NASA Contractor Report 165730

STAR ADAPTATION OF QR ALGORITHM

Shantilal N. Shah

HAMPTON INSTITUTE
Hampton, Virginia 23668

June 1981

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

SUMMARY

The QR algorithm used to solve over-determined systems of linear equations was adapted to execute efficiently on the Control Data STAR-100 computer. Using the new vectorized algorithm, the STAR-100 computer solved a system of 250 equations in 50 unknowns in less than 8.5% of the time it took using the original scalar version. This paper describes how the scalar program was adapted for the STAR-100 and indicates why these adaptations yielded an efficient STAR program. Program listings of the old scalar version and of the new vectorized SL/1 version are presented in the appendices. Execution times for the two versions, applied to the same system of linear equations, are compared.

INTRODUCTION

Programs written in standard FORTRAN language for serial computers do run on the Control Data STAR-100 computer, but very inefficiently. To take advanatage of the architecture and vector-processing capabilities of the STAR-100 computer it is necessary to vectorize the algorithms in these programs. Frequently one must rearrange the data and computations. This paper describes how the QR algorithm to solve over-determined systems of linear equations was vectorized and what factors were considered in developing an efficient STAR program.

The vectorized program utilizes SL/1, a high level language developed by NASA's Langley Research Center for the STAR-100 computer. SL/1 incorporates many features designed to see that programs it compiles take full advantage of the STAR's architecture and capabilities, including half-word storage and arithmetic. SL/1 is compatible with FORTRAN in the sense that programs written in either language can call subroutines written in the other. In utilizing the program presented in this paper, familarity with some of the notations used in the SL/1 language will be helpful.

General suggestions concerning the adaptations of algorithms for efficient use on the STAR may be found in the paper, "Star Adaptation for Two Algorithms used on Serial Computer," by Lona M. Howser and Jules J. Lambiotte (see ref. 1).

ADAPTATION OF QR ALGORITHM TO SOLVE OVER-DETERMINED SYSTEMS OF LINEAR EQUATIONS

The NASA computer mathematics library presently has a subroutine called QRASOS, written in FORTRAN for serial computers, to solve an over-determined system of linear equations. This subroutine decomposes the matrix A of the system AX=B using Householder transformations. (For details of this algorithm see ref. 2). To compute these transformations, it uses subroutines SAXPY, SSCAL, SCOPY and function SDOT from the Basic Linear Algebra Subroutines (BLAS). For an mxn ($m \geq n$) matrix A it makes n^2 calls to these subroutines and functions to solve the given system. Subroutine calls are very expensive on the STAR-100 computer.

In SL/1, a matrix can be stored either column-wise or row-wise. Column storage means that elements in one column of the matrix are stored as one vector (contiguous locations). Similarly, row-storage means that elements in one row are stored as one vector (contiguous locations). In vectorizing this algorithm both row and column storage of matrices A and B were tried.

With row-wise storage, reordering of the scalar-version computations is required but use of the inner product macro to decompose matrix A is avoided. With column-wise storage the computational steps are the same as in the scalar versions with vector instructions replacing scalar instructions, but use of the dot product macro is required. It was expected that, because of the avoidance of the dot product macro, the row-wise approach would offer a considerable saving in CPU time.

Test results show that in using the STAR computer for this algorithm both vectorized versions offer considerable CPU time savings over the scalar program, but that contrary to expectations column-wise storage is more efficient than row-wise storage (see table).

	CPU TIME IN SECONDS TO DECOMPOSE		CPU TIME IN SECONDS TO DECOMPOSE AND SOLVE	
Size of Matrix A	New Vectorized Version (Column Storage)	New Vectorized Version (Row Storage)	Old Scalar Version	New Vectorized Version (Column Storage)
250 x 200	2.169	2.695	26.34	2.239
120 x 100	.405	.588	3.396	.433
250 x 50	.158	.863	2.223	.178
100 x 10	.006	.069	0.056	.009
200 x 30	.053	.416	0.698	.063

Algorithms for both row-wise storage and column storage to decompose the matrix A are given. Back substitutions are not discussed here. In both algorithms, A is an mxn matrix and WK is a vector of length n.

COLUMN STORAGE

When matrix A is stored column-wise, the decomposition of A is achieved as follows: (Note: All references to i^{th} column refer to column entries on and below the diagonal).

(1) Take the inner product of i^{th} column with itself and store in WK_i

(2) Take the square root of WK_i

(3) If $WK_i = 0$, go to step (10)

(4) $WK_i = WK_i$ x Sign of $A_{i,i}$

(5) Divide column i by WK_i

(6) Add 1 to $A_{i,i}$

These 6 steps compute the Householder transformation for column i.

To apply this transformation to the columns K = i+1,....n do the following steps:

(7) Take the inner product of column i with column K and store the result in t

(8) Divide t by $A_{i,i}$ and then store the negative of the result in t

(9) Multiply column i by t and add the result to column K

(10) Store $A_{i,i}$ in t, $-WK_i$ in $A_{i,i}$ and t in WK_i

When i=n perform steps 1 thru 6 and 10.

ROW STORAGE

When the matrix A is stored row-wise, the decomposition of A is achieved as follows: (Note: In the steps 1 thru 6 below, all references to the row j in the i^{th} step of decomposition refer to the entries on and to the right of the diagonal. All references to the vector WK refer to its i^{th}, $(i+1)^{th}$ ....n^{th} elements. In steps 7 thru 10 all references to the row j in the i^{th} step of decomposition refer to the entries to the right of the diagonal and all references to the vector WK refer to its $(i+1)^{th}$, $(i+2)^{th}$....n^{th} elements).

At i^{th} step of decomposition (i=1,2,....n-1).

(1) Set WK=0

(2) For j=1,2....m, multiply row j by $A_{j,i}$ and add the result to WK

(3) Take the square root of WK_i

(4) If WK_i=0 go to step 11

(5) Multiply WK_i by sign of $A_{i,i}$

(6) Divide $A_{i,i}$ by WK_i and add 1 to the result

(7) Divide WK by WK_i and add row i of A to WK

(8) Divide WK by $-A_{i,i}$

(9) For j=i+1, i+2,....m
Divide $A_{j,i}$ by WK_i

(10) For j=i, i+1,....m, multiply WK by $A_{j,i}$ and add the result to row j of A

(11) Store $A_{i,i}$ in t, -WK in $A_{i,i}$ and t in WK_i

When i=n, perform Steps 1 thru 6 and 11.

WHY ROW-STORAGE IS SLOWER THAN COLUMN-STORAGE

As pointed out earlier, if the matrix A is stored row-wise, the use of the inner product macro is avoided and the computation of the Householder transformations and their application to other columns at each step of the decomposition is accomplished by the use of a vector multiplication by a scalar and then a vector addition. This should result in a considerable savings of the CPU time for a large matrx. However, our numerical experiments show just the opposite. This can be explained as follows: When an mxn (m>=n) matrix A is stored row-wise, the vector lengths in that algorithm are proportional to n, the smaller dimension. On the other hand, for column-wise storage the vector lengths are proportional to m, the larger dimension. Equivalently, we see that the row-stored algorithm requires more vector start-ups ((m-n)(m-n+1)/2 more) to do the same number of total computations as the column-stored algorithm, thus requiring more CPU time to do the same amount of work.

Another factor which makes the row-stored algorithm slower is that the transformation elements are stored in the columns of the decomposed matrix. If the matrix is stored row-wise, this leads to additional scalar computations, notably in step 9 of the algorithm. This slows down the computations considerably. Also, if m is large, then not all m vectors in row-wise storage reside in the memory at the same time. Because of need to reference different columns at different steps of algorithm, this could lead to excessive paging. Thus, any advantage gained by avoiding the use of the inner product in the row-wise storage is offset by the need to perform many scalar operations, more iterations and excessive paging.

REFERENCES

Lona M. Howser and Jules J. Lambiotte, Jr., "STAR Adaptation for Two Algorithms Used on Serial Computer," NASA TM X-3003. 1974

J. H. Wilkinson and Reinsch, Linear Algebra, Springer-Verlag, Berlin, 1971 .

APPENDIX A

SL/1 Coding of QR Algorithm

```
MODULE M1     ;               $OPT=1;$SOURCE(1,72);
/***************************************************************************** */
/*                                                                            */
/*PURPOSE                                                                     */
/*        TO SOLVE M SIMULTANEOUS EQUATIONS IN N UNKNOWNS WITH IP             */
/*        RIGHT HAND SIDES SO THAT THE SOLUTIONS ARE THE BEST POSSIBLE*/
/*        FIT IN THE LEAST SQUARES SENSE.  THE ROUTINE USES HOUSE-            */
/*        HOLDER TRANSFORMATIONS TO PERFORM THE QR DECOMPOSITION              */
/*        OF THE COEFFICIENT MATRIX.                                          */
/*                                                                            */
/*USE                                                                         */
/*                                                                            */
/*   CALL Q4QRASOS(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR)             */
/*                                                                            */
/*PARAMETERS                                                                  */
/*                                                                            */
/*      MAXM  AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE          */
/*            A,B, AND RSD ARRAYS IN THE CALLING PROGRAM.  MAXM MUST          */
/*            BE GREATER THAN OR EQUAL TO M.                                  */
/*                                                                            */
/*      MAXN  AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE          */
/*            X ARRAY IN THE CALLING PROGRAM.  MAXN MUST BE GREATER           */
/*            THAN OR EQUAL TO N.                                             */
/*                                                                            */
/*         M  AN INPUT INTEGER SPECIFYING THE NUMBER OF ROWS OF THE           */
/*            A AND B ARRAYS.  M MUST BE GREATER THAN OR EQUAL TO N.          */
/*                                                                            */
/*         N  AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF            */
/*            THE A ARRAY.                                                    */
/*                                                                            */
/*        IP  AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF            */
/*            THE B ARRAY.                                                    */
/*                                                                            */
/*         A  AN INPUT/OUTPUT TWO-DIMENSIONAL ARRAY WITH FIRST DIMEN-         */
/*            SION EQUAL TO MAXM AND SECOND DIMENSION AT LEAST N.             */
/*            ON INPUT, A MUST CONTAIN THE MATRIX OF COEFFICIENTS OF          */
/*            THE SYSTEM OF EQUATIONS.  ON OUTPUT, A CONTAINS INFOR-          */
/*            MATION DESCRIBING THE QR DECOMPOSITION OF A.                    */
/*                                                                            */
/*         B  AN INPUT TWO-DIMENSIONAL ARRAY WITH FIRST DIMENSION             */
/*            EQUAL TO MAXM AND SECOND DIMENSION AT LEAST IP.                 */
/*            THE COLUMNS OF B MUST CONTAIN THE IP RIGHT HAND SIDE            */
/*            VECTORS.                                                        */
/*                                                                            */
/*        WT  AN INPUT ONE-DIMENSIONAL ARRAY OF WEIGHTS.  IT MUST             */
/*            HAVE LENGTH AT LEAST M.  IF WEIGHTING IS DESIRED,               */
/*            THE FIRST M LOCATIONS MUST CONTAIN REAL NUMBERS GREATER */
/*            THAN ZERO.  IF WEIGHTING IS NOT DESIRED, WT [1] MUST BE */
/*            A NEGATIVE REAL NUMBER.                                         */
/*                                                                            */
/*       JOB  AN INPUT INTEGER SPECIFYING RESULTS TO BE COMPUTED.             */
/*                                                                            */
/*                =1  COMPUTE SOLUTIONS ONLY.                                 */
/*                =2  COMPUTE RESIDUALS ONLY.                                 */
/*                =3  COMPUTE BOTH SOLUTIONS AND RESIDUALS.                   */
/*                                                                            */
/*         X  AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE SOLU-            */
/*            TIONS.   X MUST BE DIMENSIONED WITH FIRST DIMENSION             */
/*            EQUAL TO MAXN AND SECOND DIMENSION AT LEAST IP. IF              */
/*            SOLUTIONS ARE DESIRED INTO  MATRIX B THEN MAXN MUST BE          */
/*            EQUAL TO MAXM FOR THIS PARTICULAR CASE.                         */
```

```
/*        RSD  AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE RESID-    */
/*             UALS.   RSD MUST BE DIMENSIONED WITH FIRST DIMENSION     */
/*             EQUAL TO MAXM AND SECOND DIMENSION AT LEAST IP.          */
/*                                                                      */
/*        SUM  AN OUTPUT ONE-DIMENSIONAL ARRAY CONTAINING THE WEIGHTED  */
/*             SUMS OF SQUARES OF THE RESIDUALS.  SUM MUST BE DIMEN-    */
/*             SIONED AT LEAST IP.                                      */
/*                                                                      */
/*         WK  A ONE-DIMENSIONAL WORK ARRAY WHICH MUST BE DIMENSIONED   */
/*             AT LEAST N.  ON OUTPUT, WK CONTAINS INFORMATION ON THE   */
/*             QR DECOMPOSITION OF A.                                   */
/*                                                                      */
/*       IERR  AN INTEGER ERROR CODE.                                   */
/*                                                                      */
/*                =0  NO ERROR DETECTED.                                */
/*                =1  N IS GREATER THAN M.                              */
/*                =2  THE DECOMPOSED MATRIX IS SINGULAR.                */
/*                =3  WEIGHTING WAS REQUESTED AND ONE OR MORE WEIGHTS   */
/*                    IS NEGATIVE.                                      */
/*                                                                      */
/*                                                                      */
/*       SOURCE      HAMPTON INSTITUTE, HAMPTON VA.                     */
/*                                                                      */
/*       LANGUAGE    SL/1.                                              */
/*                                                                      */
/*     DATE RELEASED    JANUARY 18,1980.                                */
/*                                                                      */
/*                                                                      */
/************************************************************************/
   ENTRY PROCEDURE Q4QRASOS (MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,
                             SUM,WK,IERR);
                        REAL VECTOR [MAXM] ARRAY(N) A;
                        REAL VECTOR  [MAXN] ARRAY(IP) X;
                        REAL VECTOR  [MAXM] ARRAY(IP) B,RSD;
                        REAL VECTOR  [M] WT;
                        REAL VECTOR  [N] WK;
                        REAL VECTOR  [IP] SUM;
                        AUTOMATIC REAL T;
                        INTEGER I,J,K,L,M,N,IP,MAXM,IERR,MAXN,JOB;
/*                                                                      */
/*        CHECK FOR M LESS THAN N.                                      */
/*                                                                      */
          IF M < N THEN  IERR:= 1,
                         GO  TO LAB1
          ELSE
/*                                                                      */
/*           CHECK FOR WEIGHTING                                        */
/*                                                                      */
               IF WT[1] >=0 THEN
/*                                                                      */
/*           CHECK FOR ILLEGAL WEIGHTS                                  */
/*                                                                      */
                    I:= SELLT(WT,0);
                    IF I < M THEN IERR:=3;
                           GO TO LAB1
                    ELSE
                       WT[1:M]:= SQRT(WT[1:M]);
                          FOR I:=1 TO N DO
                              A(I)[1:M]:=A(I)[1:M]*WT[1:M];
                          ENDF;
                          FOR I:=1 TO IP DO
                              B(I)[1:M]:=B(I)[1:M]*WT[1:M];
                          ENDF;
```

```
                    ENDI;
                ENDI;
            ENDI;
 /*                                                                  */
 /*  CALL G4SQRDC TO DECOMPOSE MATRIX A.                             */
 /*                                                                  */
       CALL G4SQRDC(A,MAXM,M,N,WK);
 /*                                                                  */
 /*  CALL G4SQRSL TO SOLVE IP RIGHT HAND SIDES.                      */
 /*                                                                  */
       CALL G4SQRSL(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR);
         IF IERR>0 THEN IERR:=2
         ENDI;
    LAB1: ENDP;
 /******************************************************************/
 /*                                                                  */
          PROCEDURE G4SQRDC (A,MAXM,M,N,WK);
                    REAL VECTOR [MAXM] ARRAY(N) A;
                    REAL VECTOR  [N] WK;
                    AUTOMATIC REAL T;
                      INTEGER I,J,K,L,M,N,IP,MAXM,IERR,MAXN,JOB;
 /*                                                                  */
 /*    COMPUTE HH TRANSFORMATION FOR COLUMN I                        */
 /*                                                                  */
                FOR I:=1 TO N-1 DO
                    WK[I]:= A(I)[I:M] .DOT. A(I)[I:M];
                    WK[I]:= SQRT(WK[I]);
                    IF WK[I] > 0 THEN
                        WK[I]:= WK[I]*ABS(A(I)[I])/A(I)[I];
                        A(I)[I:M]:=A(I)[I:M]/WK[I];
                        A(I)[I]:=A(I)[I]+1.;
 /*                                                                  */
 /*     APPLY HH TRANSFORMATION TO REST OF THE COLUMNS               */
 /*                                                                  */
                        J:= I+1;
                        FOR K:= J TO N DO
                            T:= A(I)[I:M] .DOT. A(K)[I:M];
                            T:=-T/A(I)[I];
                            A(K)[I:M]:= A(K)[I:M] + T* A(I)[I:M];
                        ENDF;
                    ENDI;
                ENDF;
                    WK[N]:= A(N)[N:M] .DOT. A(N)[N:M];
                    WK[N]:= SQRT(WK[N]);
                    IF WK[N] > 0 THEN
                        WK[N]:= WK[N]*ABS(A(N)[N])/A(N)[N];
                        A(N)[N:M]:= A(N)[N:M]/WK[N];
                        A(N)[N]:= A(N)[N]+1;
                    ENDI;
        ENDP;
 /******************************************************************/
 /*                                                                  */
        PROCEDURE G4SQRSL (MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,
                          SUM,WK,IERR);
                   REAL VECTOR [MAXM] ARRAY(N) A;
                   REAL VECTOR  [MAXN] ARRAY (IP) X;
                   REAL VECTOR  [MAXM]  ARRAY (IP) B,RSD;
                   REAL VECTOR  [N] WK;
                   REAL VECTOR [M] WT;
                   REAL VECTOR  [IP] SUM;
                   INTEGER I,J,K,L,M,N,IP,MAXM,IERR,MAXN,JOB;
                   AUTOMATIC REAL T;
                   IERR:=0;
```

```
/*                                                                    */
/*         SPECIAL ACTION WHEN M=1                                    */
/*                                                                    */
        IF M = 1 THEN
              IF WK[1] = 0 THEN
                 IERR:=1;
                 GO TO LAB4
              ENDI;
              IF JOB <>2  THEN
                 FOR I:=1 TO IP DO
                    X(I)[1]:= B(I)[1]/A(1)[1];
                 ENDF;
              ENDI;
              IF JOB <> 1 THEN
                    RSD(1)[1:IP]:=0.0;
              ENDI;
              GO TO LAB4;
        ENDI;
/*                                                                    */
/*                    COMPUTE TRANS(Q)*B                              */
/*                                                                    */

               FOR I :=1 TO N DO
                 IF WK[I] <> 0 THEN
                   FOR J:=1 TO IP DO
                      T:= A(I)[I:M] .DOT. B(J)[I:M];
                      T:= -T/A(I)[I];
                      B(J)[I:M]:= B(J)[I:M] + T*A(I)[I:M];
                   ENDF;
                 ENDI;
               ENDF;
               FOR I:=1 TO IP DO
                    X(I)[1:N] :=B(I)[1:N];
                ENDF;
          IF JOB > 1 THEN
/*                                                                    */
/*         COMPUTE THE RESIDUES                                       */
/*                                                                    */
              FOR I :=1 TO IP DO
                 RSD(I)[1:M]:=0.0;
              ENDF;
              FOR I :=1 TO IP DO
                  K:=N+1;
                  RSD(I)[K:M]:=B(I)[K:M];
              ENDF;
              FOR K := N DOWNTO 1 DO
                    FOR L :=1 TO IP DO
                       T:=A(K)[K:M] .DOT. RSD(L)[K:M];
                       IF WK[K]=0 THEN
                            IERR:=K; GO TO LAB4
                       ENDI;
                       T:=-T/A(K)[K];
                       RSD(L)[K:M]:=RSD(L)[K:M] + T*A(K)[K:M];
                       SUM[L]:=RSD(L)[1:M] .DOT. RSD(L)[1:M];
                    ENDF;
              ENDF;
              IF WT[1] > 0 THEN
                 FOR I := 1 TO IP DO
                    RSD(I)[1:M]:= RSD(I)[1:M]/WT[1:M];
                 ENDF;
                 WT[1:M]:=WT[1:M]*WT[1:M];
              ENDI;
          ENDI;
```

```
            IF JOB <>2 THEN
/*                                                                          */
/*          COMPUTE THE SOLUTIONS                                           */
/*                                                                          */
                    FOR I:= N DOWNTO 2 DO
                        IF WK[I]=0 THEN
                           IERR:=I; GO TO LAB4
                        ELSE
                          K:=I-1;
                          FOR J:= 1 TO IP DO
                              X(J)[I]:= -X(J)[I]/WK[I];
                              T:= -X(J)[I];
                              X(J)[1:K]:= X(J)[1:K] + T*A(I)[1:K];
                          ENDF;
                        ENDI;
                    ENDF;
                    FOR I:=1 TO IP DO
                       IF WK[1]=0 THEN
                           IERR:=1; GO TO LAB4
                        ELSE
                             X(I)[1]:= -X(I)[1]/WK[1];
                        ENDI;
                    ENDF;
            ENDI;
/*                                                                          */
/*                         SAVE THE TRANSFORMATION                          */
/*                                                                          */
      LAB4:         FOR I:=1 TO N DO
                       T:=A(I)[I]; A(I)[I]:=-WK[I]; WK[I]:=T;
                    ENDF;
     ENDP;
  ENDM;
[illegible]
```

APPENDIX B

FORTRAN Coding of QR Algorithm

```
      SUBROUTINE QRASOS(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR)  QRAS0010
C***********************************************************************QRAS0020
C*                                                                     *QRAS0030
C* PURPOSE                                                             *QRAS0040
C*          TO SOLVE M SIMULTANEOUS EQUATIONS IN N UNKNOWNS WITH IP    *QRAS0050
C*          RIGHT HAND SIDES SO THAT THE SOLUTIONS ARE THE BEST POSSIBLE*QRAS0060
C*          FIT IN THE LEAST SQUARES SENSE.  THE ROUTINE USES HOUSE-   *QRAS0070
C*          HOLDER TRANSFORMATIONS TO PERFORM THE QR DECOMPOSITION     *QRAS0080
C*          OF THE COEFFICIENT MATRIX.                                 *QRAS0090
C*                                                                     *QRAS0100
C* USE                                                                 *QRAS0110
C*                                                                     *QRAS0120
C*    CALL QRASOS(MAXM,MAXN,M,N,IP,A,B,WT,JOB,X,RSD,SUM,WK,IERR)       *QRAS0130
C*                                                                     *QRAS0140
C* PARAMETERS                                                          *QRAS0150
C*                                                                     *QRAS0160
C*       MAXM  AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE  *QRAS0170
C*             A,B, AND RSD ARRAYS IN THE CALLING PROGRAM.  MAXM MUST  *QRAS0180
C*             BE GREATER THAN OR EQUAL TO M.                          *QRAS0190
C*                                                                     *QRAS0200
C*       MAXN  AN INPUT INTEGER SPECIFYING THE FIRST DIMENSION OF THE  *QRAS0210
C*             X ARRAY IN THE CALLING PROGRAM.  MAXN MUST BE GREATER   *QRAS0220
C*             THAN OR EQUAL TO N.                                     *QRAS0230
C*                                                                     *QRAS0240
C*          M  AN INPUT INTEGER SPECIFYING THE NUMBER OF ROWS OF THE   *QRAS0250
C*             A AND B ARRAYS.  M MUST BE GREATER THAN OR EQUAL TO N.  *QRAS0260
C*                                                                     *QRAS0270
C*          N  AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF    *QRAS0280
C*             THE A ARRAY.                                            *QRAS0290
C*                                                                     *QRAS0300
C*         IP  AN INPUT INTEGER SPECIFYING THE NUMBER OF COLUMNS OF    *QRAS0310
```

```
C*          THE B ARRAY.                                                *QRAS0320
C*                                                                      *QRAS0330
C*        A AN INPUT/OUTPUT TWO-DIMENSIONAL ARRAY WITH FIRST DIMEN-     *QRAS0340
C*          SION EQUAL TO MAXM AND SECOND DIMENSION AT LEAST N.         *QRAS0350
C*          ON INPUT, A MUST CONTAIN THE MATRIX OF COEFFICIENTS OF      *QRAS0360
C*          THE SYSTEM OF EQUATIONS.  ON OUTPUT, A CONTAINS INFOR-      *QRAS0370
C*          MATION DESCRIBING THE QR DECOMPOSITION OF A.                *QRAS0380
C*                                                                      *QRAS0390
C*        B AN INPUT TWO-DIMENSIONAL ARRAY WITH FIRST DIMENSION         *QRAS0400
C*          EQUAL TO MAXM AND SECOND DIMENSION AT LEAST IP.             *QRAS0410
C*          THE COLUMNS OF B MUST CONTAIN THE IP RIGHT HAND SIDE        *QRAS0420
C*          VECTORS.                                                    *QRAS0430
C*                                                                      *QRAS0440
C*       WT AN INPUT ONE-DIMENSIONAL ARRAY OF WEIGHTS.  IF WEIGHT-      *QRAS0450
C*          ING IS DESIRED,  WT MUST HAVE LENGTH AT LEAST M, AND        *QRAS0460
C*          THE FIRST M LOCATIONS MUST CONTAIN REAL NUMBERS GREATER     *QRAS0470
C*          THAN ZERO.  IF WEIGHTING IS NOT DESIRED, WT CAN CONSIST     *QRAS0480
C*          OF A SINGLE LOCATION WHICH MUST CONTAIN A NEGATIVE REAL     *QRAS0490
C*          NUMBER.                                                     *QRAS0500
C*                                                                      *QRAS0510
C*      JOB AN INPUT INTEGER SPECIFYING RESULTS TO BE COMPUTED.         *QRAS0520
C*                                                                      *QRAS0530
C*             =1  COMPUTE SOLUTIONS ONLY.                              *QRAS0540
C*             =2  COMPUTE RESIDUALS ONLY.                              *QRAS0550
C*             =3  COMPUTE BOTH SOLUTIONS AND RESIDUALS.                *QRAS0560
C*                                                                      *QRAS0570
C*        X AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE SOLU-        *QRAS0580
C*          TIONS.  IF JOB=1 OR JOB=3, X MUST BE DIMENSIONED WITH       *QRAS0590
C*          FIRST DIMENSION EQUAL TO MAXN AND SECOND DIMENSION          *QRAS0600
C*          AT LEAST IP.  IF JOB=2, X CAN BE A DUMMY PARAMETER.         *QRAS0610
C*                                                                      *QRAS0620
```

```
C*       RSD  AN OUTPUT TWO-DIMENSIONAL ARRAY CONTAINING THE RESID-   *QRAS0630
C*            UALS.  IF JOB=2 OR JOB=3, RSD MUST BE DIMENSIONED WITH  *QRAS0640
C*            FIRST DIMENSION EQUAL TO MAXM AND SECOND DIMENSION      *QRAS0650
C*            AT LEAST IP.  IF JOB=1, RSD CAN BE A DUMMY PARAMETER.   *QRAS0660
C*                                                                    *QRAS0670
C*       SUM  AN OUTPUT ONE-DIMENSIONAL ARRAY CONTAINING THE WEIGHTED *QRAS0680
C*            SUMS OF SQUARES OF THE RESIDUALS.  IF JOB=2 OR JOB=3,   *QRAS0690
C*            SUM MUST BE DIMENSIONED AT LEAST IP.  IF JOB=1, SUM     *QRAS0700
C*            CAN BE A DUMMY PARAMETER.                               *QRAS0710
C*                                                                    *QRAS0720
C*        WK  A ONE-DIMENSIONAL WORK ARRAY WHICH MUST BE DIMENSIONED  *QRAS0730
C*            AT LEAST N.  ON OUTPUT, WK CONTAINS INFORMATION ON THE  *QRAS0740
C*            QR DECOMPOSITION OF A.                                  *QRAS0750
C*                                                                    *QRAS0760
C*      IERR  AN INTEGER ERROR CODE.                                  *QRAS0770
C*                                                                    *QRAS0780
C*               =0  NO ERROR DETECTED.                               *QRAS0790
C*               =1  N IS GREATER THAN M.                             *QRAS0800
C*               =2  THE DECOMPOSED MATRIX IS SINGULAR.               *QRAS0810
C*               =3  WEIGHTING WAS REQUESTED AND ONE OR MORE WEIGHTS  *QRAS0820
C*                   IS NEGATIVE.                                     *QRAS0830
C*                                                                    *QRAS0840
C*   REQUIRED ROUTINES        NORMS,SQRDC2,SQRSL2,SAXPY1,SDOT1,SSCAL  *QRAS0850
C*                            SCOPY                                   *QRAS0860
C*                                                                    *QRAS0870
C*   FORTRAN FUNCTIONS        ABS,AMAX1,MIN0,MOD,SIGN,SQRT            *QRAS0880
C*                                                                    *QRAS0890
C*   SOURCE                   COMPUTER SCIENCES CORPORATION,          *QRAS0900
C*                            HAMPTON, VA.                            *QRAS0910
C*                                                                    *QRAS0920
C*   LANGUAGE                 FORTRAN                                 *QRAS0930
                                                                      *QRAS0940
```

```
C*    DATE RELEASED            AUGUST 1, 1978
C*
C*    LATEST REVISION          OCTOBER 10, 1978
C*
C*
C***********************************************************************
      DIMENSION A(MAXM,1),B(MAXM,1),X(MAXN,1),RSD(MAXM,1),WT(1),WK(1)
      DIMENSION SUM(1)
      IERR = 0
C
C
C              CHECK FOR M LESS THAN N.
C
      IF(M .GE. N) GO TO 10
         IERR = 1
         GO TO 160
C
C              CHECK FOR NO WEIGHTING
C
   10 IF(WT(1) .LT. 0.0) GO TO 80
C
C              CHECK FOR ILLEGAL WEIGHTS
C
         DO 20 I = 2, M
            IF(WT(I) .LE. 0.0) GO TO 30
   20    CONTINUE
         GO TO 40
   30    IERR = 3
         GO TO 160
C
C           WEIGHT THE A AND B ARRAYS BY THE SQUARE ROOT
```

```
C          OF THE WEIGHT ARRAY.
C
   40    DO 70 I = 1, M
            WT(I) = SQRT(WT(I))
            DO 50 J = 1, N
               A(I,J) = WT(I)*A(I,J)
   50       CONTINUE
            DO 60 J = 1, IP
               B(I,J) = WT(I)*B(I,J)
   60       CONTINUE
   70    CONTINUE
   80 CONTINUE
C
C          CALL SQRDC2 TO DECOMPOSE A
C
      CALL SQRDC2(A,MAXM,M,N,WK)
C
C          CALL SQRSL2 TO SOLVE FOR IP RIGHT HAND SIDES
C
      CALL SQRSL2(A,MAXM,M,N,MAXN,IP,WK,B,X,RSD,JOB,IERR)
      IF(IERR .EQ. 0) GO TO 90
         IERR = 2
         GO TO 160
   90 CONTINUE
C
C          COMPUTE THE SUM OF WEIGHTED SQUARES OF RESIDUALS.
C
      IF(JOB .EQ. 1) GO TO 140
      DO 110 J = 1, IP
         SUM(J) = 0.0
         DO 100 I = 1, M
            SUM(J) = SUM(J) + RSD(I,J)*RSD(I,J)
```

```
 100    CONTINUE
 110 CONTINUE
C
C         COMPUTE UNWEIGHTED RESIDUALS
C
      IF(WT(1) .LT. 0.0) GO TO 160
         DO 130 I = 1, M
            DO 120 J = 1, IP
               RSD(I,J) = RSD(I,J)/WT(I)
 120        CONTINUE
 130     CONTINUE
 140 CONTINUE
      IF(WT(1) .LT. 0.0) GO TO 160
      DO 150 I=1,M
         WT(I) = WT(I)*WT(I)
 150 CONTINUE
 160 CONTINUE
      RETURN
      END
      SUBROUTINE SQRDC2(X,LDX,N,P,QRAUX)
      INTEGER LDX,N,P
      REAL X(LDX,1),QRAUX(1)
C
C     SQRDC2 USES HOUSEHOLDER TRANSFORMATIONS TO COMPUTE THE QR
C     FACTORIZATION OF AN N BY P MATRIX X.
C
C     ON ENTRY
C
C        X       REAL(LDX,P), WHERE LDX .GE. N.
C                X CONTAINS THE MATRIX WHOSE DECOMPOSITION IS TO BE
C                COMPUTED.
```

```
C
C        LDX     INTEGER.
C                LDX IS THE LEADING DIMENSION OF THE ARRAY X.
C
C        N       INTEGER.
C                N IS THE NUMBER OF ROWS OF THE MATRIX X.
C
C        P       INTEGER.
C                P IS THE NUMBER OF COLUMNS OF THE MATRIX X.
C
C
C     ON RETURN
C
C        X       X CONTAINS IN ITS UPPER TRIANGLE THE UPPER
C                TRIANGULAR MATRIX R OF THE QR FACTORIZATION.
C                BELOW ITS DIAGONAL X CONTAINS INFORMATION FROM
C                WHICH THE ORTHOGONAL PART OF THE DECOMPOSITION
C                CAN BE RECOVERED.
C
C        QRAUX   REAL(P).
C                QRAUX CONTAINS FURTHER INFORMATION REQUIRED TO RECOVER
C                THE ORTHOGONAL PART OF THE DECOMPOSITION.
C
C
C     LINPACK SUBROUTINE SQRDC VERSION DATED 07/14/77, REVISED BY
C     COMPUTER SCIENCES CORPORATION, HAMPTON, VA. 10/10/78.
C
C     BLAS SAXPY1,SDOT1,SSCAL   LRC NORMS
C     FORTRAN ABSSIGN,SQRT,MOD
C
C     INTERNAL VARIABLES
C
```

```
      INTEGER J,L,LP1
      REAL SDOT,NRMXL,T
C
C
C
C     PERFORM THE HOUSEHOLDER REDUCTION OF X.
C
      DO 190 L = 1, P
         QRAUX(L) = 0.0E0
         IF (L .EQ. N) GO TO 170
C
C           COMPUTE THE HOUSEHOLDER TRANSFORMATION FOR COLUMN L.
C
            NLEN = N-L+1
            CALL NORMS(NLEN,NLEN,1,X(L,L),2,NRMXL)
            IF (NRMXL .EQ. 0.0E0) GO TO 160
               IF (X(L,L) .NE. 0.0E0) NRMXL = SIGN(NRMXL,X(L,L))
               CALL SSCAL(N-L+1,1.0E0/NRMXL,X(L,L),1)
               X(L,L) = 1.0E0 + X(L,L)
C
C              APPLY THE TRANSFORMATION TO THE REMAINING COLUMNS.
C
               LP1 = L + 1
               IF (P .LT. LP1) GO TO 150
               DO 140 J = LP1, P
                  T = -SDOT1(N-L+1,X(L,L),X(L,J))/X(L,L)
                  CALL SAXPY1(N-L+1,T,X(L,L),X(L,J))
  140          CONTINUE
  150          CONTINUE
C
C
```

```
C                 SAVE THE TRANSFORMATION.
                  QRAUX(L) = X(L,L)
                  X(L,L) = -NRMXL
  160          CONTINUE
  170       CONTINUE
  180 CONTINUE
      RETURN
      END
      SUBROUTINE SQRSL2(X,LDX,N,K,LDB,IP,QRAUX,Y,BETA,RSD,JOB,INFO)
      INTEGER LDX,N,K,LDB,IP,JOB,INFO
      REAL X(LDX,1),QRAUX(1),Y(LDX,1),BETA(LDB,1),RSD(LDX,1)
C
C     SQRSL2 APPLIES THE OUTPUT OF THE SUBROUTINE SQRDC2 TO
C     COMPUTE A SET OF IP LEAST SQUARES SOLUTIONS AND RESIDUALS.  THE
C     OUTPUT OF SQRDC2 IS THE DECOMPOSITION OF THE N BY K MATRIX
C     X IN THE FORM
C
C                  X = Q * (R)
C                          (0)
C
C     WHERE Q IS ORTHOGONAL AND  R IS UPPER TRIANGULAR.  THIS
C     INFORMATION IS CONTAINED IN CODED FORM IN THE ARRAY X
C     AND THE ARRAY QRAUX.
C
C     ON ENTRY
C
C        X        REAL(LDX,K), WHERE LDX .GE. N.
C                 X CONTAINS THE OUTPUT FROM SQRDC.
C
C        LDX      INTEGER.
C                 LDX IS THE LEADING DIMENSION OF THE ARRAY X.
C
```

```
C        N        INTEGER.
C                 N IS THE NUMBER OF ROWS OF THE MATRIX X.
C
C        K        INTEGER.
C                 K IS THE NUMBER OF COLUMNS OF THE MATRIX X.
C
C        LDB      INTEGER.
C                 LDB IS THE LEADING DIMENSION OF THE ARRAY BETA.
C
C        IP       INTEGER.
C                 IP IS THE NUMBER OF RIGHT HAND SIDES.
C
C        QRAUX    REAL(K)
C                 QRAUX CONTAINS THE OUTPUT FROM SQRDC2.
C
C        Y        REAL(LDX,IP).
C                 Y IS THE N BY IP RIGHT HAND SIDE MATRIX THAT IS
C                 MANIPULATED BY SQRSL2.
C
C
C        JOB      INTEGER.
C                 JOB IS A PARAMETER THAT CONTROLS WHAT IS TO BE
C                 COMPUTED.
C
C                    IF JOB .EQ. 1  COMPUTE SOLUTIONS ONLY.
C                    IF JOB .EQ. 2  COMPUTE RESIDUALS ONLY.
C                    IF JOB .EQ. 3  COMPUTE SOLUTIONS AND RESIDUALS.
C
C     ON RETURN
C
C        BETA     REAL(LDB,IP).
```

```
C               BETA CONTAINS THE SOLUTIONS OF THE LEAST SQUARES      QRAS3150
C               PROBLEMS                                              QRAS3160
C                  MINIMIZE NORM2(Y(I) - X*BETA(I)), I=1,2,...,IP     QRAS3170
C               IF THEIR COMPUTATION HAS BEEN REQUESTED.              QRAS3180
C                                                                     QRAS3190
C        RSD    REAL(LDX,IP)                                          QRAS3200
C               RSD CONTAINS THE LEAST SQUARES RESIDUALS              QRAS3210
C                  Y(I) - X*BETA(I), I=1,2,...,IP                     QRAS3220
C               IF THEIR COMPUTATION HAS BEEN REQUESTED.              QRAS3230
C                                                                     QRAS3240
C        INFO   INTEGER                                               QRAS3250
C               INFO IS ZERO UNLESS THE CALCULATION OF BETA HAS BEEN  QRAS3260
C               REQUESTED AND P IS SINGULAR, IN WHICH CASE INFO IS    QRAS3270
C               THE INDEX OF THE FIRST ZERO DIAGONAL ELEMENT OF R.    QRAS3280
C               IN THIS CASE BETA IS UNALTERED.                       QRAS3290
C                                                                     QRAS3300
C     LINPACK SUBROUTINE SQRSL VERSION DATED 07/14/77, REVISED BY     QRAS3310
C     COMPUTER SCIENCES CORPORATION, HAMPTON, VA. 10/10/78.           QRAS3320
C                                                                     QRAS3330
C     BLAS SAXPY1,SCOPY,SDOT1                                         QRAS3340
C     FORTRAN ABS,MIN0,MOD                                            QRAS3350
C                                                                     QRAS3360
C     INTERNAL VARIABLES                                              QRAS3370
C                                                                     QRAS3380
      INTEGER I,J,JJ,JU,KP1                                           QRAS3390
      REAL SDOT,T,TEMP                                                QRAS3400
C                                                                     QRAS3410
C                                                                     QRAS3420
C     SET INFO FLAG                                                   QRAS3430
C                                                                     QRAS3440
      INFO = 0                                                        QRAS3450
      JU = MIN0(K,N-1)                                                QRAS3460
```

```
C
C     SPECIAL ACTION WHEN N=1
C
      IF (JU .NE. 0) GO TO 20
         IF(X(1,1) .NE. 0.0) GO TO 5
            INFO = 1
            GO TO 220
    5    CONTINUE
         DO 10 L = 1, IP
            IF(JOB .NE. 2) BETA(1,L) = Y(1,L)/X(1,1)
            IF(JOB .NE. 1) RSD(1,L) = 0.0E0
   10    CONTINUE
      GO TO 220
   20 CONTINUE
C
C           COMPUTE TRANS(Q)*Y
C
            DO 50 J = 1, JU
               IF (QRAUX(J) .EQ. 0.0E0) GO TO 40
                  TEMP = X(J,J)
                  X(J,J) = QRAUX(J)
                  DO 30 L = 1, IP
                     T = -SDOT1(N-J+1,X(J,J),Y(J,L))/X(J,J)
                     CALL SAXPY1(N-J+1,T,X(J,J),Y(J,L))
   30             CONTINUE
                  X(J,J) = TEMP
   40          CONTINUE
   50       CONTINUE
         KP1 = K + 1
         IF (JOB .EQ. 1 .OR. K .EQ. N) GO TO 70
            DO 60 L = 1, IP
```

```
            CALL SCOPY(N-K,Y(KP1,L),1,RSD(KP1,L),1)
   60    CONTINUE
   70 CONTINUE
      IF (JOB .EQ. 2) GO TO 120
C
C        COMPUTE BETA
C
         DO 75 L = 1, IP
            CALL SCOPY(K,Y(1,L),1,BETA(1,L),1)
   75    CONTINUE
         DO 100 JJ = 1, K
            J = K - JJ + 1
            IF (X(J,J) .NE. 0.0E0) GO TO 80
               INFO = J
C           ......EXIT
               GO TO 220
   80       CONTINUE
            DO 95 L = 1, IP
               BETA(J,L) = BETA(J,L)/X(J,J)
               IF (J .EQ. 1) GO TO 90
                  T = -BETA(J,L)
                  CALL SAXPY1(J-1,T,X(1,J),BETA(1,L))
   90          CONTINUE
   95       CONTINUE
  100    CONTINUE
  110    CONTINUE
  120 CONTINUE
      IF (JOB .EQ. 1) GO TO 210
C
C        COMPUTE RSD IF REQUIRED
C
         DO 160 L = 1, IP
```

```
             DO 150 I = 1, K
                RSD(I,L) = 0.0E0
150          CONTINUE
160       CONTINUE
          DO 200 JJ = 1, JU
             J = JU - JJ + 1
             IF (GRAUX(J) .EQ. 0.0E0) GO TO 190
                TEMP = X(J,J)
                X(J,J) = GRAUX(J)
                DO 170 L = 1, IP
                   T = -SDOT1(N-J+1,X(J,J),RSD(J,L))/X(J,J)
                   CALL SAXPY1(N-J+1,T,X(J,J),RSD(J,L))
170             CONTINUE
                X(J,J) = TEMP
190          CONTINUE
200       CONTINUE
210    CONTINUE
220 CONTINUE
    RETURN
    END
```

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-165730		

4. Title and Subtitle	5. Report Date
STAR-ADAPTATION OF QR ALGORITHM	June 1981
	6. Performing Organization Code
7. Author(s) SHANTILAL N. SHAH	8. Performing Organization Report No.
9. Performing Organization Name and Address Hampton Institute Hampton, Virginia 23668	10. Work Unit No.
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	13. Type of Report and Period Covered Contractor Report
	14. Sponsoring Agency Code

15. Supplementary Notes

This work was performed under Cooperative Agreement NCC1-7 with the NASA Langley Research Center and Hampton Institute.

16. Abstract

The QR algorithm used on a serial computer and presently executed on the Control Data Corporation 6000 Computer was adapted to execute efficiently on the Control Data STAR-100 computer. This paper describes how the scalar program was adapted for the STAR-100 and indicates why these adaptations yielded an efficient STAR program. Program listings of the old scalar version and the new vectorized SL/1 version are presented in the appendices. Execution times for the two versions applied to the same system of linear equations, are compared.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
STAR computer QR algorithm SL/1, LINPACK, BLAS	Unclassified - Unlimited Subject Category 64

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	28	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161