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.

PROGRAMMING LANGUAGES

The SOFA routines are available in two programming languages at present: Fortran 77 and ANSI C.

There is a one-to-one relationship between the two language versions. The naming convention is such that a SOFA routine referred to generically as "EXAMPL" exists as a Fortran subprogram iau_EXAMPL and a C function iauExampl. The calls for the two versions are very similar, with the same arguments in the same order. In a few cases, the C equivalent of a Fortran SUBROUTINE subprogram uses a return value rather than an argument.

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: FORTRAN VERSION

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

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

CALLS: C VERSION

```
        iauA2af  ( ndp, angle, &sign, idmsf );
        iauA2tf  ( ndp, angle, &sign, ihmsf );
   d =  iauAnp   ( a );
   d =  iauAnpm  ( a );
        iauC2s   ( p, &theta, &phi );
        iauCp    ( p, c );
        iauCpv   ( pv, c );
        iauCr    ( r, c );
        iauD2tf  ( ndp, days, &sign, ihmsf );
        iauIr    ( r );
        iauP2pv  ( p, pv );
        iauP2s   ( p, &theta, &phi, &r );
   d =  iauPap   ( a, b );
   d =  iauPas   ( al, ap, bl, bp );
   d =  iauPdp   ( a, b );
   d =  iauPm    ( p );
        iauPmp   ( a, b, amb );
        iauPn    ( p, &r, u );
        iauPpp   ( a, b, apb );
        iauPpsp  ( a, s, b, apsb );
        iauPv2p  ( pv, p );
        iauPv2s  ( pv, &theta, &phi, &r, &td, &pd, &rd );
        iauPvdpv ( a, b, adb );
        iauPvm   ( pv, &r, &s );
        iauPvmpv ( a, b, amb );
        iauPvppv ( a, b, apb );
        iauPvu   ( dt, pv, upv );
        iauPvup  ( dt, pv, p );
        iauPvxpv ( a, b, axb );
        iauPxp   ( a, b, axb );
        iauRm2v  ( r, p );
        iauRv2m  ( p, r );
        iauRx    ( phi, r );
        iauRxp   ( r, p, rp );
        iauRxpv  ( r, pv, rpv );
        iauRxr   ( a, b, atb );
        iauRy    ( theta, r );
        iauRz    ( psi, r );
        iauS2c   ( theta, phi, c );
        iauS2p   ( theta, phi, r, p );
        iauS2pv  ( theta, phi, r, td, pd, rd, pV );
        iauS2xpv ( s1, s2, pv );
    d = iauSepp  ( a, b );
    d = iauSeps  ( al, ap, bl, bp );
        iauSxp   ( s, p, sp );
        iauSxpv  ( s, pv, spv );
```

```
iauTr    ( r, rt );
iauTrxp  ( r, p, trp );
iauTrxpv ( r, pv, trpv );
iauZp    ( p );
iauZpv   ( pv );
iauZr    ( r );
```