GL-TR-90-0327

COORDINATE SYSTEM CONVERSION SOFTWARE USER'S GUIDE FOR ATTITUDE, QUATERNION, M50 AND RELATED APPLICATIONS

M. J. Kendra
N. A. Bonito

Radex, Inc.
Three Preston Court
Bedford, MA 01730

November 30, 1990

Scientific Report No. 5

GEOPHYSICS LABORATORY
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731-5000

"This technical report has been reviewed and is approved for publication"

EDWARD C. ROBINSON
Contract Manager
Data Systems Branch
Aerospace Engineering Division

ROBERT E. McINERNEY, Chief
Data Systems Branch
Aerospace Engineering Division

FOR THE COMMANDER

C. NEALON STARK, Director
Aerospace Engineering Division

This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify GL/IMA, Hanscom AFB, MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) RX-R-90111	5. MONITORING ORGANIZATION REPORT NUMBER(S) GL-TR-90-0327
6a. NAME OF PERFORMING ORGANIZATION RADEX, Inc. — 6b. OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION Geophysics Laboratory
6c. ADDRESS (*City, State, and ZIP Code*) Three Preston Court Bedford, MA 01730	7b. ADDRESS (*City, State, and ZIP Code*) Hanscom AFB Massachusetts 01731-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract F19628-89-C-0068
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62101F — PROJECT NO. 7659 — TASK NO. 05 — WORK UNIT ACCESSION NO. AB

11. TITLE (*Include Security Classification*)
Coordinate System Conversion Software User's Guide for Attitude, Quaternion, M50 and Related Applications

12. PERSONAL AUTHOR(S)
M. Kendra, N. Bonito

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Scientific Report #5	FROM 6/89 TO 11/90	1990, November 30	26

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Quaternions, transformations, attitude, PATH, rotations, matrices, M50, true-of-date, coordinate system conversion software, CSCS

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

The Coordinate System Conversion Software is a collection of general purpose subroutines and functions which may be used to transform quantities from one coordinate system to another. The report describes these routines and gives several simple examples of their use. A section is devoted to derivations using NASA quaternion operations. These differ from what is used in other literature.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL E. C. Robinson	22b. TELEPHONE (*Include Area Code*) (617)377-3840 — 22c. OFFICE SYMBOL GL/LCY

TABLE OF CONTENTS

LIST OF TABLES

1. INTRODUCTION

The Coordinate Systems Conversion Software (CSCS) is a collection of general purpose subroutines and functions which may be used to transform vector quantities from one coordinate system to another. It is intended for use by persons with a good working knowledge of the coordinate systems and parameters of interest. The routines are low level in the sense that a single call is only a single step in the conversion process. By combining several steps, however, the user will find the CSCS to be a powerful, general purpose tool which is capable of solving a wide variety of problems.

The CSCS collection contains routines which fall into five categories: quaternion conversion, rotation matrix construction, attitude manipulation, M50 to Aries true-of-date conversion, and vector manipulation. Some routines may fall entirely within a single category. Others span multiple categories, thus forming a bridge between them. By using such bridges the user may, for example, convert a quaternion into a rotation matrix, which is then used to manipulate a vector. The following sections describe the routines in the collection and give several example programs to demonstrate their application.

Attitude sequences, quaternions, and rotation matrices are used by the space community to store and transmit coordinate system conversion information. Each method has its own distinct advantages and disadvantages, and the adoption of one method in preference to another takes such considerations into account. In theory, attitude sequences contain all of the conversion information in the form of three rotation angles. They are easy to understand in human terms, and translate easily into motions such as turning one's body, looking up, and tilting one's head. If the sequence of rotations is not known, however, their meaning is lost. In addition, they must be converted using trigonometric functions before they can be used for vector analysis. Quaternions store the conversion information as four unambiguous numbers. They may be converted to a form suitable for vector analysis somewhat more easily than attitude sequences. Thus they are favored in situations where speed and machine storage considerations dominate. They are not easy to understand in human terms and cannot be used directly in vector computations. Rotation matrices consist of a 3x3 array of nine numbers. They can be used directly in vector multiplication and thus are preferred for data analysis using computing machines. The CSCS is capable of working with data in any of these forms and converting it to any of the other forms to meet the needs of the user.

In selecting a reference ECI coordinate system the space community has commonly employed two different systems, the M50 and the Aries true-of-date. A particular mission will generally maintain orbital and ephemeris information in one reference system but not the other. The CSCS provides the ability to convert data to the other ECI system for any date and time selected by the user. The M2000 system is likely to become more prevalent in the future, and the M50 conversion procedures may easily be modified for application to M2000.

Finally, the CSCS contains vector manipulation routines. One routine allows the user to convert vectors from one coordinate system to another. A different routine gives the user the ability to reverse the process; that is, to align an axis with a vector and obtain the associated rotation matrix. With this capability the CSCS becomes a powerful tool which lets the user construct coordinate systems of interest together with their related inter-conversion.

The Coordinate System Conversion Software described in this document was developed for IBM PC compatible systems using Microsoft FORTRAN. All floating point variables and calculations are double precision to ensure high accuracy. Constants, such as pi, are defined by the operating environment whenever possible. ANSI FORTRAN-77 standards were adhered to so that the code may be transported to other environments with a minimum of conversion effort.

2. NOTES ON THE USE OF QUATERNIONS

Two references were used in the development of the software contained in this document. The NASA PATH documentation [*Cooper, et al.*, 1985] was referred to for all routines which use quaternions. Escobal [1965] was referred to for routines which involve attitude conversions and coordinate system rotations. The Appendices of Escobal were especially helpful in this respect. Escobal also discusses quaternions in great detail and relates them to these other areas. NASA and Escobal have defined the order of operations differently, however, and the unwary user will quickly fall victim to this unfortunate circumstance. The NASA definition, $V' = q\ V\ q^*$, works just as well as that of Escobal, $V' = q^*\ V\ q$, as long as one does not mix conventions.

In the development of the quaternion routines presented here the NASA convention was followed. Much of the material discussed by Escobal is still relevant, but the derivations look quite different in the end. If an Escobal convention quaternion is encountered, it is recommended that the user convert it to its conjugate before proceeding. This may be done with the QTPOSE routine, which will return a quaternion consistent with the NASA convention.

3. DERIVATIONS OF COORDINATE SYSTEM TRANSFORMATIONS

We introduce the quaternion, q, in terms of four parameters which are components of q as follows:

$$q = (q_0, q_1, q_2, q_3). \tag{1}$$

We also define the multiplication of two quaternions, $z = xy$ [*Hollinger*, 1989]

(2)

It is sometimes more convenient to think of quaternions as being comprised of scalar and vector components. With scalar component q_0 and vector components $q_1\mathrm{i}+q_2\mathrm{j}+q_3k$, we define the quaternion q [*Escobal*, 1965] as

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{3}$$

where i, j, and k are the hyperimaginary numbers satisfying the conditions

$$\begin{aligned} i^2 = j^2 = k^2 = -1 \\ ij = -ji = k \\ jk = -kj = i \\ ki = -ik = j. \end{aligned} \tag{4}$$

Using this definition, it is now possible to carry out quaternion multiplication in the same manner as polynomial multiplication. We must take care, however, to preserve the order of operations, since the hyperimaginary numbers clearly do not commute.

We now define the conjugate of a quaternion as

$$q^* = q_0 - q_1 i - q_2 j - q_3 k. \tag{5}$$

We note that a vector in three-dimensional space with components $q_1 i + q_2 j + q_3 k$ may be expressed as a quaternion with a scalar component of zero in quaternion space

$$q = 0 + q_1 i + q_2 j + q_3 k. \tag{6}$$

3.1 CONVERSION OF A QUATERNION INTO A ROTATION MATRIX

The transformation of a vector V to V' is accomplished by the quaternion multiplication [*Cooper, et al.*, 1985]

$$V' = q V q^*. \tag{7}$$

This is equivalent to multiplication of V by the rotation matrix R

$$V' = RV. \tag{8}$$

Carrying out the quaternion multiplication,

$$V' = (q_0 + q_1 i + q_2 j + q_3 k)\ (v_1 i + v_2 j + v_3 k)\ (q_0 - q_1 i - q_2 j - q_3 k) \tag{9}$$

we reduce this expression to

$$V' = \left[(q_0^2+q_1^2-q_2^2-q_3^2)v_1+2(q_1q_2-q_0q_3)v_2+2(q_0q_2+q_1q_3)v_3\right]i$$

$$+\left[2(q_0q_3+q_1q_2)v_1+(q_0^2-q_1^2+q_2^2-q_3^2)v_2+2(q_2q_3-q_0q_1)v_3\right]j \qquad (10)$$

$$+\left[2(q_1q_3-q_0q_2)v_1+2(q_0q_1+q_2q_3)v_2+(q_0^2-q_1^2-q_2^2+q_3^2)v_3\right]k.$$

The elements of R are easily identified as

$$R = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_0q_3+q_1q_2) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}. \qquad (11)$$

The transformation from the prime to the unprimed system is accomplished using the transpose of R,

$$V = R^{\mathrm{T}}V'. \qquad (12)$$

The quaternion equivalent is

$$V = q^* V' q. \qquad (13)$$

We see that taking the quaternion conjugate is the equivalent of matrix transposition, and use these terms interchangeably.

3.2 CONVERSION OF A ROTATION MATRIX TO A QUATERNION

Using the quaternion expressions for the elements of our rotation matrix derived above, we establish

$$\begin{aligned} R_{11}+R_{22}+R_{33}+1 &= 4q_0^2 \\ R_{11}-R_{22}-R_{33}+1 &= 4q_1^2 \\ -R_{11}+R_{22}-R_{33}+1 &= 4q_2^2 \\ -R_{11}-R_{22}+R_{33}+1 &= 4q_3^2 . \end{aligned} \qquad (14)$$

Adding these, we show

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1, \tag{15}$$

a useful identity.

The relationships above give the magnitude of the quaternion components, but not their signs. Changing the signs of all four quaternion components does not affect our rotation matrix, so we may make our first choice arbitrarily. If the square root of the expression for q_0 above is not zero, we select the positive root. We then solve for the remaining quaternion components by referring to our rotation matrix and seeing that

$$q_1 = \frac{R_{32} - R_{23}}{4 q_0}$$

$$q_2 = \frac{A_{13} - A_{31}}{4 q_0} \tag{16}$$

$$q_3 = \frac{A_{21} - A_{12}}{4 q_0}.$$

If q_0 is zero, we use the square root of the expression for q_1 above and proceed in a similar manner.

3.3 THE PRODUCT OF TWO QUATERNIONS

We are interested in finding the equivalent quaternion, q'', for the transformations q followed by q'. In terms of a vector, V, two successive applications of the quaternion transformation operator give

$$V'' = q'(q V q^*) q'^*. \tag{17}$$

It can be shown that the hyperimaginary numbers are associative, thus

$$V'' = (q' q) V(q^* q'^*). \tag{18}$$

It follows that

$$
\begin{aligned}
q' q &= (q_0' + q_1' i + q_2' j + q_3' k)(q_0 + q_1 i + q_2 j + q_3 k) \\
&= (q_0 q_0' - q_1 q_1' - q_2 q_2' - q_3 q_3') \\
&+ (q_0 q_1' + q_0' q_1 + q_2' q_3 - q_3' q_2)\, i \\
&+ (q_0 q_2' + q_0' q_2 - q_1' q_3 + q_3' q_1)\, j \\
&+ (q_0 q_3' + q_0' q_3 - q_1 q_2' + q_1' q_2)\, k
\end{aligned}
\tag{19}
$$

and

$$
\begin{aligned}
q^* q'^* &= (q_0 - q_1 i - q_2 j - q_3 k)(q_0' - q_1' i - q_2' j - q_3' k) \\
&= (q_0 q_0' - q_1 q_1' - q_2 q_2' - q_3 q_3') \\
&+ (-q_0 q_1' - q_0' q_1 - q_2' q_3 + q_3' q_2)\, i \\
&+ (-q_0 q_2' - q_0' q_2 + q_1' q_3 - q_3' q_1)\, j \\
&+ (-q_0 q_3' - q_0' q_3 + q_1 q_2' - q_1' q_2)\, k \quad .
\end{aligned}
\tag{20}
$$

From these, we identify the components of q'':

$$
\begin{aligned}
q_0'' &= q_0 q_0' - q_1 q_1' - q_2 q_2' - q_3 q_3' \\
q_1'' &= q_0 q_1' + q_1 q_0' - q_2 q_3' + q_3 q_2' \\
q_2'' &= q_0 q_2' + q_1 q_3' + q_2 q_0' - q_3 q_1' \\
q_3'' &= q_0 q_3' - q_1 q_2' + q_2 q_1' + q_3 q_0' \; .
\end{aligned}
\tag{21}
$$

We may now use q'' to operate on V directly

$$
V'' = q'' V q''^* . \tag{22}
$$

3.4 ROTATE AN AXIS TO BE COLLINEAR WITH A VECTOR

This problem is treated in a general way by Escobal [1965]. A single rotation about an arbitrary rotation axis is always sufficient to make one vector collinear with another. Defining a unit vector $\hat{e}$ along the rotation axis, and a rotation angle Φ,

$$R = \begin{bmatrix} \cos\Phi + e_1^2(1-\cos\Phi) & e_1 e_2(1-\cos\Phi) + e_3\sin\Phi & e_1 e_3(1-\cos\Phi) - e_2\sin\Phi \\ e_1 e_2(1-\cos\Phi) - e_3\sin\Phi & \cos\Phi + e_2^2(1-\cos\Phi) & e_2 e_3(1-\cos\Phi) + e_1\sin\Phi \\ e_1 e_3(1-\cos\Phi) + e_2\sin\Phi & e_2 e_3(1-\cos\Phi) - e_1\sin\Phi & \cos\Phi + e_3^2(1-\cos\Phi) \end{bmatrix}. \quad (23)$$

We simplify the problem by requiring that the first vector, U, be equal in magnitude to the second vector, V. Furthermore, we require U to point in the direction of the n axis. The angle Φ is the direction cosine between V and the n axis:

$$\Phi = \arccos\frac{v_n}{|V|}. \quad (24)$$

The transformation $V = RU$ gives three equations with three unknowns, since $u_n = |V|$ and the two other components of U are each zero.

For example, if we select U to point along the z axis (see illustration), we have

$$V = R\begin{bmatrix} 0 \\ 0 \\ |V| \end{bmatrix} = \begin{bmatrix} |V|\{e_1 e_3(1-\cos\Phi) - e_2\sin\Phi\} \\ |V|\{e_2 e_3(1-\cos\Phi) + e_1\sin\Phi\} \\ |V|\{\cos\Phi + e_3^2(1-\cos\Phi)\} \end{bmatrix}. \quad (25)$$

Solving for e_3

$$e_3 = \left\{\left(\frac{v_3}{|V|} - \cos\Phi\right)/(1-\cos\Phi)\right\} = 0 \quad (26)$$

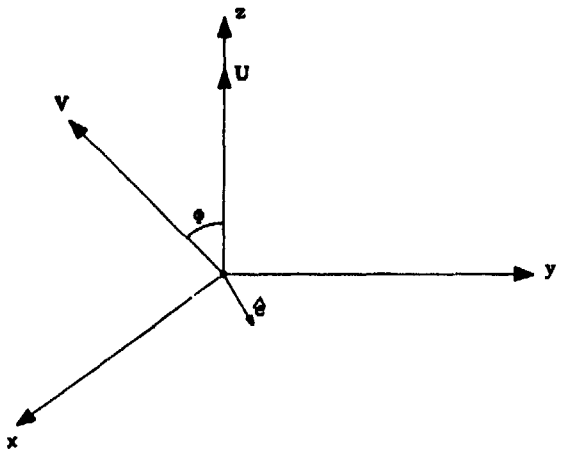
as expected, since the axis of rotation must lie in the xy plane. Solutions for the other components become

$$e_1 = \frac{v_2}{|V|\sin\Phi} \tag{27}$$

and

$$e_2 = \frac{-v_1}{|V|\sin\Phi}. \tag{28}$$

Calculating values for the elements of R is now a simple matter.



4. GENERAL DESCRIPTION

4.1 QUAT.FOR

AEQA — This function compares two attitude sequences. A value of .TRUE. is returned if they are equivalent and a value of .FALSE. if they are not. Acceptable error limits are specified in the logical function REQR. All angles are in radians. See description of subroutine ATOR for details of parameters.

ATOR — This subroutine converts rotation sequence about IAXIS(1), IAXIS(2), IAXIS(3) into the corresponding nine element rotation matrix. Allowed values are 1=roll=x-axis, 2=pitch=y-axis, and 3=yaw=z-axis. The same rotation axis cannot be used twice. All attitudes passed to this routine must be in radians. For example, IAXIS(1)=2, IAXIS(2)=3, IAXIS(3)=1 specifies a pitch, yaw, roll sequence of angles ATT(1), ATT(2), ATT(3) respectively. Returns the rotation matrix R.

QEQQ — This function compares two quaternions. A value of .TRUE. is returned if they are equivalent and a value of .FALSE. if they are not. Absolute errors on the order of 1E-6 are acceptable.

QMULTQ — This subroutine multiplies two quaternions, Q(3)=Q(2)Q(1). It is equivalent to two successive rotations, Q(1) followed by Q(2). Note that quaternion multiplication is the equivalent of matrix multiplication R(3)=R(2)R(1) for the same rotations.

QTOR — This subroutine converts a four element quaternion into a nine element rotation matrix for coordinate system transformations.

QTPOSE — This subroutine transposes the quaternion Q1 into Q2.

REQR — This function compares two rotation matrices. A value of .TRUE. is returned if they are equal and a value of .FALSE. if they are not. Absolute errors on the order of 1E-6 are acceptable.

RMULTR — This subroutine multiplies two rotation matrices, R3 = (R2)(R1). It is equivalent to two successive rotations, R1 followed by R2.

RMULTV — This subroutine multiplies a vector by a rotation matrix, V2 = (R)(V1).

QUAT.FOR (CONT.)

RTOA	This subroutine converts a nine element rotation matrix into the corresponding rotation sequence about IAXIS(1), IAXIS(2), IAXIS(3) where 1=roll=x-axis, 2=pitch=y-axis, and 3=yaw=z-axis. The same rotation axis cannot be used twice. Inputs are R and IAXIS. Returns the corresponding rotation sequence ATT(1) (0 to 2PI), ATT(2) (-PI to PI), ATT(3) (0 to 2PI) in radians.
RTOQ	This subroutine converts a nine element rotation matrix into a four element quaternion.
RTPOSE	This subroutine transposes the rotation matrix R1 into matrix R2.
VTOR	This subroutine converts a three element vector into a nine element rotation matrix for coordinate system transformations. The NAXIS axis is subjected to a single rotation which points it in the direction of the vector. NAXIS values of 1, 2, and 3 correspond to the x, y, and z axes respectively. The resulting rotation matrix R is returned. R may then be used to transform to the new system: V' = (R)(V).

4.2 M50TOT.FOR

ASTRO	Calculates M50 to true-of-date rotation matrix by constructing the appropriate precession and nutation rotation matrices, then performing matrix multiplication.
GMPRD	Multiplies matrices.
M50TOT	Subroutine to build a rotation matrix for M50 to true-of-date coordinate conversion. Input four digit year, month, day, and GMT (sec). Returns the 3x3 rotation matrix called RESULT.
MJDT	Calculates a modified Julian day number.
NUTATE	Builds the nutation rotation matrix.
PRECSS	Builds the precession rotation matrix.
RMAT	Calculates trigonometry values, then builds rotation matrix.
RMFU	Builds rotation matrix.
SETCON	Initializes the math constants used by the M50 converter.

5. EXAMPLES

5.1 Example 1.

Convert a yaw, pitch, roll sequence to the corresponding pitch, yaw, roll sequence. Show that the two sequences are equivalent.

This example is illustrated in the program ATT. Rotation angles are given values in a data statement and are stored as ATT1. IAXIS1 holds the rotation sequence associated with ATT1. The values 3,2,1 specify the z,y,x axes respectively. This corresponds to a yaw, pitch, roll sequence. Similarly, IAXIS2 holds the rotation sequence associated with ATT2. Values of 2,3,1 correspond to a pitch, yaw, roll sequence.

After changing the ATT1 angles from degrees to radians we are ready to begin the conversion. The attitude sequence defined by ATT1 and IAXIS1 is converted to a rotation matrix using a call to ATOR. The resulting matrix, TEMP, is then converted back to the rotation sequence specified by IAXIS2 using the call to RTOA. The values of interest are returned in ATT2.

To show that these two attitude sequences are equivalent, we compare them using the logical function AEQA. A value of true or false is returned and stored as RESULT. Although the use of AEQA is somewhat trivial in this example program, knowing whether or not two attitude sequences are equivalent is a common need and so it is demonstrated here.

The rotation sequences and their associated values are displayed after changing back from radians to degrees. The user should note that angles and axes are always given in the order in which they occur. It is the values of IAXIS which contain the information concerning the rotation sequence.

Finally, the equality of the two sequences is displayed using RESULT. A sample output follows the program listing.

STS MSID's of interest are listed in Table 1.

```
      PROGRAM ATT
C-----------------------------------------------------------------------
C     THIS PROGRAM CONVERTS A YAW, PITCH, ROLL SEQUENCE TO A PITCH, YAW, ROLL
C     SEQUENCE.  IT THEN SHOWS THAT THE TWO SEQUENCES ARE EQUIVALENT.
C-----------------------------------------------------------------------
C
      IMPLICIT  REAL*8(A-H,O-Z)
      LOGICAL   RESULT, AEQA
      DIMENSION ATT1(3), IAXIS1(3), ATT2(3), IAXIS2(3), TEMP(3,3)
      CHARACTER LEGEND(3)*8
      DATA LEGEND / 'ROLL  = ', 'PITCH = ', 'YAW   = ' /
C
C     IAXIS1 VALUES SPECIFY YAW, PITCH, ROLL SEQUENCE FOR ATT1.
C     IAXIS2 VALUES SPECIFY PITCH, YAW, ROLL SEQUENCE FOR ATT2.
C     ATT1 DATA IS SUPPLIED IN DATA STATEMENTS, BUT COULD EASILY BE READ FROM A
C     DATA BASE.
C
      DATA IAXIS1 / 3, 2, 1 /
      DATA ATT1   / 0.3582767D3, 0.2380823D0, 0.8965007D2 /
      DATA IAXIS2 / 2, 3, 1 /
C
C     CONVERT DEGREES TO RADIANS FIRST.
      TWOPI = 8.D0 * DATAN(1.D0)
      DTR = TWOPI / 360.D0
      RTD = 1.D0 / DTR
      DO 100 I=1,3
         ATT1(I) = ATT1(I) * DTR
  100 CONTINUE
C
C     CONVERT ATTITUDE SEQUENCE 1 TO A ROTATION MATRIX, THEN CONVERT THIS TO
C     ATTITUDE SEQUENCE 2.
C
      CALL ATOR(ATT1,IAXIS1,TEMP)
      CALL RTOA(TEMP,ATT2,IAXIS2)
      RESULT = AEQA(ATT1,IAXIS1,ATT2,IAXIS2)
C
      DO 110 I=1,3
         ATT1(I) = ATT1(I) * RTD
         ATT2(I) = ATT2(I) * RTD
  110 CONTINUE
C
C     DISPLAY RESULTS TO USER.
C
      WRITE(*,200) (LEGEND(IAXIS1(I)),ATT1(I),I=1,3),
     +             (LEGEND(IAXIS2(I)),ATT2(I),I=1,3)
      IF(RESULT) THEN
         WRITE(*,*) '     EQUAL'
      ELSE
         WRITE(*,*) '     NOT EQUAL'
      END IF
  200 FORMAT(/,/,2(1X,3(5X,A,F12.6),/,/),/)
      END
```

```
      YAW   =   358.276700     PITCH =      .238082     ROLL  =    89.650070

      PITCH =      .238190     YAW   =    -1.723285     ROLL  =    89.657233

      EQUAL
```

5.2 Example 2.

Convert M50 position and velocity in feet per second to Aries true-of-date position and velocity in kilometers per second.

This example is illustrated in the program CNVM50. In this example, the user is first asked to specify input and output file names. The time, position, and velocity are then read from the specified input file. A printout of the file selected for this example, named CNVM50.DAT, is included at the end of the program listing.

The time of day is converted to seconds, then a call is made to M50TOT. This subroutine uses the time information to calculate rotation matrix values. These values are returned as the 3x3 array M50TOD. The user may be interested to know that IMON, IDAY may be passed as either 1, day of year or month, day of month.

We use this rotation matrix to multiply the vector of interest using RMULTV. In this way the M50 position is converted to TOD position, then the M50 velocity to TOD velocity. These values are converted to kilometers and kilometers per second respectively, and displayed. A sample display is shown following the program and input file listings.

STS MSID's of interest are listed in Table 1.

```
      PROGRAM CNVM50
C-----------------------------------------------------------------------
C     THIS PROGRAM CONVERTS POSITION AND VELOCITY VECTORS FROM FT/SEC IN M50 TO
C     KM/SEC IN ARIES TRUE OF DATE COORDINATES.
C-----------------------------------------------------------------------
      IMPLICIT  REAL*8 (A-H,O-Z)
      REAL*8 M50TOD
      DIMENSION POSM50(3), VELM50(3), POSTOD(3), VELTOD(3), M50TOD(3,3)
      CHARACTER*12 IFILE, OFILE
C
C     FT/KM CONVERSION
C
      DATA FTKM / 3280.833D0 /
C
C     USER SELECTS INPUT AND OUTPUT FILE NAMES
C
      WRITE(*,*) ' ENTER INPUT FILE NAME '
      READ(*,'(A)') IFILE
      OPEN(UNIT=1,FILE=IFILE,STATUS='OLD',FORM='FORMATTED')
      WRITE(*,*) ' ENTER OUTPUT FILE NAME '
      READ(*,'(A)') OFILE
      OPEN(UNIT=3,FILE=OFILE,STATUS='NEW',FORM='FORMATTED')
C
C     READ TIME AND VECTOR INPUTS
C
      READ(1,*) IYR,IMON,IDAY,IHR,IMIN,SEC
      READ(1,*) (POSM50(I),I=1,3)
      READ(1,*) (VELM50(I),I=1,3)
C
C     CONVERT TIME TO SECONDS, THEN CONVERT VECTORS FROM M50 TO TRUE USING
C     ROTATION MATRIX.
C
      GMT=IHR*3600.D0+IMIN*60.D0+SEC
      CALL M50TOT(IYR,IMON,IDAY,GMT,M50TOD)
      CALL RMULTV(POSM50,M50TOD,POSTOD)
      CALL RMULTV(VELM50,M50TOD,VELTOD)
C
C     CONVERT TO KM
C
      DO 100 I=1,3
        POSTOD(I) = POSTOD(I) / FTKM
        VELTOD(I) = VELTOD(I) / FTKM
  100 CONTINUE
C
C     WRITE RESULTS
C
      WRITE(*,*) POSTOD,VELTOD
      WRITE(3,*) IYR,IMON,IDAY,IHR,IMIN,SEC
      WRITE(3,*) (POSTOD(I),I=1,3)
      WRITE(3,*) (VELTOD(I),I=1,3)
      CLOSE(UNIT=1)
      CLOSE(UNIT=3)
      END
```

```
           1990     1              306      8                    15
 0.000000000000000E+000
   750.624415039622700    5649.784942912215000    3398.785550422316000
    -5.275955101502905      -2.398301548788066       5.146722574414398
```

5.3 Example 3.

Use an M50 to body axis quaternion and body axis to UVW Euler angles, construct the M50 to UVW rotation matrix.

This example is illustrated in the program QM50. The quaternion values, Euler angles, and axis sequence information were taken from a NASA PATH tape. They appear as data statements at the beginning of the program.

The M50 to Body quaternion is converted to a rotation matrix using QTOR. Values are returned as the 3x3 array M50B. After changing the Euler angles from degrees to radians, the Body to UVW attitude sequence is converted to a rotation matrix using ATOR. The result is returned as the 3x3 array BUVW. These two rotation matrices are multiplied using RMULTR, which returns the 3x3 array M50UVW. This last rotation matrix is the result of two successive rotations, M50B followed by BUVW, and is the desired result.

These values are then displayed to the user. A sample display follows the program listing. The reader might be interested in comparing the values of the M50UVW array to those calculated using a different method and shown in the next example.

STS MSID's of interest are listed in Table 1.

```
      PROGRAM QM50
C-----------------------------------------------------------------------
C     THIS PROGRAM USES A QUATERNION TO CONSTRUCT AN M50 TO BODY AXIS ROTATION
      MATRIX.  IT THEN USES EULER ANGLES TO CONSTRUCT A BODY AXIS TO UVW ROTATION
      MATRIX.  THESE MATRICES ARE MULTIPLIED TO GIVE AN M50 TO UVW ROTATION
      MATRIX.
C-----------------------------------------------------------------------
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8    M50UVW, M50B
      DIMENSION QM50B(4), M50B(3,3), BUVW(3,3), M50UVW(3,3), IAXIS(3),
     +          ATT(3)
C
C     FOR THIS EXAMPLE, DATA IS SUPPLIED IN DATA STATEMENTS.  IT COULD EASILY BE
      READ IN FROM A DATA BASE INSTEAD.  SOURCE OF DATA IS NASA PATH TAPE
      51F-MISSION TDRS 41D WEST - SEG. NO. 14. IAXIS VALUES SPECIFY YAW, PITCH,
      ROLL SEQUENCE FOR ATT.
C
      DATA QM50B / 0.2599793D0, 0.5427552D-1, 0.3427433D0, -0.9011060 /
      DATA IAXIS / 3, 2, 1 /
      DATA ATT   / 0.3582767D3, 0.2380823D0, 0.8965007D2 /
C
C     CONVERT DEGREES TO RADIANS FIRST.
      TWOPI = 8.D0 * DATAN(1.D0)
      DTR = TWOPI / 360.D0
      DO 100 I=1,3
        ATT(I) = ATT(I) * DTR
  100 CONTINUE
C
      CALL QTOR(QM50B,M50B)
      CALL ATOR(ATT,IAXIS,BUVW)
      CALL RMULTR(M50B,BUVW,M50UVW)
C
C     DISPLAY RESULTS TO USER.
C
      WRITE(*,200) 'M50UVW = ',((M50UVW(I,J),J=1,3),I=1,3)
  200 FORMAT(/,/,1X,A,3(T15,3(5X,F10.6),/),/,/)
      END
```

```
M50UVW =        -.844416        .526901        .096629
                -.282325       -.591032        .755628
                 .455252        .610783        .647834
```

5.4 Example 4.

Use M50 position and velocity vectors to calculate the M50 to UVW rotation matrix. Use Aries true-of-date position and velocity vectors to calculate the True to UVW rotation matrix. Combine these to obtain the M50 to True rotation matrix, then check independently.

This example is illustrated in the program UVW. Values for M50 and True positions and velocities were taken from a NASA PATH tape. They appear as data statements at the beginning of the program.

The M50 to UVW rotation matrix is constructed from the M50 position and velocity vectors using VTOR. In the first call to VTOR, the M50 coordinate system is rotated to point the 1, or x, axis in the direction of the position vector. This rotation makes the x′ axis collinear with the U axis. The rotation matrix necessary for this operation is returned as the 3x3 array R1. The M50 velocity vector is then transformed to the new system using RMULTV, which returns the transformed velocity as TEMP. This transformed velocity is now projected onto the y′z′ plane by setting the x′ component to zero. Next the 2, or y′, axis is pointed in the direction of our transformed, projected velocity vector by using VTOR a second time. The rotation matrix for this transformation is returned as the 3x3 array R2. This second rotation can be described as a rotation about the x′ axis which results in a z″ velocity component of zero. These two rotations establish conditions sufficient to define the UVW system. The rotation matrices R1 and R2 are multiplied using RMULTR. The product, returned as the 3x3 array M50UVW, is the M50 to UVW rotation matrix of interest.

The true-of-date to UVW rotation matrix is constructed from True position and velocity vectors in a similar manner, and returned as the 3x3 array TODUVW. Since we are interested in going from UVW to True, RTPOSE is used to transpose the TODUVW array. The result is returned as the 3x3 array UVWTOD.

To complete the construction of the M50 to True rotation matrix, the rotation matrices M50UVW and UVWTOD are multiplied using RMULTR. The product is returned as the 3x3 array M50TOD.

Next, the M50 to True rotation matrix is constructed independently using the date and time associated with the position and velocity vectors. These values have been placed in a data statement at the beginning of the program. This information is converted to a rotation matrix using M50TOT, and the results are returned in the 3x3 array TEST.

Finally, we check to see if the two rotation matrices are equal using the logical function REQR. A value of true or false is returned and stored as RESULT. The values of interest are then displayed to the user. A sample display follows the program listing.

STS MSID's of interest are listed in Table 1.

```
      PROGRAM UVW
C-----------------------------------------------------------------------
C     PROGRAM TO CALCULATE M50 TO UVW AND ARIES TRUE-OF-DATE TO UVW ROTATION
C     MATRICES FROM POSITION AND VELOCITY VECTORS.  THE TWO MATRICES ARE THEN
C     MULTIPLIED TO GIVE A M50 TO TRUE CONVERSION MATRIX.  THIS IS CHECKED
C     AGAINST THE M50TOT SUBROUTINE VALUES.
C-----------------------------------------------------------------------
C
      IMPLICIT  REAL*8(A-H,O-Z)
      LOGICAL   REQR, RESULT
      REAL*8    M50UVW, M50TOD
      DIMENSION POSM50(3), VELM50(3), POSTOD(3), VELTOD(3), TEMP(3)
      DIMENSION R1(3,3), R2(3,3), M50UVW(3,3), TODUVW(3,3),
     +          UVWTOD(3,3), M50TOD(3,3), TEST(3,3)
C
C     FOR THIS EXAMPLE, DATA IS SUPPLIED IN DATA STATEMENTS.  IT COULD EASILY BE
C     READ FROM A DATA BASE INSTEAD.  SOURCE OF DATA IS NASA PATH TAPE
C     51F-MISSION TDRS 41D WEST - SEG. NO. 14.
C
      DATA IYR, IMON, IDAY, GMTSEC / 1985, 8, 1, 1001.86957D0 /
C
      DATA POSM50 / -0.5652093D4,  0.3526012D4, 0.6467874D3 /
      DATA VELM50 / -0.2178853D1, -0.4563907D1, 0.5834034D1 /
      DATA POSTOD / -0.5681994D4,  0.3481981D4, 0.6274173D3 /
      DATA VELTOD / -0.2162737D1, -0.4581256D1, 0.5826428D1 /
C
C     CONSTRUCT THE M50 TO UVW ROTATION MATRIX.
C
C     FIND THE ROTATION MATRIX WHICH POINTS THE X AXIS IN THE DIRECTION OF THE
C     M50 RADIUS VECTOR.  THIS TRANSFORMATION DEFINES THE U AXIS.
      CALL VTOR(POSM50,R1,1)
C     USE THIS MATRIX TO TRANSFORM THE VELOCITY VEL' = (R1) (VEL).
      CALL RMULTV(VELM50,R1,TEMP)
C     NOW PROJECT VEL' ONTO V,W PLANE BY SETTING THE U COMPONENT TO ZERO.  THEN
C     FIND ROTATION MATRIX WHICH POINTS THE V AXIS IN THE DIRECTION OF THIS
C     VELOCITY PROJECTION.  THIS IS EQUIVALENT TO A ROTATION ABOUT THE U AXIS
C     WHICH RESULTS IN NO W COMPONENT OF VELOCITY.
      TEMP(1)=0.D0
      CALL VTOR(TEMP,R2,2)
C     COMBINE THESE TWO ROTATIONS.
      CALL RMULTR(R1,R2,M50UVW)
C
C     CONSTRUCT THE TOD TO UVW ROTATION MATRIX IN A SIMILAR MANNER.
C
      CALL VTOR(POSTOD,R1,1)
      CALL RMULTV(VELTOD,R1,TEMP)
      TEMP(1)=0.D0
      CALL VTOR(TEMP,R2,2)
      CALL RMULTR(R1,R2,TODUVW)
C
C     NOW CONSTRUCT THE M50 TO TOD ROTATION MATRIX.
C
      CALL RTPOSE(TODUVW,UVWTOD)
      CALL RMULTR(M50UVW,UVWTOD,M50TOD)
C
C     FIND THIS VALUE INDEPENDENTLY USING SUBROUTINE M50TOT.
C
      CALL M50TOT(IYR,IMON,IDAY,GMTSEC,TEST)
C
C     TEST FOR EQUALITY.  ACCEPTABLE ERRORS OF 1D-6 ARE SPECIFIED IN LOGICAL
C     FUNCTION REQR.
C
```

```
      RESULT = REQR(TEST,M50TOD)
C
C     DISPLAY RESULTS TO USER.
C
      WRITE(*,100) 'M50UVW = ', ((M50UVW(I,J),J=1,3),I=1,3),
     +             'UVWTOD = ', ((UVWTOD(I,J),J=1,3),I=1,3),
     +             'M50TOD = ', ((M50TOD(I,J),J=1,3),I=1,3),
     +             'M50TOT = ', ((  TEST(I,J),J=1,3),I=1,3)
      IF(RESULT) THEN
        WRITE(*,*) ' M50TOD EQUALS M50TOT'
      ELSE
        WRITE(*,*) ' M50TOD DOES NOT EQUAL M50TOT'
      END IF
  100 FORMAT(/,/,4(1X,A,3(T15,3(5X,F10.6),/),/))
      END
```

```
M50UVW =          -.844416        .526901        .096629
                  -.282325       -.591032        .755628
                   .455252        .610783        .647834

UVWTOD =          -.848883       -.280239        .448179
                   .520204       -.593280        .614335
                   .093735        .754643        .649406

M50TOD =           .999963       -.007907       -.003437
                   .007907        .999969       -.000045
                   .003438        .000018        .999994

M50TOT =           .999963       -.007907       -.003437
                   .007907        .999969       -.000045
                   .003438        .000018        .999994

 M50TOD EQUALS M50TOT
```

Table 1. STS Measurement/Stimuli Identification for Atlantis

ID#	Description	Units
V90W2310C	M50 TO BODY QUAT TIME	S
V90U2240C	M50-TO-MEASURED BODY QUAT ELE 1	ND
V90U2241C	M50-TO-MEASURED BODY QUAT ELE 2	ND
V90U2242C	M50-TO-MEASURED BODY QUAT ELE 3	ND
V90U2243C	M50-TO-MEASURED BODY QUAT ELE 4	ND
V90U2641C	M50 WRT LVLH QUAT 1	ND
V90U2642C	M50 WRT LVLH QUAT 2	ND
V90U2643C	M50 WRT LVLH QUAT 3	ND
V90U2644C	M50 WRT LVLH QUAT 4	ND
V97U2218C	ADI REF QUAT ELEM 1	ND
V97U2219C	ADI REF QUAT ELEM 2	ND
V97U2220C	ADI REF QUAT ELEM 3	ND
V97U2221C	ADI REF QUAT ELEM 4	ND
V98H1248C	BODY WRT INERTIAL Q WD 1	ND
V98H1249C	BODY WRT INERTIAL Q WD 2	ND
V98H1250C	BODY WRT INERTIAL Q WD 3	ND
V98H1251C	BODY WRT INERTIAL Q WD 4	ND
V90H2141C	BODY ATTITUDE ERROR-PITCH	DEG
V90H2142C	BODY ATTITUDE ERROR-YAW	DEG
V90H2143C	BODY ATTITUDE ERROR-ROLL	DEG
V90H2202C	BODY ROLL ATTITUDE EULER ANGLE	DEG
V90H2217C	BODY PITCH ATTITUDE EULER ANGLE	DEG
V90H2230C	BODY YAW ATTITUDE EULER ANGLE	DEG
V92H3333C	POR P ATT ORB STR	DEG
V92H3334C	POR YAW ATT ORB STR	DEG
V92H3335C	POR R ATT ORB STR	DEG
V95H7467C	REQD INERTIAL ROLL ANGLE	RAD
V95H7468C	REQD INERTIAL PITCH ANGLE	RAD
V95H7473C	CURRENT INERTIAL ROLL ANGLE	RAD
V95H7474C	CURRENT INERTIAL PITCH ANGLE	RAD
V95H7475C	CURRENT INERTIAL YAW ANGLE	RAD
V95H7484C	REQD INERTIAL YAW ANGLE	RAD
V72R0916C	LH ADI ROLL RATE	DEG/S
V72R0917C	LH ADI PITCH RATE	DEG/S
V72R0918C	LH ADI YAW RATE	DEG/S
V72R1116C	AFT ADI ROLL RATE	DEG/S

Table 1. (cont'd)

ID#	Description	Units
V72R1117C	AFT ADI PITCH RATE	DEG/S
V72R1118C	AFT ADI YAW RATE	DEG/S
V90R2223C	IMU DERIVED BODY RATE X-AXIS	DEG/S
V90R2224C	IMU DERIVED BODY RATE Y-AXIS	DEG/S
V90R2225C	IMU DERIVED BODY RATE Z-AXIS	DEG/S
V92R3323C	ACT POR P ROT RATE	DEG/S
V92R3324C	ACT POR YAW ROT RATE	DEG/S
V92R3325C	ACT POR R ROT RATE	DEG/S
V95R7476C	IMU BODY RATE AROUND X-AXIS	DEG/S
V95R7477C	IMU BODY RATE AROUND Y-AXIS	DEG/S
V95R7487C	IMU BODY RATE AROUND Z-AXIS	DEG/S
V90H4277C	X-COMP OF FLTRS CURR POS VCTR-TLM	FT
V90H4278C	Y-COMP OF FLTRS CURR POS VCTR-TLM	FT
V90H4279C	Z-COMP OF FLTRS CURR POS VCTR-TLM	FT
V92H3417C	POR X-POSITION DISPLAY	IN
V92H3418C	POR Y-POSITION DISPLAY	IN
V92H3419C	POR Z-POSITION DISPLAY	IN
V95H0185C	X-COMP OF CURRENT SHUTTLE POS VCTR	FT
V95H0186C	Y-COMP OF CURRENT SHUTTLE POS VCTR	FT
V95H0187C	Z-COMP OF CURRENT SHUTTLE POS VCTR	FT
V95H0862C	X-COMP OF CURRENT TARGET POS VCTR	FT
V95H0863C	Y-COMP OF CURRENT TARGET POS VCTR	FT
V95H0864C	Z-COMP OF CURRENT TARGET POS VCTR	FT
V90L2557C	SEL TOTAL X VEL M50	FT/S
V90L2558C	SEL TOTAL Y VEL M50	FT/S
V90L2559C	SEL TOTAL Z VEL M50	FT/S
V92R3320C	ACT POR X TRANS RATE	FT/S
V92R3321C	ACT POR Y TRANS RATE	FT/S
V92R3322C	ACT POR Z TRANS RATE	FT/S
V95L0190C	X-COMP OF CURRENT SHUTTLE VEL VCTR	FT/S
V95L0191C	Y-COMP OF CURRENT SHUTTLE VEL VCTR	FT/S
V95L0192C	Z-COMP OF CURRENT SHUTTLE VEL VCTR	FT/S

6. REFERENCES

Cooper, D.H., Parker, K.C., and Torian, J.G., "OPS On-Orbit Postflight Attitude and Trajectory History (PATH) Product Description", JSC-18645, 1985.

Escobal, P.R., "Methods of Orbit Determination", John Wiley & Sons, New York, 1965.

Hollinger, H. B., Private Communication, 1989.