,I)                                                     GIL60410
      A(I,I)=A(L,I)                                                      GIL60420
      A(L,I)=TEMP1                                                       GIL60430
120   CONTINUE                                                           GIL60440
C*****STORE PIVOT IN A(I,J). MULTIPLY A AND B BY PIVOT.                  GIL60450
      IP1=I+1                                                            GIL60460
C*****OBTAIN COLUMN OF PIVOTS
      DO 128 K=IP1,N                                                     GIL60470
  128 A(K,I)=A(K,I)/A(I,I)                                               GIL60480
C*****INTERCHANGE ELEMENTS ONLY IN COLUMN J
      DO 132 J=1,NRHS                                                    GIL60490
      TEMP1= B(I,J)                                                      GIL60500
      B(I,J)=B(L,J)                                                      GIL60510
      B(L,J)=TEMP1                                                       GIL60520
      DO 130 K=IP1,N                                                     GIL60530
  130 B(K,J)=B(K,J) - A(K,I)*B(I,J)                                      GIL60540
  132 CONTINUE                                                           GIL60550
C*****INTERCHANGE ELEMENTS ONLY IN COLUMN J
      DO140 J=IP1,N                                                      GIL60560
      TEMP1 = A(I,J)                                                     GIL60570
      A(I,J)=A(L,J)                                                      GIL60580
      A(L,J)=TEMP1                                                       GIL60590
      DO135 K=IP1,N                                                      GIL60600
```

```
  135 A(K,J)=A(K,J)-A(K,I)*A(I,J)                                    GIL60610
  140 CONTINUE                                                       GIL60620
      IF (IDP.EQ.0) GO TO 200                                        GIL60630
C                                                                    GIL60640
C*****SCALE THE DETERMINANT                                          GIL60650
C                                                                    GIL60660
      PIVOTI=A(I,I)                                                  GIL60670
 1005 IF(ABS(DETERM)-R1)1030,1010,1010                               GIL60680
 1010 DETERM=DETERM/R1                                               GIL60690
      ISCALE=ISCALE+1                                                GIL60700
      IF(ABS(DETERM)-R1)1060,1020,1020                               GIL60710
 1020 DETERM=DETERM/R1                                               GIL60720
      ISCALE=ISCALE+1                                                GIL60730
      GO TO 1060                                                     GIL60740
 1030 IF(ABS(DETERM)-R2)1040,1040,1060                               GIL60750
 1040 DETERM=DETERM*R1                                               GIL60760
      ISCALE=ISCALE-1                                                GIL60770
      IF(ABS(DETERM)-R2)1050,1050,1060                               GIL60780
 1050 DETERM=DETERM*R1                                               GIL60790
      ISCALE=ISCALE-1                                                GIL60800
 1060 IF(ABS(PIVOTI)-R1)1090,1070,1070                               GIL60810
 1070 PIVOTI=PIVOTI/R1                                               GIL60820
      ISCALE=ISCALE+1                                                GIL60830
      IF(ABS(PIVOTI)-R1)320,1080,1080                                GIL60840
 1080 PIVOTI=PIVOTI/R1                                               GIL60850
      ISCALE=ISCALE+1                                                GIL60860
      GO TO 320                                                      GIL60870
 1090 IF(ABS(PIVOTI)-R2)2000,2000,320                                GIL60880
 2000 PIVOTI=PIVOTI*R1                                               GIL60890
      ISCALE=ISCALE-1                                                GIL60900
      IF(ABS(PIVOTI)-R2)2010,2010,320                                GIL60910
 2010 PIVOTI=PIVOTI*R1                                               GIL60920
      ISCALE=ISCALE-1                                                GIL60930
C*****OBTAIN PARTIAL PRODUCT FOR DETERMINANT
  320 DETERM=DETERM*PIVOTI                                           GIL60940
      IF (I.NE.NM1) GO TO 200                                        GIL60950
      IF (ID.EQ.1) GO TO 200                                         GIL60960
      PIVOTI=A(N,N)                                                  GIL60970
      ID=1                                                           GIL60980
      GO TO 1005                                                     GIL60990
200   CONTINUE                                                       GIL61000
C                                                                    GIL61010
C*****PERFORM BACK SUBSTITUTION                                      GIL61020
  340 CONTINUE                                                       GIL61030
      DO 450 KK=1,N                                                  GIL61040
      K=NP1-KK                                                       GIL61050
C*****SCALAR DIVIDE
      DO 430 J=1,NRHS                                                GIL61060
      B(K,J)=B(K,J)/A(K,K)                                           GIL61070
      IF (K.EQ.1) GO TO 430                                          GIL61080
      KM1=K-1                                                        GIL61090
C*****COLUMN K BY UNKNOWN AND SUBTRACT
      DO 420 I=1,KM1                                                 GIL61100
  420 B(I,J) = B(I,J) - A(I,K)* B(K,J)                               GIL61110
  430 CONTINUE                                                       GIL61120
  450 CONTINUE                                                       GIL61130
      DETERM=DETERM*SIGN                                             GIL61140
      RETURN                                                         GIL61150
  600 IF (A(1,1).EQ.0) GO TO 60                                      GIL61160
      DO 800  J=1,NRHS                                               GIL61170
  800 B(1,J)=B(1,J)/A(1,1)                                           GIL61180
      DETERM=A(1,1)                                                  GIL61190
      RETURN                                                         GIL61200
      END                                                            GIL61210
```

STAR Version in STAR FORTRAN

```
      SUBROUTINE GELIM(A,N,B,NRHS,MAXN,IPIVOT,IOP,DETERM,ISCALE)          GLIS0010
      DIMENSION A(MAXN,MAXN),B(MAXN,NRHS),IPIVOT(MAXN)                   GLIS0020
C     A(MAXN,MAXN) = SQUARE MATRIX OF COEFFICIENTS(A IS DESTROYED)       GLIS0030
C     N = NUMBER OF ROWS AND COLUMNS IN A                                GLIS0040
C     B(MAXN,NRHS) = MATRIX OF CONSTANTS(REPLACED BY SOLUTION MATRIX)    GLIS0050
C     NRHS = NUMBER OF COLUMNS IN B                                      GLIS0060
C     MAXN = MAXIMUM NUMBER OF ROWS AND COLUMNS IN A                     GLIS0070
C     IPIVOT(MAXN) = RECORD OF ROW INTERCHANGES                          GLIS0080
C     IOP - IOP=1,EVALUATE DETERMINANT. IOP=0,SKIP DETERMINANT EVALUATIOGLIS0090
C     DETERM - GIVES VALUE OF DETERMINAT(DETERM=(10**100)**ISCALE*DETERMGLIS0100
C     ISCALE = SCALE FACTOR COMPUTED BY SUBROUTINE IN DETERM EVALUATION GLIS0110
C              TO KEEP DETERM WITHIN THE FLOATING POINT WORD SIZE OF THEGLIS0120
C              COMPUTER                                                  GLIS0130
      ID=0                                                               GLIS0140
      SIGN=1.0                                                           GLIS0150
      DO 3 I=1,N                                                         GLIS0160
    3 IPIVOT(I)=I                                                        GLIS0170
    5 ISCALE=0                                                           GLIS0180
    6 R1=10.0**100                                                       GLIS0190
    7 R2=1.0/R1                                                          GLIS0200
   10 DETERM=1.0                                                         GLIS0210
      NM1=N-1                                                            GLIS0220
      IF (NM1.EQ.0) GO TO 600                                            GLIS0230
      NP1=N+1                                                            GLIS0240
C*****FIND LARGEST REMAINING TERM IN I-TH COLUMN FOR PIVOT               GLIS0250
      DO 200 I=1,NM1                                                     GLIS0260
C                                                                        GLIS0270
C  NEED VECTOR ABSOLUTE , MAXIMUM  BECAUSE YOU ARE SEARCHING COLUMN      GLIS0280
C                                                                        GLIS0290
      BIG=0.                                                             GLIS0300
      DO 50 K=I,N                                                        GLIS0310
      TERM=ABS(A(K,I))                                                   GLIS0320
      IF(TERM-BIG)50,50,30                                               GLIS0330
30    BIG=TERM                                                           GLIS0340
      L=K                                                                GLIS0350
50    CONTINUE                                                           GLIS0360
      IPIVOT(I)=L                                                        GLIS0370
      IF(BIG)80,60,80                                                    GLIS0380
60    DETERM=0.0                                                         GLIS0390
      RETURN                                                             GLIS0400
80    IF(I-L)90,120,90                                                   GLIS0410
90    SIGN=-SIGN                                                         GLIS0430
C*****INTERCHANGE ELEMENTS IN COLUMN I ONLY
      TEMP1= A(I,I)                                                      GLIS0440
      A(I,I)=A(L,I)                                                      GLIS0450
      A(L,I)=TEMP1                                                       GLIS0460
120   CONTINUE                                                           GLIS0470
C*****STORE PIVOT IN A(I,J). MULTIPLY A AND B BY PIVOT.                  GLIS0480
      IP1=I+1                                                            GLIS0490
C*****OBTAIN COLUMN OF PIVOTS
      A(IP1:N,I)= A(IP1:N,I)/A(I,I)                                      GLIS0500
C*****INTERCHANGE ELEMENTS ONLY IN COLUMN J
      DO131 J=1,NRHS                                                     GLIS0510
      TEMP1= B(I,J)                                                      GLIS0520
      B(I,J)=B(L,J)                                                      GLIS0530
      B(L,J)=TEMP1                                                       GLIS0540
  131 B(IP1:N,J) = B(IP1:N,J) -A(IP1:N,I) * B(I,J)                       GLIS0550
C*****INTERCHANGE ELEMENTS ONLY IN COLUMN J
      DO134 J=IP1,N                                                      GLIS0560
      TEMP1= A(I,J)                                                      GLIS0570
      A(I,J)=A(L,J)                                                      GLIS0580
      A(L,J)=TEMP1                                                       GLIS0590
```

```
  134 A(IP1:N,J)= A(IP1:N,J) -A(IP1:N,I)* A(I,J)                    GLIS0600
      IF (IOP.EQ.0) GO TO 200                                       GLIS0610
C                                                                   GLIS0620
C*****SCALE THE DETERMINANT                                         GLIS0630
C                                                                   GLIS0640
      PIVOTI=A(I,I)                                                 GLIS0650
 1005 IF(ABS(DETERM)-R1)1030,1010,1010                              GLIS0660
 1010 DETERM=DETERM/R1                                              GLIS0670
      ISCALE=ISCALE+1                                               GLIS0680
      IF(ABS(DETERM)-R1)1060,1020,1020                              GLIS0690
 1020 DETERM=DETERM/R1                                              GLIS0700
      ISCALE=ISCALE+1                                               GLIS0710
      GO TO 1060                                                    GLIS0720
 1030 IF(ABS(DETERM)-R2)1040,1040,1060                              GLIS0730
 1040 DETERM=DETERM*R1                                              GLIS0740
      ISCALE=ISCALE-1                                               GLIS0750
      IF(ABS(DETERM)-R2)1050,1050,1060                              GLIS0760
 1050 DETERM=DETERM*R1                                              GLIS0770
      ISCALE=ISCALE-1                                               GLIS0780
 1060 IF(ABS(PIVOTI)-R1)1090,1070,1070                              GLIS0790
 1070 PIVOTI=PIVOTI/R1                                              GLIS0800
      ISCALE=ISCALE+1                                               GLIS0810
      IF(ABS(PIVOTI)-R1)320,1080,1080                               GLIS0820
 1080 PIVOTI=PIVOTI/R1                                              GLIS0830
      ISCALE=ISCALE+1                                               GLIS0840
      GO TO 320                                                     GLIS0850
 1090 IF(ABS(PIVOTI)-R2)2000,2000,320                               GLIS0860
 2000 PIVOTI=PIVOTI*R1                                              GLIS0870
      ISCALE=ISCALE-1                                               GLIS0880
      IF(ABS(PIVOTI)-R2)2010,2010,320                               GLIS0890
 2010 PIVOTI=PIVOTI*R1                                              GLIS0900
      ISCALE=ISCALE-1                                               GLIS0910
C*****OBTAIN PARTIAL PRODUCT FOR DETERMINANT
  320 DETERM=DETERM*PIVOTI                                          GLIS0920
      IF (I.NE.NM1) GO TO 200                                       GLIS0930
      IF (ID.EQ.1) GO TO 200                                        GLIS0940
      PIVOTI=A(N,N)                                                 GLIS0950
      ID=1                                                          GLIS0960
      GO TO 1005                                                    GLIS0970
  200 CONTINUE                                                      GLIS0980
C                                                                   GLIS0990
C*****PERFORM BACK SUBSTITUTION                                     GLIS1000
  340 CONTINUE                                                      GLIS1010
      DO 450 KK=1,N                                                 GLIS1020
      K=NP1-KK                                                      GLIS1030
C*****SCALAR DIVIDE
      DO 430 J=1,NRHS                                               GLIS1040
      B(K,J)=B(K,J)/A(K,K)                                          GLIS1050
      IF (K.EQ.1) GO TO 430                                         GLIS1060
      KM1=K-1                                                       GLIS1070
C*****COLUMN K BY UNKNOWN AND SUBTRACT
      B(1:KM1,J)= B(1:KM1,J) -A(1:KM1,K)* B(K,J)                    GLIS1080
  430 CONTINUE                                                      GLIS1090
  450 CONTINUE                                                      GLIS1100
      DETERM=DETERM*SIGN                                            GLIS1110
      RETURN                                                        GLIS1120
  600 IF (A(1,1).EQ.0) GO TO 60                                     GLIS1130
      B(1,1:NRHS)= B(1,1:NRHS)/A(1,1)                               GLIS1140
      DETERM=A(1,1)                                                 GLIS1150
      RETURN                                                        GLIS1160
      END                                                           GLIS1170
```

APPENDIX B

FORTRAN CODING OF 6000 AND STAR VERSIONS OF ALGORITHM FOR NUMERICAL INTEGRATION EVALUATION

6000 Version in 6000 FORTRAN

```
      SUBROUTINE GLEGEN(SUM,ICODE,A,B,FX,FOFX,NFX,NQ,NPQ)                GLEGN  1
C*****************************************************************************GLEGN  2
C*                                                                           *GLEGN  3
C*    PURPOSE:          TO COMPUTE THE INTEGRALS, F(I) OF X * DX FROM      *GLEGN  4
C*                      A TO B USING THE GAUSS-LEGENDRE QUADRATURE         *GLEGN  5
C*                      FORMULA.                                           *GLEGN  6
C*                                                                         *GLEGN  7
C*        USE:          CALL GLEGEN (SUM,ICODE,A,B,FX,FOFX,NFX,NQ,NPQ)     *GLEGN  8
C*                                                                         *GLEGN  9
C* PARAMETERS:                                                             *GLEGN 10
C*               SUM    THE ARRAY FOR THE VALUE(S) OF THE INTEGRAL(S)      *GLEGN 11
C*                                                                         *GLEGN 12
C*             ICODE    AN INTEGER TEST CODE RETURNED BY THE ROUTINE       *GLEGN 13
C*                      = 0, NORMAL RETURN                                 *GLEGN 14
C*                      = 1, NFX NOT PROPERLY SPECIFIED                    *GLEGN 15
C*                      = 2, NPQ NOT PROPERLY SPECIFIED (NOT 3 OR 10)      *GLEGN 16
C*                      = 3, NQ NOT PROPERLY SPECIFIED                     *GLEGN 17
C*                                                                         *GLEGN 18
C*                      THIS PARAMETER SHOULD BE TESTED UPON RETURN        *GLEGN 19
C*                      BY THE CALLING PROGRAM                             *GLEGN 20
C*                                                                         *GLEGN 21
C*                A     LOWER LIMIT OF INTEGRATION                         *GLEGN 22
C*                                                                         *GLEGN 23
C*                B     UPPER LIMIT OF INTEGRATION                         *GLEGN 24
C*                                                                         *GLEGN 25
C*               FX     THE NAME OF A USER SUPPLIED SUBROUTINE WITH        *GLEGN 26
C*                      ARGUMENTS X AND FOFX TO EVALUATE THE FUNCTIONS     *GLEGN 27
C*                      IT MUST BE DECLARED AS EXTERNAL.                   *GLEGN 28
C*                                                                         *GLEGN 29
C*             FOFX     THE ARRAY TO STORE THE VALUES OF THE FUNCTIONS     *GLEGN 30
C*                                                                         *GLEGN 31
C*              NFX     AN INTEGER, THE NO. OF FUNCTIONS TO INTEGRATE      *GLEGN 32
C*                                                                         *GLEGN 33
C*               NQ     AN INTEGER, THE NUMBER OF QUADRATURES              *GLEGN 34
C*                                                                         *GLEGN 35
C*              NPQ     AN INTEGER, THE NO. OF POINTS PER QUADRATURE       *GLEGN 36
C*                      IT MUST BE 3 OR 10                                 *GLEGN 37
C*                                                                         *GLEGN 38
C* REQUIRED ROUTINES:   NONE                                               *GLEGN 39
C*                                                                         *GLEGN 40
C* SOURCE/IMPLEMENTER:  COMPUTER SCIENCES CORP./ G. W. HAIGLER             *GLEGN 41
C*                                                                         *GLEGN 42
C* DATE RELEASED:       NOV. 14,1972                                       *GLEGN 43
C*                                                                         *GLEGN 44
C* LATEST REVISION:     NOV. 15,1972                                       *GLEGN 45
```

```
C*********************************************************************** GLEGN 46
C                                                                       GLEGN 47
      DIMENSION U1(3),U2(10),R1(3),R2(10),U(13),R(13),SUM(1),FOFX(1)    GLEGN 48
C                                                                       GLEGN 49
      EQUIVALENCE (U1(1),U(1)),(U2(1),U(4)),(R1(1),R(1)),(R2(1),R(4))   GLEGN 50
C                                                                       GLEGN 51
      DATA U1/.112701665379259,.5,.887298334620742/,U2/.013046735741414,GLEGN 52
     1.067468316655507,.160295215850488,.283302302935376,.425562830509186LEGN 53
     25,.574437169490816,.716697697064624,.839704784149512,.932531683344GLEGN 54
     3493,.986953264258586/,R1/.277777777777778,.444444444444444,.277777GLEGN 55
     4777777778/,R2/.033335672154344,.074725674575291,.109543181257991, GLEGN 56
     5.134633359654998,.147762112357377,.147762112357377,.134633359654999GLEGN 57
     68,.109543181257991,.074725674575291,.033335672154344/             GLEGN 58
C                                                                       GLEGN 59
      ICODE = 0                                                         GLEGN 60
      IF(NFX.LE.0)ICODE = 1                                             GLEGN 61
      IF(NQ.LE.0)ICODE = 3                                              GLEGN 62
      IF(NPQ.EQ.3)GO TO 5                                               GLEGN 63
      IF(NPQ.NE.10)ICODE = 2                                            GLEGN 64
    5 IF(ICODE.GT.0)RETURN                                              GLEGN 65
      J=3                                                               GLEGN 66
      IF  (NPQ  .EQ.  3)  J=0                                           GLEGN 67
      DELT= (B-A)/FLOAT(NQ)                                             GLEGN 68
      DO 10 I = 1,NFX                                                   GLEGN 69
   10 SUM(I) = 0.0                                                      GLEGN 70
      DO 80 K=1,NQ                                                      GLEGN 71
      XI = K - 1                                                        GLEGN 72
      FF = A + XI*DELT                                                  GLEGN 73
      DO 80 L=1,NPQ                                                     GLEGN 74
      UU=U(J+L)*DELT+FF                                                 GLEGN 75
      CALL FX(UU,FOFX)                                                  GLEGN 76
      FACT=R(J+L)                                                       GLEGN 77
      DO 80 J8=1,NFX                                                    GLEGN 78
   80 SUM(J8) =FOFX(J8)*FACT+SUM(J8)                                    GLEGN 79
      DO 100 II=1,NFX                                                   GLEGN 80
  100 SUM(II) = SUM(II)*DELT                                            GLEGN 81
      RETURN                                                            GLEGN 82
      END                                                               GLEGN 83
```

STAR Version in 6000 FORTRAN

```
      SUBROUTINE GLEGENS(SUM,ICODE,A,B,FX,FOFX,NFX,NQ,NPQ,UU,FF,UD,MAXI)GLENS001
C*********************************************************************** GLENS002
C*                                                                     * GLENS003
C*     PURPOSE:          TO COMPUTE THE INTEGRALS, F(I) OF X * DX FROM * GLENS004
C*                       A TO B USING THE GAUSS-LEGENDRE QUADRATURE    * GLENS005
C*                       FORMULA.                                      * GLENS006
C*                                                                     * GLENS007
C*          USE:         CALL GLEGEN (SUM,ICODE,A,B,FX,FOFX,NFX,NQ,NPQ)* GLENS008
C*                                                                     * GLENS009
C* PARAMETERS:                                                         * GLENS010
C*               SUM     THE ARRAY FOR THE VALUE(S) OF THE INTEGRAL(S) * GLENS011
C*                                                                     * GLENS012
C*             ICODE     AN INTEGER TEST CODE RETURNED BY THE ROUTINE  * GLENS013
C*                       = 0, NORMAL RETURN                            * GLENS014
C*                       = 1, NFX NOT PROPERLY SPECIFIED               * GLENS015
C*                       = 2, NPQ NOT PROPERLY SPECIFIED (NOT 3 OR 10) * GLENS016
C*                       = 3, NQ NOT PROPERLY SPECIFIED                * GLENS017
C*                                                                     * GLENS018
C*                       THIS PARAMETER SHOULD BE TESTED UPON RETURN   * GLENS019
C*                       BY THE CALLING PROGRAM                        * GLENS020
C*                                                                     * GLENS021
```

```
C*                  A     LOWER LIMIT OF INTEGRATION                        * GLENS022
C*                                                                          * GLENS023
C*                  B     UPPER LIMIT OF INTEGRATION                        * GLENS024
C*                                                                          * GLENS025
C*                 FX     THE NAME OF A USER SUPPLIED SUBROUTINE WITH       * GLENS026
C*                        ARGUMENTS X AND FOFX TO EVALUATE THE FUNCTIONS    * GLENS027
C*                        IT MUST BE DECLARED AS EXTERNAL.                  * GLENS028
C*                                                                          * GLENS029
C                 FOFX    A TWO DIMENSIONED ARRAY TO STORE VALUES OF THE      GLENS030
C                         FUNCTIONS, FOFX(MAXI,NFX) IN CALLING PROGRAM        GLENS031
C*                                                                          * GLENS032
C*                 NFX     AN INTEGER, THE NO. OF FUNCTIONS TO INTEGRATE     * GLENS033
C*                                                                          * GLENS034
C*                  NQ     AN INTEGER, THE NUMBER OF QUADRATURES             * GLENS035
C*                                                                          * GLENS036
C*                 NPQ     AN INTEGER, THE NO. OF POINTS PER QUADRATURE      * GLENS037
C*                         IT MUST BE 3 OR 10                                * GLENS038
C*                                                                          * GLENS039
C*                  UU     ARRAY TO STORE ALL POINTS AT WHICH FUNCTION IS     GLENS040
C*                         EVALUATED ,ALSO STORES WEIGHTS                      GLENS041
C*                  FF     ARRAY OF WORKING SPACE, CONTAINS LOWER LIMIT OF    GLENS042
C*                          INTEGRATION FOR EACH QUADRATURE                    GLENS043
C*                  UD     ARRAY OF WORKING SPACE, ABSCISSAS* DELT            GLENS044
C                 MAXI     MAXIMUM ROW DIMENSIONS OF FOFX IN CALLING PROG.    GLENS045
C                                                                             GLE6S046
C*                                                                          * GLE6S047
C* REQUIRED ROUTINES:    NONE                                              * GLE6S048
C*                                                                          * GLE6S049
C* SOURCE/IMPLEMENTER:   COMPUTER SCIENCES CORP./ G. W. HAIGLER            * GLE6S050
C*                                                                          * GLE6S051
C* DATE RELEASED:        NOV. 14,1972                                      * GLE6S052
C*                                                                          * GLE6S053
C* LATEST REVISION:      NOV. 15,1972                                      * GLE6S054
C*************************************************************************** GLE6S055
C                                                                             GLE6S056
      DIMENSION U1(3),U2(10),R1(3),R2(10),U(13),R(13),SUM(1),                 GLE6S057
     1  FOFX(MAXI,1  ),UU(1  ),UD(1  ),FF(1  )                                GLE6S058
C                                                                             GLE6S059
      EQUIVALENCE (U1(1),U(1)),(U2(1),U(4)),(R1(1),R(1)),(R2(1),R(4))         GLE6S060
C                                                                             GLE6S061
      DATA U1/.112701665379259,.5,.887298334620742/,U2/.013046735741414,GLE6S062
     1.067468316655507,.160295215850488,.283302302935376,.425562830509186LE6S063
     25,.574437169490816,.716697697064624,.839704784149512,.932531683344GLE6S064
     3493,.986953264258586/,R1/.277777777777778,.444444444444444,.277777GLE6S065
     4777777778/,R2/.033335672154344,.074725674575291,.109543181257991, GLE6S066
     5.134633359654998,.147762112357377,.147762112357377,.134633359654990LE6S067
     68,.109543181257991,.074725674575291,.033335672154344/                 GLE6S068
C                                                                             GLE6S069
      ICODE = 0                                                               GLE6S070
C  NT = TOTAL  NUMBER OF POINTS AT WHICH FUNCTION WILL BE EVALUATED           GLE6S071
      NT=NQ*NPQ                                                               GLE6S072
      IF(NFX.LE.0)ICODE = 1                                                   GLE6S073
      IF(NQ.LE.0)ICODE = 3                                                    GLE6S074
      IF(NPQ.EQ.3)GO TO 5                                                     GLE6S075
      IF(NPQ.NE.10)ICODE = 2                                                  GLE6S076
    5 IF(ICODE.GT.0)RETURN                                                    GLE6S077
      J=3                                                                     GLE6S078
      IF  (NPQ  .EQ.  3)  J=0                                                 GLE6S079
      DELT= (B-A)/FLOAT(NQ)                                                   GLE6S080
C*****COMPUTE ARRAY TO BE ADDED TO FIRST POINT IN EACH QUADRATURE
      DO 20 K=1,NPQ                                                           GLE6S081
      J1 =J+K                                                                 GLE6S082
   20 UD(K)= U(J1)*DELT                                                       GLE6S083
```

```
C*****COMPUTE FIRST POINT IN EACH QUADRATURE
      DO 30 K=1,NQ                                                    GLE6S084
      XI =K-1                                                         GLE6S085
   30 FF(K) =A + XI*DELT                                              GLE6S086
C  CREATE VECTOR OF ALL POINTS FOR ALL QUADRATURES                    GLE6S087
      DO 40 K=1,NQ                                                    GLE6S088
      K1 =(K-1)*NPQ                                                   GLE6S089
      DO 35 I=1,NPQ                                                   GLE6S090
   35 UU(K1+I) = FF(K)+ UD(I)                                         GLE6S091
   40 CONTINUE                                                        GLE6S092
C  EVALUATE FUNCTION AT ALL POINTS                                    GLE6S093
C  NOTE CHANGE IN CALLING SEQUENCE FOR FUNCTION EVALUATION SUBROUTINE GLE6S094
      CALL FXS(UU,FOFX,MAXI,NT)                                       GLE6S095
C  CREATE VECTOR OF WEIGHTS                                           GLE6S096
      DO 50 K=1,NQ                                                    GLE6S097
      K1 =(K-1)*NPQ                                                   GLE6S098
      DO 45 I= 1,NPQ                                                  GLE6S099
   45 UU(K1+I) = R(J+I)                                               GLE6S100
   50 CONTINUE                                                        GLE6S101
      DO 80 I=1,NFX                                                   GLE6S102
      SUM(I)=0.0                                                      GLE6S103
      DO 70 K=1,NT                                                    GLE6S104
   70 SUM(I)= SUM(I) +UU(K)* FOFX(K,I)                                GLE6S105
      SUM(I) = DELT * SUM(I)                                          GLE6S106
   80 CONTINUE                                                        GLE6S107
      RETURN                                                          GLE6S108
      END                                                             GLE6S109
```

STAR Version in STAR FORTRAN

```
      SUBROUTINE GLEGENS(SUM,ICODE,A,B,FX,FOFX,NFX,NQ,NPQ,UU,FF,UD,MAXI)GLE6S001
C*********************************************************************** GLE6S002
C*                                                                     * GLE6S003
C*    PURPOSE:          TO COMPUTE THE INTEGRALS, F(I) OF X * DX FROM  * GLE6S004
C*                      A TO B USING THE GAUSS-LEGENDRE QUADRATURE     * GLE6S005
C*                      FORMULA.                                       * GLE6S006
C*                                                                     * GLE6S007
C*        USE:          CALL GLEGEN (SUM,ICODE,A,B,FX,FOFX,NFX,NQ,NPQ) * GLE6S008
C*                                                                     * GLE6S009
C* PARAMETERS:                                                         * GLE6S010
C*               SUM    THE ARRAY FOR THE VALUE(S) OF THE INTEGRAL(S)  * GLE6S011
C*                                                                     * GLE6S012
C*             ICODE    AN INTEGER TEST CODE RETURNED BY THE ROUTINE   * GLE6S013
C*                      = 0, NORMAL RETURN                             * GLE6S014
C*                      = 1, NFX NOT PROPERLY SPECIFIED                * GLE6S015
C*                      = 2, NPQ NOT PROPERLY SPECIFIED (NOT 3 OR 10)  * GLE6S016
C*                      = 3, NQ NOT PROPERLY SPECIFIED                 * GLE6S017
C*                                                                     * GLE6S018
C*                      THIS PARAMETER SHOULD BE TESTED UPON RETURN    * GLE6S019
C*                      BY THE CALLING PROGRAM                         * GLE6S020
C*                                                                     * GLE6S021
C*                A     LOWER LIMIT OF INTEGRATION                     * GLE6S022
C*                                                                     * GLE6S023
C*                B     UPPER LIMIT OF INTEGRATION                     * GLE6S024
C*                                                                     * GLE6S025
C*               FX     THE NAME OF A USER SUPPLIED SUBROUTINE WITH    * GLE6S026
C*                      ARGUMENTS X AND FOFX TO EVALUATE THE FUNCTIONS* GLE6S027
C*                      IT MUST BE DECLARED AS EXTERNAL.               * GLE6S028
C*                                                                     * GLE6S029
C              FOFX     A TWO DIMENSIONED ARRAY TO STORE VALUES OF THE  GLE6S030
C                       FUNCTIONS, FOFX(MAXI,NFX) IN CALLING PROGRAM    GLE6S031
```

```
C*                                                                    * GLE6S032
C*               NFX     AN INTEGER, THE NO. OF FUNCTIONS TO INTEGRATE * GLE6S033
C*                                                                    * GLE6S034
C*                NQ     AN INTEGER, THE NUMBER OF QUADRATURES        * GLE6S035
C*                                                                    * GLE6S036
C*               NPQ     AN INTEGER, THE NO. OF POINTS PER QUADRATURE * GLE6S037
C*                       IT MUST BE 3 OR 10                           * GLE6S038
C*                UU     ARRAY TO STORE ALL POINTS AT WHICH FUNCTION IS GLE6S039
C*                       EVALUATED ,ALSO STORES WEIGHTS               GLE6S040
C*                FF     ARRAY OF WORKING SPACE, CONTAINS LOWER LIMIT OF GLE6S041
C*                        INTEGRATION FOR EACH QUADRATURE             GLE6S042
C*                UD     ARRAY OF WORKING SPACE, ABSCISSAS* DELT      GLE6S043
C*                                                                    GLE6S044
C*             MAXI      MAXIMUM ROW DIMENSIONS OF FOFX IN CALLING PROG. GLE6S045
C*                                                                    GLENS046
C* REQUIRED ROUTINES:    NONE                                         * GLENS047
C*                                                                    * GLENS048
C* SOURCE/IMPLEMENTER:   COMPUTER SCIENCES CORP./ G. W. HAIGLER       * GLENS049
C*                                                                    * GLENS050
C* DATE RELEASED:        NOV. 14,1972                                 * GLENS051
C*                                                                    * GLENS052
C* LATEST REVISION:      NOV. 15,1972                                 * GLENS053
C********************************************************************** GLENS054
C                                                                       GLENS055
      DIMENSION U1(3),U2(10),R1(3),R2(10),U(13),R(13),SUM(1),           GLENS056
     1  FOFX(MAXI,1  ),UU(1 ),UD(1  ),FF(1  )                          GLENS057
C                                                                       GLENS058
      EQUIVALENCE (U1(1),U(1)),(U2(1),U(4)),(R1(1),R(1)),(R2(1),R(4))   GLENS059
C                                                                       GLENS060
      DATA U1/.112701665379259,.5,.887298334620742/,U2/.0130467357414,GLENS061
     1.067468316655507,.160295215850488,.283302302935376,.4255628305091 8GLENS062
     25,.574437169490816,.716697697064624,.839704784149512,.932531683344GLENS063
     3493,.986953264258586/,R1/.277777777777778,.444444444444444,.277777GLENS064
     4777777778/,R2/.033335672154344,.074725674575291,.109543181257991, GLENS065
     5.134633359654998,.147762112357377,.147762112357377,.1346333596549 9GLENS066
     68,.109543181257991,.074725674575291,.033335672154344/             GLENS067
C                                                                       GLENS068
      ICODE = 0                                                         GLENS069
C  NT = TOTAL  NUMBER OF POINTS AT WHICH FUNCTION WILL BE EVALUATED     GLENS070
      NT=NQ*NPQ                                                         GLENS071
      IF(NFX.LE.0)ICODE = 1                                             GLENS072
      IF(NQ.LE.0)ICODE = 3                                              GLENS073
      IF(NPQ.EQ.3)GO TO 5                                               GLENS074
      IF(NPQ.NE.10)ICODE = 2                                            GLENS075
    5 IF(ICODE.GT.0)RETURN                                              GLENS076
      J=3                                                               GLENS077
      IF  (NPQ  .EQ.  3)  J=0                                           GLENS078
      DELT= (B-A)/FLOAT(NQ)                                             GLENS079
      J1 = J+1                                                          GLENS080
      NJ= J +NPQ                                                        GLENS081
C*****COMPUTE ARRAY TO BE ADDED TO FIRST POINT IN EACH QUADRATURE      GLENS082
      UD(1:NPQ) = U(J1:NJ) * DELT                                       GLENS083
C*****COMPUTE FIRST POINT IN EACH QUADRATURE                           GLENS084
      DO 30 K=1,NQ                                                      GLENS085
      XI =K-1                                                           GLENS086
   30 FF(K) =A + XI*DELT                                                GLENS087
C*****COMPUTE ARRAY OF ALL POINTS FOR ALL QUADRATURES                  GLENS088
      DO 40 K=1,NQ                                                      GLENS089
      K1 =(K-1)*NPQ                                                     GLENS090
      NK= K  + NPQ                                                      GLENS091
      UU(K1+1:NK) = FF(K) + UD(1:NPQ)                                   GLENS092
   40 CONTINUE                                                          GLENS093
C  EVALUATE FUNCTION AT ALL POINTS                                      GLENS094
```

```
C  NOTE CHANGE IN CALLING SEQUENCE FOR FUNCTION EVALUATION SUBROUTINE   GLENS095
      CALL FXS(UU,FOFX,MAXI,NT)                                        GLENS096
C  CREATE VECTOR OF WEIGHTS                                            GLENS097
      DO 50 K=1,NQ                                                     GLENS098
      K1 =(K-1)*NPQ                                                    GLENS099
      NK = K+ NPQ                                                      GLENS100
      UU(K1+1:NK) = R(J1:NJ)                                           GLENS101
   50 CONTINUE                                                         GLENS102
C  THESE STATEMENTS CAN BE REPLACED BY DOT PRODUCT FUNCTION IF AVAILABLEGLENS103
C   SUCH AS                                                            GLENS104
C     SUM(I) = DOTP (UU(1:NT) ,FOFX(1:NT, I)                          GLENS105
C                                                                      GLENS106
C **                                                                   GLENS107
      DO 80 I=1,NFX                                                    GLENS108
      SUM(I)=0.0                                                       GLENS109
      DO 70 K=1,NT                                                     GLENS110
   70 SUM(I)= SUM(I) +UU(K)* FOFX(K,I)                                 GLENS111
      SUM(I) = DELT * SUM(I)                                           GLENS112
   80 CONTINUE                                                         GLENS113
C **                                                                   GLENS114
      RETURN                                                           GLENS115
      END                                                              GLENS116
```

REFERENCES

1. Anon.: Control Data STAR Computer System FORTRAN Reference Manual. 60384500 A, Control Data Corp., c.1973.

2. Anon.: Control Data STAR-100 Computer System Hardware Reference Manual. 60256000 06, Control Data Corp., c.1973.

3. Denning, Peter J.: Virtual Memory. Comput. Surv., vol. 2, no. 3, Sept. 1970, pp. 153-189.

4. Lambiotte, Jules J., Jr.; and Howser, Lona M.: Vectorization on the STAR Computer of Several Numerical Methods for a Fluid Flow Problem. NASA TN D-7545, 1974.

5. Hildebrand, F. B.: Introduction to Numerical Analysis. McGraw-Hill Book Co., Inc., 1956.