SOFA Vector/Matrix Library

PREFACE

The routines described here comprise the SOFA vector/matrix library. Their general appearance and coding style conforms to conventions agreed by the SOFA Review Board, and their functions, names and algorithms have been ratified by the Board. Procedures for soliciting and agreeing additions to the library are still evolving.

At present the routines are all written in Fortran 77, complying with the ANSI standard (X3.9-1978) except in two respects:

(1) All routine names are prefixed with the string "iau_". If necessary, the string can be removed globally; the result is correctly functioning code.

(2) All routines include an IMPLICIT NONE statement. This can be removed without affecting the behaviour of the code.

If the "iau_" string and/or the IMPLICIT NONE statements are removed globally, the resulting code is fully ANSI-compliant and is functionally unaffected.

GENERAL PRINCIPLES

The library consists mostly of routines which operate on ordinary Cartesian vectors (x,y,z) and 3x3 rotation matrices. However, there is also support for vectors which represent velocity as well as position and vectors which represent rotation instead of position. The vectors which represent both position and velocity may be considered still to have dimensions (3), but to comprise elements each of which is two numbers, representing the value itself and the time derivative. Thus:

* "Position" or "p" vectors (or just plain 3-vectors) have dimension (3) in Fortran and [3] in C.

* "Position/velocity" or "pv" vectors have dimensions (3,2) in Fortran and [2][3] in C.

* "Rotation" or "r" matrices have dimensions (3,3) in Fortran and [3][3] in C. When used for rotation, they are "orthogonal"; the inverse of such a matrix is equal to the transpose. Most of the routines in this library do not assume that r-matrices are necessarily orthogonal and in fact work on any 3x3 matrix.

* "Rotation" or "r" vectors have dimensions (3) in Fortran and [3] in C. Such vectors are a combination of the Euler axis and angle and are convertible to and from r-matrices. The direction is the axis of rotation and the magnitude is the angle of rotation, in radians. Because the amount of rotation can be scaled up and down simply by multiplying the vector by a scalar, r-vectors are useful for representing spins about an axis which is fixed.

* The above rules mean that in terms of memory address, the three velocity components of a pv-vector follow the three position components. Application code is permitted to exploit this and all other knowledge of the internal layouts: that x, y and z appear in that order and are in a right-handed Cartesian coordinate system etc. For example, the cp function (copy a p-vector) can be used to copy the velocity component of a pv-vector (indeed, this is how the CPV routine is coded).

* The routines provided do not completely fill the range of operations that link all the various vector and matrix options, but are confined to functions that are required by other parts of the SOFA software or which are likely to prove useful.

In addition to the vector/matrix routines, the library contains some routines related to spherical angles, including conversions to and from sexagesimal format.

Using the library requires knowledge of vector/matrix methods, spherical trigonometry, and methods of attitude representation. These topics are covered in many textbooks, including "Spacecraft Attitude Determination and Control", James R. Wertz (ed.), Astrophysics and Space Science Library, Vol. 73, D. Reidel Publishing Company, 1986.

OPERATIONS INVOLVING P-VECTORS AND R-MATRICES

Initialize

```
ZP        zero p-vector
ZR        initialize r-matrix to null
IR        initialize r-matrix to identity
```

Copy/extend/extract

```
CP        copy p-vector
CR        copy r-matrix
```

Build rotations

```
RX        rotate r-matrix about x
RY        rotate r-matrix about y
RZ        rotate r-matrix about z
```

Spherical/Cartesian conversions

```
S2C       spherical to unit vector
C2S       unit vector to spherical
S2P       spherical to p-vector
P2S       p-vector to spherical
```

Operations on vectors

```
PPP       p-vector plus p-vector
PMP       p-vector minus p-vector
PPSP      p-vector plus scaled p-vector
PDP       inner (=scalar=dot) product of two p-vectors
PXP       outer (=vector=cross) product of two p-vectors
PM        modulus of p-vector
PN        normalize p-vector returning modulus
SXP       multiply p-vector by scalar
```

Operations on matrices

```
RXR       r-matrix multiply
TR        transpose r-matrix
```

Matrix-vector products

```
RXP       product of r-matrix and p-vector
TRXP      product of transpose of r-matrix and p-vector
```

Separation and position-angle

```
SEPP      angular separation from p-vectors
SEPS      angular separation from spherical coordinates
PAP       position-angle from p-vectors
PAS       position-angle from spherical coordinates
```

Rotation vectors

```
RV2M      r-vector to r-matrix
RM2V      r-matrix to r-vector
```

OPERATIONS INVOLVING PV-VECTORS

```
  Initialize

     ZPV       zero pv-vector

  Copy/extend/extract

     CPV       copy pv-vector
     P2PV      append zero velocity to p-vector
     PV2P      discard velocity component of pv-vector

  Spherical/Cartesian conversions

     S2PV      spherical to pv-vector
     PV2S      pv-vector to spherical

  Operations on vectors

     PVPPV     pv-vector plus pv-vector
     PVMPV     pv-vector minus pv-vector
     PVDPV     inner (=scalar=dot) product of two pv-vectors
     PVXPV     outer (=vector=cross) product of two pv-vectors
     PVM       modulus of pv-vector
     SXPV      multiply pv-vector by scalar
     S2XPV     multiply pv-vector by two scalars
     PVU       update pv-vector
     PVUP      update pv-vector discarding velocity

  Matrix-vector products

     RXPV      product of r-matrix and pv-vector
     TRXPV     product of transpose of r-matrix and pv-vector


OPERATIONS ON ANGLES

     ANP       normalize radians to range 0 to 2pi
     ANPM      normalize radians to range -pi to +pi
     A2TF      decompose radians into hms
     A2AF      decompose radians into d ' "
     D2TF      decompose days into hms


CALLS

   SUBROUTINE        iau_A2AF   ( NDP, ANGLE, SIGN, IDMSF )
   SUBROUTINE        iau_A2TF   ( NDP, ANGLE, SIGN, IHMSF )
   DOUBLE PRECISION FUNCTION
                     iau_ANP    ( A )
   DOUBLE PRECISION FUNCTION
                     iau_ANPM   ( A )
   SUBROUTINE        iau_C2S    ( P, THETA, PHI )
   SUBROUTINE        iau_CP     ( P, C )
   SUBROUTINE        iau_CPV    ( PV, C )
   SUBROUTINE        iau_CR     ( R, C )
   SUBROUTINE        iau_D2TF   ( NDP, DAYS, SIGN, IHMSF )
   SUBROUTINE        iau_IR     ( R )
   SUBROUTINE        iau_P2PV   ( P, PV )
   SUBROUTINE        iau_P2S    ( P, THETA, PHI, R )
   SUBROUTINE        iau_PAP    ( A, B, THETA )
   SUBROUTINE        iau_PAS    ( AL, AP, BL, BP, THETA )
   SUBROUTINE        iau_PDP    ( A, B, ADB )
   SUBROUTINE        iau_PM     ( P, R )
   SUBROUTINE        iau_PMP    ( A, B, AMB )
   SUBROUTINE        iau_PN     ( P, R, U )
   SUBROUTINE        iau_PPP    ( A, B, APB )
   SUBROUTINE        iau_PPSP   ( A, S, B, APSB )
   SUBROUTINE        iau_PV2P   ( PV, P )
   SUBROUTINE        iau_PV2S   ( PV, THETA, PHI, R, TD, PD, RD )
   SUBROUTINE        iau_PVDPV  ( A, B, ADB )
   SUBROUTINE        iau_PVM    ( PV, R, S )
   SUBROUTINE        iau_PVMPV  ( A, B, AMB )
   SUBROUTINE        iau_PVPPV  ( A, B, APB )
   SUBROUTINE        iau_PVU    ( DT, PV, UPV )
```

```
SUBROUTINE        iau_PVUP   ( DT, PV, P )
SUBROUTINE        iau_PVXPV  ( A, B, AXB )
SUBROUTINE        iau_PXP    ( A, B, AXB )
SUBROUTINE        iau_RM2V   ( R, P )
SUBROUTINE        iau_RV2M   ( P, R )
SUBROUTINE        iau_RX     ( PHI, R )
SUBROUTINE        iau_RXP    ( R, P, RP )
SUBROUTINE        iau_RXPV   ( R, PV, RPV )
SUBROUTINE        iau_RXR    ( A, B, ATB )
SUBROUTINE        iau_RY     ( THETA, R )
SUBROUTINE        iau_RZ     ( PSI, R )
SUBROUTINE        iau_S2C    ( THETA, PHI, C )
SUBROUTINE        iau_S2P    ( THETA, PHI, R, P )
SUBROUTINE        iau_S2PV   ( THETA, PHI, R, TD, PD, RD, PV )
SUBROUTINE        iau_S2XPV  ( S1, S2, PV )
SUBROUTINE        iau_SEPP   ( A, B, S )
SUBROUTINE        iau_SEPS   ( AL, AP, BL, BP, S )
SUBROUTINE        iau_SXP    ( S, P, SP )
SUBROUTINE        iau_SXPV   ( S, PV, SPV )
SUBROUTINE        iau_TR     ( R, RT )
SUBROUTINE        iau_TRXP   ( R, P, TRP )
SUBROUTINE        iau_TRXPV  ( R, PV, TRPV )
SUBROUTINE        iau_ZP     ( P )
SUBROUTINE        iau_ZPV    ( PV )
SUBROUTINE        iau_ZR     ( R )
```