PREFACE

The routines described here comprise the SOFA vector/matrix library. Their general appearance and coding style conforms to conventions agreed by the SOFA 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 [3][2] 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.


OPERATIONS INVOLVING P-VECTORS AND R-MATRICES

Initialize

\[
\begin{align*}
ZP & \quad \text{zero p-vector} \\
ZR & \quad \text{initialize r-matrix to null} \\
IR & \quad \text{initialize r-matrix to identity}
\end{align*}
\]

Copy/extend/extract

\[
\begin{align*}
CP & \quad \text{copy p-vector} \\
CR & \quad \text{copy r-matrix}
\end{align*}
\]

Build rotations

\[
\begin{align*}
RX & \quad \text{rotate r-matrix about x} \\
RY & \quad \text{rotate r-matrix about y} \\
RZ & \quad \text{rotate r-matrix about z}
\end{align*}
\]

Spherical/Cartesian conversions

\[
\begin{align*}
S2C & \quad \text{spherical to unit vector} \\
C2S & \quad \text{unit vector to spherical} \\
S2P & \quad \text{spherical to p-vector} \\
P2S & \quad \text{p-vector to spherical}
\end{align*}
\]

Operations on vectors

\[
\begin{align*}
PPP & \quad \text{p-vector plus p-vector} \\
PMP & \quad \text{p-vector minus p-vector} \\
PPSP & \quad \text{p-vector plus scaled p-vector} \\
PDP & \quad \text{inner (=scalar=dot) product of two p-vectors} \\
PXP & \quad \text{outer (=vector=cross) product of two p-vectors} \\
PM & \quad \text{modulus of p-vector} \\
PN & \quad \text{normalize p-vector returning modulus} \\
SXP & \quad \text{multiply p-vector by scalar}
\end{align*}
\]

Operations on matrices

\[
\begin{align*}
RXR & \quad \text{r-matrix multiply} \\
TR & \quad \text{transpose r-matrix}
\end{align*}
\]

Matrix-vector products

\[
\begin{align*}
RXP & \quad \text{product of r-matrix and p-vector} \\
TRXP & \quad \text{product of transpose of r-matrix and p-vector}
\end{align*}
\]

Separation and position-angle

\[
\begin{align*}
SEPP & \quad \text{angular separation from p-vectors} \\
SEPS & \quad \text{angular separation from spherical coordinates} \\
PAP & \quad \text{position-angle from p-vectors} \\
PAS & \quad \text{position-angle from spherical coordinates}
\end{align*}
\]

Rotation vectors

\[
\begin{align*}
RV2M & \quad \text{r-vector to r-matrix} \\
RM2V & \quad \text{r-matrix to r-vector}
\end{align*}
\]

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
PVXPV inner (=scalar=dot) 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 hours, minutes, seconds
A2AF decompose radians into degrees, arcminutes, arcseconds
AF2A degrees, arcminutes, arcseconds to radians
D2TF decompose days into hours, minutes, seconds
TF2A hours, minutes, seconds to radians
TF2D hours, minutes, seconds to days

CALLS: FORTRAN VERSION

CALL iau_A2AF  ( NDP, ANGLE, SIGN, IDMSF )
CALL iau_A2TF  ( NDP, ANGLE, SIGN, IHMSF )
CALL iau_AF2A  ( S, IDEG, IAMIN, ASEC, RAD, J )
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_TD2F2A ( NDTP, ANGLE, &SIGN, IDMSF );
CALL iauA2af ( ndp, angle, &sign, idmsf );
CALL iauA2tf ( ndp, angle, &sign, ihmsf );
i = iauAf2a ( s, ideg, iamin, asec, &rad );
d = iauAnp ( a );
d = iauAnpm ( a );
CALL iuC2s ( p, &theta, &phi );
CALL iuCp ( p, c );
CALL iuCpv ( pv, c );
CALL iuCr ( r, c );
CALL iuD2tf ( ndp, days, &sign, ihmsf );
CALL iuIr ( r );
CALL iuP2pv ( p, pv );
CALL iuP2s ( p, &theta, &phi, &r );
d = iauPap ( a, b );
d = iauPas ( al, ap, bl, bp );
d = iauPdp ( a, b );
d = iauPm ( p );
CALL iuPmp ( a, b, amb );
CALL iuPn ( p, &r, u );
CALL iuPpp ( a, b, apb );
CALL iuPsp ( a, s, b, apsb );
CALL iuPV2p ( pv, p );
CALL iuPV2s ( pv, &theta, &phi, &r, &td, &pd, &rd );
CALL iuPVdpv ( a, b, adb );
CALL iuPVm ( pv, &r, &s );
CALL iuPvpmp ( a, b, amb );
CALL iuPvppv ( a, b, apb );
CALL iuPvu ( dt, pv, upv );
CALL iuPVup ( dt, pv, p );
CALL iuPVxp ( a, b, axb );
CALL iuExp ( a, b, axb );
CALL iuRm2v ( r, p );
CALL iuRx2m ( p, r );
CALL iuRx  ( phi, r );
CALL iuRxp ( r, p, rp );
CALL iuRxpv ( r, pv, rpv );
CALL iuRxr ( a, b, atb );
CALL iuRy  ( theta, r );
CALL iuRz  ( psi, r );
CALL iuS2c ( theta, phi, c );
CALLS: C VERSION
iauS2p ( theta, phi, r, p );
iauS2pv ( theta, phi, r, td, pd, rd, pv );
iauS2xpv ( s1, s2, pv );
d = iauSepp ( a, b );
d = iauSepps ( al, ap, bl, bp );
iauSxp ( s, p, sp );
iauSxpv ( s, pv, spv );
i = iauTf2a ( s, ihour, imin, sec, &rad );
i = iauTf2d ( s, ihour, imin, sec, &days );
iauTr ( r, rt );
iauTrxp ( r, p, trp );
iauTrxpv ( r, pv, trpv );
iauZp ( p );
iauZpv ( pv );
iauZr ( r );