Mecánica Computacional Volumen XV
Compilado por A. Larreteguy y M. Vénere
Bariloche, noviembre de 1995

ON THE USE OF INTERVAL ARITHMETIC TO STUDY ERROR PROPAGATIONS IN THE ARTIFICIAL SATELLITE ORBIT DETERMINATIONS

Kondapalli Rama Rao
Div. da Mecânica Espacial e Controle - DMC
INPE, CP 515, São José dos Campos, SP Brazil

Roger A. Broucke
Dept of Aerospace Engineering
The University of Texas at Austin, U.S.A.

ABSTRACT

Two of the most important numerical methods that are widely used in the artificial satellite orbit determination are the least squares procedures to process the tracking station observations and the numerical integration of ordinary differential equations. The propagation of errors in these methods is a very important problem in the area of orbital mechanics. The recently developed *Interval Analysis* is a technique to provide lower and upper bounds for the exact values of this type of errors and can be used in these error propagations. *Interval Arithmetic* replaces the computation on real numbers by a computation on pairs (a,b) where the a and b are real numbers: a is the lower bound and b the upper bound. This is thus a convenient way of representing the error that may be associated with a number. The number is supposed to be anywhere (uniformly) between these two limits a and b. With the idea of using this technique in this problem, our work aims at comparing two different methods of least squares, using numerical calculations in *Interval Arithmetic*: the standard method with inversion of A-transpose-A and the orthogonalization method with the Gram-Schmidt decomposition. The basic problem in question, the two methods of least squares being compared, and the preliminary computer program listings using interval arithmetic alongwith the results and comments are given here. A comparison of two numerical integration methods - Runge-Kutta methods of orders 4 and 5 - is proposed to be done in a later work.

INTRODUCTION

Many a time a practical problem requiring numerical computations deals with quantities determined experimentally by approximate measurements, with only a rough estimate of these measured values. A typical computation starts with some numbers known only to a certain number of significant decimal digits. This inexact initial data can only give results with limited accuracy [1]. Often it becomes mandatory to assess the accuracy obtained in such results in order to estimate the sensitivity of final results to small changes in initial data. A number of analytical techniques are available for this purpose but their application is usually very laborious. One can alternatively repeat computations in a digital computer with several sets of nearby initial values but it is possible only in short computations.

Even if the initial values given are exact, another source of error in the final results is round-off error i.e. error due to the finite precision of machine arithmetic. Computation in a digital

computer is a finite sequence of inexact arithmetic operations and very often one needs procedures by which the computer can say how far off the results of its limited precision arithmetic operations can be [2].

CONCEPT OF INTERVAL ARITHMETIC

Out of all the techniques available for treating the error types described, a recently developed concept called *interval arithmetic* seems to give promising results in many practical problems where extremely large number of data points obtained from approximate measurements are involved. To obtain error bounds for the differences between the results and the exact solution values, one usually have a pair of numbers: an appropriate value for solution and an upper bound, or an upper bound and a lower bound to the exact result. Interval arithmetic technique is simply an extension of real arithmetic and performs arithmetic operations on numbers which are defined as ordered pairs of real numbers $[a,b]$, with $a \le b$, where the end points a and b are assumed to be known with infinite precision. In other words, the interval arithmetic deals with a *range* instead of some fixed value for a real number. According to this concept, if a real number x is defined as the result of a finite sequence of arithmetic operations beginning with a finite collection of real numbers with known decimal representations, then the execution by the computer of the corresponding finite sequence or rounded interval arithmetic operations produces an interval, X, containing the real number x and by carrying enough places, the width of X can be made arbitrarily small [2]. The aim of this work is only to apply this concept in a practical problem. A systematic introduction to the tools of interval analysis as well as a unifying presentation of the interval analytic methods are given elsewhere [3]. The set of four basic arithmetic operations for interval numbers and the width of an interval [a,b], w([a,b]), are defined by [2]:

$$[a,b] + [c,d] = [a+c,\ b+d]$$
$$[a,b] - [c,d] = [a-d,\ b-c]$$
$$[a,b] \,.\, [c,d] = [\min(ac,ad,bc,bd),\ \max(ac,ad,bc,bd)]$$
$$[a,b] / [c,d] = [a,b].[1/d,1/c] \quad \text{if } 0 \notin [c,d]$$
$$w([a,b]) = b-a.$$

BASIC PROBLEM

Orbit estimation is one of the most important problems in the theory of orbits of the artificial satellites. It involves a comparison of observations, made through tracking stations, with propagated predictions obtained from mathematical models. In a typical orbit estimation (or determination) problem, the knowledge of the orbit is improved by minimizing the difference between the observed and the computed trajectory using a batch least squares differential method [4]. Starting with two real-valued vectors $\mathbf{X}$ and $\mathbf{Y}$, the former representing physical state of a dynamic system and the latter its directly observable state, which are related by:

$$\mathbf{Y} = \mathbf{F}(\mathbf{X}),$$

the basic mathematical equation to be solved for in this process can simply be expressed as:

$$\mathbf{X}-\mathbf{X}_0 = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T(\mathbf{Y}-\mathbf{F}(\mathbf{X}_0))$$
$$\Delta\mathbf{X} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\Delta\mathbf{Y}, \qquad (1)$$

where $\mathbf{X}$ is an n-dimensional vector of satellite trajectory parameters with its initial values given by $\mathbf{X}_0$, $\mathbf{Y}$ is an m-dimensional vector of tracking observations ($m \geq n$), $\mathbf{H}$ is an mxn matrix of partial derivatives of $\mathbf{F(X)}$ with respect to $\mathbf{X}$, evaluated at $\mathbf{X}=\mathbf{X}_0$. This process involves two basic numerical procedures: a least squares method for estimating the differential correction $\Delta\mathbf{X}$ and integration of a system of differential equations in mounting the matrix $\mathbf{H}$. This paper is aimed at comparing two methods of least squares: with the standard method of matrix inversion and with the Gram-Schmidt decompostion, in order to assess the error propagations in these numerical procedures, using interval arithmetic technique. Comparison of two numerical integration methods - Runge-Kutta methods of orders 4 and 5 - will be done in a continuation of this work.

LEAST SQUARES METHOD WITH STANDARD MATRIX INVERSION

If $\mathbf{A} = [a_{ij}]$ is a square matrix, and if $\mathbf{A}_{ij}$ is the cofactor of a_{ij} in the determinant of $\mathbf{A}$, then the matrix $[\mathbf{A}_{ji}] = [\mathbf{A}_{ij}]^T$ = Transpose of $[\mathbf{A}_{ij}]$ is called the adjoint of the matrix $\mathbf{A}$ and the inverse $\mathbf{A}^{-1}$ of a nonsingular matrix $\mathbf{A} = [a_{ij}]$ is defined as the adjoint of $\mathbf{A}$ divided by the determinant of $\mathbf{A}$ [5], i.e.

$$\mathbf{A}^{-1} = [\mathbf{A}_{ij}]^T/\det \mathbf{A}. \tag{2}$$

This is the basic definition of the inverse of a matrix. In the problem of orbit estimation, starting with the vector $\mathbf{X}_0$ of initial values and having mounted the matrix $\mathbf{H}$ somehow, the inverse of the square matrix $\mathbf{H}^T\mathbf{H}$ can be computed by using the Eq. (2) and the Eq. (1) can be solved to compute $\Delta\mathbf{X}$.

LEAST SQUARES METHOD WITH GRAM-SCHMIDT DECOMPOSITION

Given a matrix $\mathbf{A} \in \Re^{mxn}$ with rank(A) = n, the Gram-Schmidt algorithm computes the factorization $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q} \in \Re^{mxn}$ has orthonormal columns and $\mathbf{R} \in \Re^{nxn}$ is an upper triangular matrix [6]. In otherwords, the matrix $\mathbf{A}$ is factorized into an easily computed unitary matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$. Then a given general system of linear equations can be transformed as follows:

$$\begin{aligned} \mathbf{AX} &= \mathbf{B} \\ \mathbf{QRX} &= \mathbf{B} \\ \mathbf{Q}^{-1}\mathbf{QRX} &= \mathbf{Q}^{-1}\mathbf{B} \\ \mathbf{RX} &= \mathbf{Q}^{-1}\mathbf{B}, \end{aligned} \tag{3}$$

since $\mathbf{Q}^{-1}\mathbf{Q} = \mathbf{I}$, an identity matrix. Since $\mathbf{R}$ is an upper triangular matrix, the resolution of the Eq.(3) becomes simple. This procedure also can be used for computing $\Delta\mathbf{X}$ of the Eq. (1).

TEST PROBLEM

Considering simply a system of nonhomogeneous linear equations similar to that of Eq. (1), expressed as:

$$\mathbf{AX} = \mathbf{B}, \tag{4}$$

where **A** is a 10 x 3 coefficient matrix, **X** is a 3 x 1 solve-for vector and **B** is a 10 x 1 vector, the elements of **A** and **B** have been generated in a random manner as interval numbers (ordered pairs of real numbers) where the width of the interval is of the order of 10^{-06}. Then the Eq. (4) is solved using the two least squares methods described before.

RESULTS AND COMMENTS

The preliminary computer program listings using interval arithmetic in the two least squares methods along with the results obtained are given at the end of the paper. In the case of using the standard method with matrix inversion, looking at the solution vector, the maximum width of the interval numbers is $3x10^{-04}$ whereas the interval width in the elements of the matrix A is $1x10^{-06}$. Thus the errors are increased by a factor of 300. In the case of using Gram-Schmidt orthogonalization, the maximum width of the interval numbers in the solution vector obtained is $4x10^{-05}$. That is, the factor by which the width is increased is 40. Considering this error propagation process, once can conclude that the orthogonalization method gives better results than the matrix inversion method in orbit estimation type problems.

REFERENCES

[1] **Moore, R. E.** *Interval analysis*, Prentice-Hall, Inc., Englewood Cliffs, N. J. 1966.

[2] **Moore, R. E.** *The automatic analysis and control of error in digital computing based on the use of interval numbers.* In: Eds. Rall, L. B. *Error in digital computation*, V. 1, John Wiley & Sons, Inc., New York, 1965 (Proceedings of an Advanced Seminar conducted by the Mathematics Research Center, United States Army, at the University of Wisconsin, Madison, October 5-7, 1964).

[3] **Alefeld, G.; Herzeberger, J.** *Introduction to interval computations*, Academic Press, New York, 1983.

[4] **Kondapalli, R. R.; Kuga, H. K.** *On the least squares differential correction method for orbit estimation problem*, INPE, São José dos Campos, SP, Brazil (INPE-4222-NTI/274).

[5] **Wylie, C. R.** *Advanced engineering mathematics*, McGraw-Hill Kogakusha Ltd., Tokyo, 1975.

[6] **Golub, G. H.; Van Loan, C. F.** *Matrix computations*, The Johns Hopkins University Press, Baltimore, Maryland, 1985.

LISTINGS OF COMPUTER PROGRAMS AND RESULTS.

```
C
C     ** NAME OF THE PROGRAM **
C        INT(ERVAL ARITHMETIC USAGE IN A LEAST SQUARES METHOD WITH)
C        M(ATRIX) IN(VERSION) - INTMIN
C
C     ** NAME OF THE PROGRAMMER **
C        KONDAPALLI RAMA RAO
C        ROGER A. BROUCKE
C
C     ** AIM **
C        TO STUDY THE PERFORMANCE OF A LEAST SQUARES METHOD WITH THE
C        STANDARD METHOD OF MATRIX INVERSION USING NUMERICAL CALCULATIONS
C        IN INTERVAL ARITHMETIC

      REAL A(2,10,3),ATA(2,3,3),B(2,10)
      REAL AT(2,9,10),ATB(2,3),AX(2,3)
      OPEN(1,FILE='LEAST1.DAT',STATUS='NEW')
      WRITE(1,'(1H ,9A1)') CHAR(15),CHAR(27),CHAR(48)
      WRITE(1,'(/,A)')'**** INTERVAL ARITHMETIC - MATRIX INVERSION ****'
      WRITE(6,*) '  INPUT EPSILON: '
      READ(5,*) EPS
      IF(EPS .GT. 10.0) STOP
      WRITE (1,*)
      WRITE(1,*) 'EPSLON:',EPS
      M=10
      N=3
      IRAN = 1+1.0E+08
      DO 5 I=1,M
      B(1,I)=RAN(IRAN)
      B(2,I)=B(1,I)+EPS*RAN(IRAN)
      DO 5 J=1,N
         A(1,I,J)=RAN(IRAN)
    5    A(2,I,J)=A(1,I,J)+EPS*RAN(IRAN)
      WRITE (1,'(/,A)') '**** THE RECTANGULAR MATRIX A ****'
      CALL PRM1(A,M,N)
      WRITE (1,'(/,A)') '**** THE RIGHT-HAND SIDE B ****'
      CALL PRM1(B,M,1)
      CALL ATAM(A,ATA,M,N)
      WRITE (1,'(/,A)') '**** THE A-TRANSPOSE-A MATRIX ****'
      CALL PRM1(ATA,N,N)
      CALL INVERT(ATA,N)
      WRITE (1,'(/,A)') '**** THE INVERSE OF A-TRANSPOSE-A ****'
      CALL PRM1(ATA,N,N)
      CALL ATR(A,AT,M,N)
      CALL MULM(AT,B,ATB,N,M,1)
      CALL MULM(ATA,ATB,AX,N,N,1)
      WRITE (1,'(/,A)') '**** THE SOLUTION VECTOR ****'
      CALL PRM1(AX,N,1)
      STOP
      END
      SUBROUTINE ATAM(A,ATA,M,N)
      REAL A(2,M,N),ATA(2,N,N),TEMP(2)
      DO 1 I=1,N
      DO 1 J=1,N
      ATA(1,I,J)=0.0
      ATA(2,I,J)=0.0
      DO 1 K=1,M
         CALL INA(A(1,K,I),A(1,K,J),TEMP,3)
    1    CALL INA(ATA(1,I,J),TEMP,ATA(1,I,J),1)
      RETURN
      END
      SUBROUTINE INVERT(A,N)
      REAL A(2,N,N),PIV(2),Z(2),TEMP(2)
      DO 2 K=1,N
         IT=K
         JT=K
         PIV(1)=A(1,IT,JT)
         PIV(2)=A(2,IT,JT)
         A(1,IT,JT)=1.0
         A(2,IT,JT)=1.0
         DO 1 J=1,N
    1       CALL INA(A(1,IT,J),PIV,A(1,IT,J),4)
      DO 2 I=1,N
         IF(I.EQ.IT) GO TO 2
         Z(1)=A(1,I,JT)
         Z(2)=A(2,I,JT)
         A(1,I,JT)=0.0
         A(2,I,JT)=0.0
      DO 3 J=1,N
         CALL INA(Z,A(1,IT,J),TEMP,3)
    3    CALL INA(A(1,I,J),TEMP,A(1,I,J),2)
    2 CONTINUE
      RETURN
      END
      SUBROUTINE PRM1(A,M,N)
      REAL A(2,M,N)
   10 FORMAT(1H,4(E16.9,1H/,E16.9,1X))
      WRITE(1,10)
      DO 2 I=1,M
    2 WRITE(1,10) (A(1,I,J),A(2,I,J),J=1,N)
      RETURN
      END
```

```
      SUBROUTINE MULM(A,B,C,M,N,L)
      REAL A(2,M,N),B(2,N,L),C(2,M,L),TEMP(2)
      DO 1 I=1,M
         DO 1 J=1,L
         C(1,I,J)=0.0
         C(2,I,J)=0.0
         DO 1 K=1,N
            CALL INA(A(1,I,K),B(1,K,J),TEMP,3)
    1 CALL INA(C(1,I,J),TEMP,C(1,I,J),1)
      RETURN
      END
      SUBROUTINE ATR(A,AT,M,N)
      REAL A(2,M,N),AT(2,N,M)
      DO 1 I=1,M
         DO 1 J=1,N
         AT(1,J,I)=A(1,I,J)
    1 AT(2,J,I)=A(2,I,J)
      RETURN
      END
      SUBROUTINE INA(A,B,C,N)
      REAL A(2),B(2),C(2)
      IF(N.LT.1 .OR. N.GT.4) RETURN
      GO TO (1,2,3,4),N
    1 C(1)=A(1)+B(1)
      C(2)=A(2)+B(2)
      GOTO 100
    2 C111=A(1)-B(2)
      C(2)=A(2)-B(1)
      C(1)=C111
      GOTO 100
    3 C111=AMIN1(A(1)*B(1),A(1)*B(2),A(2)*B(1),A(2)*B(2))
      C(2)=AMAX1(A(1)*B(1),A(1)*B(2),A(2)*B(1),A(2)*B(2))
      C(1)=C111
      GOTO 100
    4 IF(B(1)*B(2) .LE. 0.0) GO TO 99
      C111=AMIN1(A(1)/B(1),A(1)/B(2),A(2)/B(1),A(2)/B(2))
      C(2)=AMAX1(A(1)/B(1),A(1)/B(2),A(2)/B(1),A(2)/B(2))
      C(1)=C111
  100 RETURN
   99 N=99
      RETURN
      END
```

```
*** INTERVAL ARITHMETIC - MATRIX INVERSION ****

EPSLON:  1.0000000E-06

*** THE RECTANGULAR MATRIX A ****

 0.114268780E+00/ 0.114269212E+00  0.115364492E+00/ 0.115364604E+00  0.364190638E+00/
0.364190936E+00
 0.650560141E+00/ 0.650560677E+00  0.622775733E+00/ 0.622776210E+00  0.360372066E-01/
0.360372625E-01
 0.816157699E+00/ 0.816157877E+00  0.921398044E+00/ 0.921398103E+00  0.207748771E+00/
0.207748771E+00
 0.387930334E+00/ 0.387931287E+00  0.516173363E+00/ 0.516173959E+00  0.868077636E+00/
0.868077874E+00
 0.937119424E+00/ 0.937120318E+00  0.887568533E+00/ 0.887569010E+00  0.658547282E+00/
0.658547461E+00
 0.655510843E+00/ 0.655511320E+00  0.364622176E+00/ 0.364622265E+00  0.668373883E+00/
0.668374777E+00
 0.619106591E+00/ 0.619106650E+00  0.935491562E+00/ 0.935492039E+00  0.666888416E+00/
0.666888714E+00
 0.489297032E+00/ 0.489297301E+00  0.995499492E+00/ 0.995499671E+00  0.773458421E+00/
0.773458421E+00
 0.651532531E+00/ 0.651533246E+00  0.326811671E-01/ 0.326814242E-01  0.564910114E+00/
0.564910889E+00
 0.791495383E+00/ 0.791496158E+00  0.250831068E+00/ 0.250831723E+00  0.620449901E+00/
0.620450735E+00

*** THE RIGHT-HAND SIDE B ****

 0.137972653E+00/ 0.137973279E+00
 0.816119015E+00/ 0.816119552E+00
 0.562035322E+00/ 0.562035561E+00
 0.130449951E+00/ 0.130449995E+00
 0.291655064E+00/ 0.291655391E+00
 0.485803902E+00/ 0.485804886E+00
 0.414707184E+00/ 0.414707601E+00
 0.571130455E+00/ 0.571130872E+00
 0.705590487E+00/ 0.705590904E+00
 0.230652630E+00/ 0.230653584E+00

*** THE A-TRANSPOSE-A MATRIX ****

 0.423444080E+01/ 0.423444748E+01  0.372744107E+01/ 0.372744584E+01  0.327709937E+01/
0.327710509E+01
 0.372744107E+01/ 0.372744584E+01  0.436744356E+01/ 0.436744738E+01  0.310010171E+01/
0.310010457E+01
 0.327709937E+01/ 0.327710509E+01  0.310010171E+01/ 0.310010457E+01  0.355811977E+01/
0.355812407E+01
```

```
*** THE INVERSE OF A-TRANSPOSE-A ****

 0.131633759E+01/ 0.131638610E+01 -0.688993096E+00/-0.688959599E+00 -0.612125099E+00
0.612090528E+00
-0.688993037E+00/-0.688959658E+00  0.960689723E+00/ 0.960713029E+00 -0.202485830E+00
0.202462316E+00
-0.612125158E+00/-0.612090647E+00 -0.202485830E+00/-0.202462316E+00  0.102120888E+01
0.102123404E+01

*** THE SOLUTION VECTOR ****

 0.663610220E+00/ 0.663917542E+00
 0.111516654E+00/ 0.111727744E+00
-0.122313738E+00/-0.122095108E+00
```

```
C
C     ** NAME OF THE PROGRAM **
C        INT(ERVAL ARITHMETIC USAGE IN A LEAST SQUARES METHOD WITH)
C        ORT(HOGONALIZATION OF GRAM-SCHMIDT) - INTORT
C
C     ** NAME OF THE PROGRAMMER **
C        KONDAPALLI RAMA RAO
C        ROGER A. BROUCKE
C
C     ** AIM **
C        TO STUDY THE PERFORMANCE OF A LEAST SQUARES METHOD WITH THE
C        GRAM-SCHMIDT ORTHOGONALIZATION USING NUMERICAL CALCULATIONS
C        IN INTERVAL ARITHMETIC

      REAL A(2,10,3),B(2,10),X(2,3),R(2,10),C(2,3,3),D(2,3)
      OPEN (1,FILE='LEAST2.DAT',STATUS='NEW')
      WRITE(1,'(1H ,9A1)') CHAR(15),CHAR(27),CHAR(48)
      WRITE(1,'(/,A)')'**** INTERVAL ARITHMETIC - ORTHOGONALIZATION ****'
      WRITE (6,*) 'INPUT EPSLON'
      READ(5,*) EPS
      WRITE (1,*)
      WRITE (1,*) 'EPSLON:', EPS
      M = 10
      N = 3
      IRAN = 1+1.0E+08
      DO 10 I = 1,M
         B(1,I) = RAN(IRAN)
         B(2,I) = B(1,I)+EPS*RAN(IRAN)
         DO 5 J = 1,N
            A(1,I,J) = RAN(IRAN)
            A(2,I,J) = A(1,I,J)+EPS*RAN(IRAN)
    5    CONTINUE
   10 CONTINUE
      WRITE (1,'(/,A)') '***** THE RECTANGULAR MATRIX A *****'
      CALL PRM1(A,M,N)
      WRITE (1,'(/,A)') '***** THE RIGHT-HAND SIDE B *****'
      CALL PRM1(B,M,1)
      CALL ORTHOG(A,B,X,R,C,D,M,N)
      WRITE (1,'(/,A)') '***** THE SOLUTION VECTOR *****'
      CALL PRM1(X,N,1)
      CLOSE(1)
      STOP
      END
      SUBROUTINE PRM1(A,M,N)
      REAL A(2,M,N)
   10 FORMAT (1H,4(E16.9,1H/,E16.9,1X))
      WRITE (1,10)
      DO 2 I = 1,M
         WRITE (1,10) (A(1,I,J),A(2,I,J),J=1,N)
    2 CONTINUE
      RETURN
      END
      SUBROUTINE ORTHOG(A,V,X,R,C,D,NR,NC)
      REAL A(2,NR,NC),V(2,NR),X(2,NC),R(2,NR),C(2,NC,NC),D(2,NC),
     1     A1(2),A2(2),C1(2),C2(2),A12(2),U(2),V1(2),D1(2),H(2),
     2     X1(2),XC(2)
      DO 14 K = 1,NC
         IF (K.EQ.1) GO TO 25
         DO 24 J = 1,K-1
            C(1,J,K) = 0.0
            C(2,J,K) = 0.0
            DO 34 I = 1,NR
               A1(1) = A(1,I,J)
               A1(2) = A(2,I,J)
               A2(1) = A(1,I,K)
               A2(2) = A(2,I,K)
               C1(1) = C(1,J,K)
               C1(2) = C(2,J,K)
               CALL INA(A1,A2,A12,3)
               CALL INA(C1,A12,C1,1)
               C(1,J,K) = C1(1)
               C(2,J,K) = C1(2)
   34       CONTINUE
```

```
            DO 44 IS = 1,NR
               A1(1) = A(1,IS,J)
               A1(2) = A(2,IS,J)
               C1(1) = C(1,J,K)
               C1(2) = C(2,J,K)
               A2(1) = A(1,IS,K)
               A2(2) = A(2,IS,K)
               CALL INA(C1,A1,C2,3)
               CALL INA(A2,C2,A2,2)
               A(1,IS,K) = A2(1)
               A(2,IS,K) = A2(2)
   44       CONTINUE
   24    CONTINUE
   25    U(1) = 0.0
         U(2) = 0.0
         DO 54 I = 1,NR
            A1(1) = A(1,I,K)
            A1(2) = A(2,I,K)
C           CALL INA(A1,A1,A2,3)
            A2(1) = A1(1)*A1(1)
            A2(2) = A1(2)*A1(2)
            CALL INA(U,A2,U,1)
   54    CONTINUE
         C1(1) = SQRT(U(1))
         C1(2) = SQRT(U(2))
         C(1,K,K) = C1(1)
         C(2,K,K) = C1(2)
C        CALL INA(U,U,C1,5)
         DO 64 IS = 1,NR
            A2(1) = A(1,IS,K)
            A2(2) = A(2,IS,K)
            C1(1) = C(1,K,K)
            C1(2) = C(2,K,K)
            CALL INA(A2,C1,A2,4)
            A(1,IS,K) = A2(1)
            A(2,IS,K) = A2(2)
   64    CONTINUE
         DO 74 J = K+1,NC
            C(1,J,K) = 0.0
            C(2,J,K) = 0.0
   74    CONTINUE
   14 CONTINUE
      WRITE (1,'(/,A)') '***** THE MODIFIED MATRIX A *****'
      CALL PRM1(A,NR,NC)
      WRITE (1,'(/,A)') '***** THE MATRIX R *****'
      CALL PRM1(C,NC,NC)
      DO 10 J = 1,NC

         D(1,J) = 0.0
         D(2,J) = 0.0
         DO 20 I = 1,NR
            A1(1) = A(1,I,J)
            A1(2) = A(2,I,J)
            V1(1) = V(1,I)
            V1(2) = V(2,I)
            D1(1) = D(1,J)
            D1(2) = D(2,J)
            CALL INA(A1,V1,A2,3)
            CALL INA(D1,A2,D1,2)
            D(1,J) = D1(1)
            D(2,J) = D1(2)
   20    CONTINUE
         DO 30 I = 1,NR
            D1(1) = D(1,J)
            D1(2) = D(2,J)
            A1(1) = A(1,I,J)
            A1(2) = A(2,I,J)
            V1(1) = V(1,I)
            V1(2) = V(2,I)
            CALL INA(D1,A1,A2,3)
            CALL INA(V1,A2,V1,1)
            V(1,I) = V1(1)
            V(2,I) = V1(2)
   30    CONTINUE
   10 CONTINUE
      DO 40 I = 1,NR
         R(1,I) = V(1,I)
         R(2,I) = V(2,I)
   40 CONTINUE
      DO 50 I = NC,1,-1
         H(1) = D(1,I)
         H(2) = D(2,I)
         DO 60 J = NC,I+1,-1
            C1(1) = C(1,I,J)
            C1(2) = C(2,I,J)
            X1(1) = X(1,J)
            X1(2) = X(2,J)
            CALL INA(X1,C1,XC,3)
            CALL INA(H,XC,H,1)
   60    CONTINUE
         C2(1) = C(1,I,I)
         C2(2) = C(2,I,I)
         CALL INA(H,C2,XC,4)
         X(1,I) = -XC(1)
         X(2,I) = -XC(2)
   50 CONTINUE
```

```
      RETURN
      END
      SUBROUTINE INA(A,B,C,N)
      REAL A(2),B(2),C(2)
      IF (N.LT.1.OR.N.GT.4) RETURN
      GO TO (1,2,3,4),N
    1 C(1) = A(1)+B(1)
      C(2) = A(2)+B(2)
      RETURN
    2 C(1) = A(1)-B(2)
      C(2) = A(2)-B(1)
      RETURN
    3 C(1) = AMIN1(A(1)*B(1),A(1)*B(2),A(2)*B(1),A(2)*B(2))
      C(2) = AMAX1(A(1)*B(1),A(1)*B(2),A(2)*B(1),A(2)*B(2))
      RETURN
    4 IF (B(1)*B(2).LE.0.0) GO TO 99
      C(1) = AMIN1(A(1)/B(1),A(1)/B(2),A(2)/B(1),A(2)/B(2))
      C(2) = AMAX1(A(1)/B(1),A(1)/B(2),A(2)/B(1),A(2)/B(2))
      RETURN
   99 N = 99
      WRITE (6,*) N
      RETURN
      END
```

```
*** INTERVAL ARITHMETIC - ORTHOGONALIZATION ****

EPSLON:  1.0000000E-06

**** THE RECTANGULAR MATRIX A *****

 0.114268780E+00/ 0.114269212E+00  0.115364492E+00/ 0.115364604E+00  0.364190638E+00/
0.364190936E+00
 0.650560141E+00/ 0.650560677E+00  0.622775733E+00/ 0.622776210E+00  0.360372066E-01/
0.360372625E-01
 0.816157699E+00/ 0.816157877E+00  0.921398044E+00/ 0.921398103E+00  0.207748771E+00/
0.207748771E+00
 0.387930334E+00/ 0.387931287E+00  0.516173363E+00/ 0.516173959E+00  0.868077636E+00/
0.868077874E+00
 0.937119424E+00/ 0.937120318E+00  0.887568533E+00/ 0.887569010E+00  0.658547282E+00/
0.658547461E+00
 0.655510843E+00/ 0.655511320E+00  0.364622176E+00/ 0.364622265E+00  0.668373883E+00/
0.668374777E+00
 0.619106591E+00/ 0.619106650E+00  0.935491562E+00/ 0.935492039E+00  0.666888416E+00/
0.666888714E+00
 0.489297032E+00/ 0.489297301E+00  0.995499492E+00/ 0.995499671E+00  0.773458421E+00/
0.773458421E+00
 0.651532531E+00/ 0.651533246E+00  0.326811671E-01/ 0.326814242E-01  0.564910114E+00/
0.564910889E+00
 0.791495383E+00/ 0.791496158E+00  0.250831068E+00/ 0.250831723E+00  0.620449901E+00/
0.620450735E+00

**** THE RIGHT-HAND SIDE B *****

 0.137972653E+00/ 0.137973279E+00
 0.816119015E+00/ 0.816119552E+00
 0.562035322E+00/ 0.562035561E+00
 0.130449951E+00/ 0.130449995E+00
 0.291655064E+00/ 0.291655391E+00
 0.485803902E+00/ 0.485804886E+00
 0.414707184E+00/ 0.414707601E+00
 0.571130455E+00/ 0.571130872E+00
 0.705590487E+00/ 0.705590904E+00
 0.230652630E+00/ 0.230653584E+00

**** THE MODIFIED MATRIX A *****

 0.555301942E-01/ 0.555304512E-01  0.141777787E-01/ 0.141785378E-01  0.275704771E+00/
0.275707453E+00
 0.316147000E+00/ 0.316147506E+00  0.480760261E-01/ 0.480785482E-01 -0.482418150E+00/
0.482412457E+00
 0.396620989E+00/ 0.396621406E+00  0.194731176E+00/ 0.194733486E+00 -0.469031513E+00/
0.469024032E+00
 0.188519090E+00/ 0.188519716E+00  0.167607144E+00/ 0.167609513E+00  0.538841248E+00/
0.538848519E+00
 0.455403686E-00/ 0.455404490E-00  0.601108037E-01/ 0.601143539E-01 -0.799639523E-01/
0.799588785E-01
 0.318552852E+00/ 0.318553329E+00 -0.203792423E+00/-0.203790307E+00  0.205317572E+00/
0.205324382E+00
 0.300861776E+00/ 0.300862074E+00  0.374678820E+00/ 0.374680966E+00  0.111487627E+00/
0.111495033E+00
 0.237779379E+00/ 0.237779707E+00  0.541888297E+00/ 0.541889966E+00  0.285785615E+00/
0.285795271E+00
 0.316619545E+00/ 0.316620141E+00 -0.518915772E+00/-0.518913329E+00  0.169676885E+00/
0.169687346E+00
 0.384636045E+00/ 0.384636730E+00 -0.427820146E+00/-0.427816898E+00  0.973173976E-01/
0.973270535E-01
```

```
 **** THE MATRIX R *****

 0.205777574E+01/ 0.205777740E+01  0.181139171E+01/ 0.181139553E+01  0.159254348E+01/
0.159254730E+01
 0.000000000E+00/ 0.000000000E+00  0.104225624E+01/ 0.104225612E+01  0.206638515E+00/
0.206650153E+00
 0.000000000E+00/ 0.000000000E+00  0.000000000E+00/ 0.000000000E+00  0.989552498E+00/
0.989557743E+00

**** THE SOLUTION VECTOR *****

 0.663782656E+00/ 0.663744509E+00
 0.111630671E+00/ 0.111613899E+00
-0.122191042E+00/-0.122217752E+00
```