SUN/267.3

Starlink Project
Starlink User Note 267.3

Tim Jenness

2015 January 1

PAL — Positional Astronomy Library
0.9.0
Programmer's Manual

Abstract

PAL provides a subset of the Fortran SLALIB library but written in C using the SLALIB C API. Where possible the PAL routines are implemented using the C SOFA/ERFA library. It is provided with a GPLv3 license.

Contents

1 Introduction **1**

2 Citing PAL **1**

A Function Descriptions **2**
A.1 SOFA Mappings . . . 2
palAddet . . . 4
palAirmas . . . 5
palAltaz . . . 6
palAmp . . . 8
palAmpqk . . . 9
palAop . . . 10
palAoppa . . . 14
palAoppat . . . 17
palAopqk . . . 18
palAtmdsp . . . 21
palCaldj . . . 23
palDafin . . . 24
palDcmpf . . . 26
palDe2h . . . 28
palDeuler . . . 29
palDfltin . . . 30
palDh2e . . . 31
palDjcal . . . 32
palDmat . . . 33
palDs2tp . . . 34
palDat . . . 35
palDmoon . . . 36
palDrange . . . 37
palDt . . . 38
palDtp2s . . . 39
palDtps2c . . . 40
palDtt . . . 42
palEcleq . . . 43
palEcmat . . . 44
palEl2ue . . . 45
palEpco . . . 48
palEpv . . . 49
palEtrms . . . 50
palEqecl . . . 51
palEqgal . . . 52
palEvp . . . 53
palFitxy . . . 54
palFk45z . . . 56
palFk524 . . . 58
palFk54z . . . 60

palGaleq 62
palGalsup 63
palGe50 64
palGeoc 65
palIntin 66
palInvf 67
palMap 69
palMappa 71
palMapqk 72
palMapqkz 74
palNut 76
palNutc 77
palOap 78
palOapqk 82
palObs 84
palPa 86
palPcd 87
palPertel 88
palPertue 91
palPlanel 94
palPlanet 97
palPlante 98
palPlantu 102
palPm 104
palPolmo 105
palPrebn 107
palPrec 108
palPreces 109
palPrenut 110
palPv2el 111
palPv2ue 114
palPvobs 116
palPxy 117
palRanorm 119
palRdplan 120
palRefco 121
palRefro 123
palRefv 126
palRefz 128
palRverot 130
palRvgalc 131
palRvlg 132
palRvlsrd 133
palRvlsrk 134
palSubet 135
palSupgal 136
palUe2el 137
palUe2pv 140

palUnpcd . . . 142
palVers . . . 143
palXy2xy . . . 144
palCldj . . . 145
palDbear . . . 146
palDaf2r . . . 147
palDav2m . . . 148
palDcc2s . . . 149
palDcs2c . . . 150
palDd2tf . . . 151
palDimxv . . . 152
palDm2av . . . 153
palDjcl . . . 154
palDmxm . . . 155
palDmxv . . . 156
palDpav . . . 157
palDr2af . . . 158
palDr2tf . . . 159
palDranrm . . . 160
palDsep . . . 161
palDsepv . . . 162
palDtf2d . . . 163
palDtf2r . . . 164
palDvdv . . . 165
palDvn . . . 166
palDvxv . . . 167
palEpb . . . 168
palEpb2d . . . 169
palEpj . . . 170
palEpj2d . . . 171
palEqeqx . . . 172
palFk5hz . . . 173
palGmst . . . 174
palGmsta . . . 175
palHfk5z . . . 176
palRefcoq . . . 177

1 Introduction

This library provides a C library designed as a API-compatible replacement for the C SLALIB library (SUN/67) and uses a GPL licence so is freely redistributable. Where possible the functions call equivalent SOFA routines (Hohenkerk, C., 2011, Scholarpedia, **6**, *11404*)[1] and use current IAU 2006 standards. This means that any functions that rely on nutation or precession will return slightly different answers to the SLA functions.

2 Citing PAL

If you use PAL in your work please consider citing it. The description paper for PAL is: *PAL: A Positional Astronomy Library*, Jenness, T. & Berry, D. S., in *Astronomical Data Anaysis Software and Systems XXII*, Friedel, D. N. (ed), ASP Conf. Ser. **475**, p307.

[1] or equivalent ERFA routines.

A Function Descriptions

By default PAL is set up to use the ERFA variant of SOFA. ERFA is an approved redistribution of the SOFA code using a BSD-license and renamed function calls. Whereas SOFA routines have a `iau` prefix the ERFA equivalents have a `era` prefix. The PAL build script will try to detect which of ERFA and SOFA is available. Wherever SOFA is mentioned in this document the ERFA equivalent can be substituted.

ERFA can be obtained from https://github.com/liberfa/erfa

A.1 SOFA Mappings

The following table lists PAL/SLA functions that have direct replacements in SOFA. Whilst these routines are implemented in the PAL library using SOFA new code should probably call SOFA directly.

SLA/PAL	SOFA
`palCldj`	`iauCal2jd`
`palDbear`	`iauPas`
`palDaf2r`	`iauAf2a`
`palDav2m`	`iauRv2m`
`palDcc2s`	`iauC2s`
`palDcs2c`	`iauS2c`
`palDd2tf`	`iauD2tf`
`palDimxv`	`iauTrxp`
`palDm2av`	`iauRm2v`
`palDjcl`	`iauJd2cal`
`palDmxm`	`iauRxr`
`palDmxv`	`iauRxp`
`palDpav`	`iauPap`
`palDr2af`	`iauA2af`
`palDr2tf`	`iauA2tf`
`palDranrm`	`iauAnp`
`palDsep`	`iauSeps`
`palDsepv`	`iauSepp`
`palDtf2d`	`iauTf2d`
`palDtf2r`	`iauTf2a`
`palDvdv`	`iauPdp`
`palDvn`	`iauPn`
`palDvxv`	`iauPxp`
`palEpb`	`iauEpb`
`palEpb2d`	`iauEpb2d`
`palEpj`	`iauEpj`
`palEpj2d`	`iauEpj2jd`
`palEqeqx`	`iauEe06a`
`palFk5hz`	`iauFk5hz` *also calls iauEpj2jd*
`palGmst`	`iauGmst06`

`palGmsta` `iauGmst06`
`palHfk5z` `iauHfk5z` *also calls iauEpj2jd*
`palRefcoq` `iauRefco`

palAddet
Add the E-terms to a pre IAU 1976 mean place

Description:

Add the E-terms (elliptic component of annual aberration) to a pre IAU 1976 mean place to conform to the old catalogue convention.

Invocation:

```
void palAddet ( double rm, double dm, double eq, double *rc, double *dc );
```

Arguments:

rm = double (Given)
RA without E-terms (radians)

dm = double (Given)
Dec without E-terms (radians)

eq = double (Given)
Besselian epoch of mean equator and equinox

rc = double * (Returned)
RA with E-terms included (radians)

dc = double * (Returned)
Dec with E-terms included (radians)

Notes:

Most star positions from pre-1984 optical catalogues (or derived from astrometry using such stars) embody the E-terms. If it is necessary to convert a formal mean place (for example a pulsar timing position) to one consistent with such a star catalogue, then the RA,Dec should be adjusted using this routine.

See Also :

Explanatory Supplement to the Astronomical Ephemeris, section 2D, page 48.

palAirmas
Air mass at given zenith distance

Description:

Calculates the airmass at the observed zenith distance.

Invocation:

```
double palAirmas( double zd );
```

Arguments:

zd = double (Given)

Observed zenith distance (radians)

Notes:

- The " observed" zenith distance referred to above means " as affected by refraction" .
- Uses Hardie's (1962) polynomial fit to Bemporad's data for the relative air mass, X, in units of thickness at the zenith as tabulated by Schoenberg (1929). This is adequate for all normal needs as it is accurate to better than 0.1% up to X = 6.8 and better than 1% up to X = 10. Bemporad's tabulated values are unlikely to be trustworthy to such accuracy because of variations in density, pressure and other conditions in the atmosphere from those assumed in his work.
- The sign of the ZD is ignored.
- At zenith distances greater than about ZD = 87 degrees the air mass is held constant to avoid arithmetic overflows.

See Also :

- Hardie, R.H., 1962, in " Astronomical Techniques" ed. W.A. Hiltner, University of Chicago Press, p180.
- Schoenberg, E., 1929, Hdb. d. Ap., Berlin, Julius Springer, 2, 268.

palAltaz
Positions, velocities and accelerations for an altazimuth telescope mount

Description:

Positions, velocities and accelerations for an altazimuth telescope mount.

Invocation:

```
palAltaz ( double ha, double dec, double phi, double *az, double *azd, double
*azdd, double *el, double *eld, double *eldd, double *pa, double *pad, double
*padd );
```

Arguments:

ha = double (Given)
Hour angle (radians)

dec = double (Given)
Declination (radians)

phi = double (Given)
Observatory latitude (radians)

az = double * (Returned)
Azimuth (radians)

azd = double * (Returned)
Azimuth velocity (radians per radian of HA)

azdd = double * (Returned)
Azimuth acceleration (radians per radian of HA squared)

el = double * (Returned)
Elevation (radians)

eld = double * (Returned)
Elevation velocity (radians per radian of HA)

eldd = double * (Returned)
Elevation acceleration (radians per radian of HA squared)

pa = double * (Returned)
Parallactic angle (radians)

pad = double * (Returned)
Parallactic angle velocity (radians per radian of HA)

padd = double * (Returned)
Parallactic angle acceleration (radians per radian of HA squared)

Notes:

- Natural units are used throughout. HA, DEC, PHI, AZ, EL and ZD are in radians. The velocities and accelerations assume constant declination and constant rate of change of hour angle (as for tracking a star); the units of AZD, ELD and PAD are radians per radian of HA, while the units of AZDD, ELDD and PADD are radians per radian of HA squared. To convert into practical degree- and second-based units:

angles $*$ 360/2pi -> degrees velocities $*$ (2pi/86400)$*$(360/2pi) -> degree/sec accelerations $*$ ((2pi/86400)$**$2)$*$(360/2pi) -> degree/sec/sec

Note that the seconds here are sidereal rather than SI. One sidereal second is about 0.99727 SI seconds.

The velocity and acceleration factors assume the sidereal tracking case. Their respective numerical values are (exactly) 1/240 and (approximately) 1/3300236.9.

- Azimuth is returned in the range 0-2pi; north is zero, and east is +pi/2. Elevation and parallactic angle are returned in the range +/-pi. Parallactic angle is +ve for a star west of the meridian and is the angle NP-star-zenith.
- The latitude is geodetic as opposed to geocentric. The hour angle and declination are topocentric. Refraction and deficiencies in the telescope mounting are ignored. The purpose of the routine is to give the general form of the quantities. The details of a real telescope could profoundly change the results, especially close to the zenith.
- No range checking of arguments is carried out.
- In applications which involve many such calculations, rather than calling the present routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude, and (for tracking a star) sine and cosine of declination.

palAmp
Convert star RA,Dec from geocentric apparaent to mean place

Description:

Convert star RA,Dec from geocentric apparent to mean place. The mean coordinate system is close to ICRS. See palAmpqk for details.

Invocation:

```
void palAmp ( double ra, double da, double date, double eq, double *rm, double
*dm );
```

Arguments:

ra = double (Given)
Apparent RA (radians)

dec = double (Given)
Apparent Dec (radians)

date = double (Given)
TDB for apparent place (JD-2400000.5)

eq = double (Given)
Equinox: Julian epoch of mean place.

rm = double * (Returned)
Mean RA (radians)

dm = double * (Returned)
Mean Dec (radians)

Notes:

- See palMappa and palAmpqk for details.

palAmpqk
Convert star RA,Dec from geocentric apparent to mean place

Description:

Convert star RA,Dec from geocentric apparent to mean place. The " mean" coordinate system is in fact close to ICRS. Use of this function is appropriate when efficiency is important and where many star positions are all to be transformed for one epoch and equinox. The star-independent parameters can be obtained by calling the palMappa function.

Invocation:

```
void palAmpqk ( double ra, double da, double amprms[21], double *rm, double *dm
)
```

Arguments:

ra = double (Given)
Apparent RA (radians).

da = double (Given)
Apparent Dec (radians).

amprms = double[21] (Given)
Star-independent mean-to-apparent parameters (see palMappa): (0) time interval for proper motion (Julian years) (1-3) barycentric position of the Earth (AU) (4-6) heliocentric direction of the Earth (unit vector) (7) (grav rad Sun)*2/(Sun-Earth distance) (8-10) abv: barycentric Earth velocity in units of c (11) sqrt(1-v*v) where v=modulus(abv) (12-20) precession/nutation (3,3) matrix

rm = double (Returned)
Mean RA (radians).

dm = double (Returned)
Mean Dec (radians).

Note :

Iterative techniques are used for the aberration and light deflection corrections so that the routines palAmp (or palAmpqk) and palMap (or palMapqk) are accurate inverses; even at the edge of the Sun's disc the discrepancy is only about 1 nanoarcsecond.

palAop
Apparent to observed place

Description:

Apparent to observed place for sources distant from the solar system.

Invocation:

```
void palAop ( double rap, double dap, double date, double dut, double elongm,
double phim, double hm, double xp, double yp, double tdk, double pmb, double
rh, double wl, double tlr, double *aob, double *zob, double *hob, double *dob,
double *rob );
```

Arguments:

rap = double (Given)
Geocentric apparent right ascension

dap = double (Given)
Geocentirc apparent declination

date = double (Given)
UTC date/time (Modified Julian Date, JD-2400000.5)

dut = double (Given)
delta UT: UT1-UTC (UTC seconds)

elongm = double (Given)
Mean longitude of the observer (radians, east +ve)

phim = double (Given)
Mean geodetic latitude of the observer (radians)

hm = double (Given)
Observer's height above sea level (metres)

xp = double (Given)
Polar motion x-coordinates (radians)

yp = double (Given)
Polar motion y-coordinates (radians)

tdk = double (Given)
Local ambient temperature (K; std=273.15)

pmb = double (Given)
Local atmospheric pressure (mb; std=1013.25)

rh = double (Given)
Local relative humidity (in the range 0.0-1.0)

wl = double (Given)
Effective wavelength (micron, e.g. 0.55)

tlr = double (Given)
Tropospheric laps rate (K/metre, e.g. 0.0065)

aob = double * (Returned)
Observed azimuth (radians: N=0; E=90)

zob = double * (Returned)
Observed zenith distance (radians)

hob = double * (Returned)
Observed Hour Angle (radians)

dob = double * (Returned)
Observed Declination (radians)

rob = double * (Returned)
Observed Right Ascension (radians)

Notes:

- This routine returns zenith distance rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.
- The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted apparent RA,Dec should be within about 0.1 arcsec for a zenith distance of less than 70 degrees. Even at a topocentric zenith distance of 90 degrees, the accuracy in elevation should be better than 1 arcmin; useful results are available for a further 3 degrees, beyond which the palRefro routine returns a fixed value of the refraction. The complementary routines palAop (or palAopqk) and palOap (or palOapqk) are self-consistent to better than 1 micro- arcsecond all over the celestial sphere.
- It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
- " Apparent" place means the geocentric apparent right ascension and declination, which is obtained from a catalogue mean place by allowing for space motion, parallax, precession, nutation, annual aberration, and the Sun's gravitational lens effect. For star positions in the FK5 system (i.e. J2000), these effects can be applied by means of the palMap etc routines. Starting from other mean place systems, additional transformations will be needed; for example, FK4 (i.e. B1950) mean places would first have to be converted to FK5, which can be done with the palFk425 etc routines.
- " Observed" Az,El means the position that would be seen by a perfect theodolite located at the observer. This is obtained from the geocentric apparent RA,Dec by allowing for Earth orientation and diurnal aberration, rotating from equator to horizon coordinates, and then adjusting for refraction. The HA,Dec is obtained by rotating back into equatorial coordinates, using the geodetic latitude corrected for polar motion, and is the position that would be seen by a perfect equatorial located at the

observer and with its polar axis aligned to the Earth' s axis of rotation (n.b. not to the refracted pole). Finally, the RA is obtained by subtracting the HA from the local apparent ST.

- To predict the required setting of a real telescope, the observed place produced by this routine would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
- This routine takes time to execute, due mainly to the rigorous integration used to evaluate the refraction. For processing multiple stars for one location and time, call palAoppa once followed by one call per star to palAopqk. Where a range of times within a limited period of a few hours is involved, and the highest precision is not required, call palAoppa once, followed by a call to palAoppat each time the time changes, followed by one call per star to palAopqk.
- The DATE argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the delta UT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap delta UT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the delta UTC changes by one second to its post-leap new value.
- The delta UT (UT1-UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep delta UT within +/- 0.9 seconds.
- IMPORTANT – TAKE CARE WITH THE LONGITUDE SIGN CONVENTION. The longitude required by the present routine is east-positive, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the palObs routine are west-positive, following astronomical usage, and must be reversed in sign before use in the present routine.
- The polar coordinates XP,YP can be obtained from IERS circulars and equivalent publications. The maximum amplitude is about 0.3 arcseconds. If XP,YP values are unavailable, use XP=YP=0.0. See page B60 of the 1988 Astronomical Almanac for a definition of the two angles.
- The height above sea level of the observing station, HM, can be obtained from the Astronomical Almanac (Section J in the 1988 edition), or via the routine palObs. If P, the pressure in millibars, is available, an adequate estimate of HM can be obtained from the expression

HM $\sim$ -29.3$*$TSL$*$LOG(P/1013.25).

where TSL is the approximate sea-level air temperature in K (see Astrophysical Quantities, C.W.Allen, 3rd edition, section 52). Similarly, if the pressure P is not known, it can be estimated from the height of the observing station, HM, as follows:

P $\sim$ 1013.25$*$EXP(-HM/(29.3$*$TSL)).

Note, however, that the refraction is nearly proportional to the pressure and that an accurate P value is important for precise work.

- The azimuths etc produced by the present routine are with respect to the celestial pole. Corrections to the terrestrial pole can be computed using palPolmo.

palAoppa
Precompute apparent to observed place parameters

Description:

Precompute apparent to observed place parameters required by palAopqk and palOapqk.

Invocation:

```
void palAoppa ( double date, double dut, double elongm, double phim, double hm,
double xp, double yp, double tdk, double pmb, double rh, double wl, double tlr,
double aoprms[14] );
```

Arguments:

date = double (Given)
UTC date/time (modified Julian Date, JD-2400000.5)

dut = double (Given)
delta UT: UT1-UTC (UTC seconds)

elongm = double (Given)
mean longitude of the observer (radians, east +ve)

phim = double (Given)
mean geodetic latitude of the observer (radians)

hm = double (Given)
observer's height above sea level (metres)

xp = double (Given)
polar motion x-coordinate (radians)

yp = double (Given)
polar motion y-coordinate (radians)

tdk = double (Given)
local ambient temperature (K; std=273.15)

pmb = double (Given)
local atmospheric pressure (mb; std=1013.25)

rh = double (Given)
local relative humidity (in the range 0.0-1.0)

wl = double (Given)
effective wavelength (micron, e.g. 0.55)

tlr = double (Given)
tropospheric lapse rate (K/metre, e.g. 0.0065)

aoprms = double [14] (Returned)

Star-independent apparent-to-observed parameters

(0) geodetic latitude (radians) (1,2) sine and cosine of geodetic latitude (3) magnitude of diurnal aberration vector (4) height (hm) (5) ambient temperature (tdk) (6) pressure (pmb) (7) relative humidity (rh) (8) wavelength (wl) (9) lapse rate (tlr) (10,11) refraction constants A and B (radians) (12) longitude + eqn of equinoxes + sidereal DUT (radians) (13) local apparent sidereal time (radians)

Notes:

- It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
- The DATE argument is UTC expressed as an MJD. This is, strictly speaking, improper, because of leap seconds. However, as long as the delta UT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap delta UT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the delta UTC changes by one second to its post-leap new value.
- The delta UT (UT1-UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep delta UT within +/- 0.9 seconds.
- IMPORTANT – TAKE CARE WITH THE LONGITUDE SIGN CONVENTION. The longitude required by the present routine is east-positive, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the palObs routine are west-positive, following astronomical usage, and must be reversed in sign before use in the present routine.
- The polar coordinates XP,YP can be obtained from IERS circulars and equivalent publications. The maximum amplitude is about 0.3 arcseconds. If XP,YP values are unavailable, use XP=YP=0.0. See page B60 of the 1988 Astronomical Almanac for a definition of the two angles.
- The height above sea level of the observing station, HM, can be obtained from the Astronomical Almanac (Section J in the 1988 edition), or via the routine palObs. If P, the pressure in millibars, is available, an adequate estimate of HM can be obtained from the expression

HM $\sim$ -29.3$*$TSL$*$log(P/1013.25).

where TSL is the approximate sea-level air temperature in K (see Astrophysical Quantities, C.W.Allen, 3rd edition, section 52). Similarly, if the pressure P is not known, it can be estimated from the height of the observing station, HM, as follows:

P $\sim$ 1013.25$*$exp(-HM/(29.3$*$TSL)).

Note, however, that the refraction is nearly proportional to the pressure and that an accurate P value is important for precise work.

- Repeated, computationally-expensive, calls to palAoppa for times that are very close together can be avoided by calling palAoppa just once and then using palAoppat for the subsequent times. Fresh calls to palAoppa will be needed only when changes in the precession have grown to unacceptable levels or when anything affecting the refraction has changed.

palAoppat
Recompute sidereal time to support apparent to observed place

Description:

This routine recomputes the sidereal time in the apparent to observed place star-independent parameter block.

Invocation:

```
void palAoppat( double date, double aoprms[14] );
```

Arguments:

date = double (Given)

UTC date/time (modified Julian Date, JD-2400000.5) (see palAoppa description for comments on leap seconds)

aoprms = double[14] (Given & Returned)

Star-independent apparent-to-observed parameters. Updated by this routine. Requires element 12 to be the longitude + eqn of equinoxes + sidereal DUT and fills in element 13 with the local apparent sidereal time (in radians).

Notes:

- See palAoppa for more information.
- The star-independent parameters are not treated as an opaque struct in order to retain compatibility with SLA.

palAopqk
Quick apparent to observed place

Description:

Quick apparent to observed place.

Invocation:

```
void palAopqk ( double rap, double dap, const double aoprms[14], double *aob,
double *zob, double *hob, double *dob, double *rob );
```

Arguments:

rap = double (Given)

Geocentric apparent right ascension

dap = double (Given)

Geocentric apparent declination

aoprms = const double [14] (Given)

Star-independent apparent-to-observed parameters.

[0] geodetic latitude (radians) [1,2] sine and cosine of geodetic latitude [3] magnitude of diurnal aberration vector [4] height (HM) [5] ambient temperature (T) [6] pressure (P) [7] relative humidity (RH) [8] wavelength (WL) [9] lapse rate (TLR) [10,11] refraction constants A and B (radians) [12] longitude + eqn of equinoxes + sidereal DUT (radians) [13] local apparent sidereal time (radians)

aob = double * (Returned)

Observed azimuth (radians: N=0,E=90)

zob = double * (Returned)

Observed zenith distance (radians)

hob = double * (Returned)

Observed Hour Angle (radians)

dob = double * (Returned)

Observed Declination (radians)

rob = double * (Returned)

Observed Right Ascension (radians)

Notes:

- This routine returns zenith distance rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.

- The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the observed RA,Dec predicted by this routine should be within about 0.1 arcsec for a zenith distance of less than 70 degrees. Even at a topocentric zenith distance of 90 degrees, the accuracy in elevation should be better than 1 arcmin; useful results are available for a further 3 degrees, beyond which the palRefro routine returns a fixed value of the refraction. The complementary routines palAop (or palAopqk) and palOap (or palOapqk) are self-consistent to better than 1 micro- arcsecond all over the celestial sphere.
- It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
- " Apparent" place means the geocentric apparent right ascension and declination, which is obtained from a catalogue mean place by allowing for space motion, parallax, precession, nutation, annual aberration, and the Sun' s gravitational lens effect. For star positions in the FK5 system (i.e. J2000), these effects can be applied by means of the palMap etc routines. Starting from other mean place systems, additional transformations will be needed; for example, FK4 (i.e. B1950) mean places would first have to be converted to FK5, which can be done with the palFk425 etc routines.
- " Observed" Az,El means the position that would be seen by a perfect theodolite located at the observer. This is obtained from the geocentric apparent RA,Dec by allowing for Earth orientation and diurnal aberration, rotating from equator to horizon coordinates, and then adjusting for refraction. The HA,Dec is obtained by rotating back into equatorial coordinates, using the geodetic latitude corrected for polar motion, and is the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth' s axis of rotation (n.b. not to the refracted pole). Finally, the RA is obtained by subtracting the HA from the local apparent ST.
- To predict the required setting of a real telescope, the observed place produced by this routine would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
- The star-independent apparent-to-observed-place parameters in AOPRMS may be computed by means of the palAoppa routine. If nothing has changed significantly except the time, the palAoppat routine may be used to perform the requisite partial recomputation of AOPRMS.
- At zenith distances beyond about 76 degrees, the need for special care with the corrections for refraction causes a marked increase in execution time. Moreover, the effect gets worse with increasing zenith distance. Adroit programming in the calling application may allow the problem to be reduced. Prepare an alternative AOPRMS array, computed for zero air-pressure; this will disable the refraction corrections and cause rapid execution. Using this AOPRMS array, a preliminary call to the present routine will, depending on the application, produce a rough position which may be enough to establish whether the full, slow calculation (using the real AOPRMS array) is worthwhile. For example, there would be no need for the full calculation if the

preliminary call had already established that the source was well below the elevation limits for a particular telescope.

- The azimuths etc produced by the present routine are with respect to the celestial pole. Corrections to the terrestrial pole can be computed using palPolmo.

palAtmdsp
Apply atmospheric-dispersion adjustments to refraction coefficients

Description:

Apply atmospheric-dispersion adjustments to refraction coefficients.

Invocation:

```
void palAtmdsp( double tdk, double pmb, double rh, double wl1, double a1, double
b1, double wl2, double *a2, double *b2 );
```

Arguments:

tdk = double (Given)
Ambient temperature, K

pmb = double (Given)
Ambient pressure, millibars

rh = double (Given)
Ambient relative humidity, 0-1

wl1 = double (Given)
Reference wavelength, micrometre (0.4 recommended)

a1 = double (Given)
Refraction coefficient A for wavelength wl1 (radians)

b1 = double (Given)
Refraction coefficient B for wavelength wl1 (radians)

wl2 = double (Given)
Wavelength for which adjusted A,B required

a2 = double * (Returned)
Refraction coefficient A for wavelength WL2 (radians)

b2 = double * (Returned)
Refraction coefficient B for wavelength WL2 (radians)

Notes:

- To use this routine, first call palRefco specifying WL1 as the wavelength. This yields refraction coefficients A1,B1, correct for that wavelength. Subsequently, calls to palAtmdsp specifying different wavelengths will produce new, slightly adjusted refraction coefficients which apply to the specified wavelength.
- Most of the atmospheric dispersion happens between 0.7 micrometre and the UV atmospheric cutoff, and the effect increases strongly towards the UV end. For this reason a blue reference wavelength is recommended, for example 0.4 micrometres.

- The accuracy, for this set of conditions:

height above sea level 2000 m latitude 29 deg pressure 793 mb temperature 17 degC humidity 50% lapse rate 0.0065 degC/m reference wavelength 0.4 micrometre star elevation 15 deg

is about 2.5 mas RMS between 0.3 and 1.0 micrometres, and stays within 4 mas for the whole range longward of 0.3 micrometres (compared with a total dispersion from 0.3 to 20.0 micrometres of about 11 arcsec). These errors are typical for ordinary conditions and the given elevation; in extreme conditions values a few times this size may occur, while at higher elevations the errors become much smaller.

- If either wavelength exceeds 100 micrometres, the radio case is assumed and the returned refraction coefficients are the same as the given ones. Note that radio refraction coefficients cannot be turned into optical values using this routine, nor vice versa.
- The algorithm consists of calculation of the refractivity of the air at the observer for the two wavelengths, using the methods of the palRefro routine, and then scaling of the two refraction coefficients according to classical refraction theory. This amounts to scaling the A coefficient in proportion to (n-1) and the B coefficient almost in the same ratio (see R.M.Green, " Spherical Astronomy" , Cambridge University Press, 1985).

palCaldj
Gregorian Calendar to Modified Julian Date

Description:

Modified Julian Date to Gregorian Calendar with special behaviour for 2-digit years relating to 1950 to 2049.

Invocation:

```
void palCaldj ( int iy, int im, int id, double *djm, int *j );
```

Arguments:

iy = int (Given)
Year in the Gregorian calendar

im = int (Given)
Month in the Gergorian calendar

id = int (Given)
Day in the Gregorian calendar

djm = double * (Returned)
Modified Julian Date (JD-2400000.5) for 0 hrs

j = status (Returned)
0 = OK. See eraCal2jd for other values.

Notes:

- Uses eraCal2jd
- Unlike eraCal2jd this routine treats the years 0-100 as referring to the end of the 20th Century and beginning of the 21st Century. If this behaviour is not acceptable use the SOFA/ERFA routine directly or palCldj. Acceptable years are 00-49, interpreted as 2000-2049, 50-99, " " 1950-1999, all others, interpreted literally.
- Unlike SLA this routine will work with negative years.

palDafin
Sexagesimal character string to angle

Description:

Extracts an angle from a sexagesimal string with degrees, arcmin, arcsec fields using space or comma delimiters.

Invocation:

```
void palDafin ( const char *string, int *ipos, double *a, int *j );
```

Arguments:

string = const char * (Given)
String containing deg, arcmin, arcsec fields

ipos = int * (Given & Returned)
Position to start decoding " string" . First character is position 1 for compatibility with SLA. After calling this routine " iptr" will be positioned after the sexagesimal string.

a = double * (Returned)
Angle in radians.

j = int * (Returned)
status: 0 = OK +1 = default, A unchanged

- 1 = bad degrees)
- 2 = bad arcminutes) (note 3)
- 3 = bad arcseconds)

Notes:

- The first three " fields" in STRING are degrees, arcminutes, arcseconds, separated by spaces or commas. The degrees field may be signed, but not the others. The decoding is carried out by the palDfltin routine and is free-format.
- Successive fields may be absent, defaulting to zero. For zero status, the only combinations allowed are degrees alone, degrees and arcminutes, and all three fields present. If all three fields are omitted, a status of +1 is returned and A is unchanged. In all other cases A is changed.
- Range checking:

The degrees field is not range checked. However, it is expected to be integral unless the other two fields are absent.

The arcminutes field is expected to be 0-59, and integral if the arcseconds field is present. If the arcseconds field is absent, the arcminutes is expected to be 0-59.9999...

The arcseconds field is expected to be 0-59.9999...

- Decoding continues even when a check has failed. Under these circumstances the field takes the supplied value, defaulting to zero, and the result A is computed and returned.
- Further fields after the three expected ones are not treated as an error. The pointer IPOS is left in the correct state for further decoding with the present routine or with palDfltin etc. See the example, above.
- If STRING contains hours, minutes, seconds instead of degrees etc, or if the required units are turns (or days) instead of radians, the result A should be multiplied as follows:

for to obtain multiply STRING A in A by

d ' " radians 1 = 1.0 d ' " turns 1/2pi = 0.1591549430918953358 h m s radians 15 = 15.0 h m s days 15/2pi = 2.3873241463784300365

Example :

argument before after

STRING ' -57 17 44.806 12 34 56.7' unchanged IPTR 1 16 (points to 12...) A ? -1.00000D0 J ? 0

palDcmpf
Decompose an [x,y] linear fit into its constituent parameters: zero points, scales, nonperpendicularity and orientation

Description:

The model relates two sets of [x,y] coordinates as follows. Naming the elements of coeffs:

coeffs[0] = A
coeffs[1] = B
coeffs[2] = C
coeffs[3] = D
coeffs[4] = E
coeffs[5] = F

the model transforms coordinates [x1,y1] into coordinates [x2,y2] as follows:

x2 = A + B ∗ x1 + C ∗ y1
y2 = D + E ∗ x1 + F ∗ y1

The transformation can be decomposed into four steps:

1) Zero points:
x' = xz + x1
y' = yz + y1

2) Scales:
x'' = xs ∗ x'
y'' = ys ∗ y'

3) Nonperpendicularity:
x''' = cos(perp / 2) ∗ x'' + sin(perp / 2) ∗ y''
y''' = sin(perp / 2) ∗ x'' + cos(perp / 2) ∗ y''

4) Orientation:
x2 = cos(orient) ∗ x''' + sin(orient) ∗ y'''
y2 = -sin(orient) ∗ y''' + cos(orient) ∗ y'''

Invocation:

```
palDcmpf ( double coeffs[6], double *xz, double *yz, double *xs, double *ys,
double *perp, double *orient )
```

Arguments:

coeffs = double[6] (Given)
transformation coefficients (see note)

xz = double (Returned)
x zero point

yz = double (Returned)
y zero point

xs = double (Returned)
x scale

ys = double (Returned)
y scale

perp = double (Returned)
nonperpendicularity (radians)

orient = double (Returned)
orientation (radians)

See also :

palFitxy, palPxy, palInvf and palXy2xy

palDe2h
Equatorial to horizon coordinates: HA,Dec to Az,E

Description:
Convert equatorial to horizon coordinates.

Invocation:

```
palDe2h( double ha, double dec, double phi, double * az, double * el );
```

Arguments:

ha = double * (Given)
Hour angle (radians)

dec = double * (Given)
Declination (radians)

phi = double (Given)
Observatory latitude (radians)

az = double * (Returned)
Azimuth (radians)

el = double * (Returned)
Elevation (radians)

Notes:

- All the arguments are angles in radians.
- Azimuth is returned in the range 0-2pi; north is zero, and east is +pi/2. Elevation is returned in the range +/-pi/2.
- The latitude must be geodetic. In critical applications, corrections for polar motion should be applied.
- In some applications it will be important to specify the correct type of hour angle and declination in order to produce the required type of azimuth and elevation. In particular, it may be important to distinguish between elevation as affected by refraction, which would require the " observed" HA,Dec, and the elevation in vacuo, which would require the " topocentric" HA,Dec. If the effects of diurnal aberration can be neglected, the " apparent" HA,Dec may be used instead of the topocentric HA,Dec.
- No range checking of arguments is carried out.
- In applications which involve many such calculations, rather than calling the present routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude, and (for tracking a star) sine and cosine of declination.

palDeuler
Form a rotation matrix from the Euler angles

Description:

A rotation is positive when the reference frame rotates anticlockwise as seen looking towards the origin from the positive region of the specified axis.

The characters of ORDER define which axes the three successive rotations are about. A typical value is ' ZXZ' , indicating that RMAT is to become the direction cosine matrix corresponding to rotations of the reference frame through PHI radians about the old Z-axis, followed by THETA radians about the resulting X-axis, then PSI radians about the resulting Z-axis.

The axis names can be any of the following, in any order or combination: X, Y, Z, uppercase or lowercase, 1, 2, 3. Normal axis labelling/numbering conventions apply; the xyz (=123) triad is right-handed. Thus, the ' ZXZ' example given above could be written ' zxz' or ' 313' (or even ' ZxZ' or ' 3xZ'). ORDER is terminated by length or by the first unrecognized character.

Fewer than three rotations are acceptable, in which case the later angle arguments are ignored. If all rotations are zero, the identity matrix is produced.

Invocation:

```
void palDeuler ( const char *order, double phi, double theta, double psi, double
rmat[3][3] );
```

Arguments:

order = const char[] (Given)
Specifies about which axes the rotation occurs

phi = double (Given)
1st rotation (radians)

theta = double (Given)
2nd rotation (radians)

psi = double (Given)
3rd rotation (radians)

rmat = double[3][3] (Given & Returned)
Rotation matrix

palDfltin
Convert free-format input into double precision floating point

Description:

Extracts a number from an input string starting at the specified index.

Invocation:

```
void palDfltin( const char * string, int *nstrt, double *dreslt, int *jflag );
```

Arguments:

string = const char * (Given)
String containing number to be decoded.

nstrt = int * (Given and Returned)
Character number indicating where decoding should start. On output its value is updated to be the location of the possible next value. For compatibility with SLA the first character is index 1.

dreslt = double * (Returned)
Result. Not updated when jflag=1.

jflag = int * (Returned)
status: -1 = -OK, 0 = +OK, 1 = null, 2 = error

Notes:

- Uses the strtod() system call to do the parsing. This may lead to subtle differences when compared to the SLA/F parsing.
- All " D" characters are converted to " E" to handle fortran exponents.
- Commas are recognized as a special case and are skipped if one happens to be the next character when updating nstrt. Additionally the output nstrt position will skip past any trailing space.
- If no number can be found flag will be set to 1.
- If the number overflows or underflows jflag will be set to 2. For overflow the returned result will have the value HUGE_VAL, for underflow it will have the value 0.0.
- For compatiblity with SLA/F -0 will be returned as " 0" with jflag == -1.
- Unlike slaDfltin a standalone " E" will return status 1 (could not find a number) rather than 2 (bad number).

Implementation Status:

- The code is more robust if the C99 copysign() function is available. This can recognize the -0.0 values returned by strtod. If copysign() is missing we try to scan the string looking for minus signs.

palDh2e
Horizon to equatorial coordinates: Az,El to HA,Dec

Description:
Convert horizon to equatorial coordinates.

Invocation:

```
palDh2e( double az, double el, double phi, double * ha, double * dec );
```

Arguments:

az = double (Given)
Azimuth (radians)

el = double (Given)
Elevation (radians)

phi = double (Given)
Observatory latitude (radians)

ha = double * (Returned)
Hour angle (radians)

dec = double * (Returned)
Declination (radians)

Notes:

- All the arguments are angles in radians.
- The sign convention for azimuth is north zero, east +pi/2.
- HA is returned in the range +/-pi. Declination is returned in the range +/-pi/2.
- The latitude is (in principle) geodetic. In critical applications, corrections for polar motion should be applied.
- In some applications it will be important to specify the correct type of elevation in order to produce the required type of HA,Dec. In particular, it may be important to distinguish between the elevation as affected by refraction, which will yield the " observed" HA,Dec, and the elevation in vacuo, which will yield the " topocentric" HA,Dec. If the effects of diurnal aberration can be neglected, the topocentric HA,Dec may be used as an approximation to the " apparent" HA,Dec.
- No range checking of arguments is done.
- In applications which involve many such calculations, rather than calling the present routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude.

palDjcal
Modified Julian Date to Gregorian Calendar

Description:

Modified Julian Date to Gregorian Calendar, expressed in a form convenient for formatting messages (namely rounded to a specified precision, and with the fields stored in a single array)

Invocation:

```
void palDjcal ( int ndp, double djm, int iymdf[4], int *j );
```

Arguments:

ndp = int (Given)
Number of decimal places of days in fraction.

djm = double (Given)
Modified Julian Date (JD-2400000.5)

iymdf[4] = int[] (Returned)
Year, month, day, fraction in Gregorian calendar.

j = status (Returned)
0 = OK. See eraJd2cal for other values.

Notes:

- Uses eraJd2cal

palDmat
Matrix inversion & solution of simultaneous equations

Description:

Matrix inversion & solution of simultaneous equations For the set of n simultaneous equations in n unknowns: A.Y = X this routine calculates the inverse of A, the determinant of matrix A and the vector of N unknowns.

Invocation:

```
void palDmat( int n, double *a, double *y, double *d, int *jf, int *iw );
```

Arguments:

n = int (Given)

Number of simultaneous equations and number of unknowns.

a = double[] (Given & Returned)

A non-singular NxN matrix (implemented as a contiguous block of memory). After calling this routine " a" contains the inverse of the matrix.

y = double[] (Given & Returned)

On input the vector of N knowns. On exit this vector contains the N solutions.

d = double * (Returned)

The determinant.

jf = int * (Returned)

The singularity flag. If the matrix is non-singular, jf=0 is returned. If the matrix is singular, jf=-1 & d=0.0 are returned. In the latter case, the contents of array " a" on return are undefined.

iw = int[] (Given)

Integer workspace of size N.

Notes:

- Implemented using Gaussian elimination with partial pivoting.
- Optimized for speed rather than accuracy with errors 1 to 4 times those of routines optimized for accuracy.

palDs2tp
Spherical to tangent plane projection

Description:

Projection of spherical coordinates onto tangent plane: " gnomonic" projection - " standard coordinates"

Invocation:

```
palDs2tp( double ra, double dec, double raz, double decz, double *xi, double
*eta, int *j );
```

Arguments:

ra = double (Given)
RA spherical coordinate of point to be projected (radians)

dec = double (Given)
Dec spherical coordinate of point to be projected (radians)

raz = double (Given)
RA spherical coordinate of tangent point (radians)

decz = double (Given)
Dec spherical coordinate of tangent point (radians)

xi = double * (Returned)
First rectangular coordinate on tangent plane (radians)

eta = double * (Returned)
Second rectangular coordinate on tangent plane (radians)

j = int * (Returned)
status: 0 = OK, star on tangent plane 1 = error, star too far from axis 2 = error, antistar on tangent plane 3 = error, antistar too far from axis

palDat
Return offset between UTC and TAI

Description:

Increment to be applied to Coordinated Universal Time UTC to give International Atomic Time (TAI).

Invocation:

```
dat = palDat( double utc );
```

Arguments:

utc = double (Given)
UTC date as a modified JD (JD-2400000.5)

Returned Value:

dat = double
TAI-UTC in seconds

Notes:

- This routine converts the MJD argument to calendar date before calling the SOFA/ERFA eraDat function.
- This routine matches the slaDat interface which differs from the eraDat interface. Consider coding directly to the SOFA/ERFA interface.
- See eraDat for a description of error conditions when calling this function with a time outside of the UTC range.
- The status argument from eraDat is ignored. This is reasonable since the error codes are mainly related to incorrect calendar dates when calculating the JD internally.

palDmoon
Approximate geocentric position and velocity of the Moon

Description:

Calculate the approximate geocentric position of the Moon using a full implementation of the algorithm published by Meeus (l' Astronomie, June 1984, p348).

Invocation:

```
void palDmoon( double date, double pv[6] );
```

Arguments:

date = double (Given)
TDB as a Modified Julian Date (JD-2400000.5)

pv = double [6] (Returned)
Moon x,y,z,xdot,ydot,zdot, mean equator and equinox of date (AU, AU/s)

Notes:

- Meeus quotes accuracies of 10 arcsec in longitude, 3 arcsec in latitude and 0.2 arcsec in HP (equivalent to about 20 km in distance). Comparison with JPL DE200 over the interval 1960-2025 gives RMS errors of 3.7 arcsec and 83 mas/hour in longitude, 2.3 arcsec and 48 mas/hour in latitude, 11 km and 81 mm/s in distance. The maximum errors over the same interval are 18 arcsec and 0.50 arcsec/hour in longitude, 11 arcsec and 0.24 arcsec/hour in latitude, 40 km and 0.29 m/s in distance.
- The original algorithm is expressed in terms of the obsolete timescale Ephemeris Time. Either TDB or TT can be used, but not UT without incurring significant errors (30 arcsec at the present time) due to the Moon's 0.5 arcsec/sec movement.
- The algorithm is based on pre IAU 1976 standards. However, the result has been moved onto the new (FK5) equinox, an adjustment which is in any case much smaller than the intrinsic accuracy of the procedure.
- Velocity is obtained by a complete analytical differentiation of the Meeus model.

palDrange
Normalize angle into range +/- pi

Description:

The result is " angle" expressed in the range +/- pi. If the supplied value for " angle" is equal to +/- pi, it is returned unchanged.

Invocation:

```
palDrange( double angle )
```

Arguments:

angle = double (Given)

The angle in radians.

palDt
Estimate the offset between dynamical time and UT

Description:

Estimate the offset between dynamical time and Universal Time for a given historical epoch.

Invocation:

```
double palDt( double epoch );
```

Arguments:

epoch = double (Given)
Julian epoch (e.g. 1850.0)

Returned Value:

palDt = double
Rough estimate of ET-UT (after 1984, TT-UT) at the given epoch, in seconds.

Notes:

- Depending on the epoch, one of three parabolic approximations is used:

before 979 Stephenson & Morrison' s 390 BC to AD 948 model 979 to 1708 Stephenson & Morrison' s 948 to 1600 model after 1708 McCarthy & Babcock' s post-1650 model

The breakpoints are chosen to ensure continuity: they occur at places where the adjacent models give the same answer as each other.

- The accuracy is modest, with errors of up to 20 sec during the interval since 1650, rising to perhaps 30 min by 1000 BC. Comparatively accurate values from AD 1600 are tabulated in the Astronomical Almanac (see section K8 of the 1995 AA).
- The use of double-precision for both argument and result is purely for compatibility with other SLALIB time routines.
- The models used are based on a lunar tidal acceleration value of -26.00 arcsec per century.

See Also :

Explanatory Supplement to the Astronomical Almanac, ed P.K.Seidelmann, University Science Books (1992), section 2.553, p83. This contains references to the Stephenson & Morrison and McCarthy & Babcock papers.

palDtp2s
Tangent plane to spherical coordinates

Description:
Transform tangent plane coordinates into spherical.

Invocation:

```
palDtp2s( double xi, double eta, double raz, double decz, double *ra, double
*dec);
```

Arguments:

xi = double (Given)
First rectangular coordinate on tangent plane (radians)

eta = double (Given)
Second rectangular coordinate on tangent plane (radians)

raz = double (Given)
RA spherical coordinate of tangent point (radians)

decz = double (Given)
Dec spherical coordinate of tangent point (radians)

ra = double * (Returned)
RA spherical coordinate of point to be projected (radians)

dec = double * (Returned)
Dec spherical coordinate of point to be projected (radians)

palDtps2c
Determine RA,Dec of tangent point from coordinates

Description:

From the tangent plane coordinates of a star of known RA,Dec, determine the RA,Dec of the tangent point.

Invocation:

```
palDtps2c( double xi, double eta, double ra, double dec, double * raz1, double
decz1, double * raz2, double decz2, int *n);
```

Arguments:

xi = double (Given)
First rectangular coordinate on tangent plane (radians)

eta = double (Given)
Second rectangular coordinate on tangent plane (radians)

ra = double (Given)
RA spherical coordinate of star (radians)

dec = double (Given)
Dec spherical coordinate of star (radians)

raz1 = double * (Returned)
RA spherical coordinate of tangent point, solution 1 (radians)

decz1 = double * (Returned)
Dec spherical coordinate of tangent point, solution 1 (radians)

raz2 = double * (Returned)
RA spherical coordinate of tangent point, solution 2 (radians)

decz2 = double * (Returned)
Dec spherical coordinate of tangent point, solution 2 (radians)

n = int * (Returned)
number of solutions: 0 = no solutions returned (note 2) 1 = only the first solution is useful (note 3) 2 = both solutions are useful (note 3)

Notes:

- The RAZ1 and RAZ2 values are returned in the range 0-2pi.
- Cases where there is no solution can only arise near the poles. For example, it is clearly impossible for a star at the pole itself to have a non-zero XI value, and hence it is meaningless to ask where the tangent point would have to be to bring about this combination of XI and DEC.

- Also near the poles, cases can arise where there are two useful solutions. The argument N indicates whether the second of the two solutions returned is useful. N=1 indicates only one useful solution, the usual case; under these circumstances, the second solution corresponds to the " over-the-pole" case, and this is reflected in the values of RAZ2 and DECZ2 which are returned.
- The DECZ1 and DECZ2 values are returned in the range +/-pi, but in the usual, non-pole-crossing, case, the range is +/-pi/2.
- This routine is the spherical equivalent of the routine sla_DTPV2C.

palDtt
Return offset between UTC and TT

Description:

Increment to be applied to Coordinated Universal Time UTC to give Terrestrial Time TT (formerly Ephemeris Time ET)

Invocation:

```
dtt = palDtt( double utc );
```

Arguments:

utc = double (Given)
UTC date as a modified JD (JD-2400000.5)

Returned Value:

dtt = double
TT-UTC in seconds

Notes:

- Consider a comprehensive upgrade to use the time transformations in SOFA's time cookbook: http://www.iausofa.org/sofa_ts_c.pdf.
- See eraDat for a description of error conditions when calling this function with a time outside of the UTC range. This behaviour differs from slaDtt.

palEcleq
Transform from ecliptic coordinates to J2000.0 equatorial coordinates

Description:

Transform from ecliptic coordinate to J2000.0 equatorial coordinates.

Invocation:

```
void palEcleq ( double dl, double db, double date, double *dr, double *dd );
```

Arguments:

dl = double (Given)
Ecliptic longitude (mean of date, IAU 1980 theory, radians)

db = double (Given)
Ecliptic latitude (mean of date, IAU 1980 theory, radians)

date = double (Given)
TT as Modified Julian Date (JD-2400000.5). The difference between TT and TDB is of the order of a millisecond or two (i.e. about 0.02 arc-seconds).

dr = double * (Returned)
J2000.0 mean RA (radians)

dd = double * (Returned)
J2000.0 mean Dec (Radians)

palEcmat
Form the equatorial to ecliptic rotation matrix - IAU 2006 precession model

Description:

The equatorial to ecliptic rotation matrix is found and returned. The matrix is in the sense V(ecl) = RMAT ∗ V(equ); the equator, equinox and ecliptic are mean of date.

Invocation:

```
palEcmat( double date, double rmat[3][3] )
```

Arguments:

date = double (Given)

TT as Modified Julian Date (JD-2400000.5). The difference between TT and TDB is of the order of a millisecond or two (i.e. about 0.02 arc-seconds).

rmat = double[3][3] (Returned)

Rotation matrix

palEl2ue
Transform conventional elements into " universal" form

Description:

Transform conventional osculating elements into " universal" form.

Invocation:

```
void palEl2ue ( double date, int jform, double epoch, double orbinc, double anode,
double perih, double aorq, double e, double aorl, double dm, double u[13], int
*jstat );
```

Arguments:

date = double (Given)
Epoch (TT MJD) of osculation (Note 3)

jform = int (Given)
Element set actually returned (1-3; Note 6)

epoch = double (Given)
Epoch of elements (TT MJD)

orbinc = double (Given)
inclination (radians)

anode = double (Given)
longitude of the ascending node (radians)

perih = double (Given)
longitude or argument of perihelion (radians)

aorq = double (Given)
mean distance or perihelion distance (AU)

e = double (Given)
eccentricity

aorl = double (Given)
mean anomaly or longitude (radians, JFORM=1,2 only)

dm = double (Given)
daily motion (radians, JFORM=1 only)

u = double [13] (Returned)
Universal orbital elements (Note 1)

- (0) combined mass (M+m)
- (1) total energy of the orbit (alpha)
- (2) reference (osculating) epoch (t0)

- (3-5) position at reference epoch (r0)
- (6-8) velocity at reference epoch (v0)
- (9) heliocentric distance at reference epoch
- (10) r0.v0
- (11) date (t)
- (12) universal eccentric anomaly (psi) of date, approx

jstat = int ∗ (Returned)
status: 0 = OK

- -1 = illegal JFORM
- -2 = illegal E
- -3 = illegal AORQ
- -4 = illegal DM
- -5 = numerical error

Notes:

- The " universal" elements are those which define the orbit for the purposes of the method of universal variables (see reference). They consist of the combined mass of the two bodies, an epoch, and the position and velocity vectors (arbitrary reference frame) at that epoch. The parameter set used here includes also various quantities that can, in fact, be derived from the other information. This approach is taken to avoiding unnecessary computation and loss of accuracy. The supplementary quantities are (i) alpha, which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of psi, the " universal eccentric anomaly" at a given date and (v) that date.
- The companion routine is palUe2pv. This takes the set of numbers that the present routine outputs and uses them to derive the object' s position and velocity. A single prediction requires one call to the present routine followed by one call to palUe2pv; for convenience, the two calls are packaged as the routine palPlanel. Multiple predictions may be made by again calling the present routine once, but then calling palUe2pv multiple times, which is faster than multiple calls to palPlanel.
- DATE is the epoch of osculation. It is in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD-2400000.5).
- The supplied orbital elements are with respect to the J2000 ecliptic and equinox. The position and velocity parameters returned in the array U are with respect to the mean equator and equinox of epoch J2000, and are for the perihelion prior to the specified epoch.
- The universal elements returned in the array U are in canonical units (solar masses, AU and canonical days).
- Three different element-format options are available:

Option JFORM=1, suitable for the major planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = longitude of perihelion, curly pi (radians) AORQ = mean distance, a (AU) E = eccentricity, e (range 0 to <1) AORL = mean longitude L (radians) DM = daily motion (radians)

Option JFORM=2, suitable for minor planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = mean distance, a (AU) E = eccentricity, e (range 0 to <1) AORL = mean anomaly M (radians)

Option JFORM=3, suitable for comets:

EPOCH = epoch of perihelion (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = perihelion distance, q (AU) E = eccentricity, e (range 0 to 10)

- Unused elements (DM for JFORM=2, AORL and DM for JFORM=3) are not accessed.
- The algorithm was originally adapted from the EPHSLA program of D.H.P.Jones (private communication, 1996). The method is based on Stumpff's Universal Variables.

See Also :

Everhart & Pitkin, Am.J.Phys. 51, 712 (1983).

palEpco
Convert an epoch into the appropriate form - ' B' or ' J'

Description:

Converts a Besselian or Julian epoch to a Julian or Besselian epoch.

Invocation:

```
double palEpco( char k0, char k, double e );
```

Arguments:

k0 = char (Given)
Form of result: ' B' =Besselian, ' J' =Julian

k = char (Given)
Form of given epoch: ' B' or ' J' .

Notes:

- The result is always either equal to or very close to the given epoch E. The routine is required only in applications where punctilious treatment of heterogeneous mixtures of star positions is necessary.
- k and k0 are case insensitive. This differes slightly from the Fortran SLA implementation.
- k and k0 are not validated. They are interpreted as follows: o If k0 and k are the same the result is e o If k0 is ' b' or ' B' and k isn' t the conversion is J to B. o In all other cases, the conversion is B to J.

palEpv
Earth position and velocity with respect to the BCRS

Description:

Earth position and velocity, heliocentric and barycentric, with respect to the Barycentric Celestial Reference System.

Invocation:

```
void palEpv( double date, double ph[3], double vh[3], double pb[3], double vb[3]
);
```

Arguments:

date = double (Given)
Date, TDB Modified Julian Date (JD-2400000.5)

ph = double [3] (Returned)
Heliocentric Earth position (AU)

vh = double [3] (Returned)
Heliocentric Earth velocity (AU/day)

pb = double [3] (Returned)
Barycentric Earth position (AU)

vb = double [3] (Returned)
Barycentric Earth velocity (AU/day)

Notes:

- See eraEpv00 for details on accuracy
- Note that the status argument from eraEpv00 is ignored

palEtrms
Compute the E-terms vector

Description:

Computes the E-terms (elliptic component of annual aberration) vector.

Note the use of the J2000 aberration constant (20.49552 arcsec). This is a reflection of the fact that the E-terms embodied in existing star catalogues were computed from a variety of aberration constants. Rather than adopting one of the old constants the latest value is used here.

Invocation:

```
void palEtrms ( double ep, double ev[3] );
```

Arguments:

ep = double (Given)
Besselian epoch

ev = double [3] (Returned)
E-terms as (dx,dy,dz)

See also :

- Smith, C.A. et al., 1989. Astr.J. 97, 265.
- Yallop, B.D. et al., 1989. Astr.J. 97, 274.

palEqecl
Transform from J2000.0 equatorial coordinates to ecliptic coordinates

Description:

Transform from J2000.0 equatorial coordinates to ecliptic coordinates.

Invocation:

```
void palEqecl( double dr, double dd, double date, double *dl, double *db);
```

Arguments:

dr = double (Given)
J2000.0 mean RA (radians)

dd = double (Given)
J2000.0 mean Dec (Radians)

date = double (Given)
TT as Modified Julian Date (JD-2400000.5). The difference between TT and TDB is of the order of a millisecond or two (i.e. about 0.02 arc-seconds).

dl = double * (Returned)
Ecliptic longitude (mean of date, IAU 1980 theory, radians)

db = double * (Returned)
Ecliptic latitude (mean of date, IAU 1980 theory, radians)

palEqgal
Convert from J2000.0 equatorial coordinates to Galactic

Description:

Transformation from J2000.0 equatorial coordinates to IAU 1958 galactic coordinates.

Invocation:

```
void palEqgal ( double dr, double dd, double *dl, double *db );
```

Arguments:

dr = double (Given)
J2000.0 RA (radians)

dd = double (Given)
J2000.0 Dec (radians

dl = double * (Returned)
Galactic longitude (radians).

db = double * (Returned)
Galactic latitude (radians).

Notes:

The equatorial coordinates are J2000.0. Use the routine palGe50 if conversion to B1950.0 'FK4' coordinates is required.

See Also :

Blaauw et al, Mon.Not.R.Astron.Soc.,121,123 (1960)

palEvp
Returns the barycentric and heliocentric velocity and position of the Earth

Description:

Returns the barycentric and heliocentric velocity and position of the Earth at a given epoch, given with respect to a specified equinox. For information about accuracy, see the function eraEpv00.

Invocation:

```
void palEvp( double date, double deqx, double dvb[3], double dpb[3], double dvh[3],
double dph[3] )
```

Arguments:

date = double (Given)
TDB (loosely ET) as a Modified Julian Date (JD-2400000.5)

deqx = double (Given)
Julian epoch (e.g. 2000.0) of mean equator and equinox of the vectors returned. If deqx <= 0.0, all vectors are referred to the mean equator and equinox (FK5) of epoch date.

dvb = double[3] (Returned)
Barycentric velocity (AU/s, AU)

dpb = double[3] (Returned)
Barycentric position (AU/s, AU)

dvh = double[3] (Returned)
heliocentric velocity (AU/s, AU)

dph = double[3] (Returned)
Heliocentric position (AU/s, AU)

palFitxy
Fit a linear model to relate two sets of [x,y] coordinates

Description:

Fits a linear model to relate two sets of [X,Y] coordinates.

Invocation:

```
palFitxy ( int itype, int np, double xye[][2], double xym[][2], double coeffs[6],
int *j )
```

Arguments:

itype = int (Given)
type of model: 4 or 6 (note 1)

np = int (Given)
number of samples (note 2)

xye = double[np][2] (Given)
expected [x,y] for each sample

xym = double[np][2] (Given)
measured [x,y] for each sample

coeffs = double[6] (Returned)
coefficients of model (note 3)

j = int * (Returned)
status:

- 0 = OK
- -1 = illegal itype
- -2 = insufficient data
- -3 = no solution

Notes:

1) itype, which must be either 4 or 6, selects the type of model fitted. Both allowed itype values produce a model coeffs which consists of six coefficients, namely the zero points and, for each of xe and ye, the coefficient of xm and ym. For itype=6, all six coefficients are independent, modelling squash and shear as well as origin, scale, and orientation. However, itype=4 selects the " solid body rotation" option; the model coeffs still consists of the same six coefficients, but now two of them are used twice (appropriately signed). Origin, scale and orientation are still modelled, but not squash or shear - the units of x and y have to be the same.

2) For itype=4, np must be at least 2. For itype=6, np must be at least 3.

3) The model is returned in the array coeffs. Naming the elements of coeffs as follows:

coeffs[0] = A
coeffs[1] = B
coeffs[2] = C
coeffs[3] = D
coeffs[4] = E
coeffs[5] = F

the model is:

$xe = A + B * xm + C * ym$
$ye = D + E * xm + F * ym$

For the " solid body rotation" option (itype=4), the magnitudes of B and F, and of C and E, are equal. The signs of these coefficients depend on whether there is a sign reversal between xe,ye and xm,ym; fits are performed with and without a sign reversal and the best one chosen.

4) Error status values j=-1 and -2 leave coeffs unchanged; if j=-3 coeffs may have been changed.

See also :

palPxy, palInvf, palXy2xy and palDcmpf

palFk45z
Convert B1950.0 FK4 star data to J2000.0 FK5 assuming zero proper motion in the FK5 frame

Description:

Convert B1950.0 FK4 star data to J2000.0 FK5 assuming zero proper motion in the FK5 frame (double precision)

This function converts stars from the Bessel-Newcomb, FK4 system to the IAU 1976, FK5, Fricke system, in such a way that the FK5 proper motion is zero. Because such a star has, in general, a non-zero proper motion in the FK4 system, the routine requires the epoch at which the position in the FK4 system was determined.

The method is from Appendix 2 of Ref 1, but using the constants of Ref 4.

Invocation:

```
palFk45z( double r1950, double d1950, double bepoch, double *r2000, double *d2000
)
```

Arguments:

r1950 = double (Given)
B1950.0 FK4 RA at epoch (radians).

d1950 = double (Given)
B1950.0 FK4 Dec at epoch (radians).

bepoch = double (Given)
Besselian epoch (e.g. 1979.3)

r2000 = double (Returned)
J2000.0 FK5 RA (Radians).

d2000 = double (Returned)
J2000.0 FK5 Dec(Radians).

Notes:

- The epoch BEPOCH is strictly speaking Besselian, but if a Julian epoch is supplied the result will be affected only to a negligible extent.
- Conversion from Besselian epoch 1950.0 to Julian epoch 2000.0 only is provided for. Conversions involving other epochs will require use of the appropriate precession, proper motion, and E-terms routines before and/or after palFk45z is called.
- In the FK4 catalogue the proper motions of stars within 10 degrees of the poles do not embody the differential E-term effect and should, strictly speaking, be handled in a different manner from stars outside these regions. However, given the general lack of homogeneity of the star data available for routine astrometry, the difficulties of

handling positions that may have been determined from astrometric fields spanning the polar and non-polar regions, the likelihood that the differential E-terms effect was not taken into account when allowing for proper motion in past astrometry, and the undesirability of a discontinuity in the algorithm, the decision has been made in this routine to include the effect of differential E-terms on the proper motions for all stars, whether polar or not. At epoch 2000, and measuring on the sky rather than in terms of dRA, the errors resulting from this simplification are less than 1 milliarcsecond in position and 1 milliarcsecond per century in proper motion.

References :

- Aoki,S., et al, 1983. Astron.Astrophys., 128, 263.
- Smith, C.A. et al, 1989. " The transformation of astrometric catalog systems to the equinox J2000.0" . Astron.J. 97, 265.
- Yallop, B.D. et al, 1989. " Transformation of mean star places from FK4 B1950.0 to FK5 J2000.0 using matrices in 6-space" . Astron.J. 97, 274.
- Seidelmann, P.K. (ed), 1992. " Explanatory Supplement to the Astronomical Almanac" , ISBN 0-935702-68-7.

palFk524
Convert J2000.0 FK5 star data to B1950.0 FK4

Description:

This function converts stars from the IAU 1976, FK5, Fricke system, to the Bessel-Newcomb, FK4 system. The precepts of Smith et al (Ref 1) are followed, using the implementation by Yallop et al (Ref 2) of a matrix method due to Standish. Kinoshita's development of Andoyer's post-Newcomb precession is used. The numerical constants from Seidelmann et al (Ref 3) are used canonically.

Invocation:

```
palFk524( double r2000, double d2000, double dr2000, double dd2000, double p2000,
double v2000, double *r1950, double *d1950, double *dr1950, double *dd1950, double
*p1950, double *v1950 )
```

Arguments:

r2000 = double (Given)
J2000.0 FK5 RA (radians).

d2000 = double (Given)
J2000.0 FK5 Dec (radians).

dr2000 = double (Given)
J2000.0 FK5 RA proper motion (rad/Jul.yr)

dd2000 = double (Given)
J2000.0 FK5 Dec proper motion (rad/Jul.yr)

p2000 = double (Given)
J2000.0 FK5 parallax (arcsec)

v2000 = double (Given)
J2000.0 FK5 radial velocity (km/s, +ve = moving away)

r1950 = double * (Returned)
B1950.0 FK4 RA (radians).

d1950 = double * (Returned)
B1950.0 FK4 Dec (radians).

dr1950 = double * (Returned)
B1950.0 FK4 RA proper motion (rad/Jul.yr)

dd1950 = double * (Returned)
B1950.0 FK4 Dec proper motion (rad/Jul.yr)

p1950 = double * (Returned)
B1950.0 FK4 parallax (arcsec)

v1950 = double ∗ (Returned)
B1950.0 FK4 radial velocity (km/s, +ve = moving away)

Notes:

- The proper motions in RA are dRA/dt rather than cos(Dec)∗dRA/dt, and are per year rather than per century.
- Note that conversion from Julian epoch 2000.0 to Besselian epoch 1950.0 only is provided for. Conversions involving other epochs will require use of the appropriate precession, proper motion, and E-terms routines before and/or after FK524 is called.
- In the FK4 catalogue the proper motions of stars within 10 degrees of the poles do not embody the differential E-term effect and should, strictly speaking, be handled in a different manner from stars outside these regions. However, given the general lack of homogeneity of the star data available for routine astrometry, the difficulties of handling positions that may have been determined from astrometric fields spanning the polar and non-polar regions, the likelihood that the differential E-terms effect was not taken into account when allowing for proper motion in past astrometry, and the undesirability of a discontinuity in the algorithm, the decision has been made in this routine to include the effect of differential E-terms on the proper motions for all stars, whether polar or not. At epoch 2000, and measuring on the sky rather than in terms of dRA, the errors resulting from this simplification are less than 1 milliarcsecond in position and 1 milliarcsecond per century in proper motion.

References :

- Smith, C.A. et al, 1989. " The transformation of astrometric catalog systems to the equinox J2000.0" . Astron.J. 97, 265.
- Yallop, B.D. et al, 1989. " Transformation of mean star places from FK4 B1950.0 to FK5 J2000.0 using matrices in 6-space" . Astron.J. 97, 274.
- Seidelmann, P.K. (ed), 1992. " Explanatory Supplement to the Astronomical Almanac" , ISBN 0-935702-68-7.

palFk54z
Convert a J2000.0 FK5 star position to B1950.0 FK4 assuming zero proper motion and parallax

Description:

This function converts star positions from the IAU 1976, FK5, Fricke system to the Bessel-Newcomb, FK4 system.

Invocation:

```
palFk54z( double r2000, double d2000, double bepoch, double *r1950, double *d1950,
double *dr1950, double *dd1950 )
```

Arguments:

r2000 = double (Given)
J2000.0 FK5 RA (radians).

d2000 = double (Given)
J2000.0 FK5 Dec (radians).

bepoch = double (Given)
Besselian epoch (e.g. 1950.0).

r1950 = double * (Returned)
B1950 FK4 RA (radians) at epoch " bepoch" .

d1950 = double * (Returned)
B1950 FK4 Dec (radians) at epoch " bepoch" .

dr1950 = double * (Returned)
B1950 FK4 proper motion (RA) (radians/trop.yr)).

dr1950 = double * (Returned)
B1950 FK4 proper motion (Dec) (radians/trop.yr)).

Notes:

- The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
- Conversion from Julian epoch 2000.0 to Besselian epoch 1950.0 only is provided for. Conversions involving other epochs will require use of the appropriate precession functions before and after this function is called.
- The FK5 proper motions, the parallax and the radial velocity are presumed zero.
- It is the intention that FK5 should be a close approximation to an inertial frame, so that distant objects have zero proper motion; such objects have (in general) non-zero proper motion in FK4, and this function returns those fictitious proper motions.

- The position returned by this function is in the B1950 reference frame but at Besselian epoch BEPOCH. For comparison with catalogues the " bepoch" argument will frequently be 1950.0.

palGaleq
Convert from galactic to J2000.0 equatorial coordinates

Description:

Transformation from IAU 1958 galactic coordinates to J2000.0 equatorial coordinates.

Invocation:

```
void palGaleq ( double dl, double db, double *dr, double *dd );
```

Arguments:

dl = double (Given)
Galactic longitude (radians).

db = double (Given)
Galactic latitude (radians).

dr = double * (Returned)
J2000.0 RA (radians)

dd = double * (Returned)
J2000.0 Dec (radians)

Notes:

The equatorial coordinates are J2000.0. Use the routine palGe50 if conversion to B1950.0 'FK4' coordinates is required.

See Also :

Blaauw et al, Mon.Not.R.Astron.Soc.,121,123 (1960)

palGalsup
Convert from galactic to supergalactic coordinates

Description:

Transformation from IAU 1958 galactic coordinates to de Vaucouleurs supergalactic coordinates.

Invocation:

```
void palGalsup ( double dl, double db, double *dsl, double *dsb );
```

Arguments:

dl = double (Given)
Galactic longitude.

db = double (Given)
Galactic latitude.

dsl = double * (Returned)
Supergalactic longitude.

dsb = double * (Returned)
Supergalactic latitude.

See Also :

- de Vaucouleurs, de Vaucouleurs, & Corwin, Second Reference Catalogue of Bright Galaxies, U. Texas, page 8.
- Systems & Applied Sciences Corp., Documentation for the machine-readable version of the above catalogue, Contract NAS 5-26490.

(These two references give different values for the galactic longitude of the supergalactic origin. Both are wrong; the correct value is L2=137.37.)

palGe50
Transform Galactic Coordinate to B1950 FK4

Description:

Transformation from IAU 1958 galactic coordinates to B1950.0 'FK4' equatorial coordinates.

Invocation:

```
palGe50( double dl, double db, double *dr, double *dd );
```

Arguments:

dl = double (Given)
Galactic longitude (radians)

db = double (Given)
Galactic latitude (radians)

dr = double * (Returned)
B9150.0 FK4 RA.

dd = double * (Returned)
B1950.0 FK4 Dec.

Notes:

- The equatorial coordinates are B1950.0 'FK4'. Use the routine palGaleq if conversion to J2000.0 coordinates is required.

See Also :

- Blaauw et al, Mon.Not.R.Astron.Soc.,121,123 (1960)

palGeoc
Convert geodetic position to geocentric

Description:

Convert geodetic position to geocentric.

Invocation:

```
void palGeoc( double p, double h, double * r, double *z );
```

Arguments:

p = double (Given)
latitude (radians)

h = double (Given)
height above reference spheroid (geodetic, metres)

r = double * (Returned)
distance from Earth axis (AU)

z = double * (Returned)
distance from plane of Earth equator (AU)

Notes:

- Geocentric latitude can be obtained by evaluating atan2(z,r)
- Uses WGS84 reference ellipsoid and calls eraGd2gc

palIntin
Convert free-format input into an integer

Description:

Extracts a number from an input string starting at the specified index.

Invocation:

```
void palIntin( const char * string, int *nstrt, long *ireslt, int *jflag );
```

Arguments:

string = const char * (Given)

String containing number to be decoded.

nstrt = int * (Given and Returned)

Character number indicating where decoding should start. On output its value is updated to be the location of the possible next value. For compatibility with SLA the first character is index 1.

ireslt = long * (Returned)

Result. Not updated when jflag=1.

jflag = int * (Returned)

status: -1 = -OK, 0 = +OK, 1 = null, 2 = error

Notes:

- Uses the strtol() system call to do the parsing. This may lead to subtle differences when compared to the SLA/F parsing.
- Commas are recognized as a special case and are skipped if one happens to be the next character when updating nstrt. Additionally the output nstrt position will skip past any trailing space.
- If no number can be found flag will be set to 1.
- If the number overflows or underflows jflag will be set to 2. For overflow the returned result will have the value LONG_MAX, for underflow it will have the value LONG_MIN.

palInvf
Invert a linear model of the type produced by the palFitxy routine

Description:

The models relate two sets of [x,y] coordinates as follows. Naming the elements of fwds:

fwds[0] = A
fwds[1] = B
fwds[2] = C
fwds[3] = D
fwds[4] = E
fwds[5] = F

where two sets of coordinates [x1,y1] and [x2,y2] are related thus:

$x2 = A + B * x1 + C * y1$
$y2 = D + E * x1 + F * y1$

the present routine generates a new set of coefficients:

bkwds[0] = P
bkwds[1] = Q
bkwds[2] = R
bkwds[3] = S
bkwds[4] = T
bkwds[5] = U

such that:

$x1 = P + Q * x2 + R * y2$
$y1 = S + T * x2 + U * y2$

Two successive calls to palInvf will thus deliver a set of coefficients equal to the starting values.

Invocation:

```
palInvf ( double fwds[6], double bkwds[6], int *j )
```

Arguments:

fwds = double[6] (Given)
model coefficients

bkwds = double[6] (Returned)
inverse model

j = int (Returned)
status: 0 = OK, -1 = no inverse

See also :

palFitxy, palPxy, palXy2xy and palDcmpf

palMap
Convert star RA,Dec from mean place to geocentric apparent

Description:

Convert star RA,Dec from mean place to geocentric apparent.

Invocation:

```
void palMap( double rm, double dm, double pr, double pd, double px, double rv,
double eq, double date, double *ra, double *da );
```

Arguments:

rm = double (Given)
Mean RA (radians)

dm = double (Given)
Mean declination (radians)

pr = double (Given)
RA proper motion, changes per Julian year (radians)

pd = double (Given)
Dec proper motion, changes per Julian year (radians)

px = double (Given)
Parallax (arcsec)

rv = double (Given)
Radial velocity (km/s, +ve if receding)

eq = double (Given)
Epoch and equinox of star data (Julian)

date = double (Given)
TDB for apparent place (JD-2400000.5)

ra = double * (Returned)
Apparent RA (radians)

dec = double * (Returned)
Apparent dec (radians)

Notes:

- Calls palMappa and palMapqk
- The reference systems and timescales used are IAU 2006.

- EQ is the Julian epoch specifying both the reference frame and the epoch of the position - usually 2000. For positions where the epoch and equinox are different, use the routine palPm to apply proper motion corrections before using this routine.
- The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
- The proper motions in RA are dRA/dt rather than cos(Dec)∗dRA/dt.
- This routine may be wasteful for some applications because it recomputes the Earth position/velocity and the precession- nutation matrix each time, and because it allows for parallax and proper motion. Where multiple transformations are to be carried out for one epoch, a faster method is to call the palMappa routine once and then either the palMapqk routine (which includes parallax and proper motion) or palMapqkz (which assumes zero parallax and proper motion).
- The accuracy is sub-milliarcsecond, limited by the precession-nutation model (see palPrenut for details).
- The accuracy is further limited by the routine palEvp, called by palMappa, which computes the Earth position and velocity. See eraEpv00 for details on that calculation.

palMappa
Compute parameters needed by palAmpqk and palMapqk

Description:

Compute star-independent parameters in preparation for transformations between mean place and geocentric apparent place.

The parameters produced by this function are required in the parallax, aberration, and nutation/bias/precession parts of the mean/apparent transformations.

The reference systems and timescales used are IAU 2006.

Invocation:

```
void palMappa( double eq, double date, double amprms[21] )
```

Arguments:

eq = double (Given)
epoch of mean equinox to be used (Julian)

date = double (Given)
TDB (JD-2400000.5)

amprms = double[21] (Returned)
star-independent mean-to-apparent parameters:

- (0) time interval for proper motion (Julian years)
- (1-3) barycentric position of the Earth (AU)
- (4-6) heliocentric direction of the Earth (unit vector)
- (7) (Schwarzschild radius of Sun)/(Sun-Earth distance)
- (8-10) abv: barycentric Earth velocity in units of c
- (11) sqrt(1-v^2) where v=modulus(abv)
- (12-20) precession/nutation (3,3) matrix

Notes:

- For date, the distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
- The vector amprms(1-3) is referred to the mean equinox and equator of epoch eq.
- The parameters amprms produced by this function are used by palAmpqk, palMapqk and palMapqkz.

palMapqk
Quick mean to apparent place

Description:

Quick mean to apparent place: transform a star RA,Dec from mean place to geocentric apparent place, given the star-independent parameters.

Use of this routine is appropriate when efficiency is important and where many star positions, all referred to the same equator and equinox, are to be transformed for one epoch. The star-independent parameters can be obtained by calling the palMappa routine.

If the parallax and proper motions are zero the palMapqkz routine can be used instead.

Invocation:

```
void palMapqk ( double rm, double dm, double pr, double pd, double px, double
rv, double amprms[21], double *ra, double *da );
```

Arguments:

rm = double (Given)
Mean RA (radians)

dm = double (Given)
Mean declination (radians)

pr = double (Given)
RA proper motion, changes per Julian year (radians)

pd = double (Given)
Dec proper motion, changes per Julian year (radians)

px = double (Given)
Parallax (arcsec)

rv = double (Given)
Radial velocity (km/s, +ve if receding)

amprms = double [21] (Given)
Star-independent mean-to-apparent parameters (see palMappa).

ra = double * (Returned)
Apparent RA (radians)

dec = double * (Returned)
Apparent dec (radians)

Notes:

- The reference frames and timescales used are post IAU 2006.

- The mean place rm, dm and the vectors amprms[1-3] and amprms[4-6] are referred to the mean equinox and equator of the epoch specified when generating the precession/nutation matrix amprms[12-20]. In the call to palMappa (q.v.) normally used to populate amprms, this epoch is the first argument (eq).
- Strictly speaking, the routine is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about 920 arcsec of the centre of the Sun's disc. The term has a maximum value of about 1.85 arcsec at this radius, and decreases to zero as the centre of the disc is approached.

palMapqkz
Quick mean to apparent place (no proper motion or parallax)

Description:

Quick mean to apparent place: transform a star RA,dec from mean place to geocentric apparent place, given the star-independent parameters, and assuming zero parallax and proper motion.

Use of this function is appropriate when efficiency is important and where many star positions, all with parallax and proper motion either zero or already allowed for, and all referred to the same equator and equinox, are to be transformed for one epoch. The star-independent parameters can be obtained by calling the palMappa function.

The corresponding function for the case of non-zero parallax and proper motion is palMapqk.

Invocation:

```
void palMapqkz( double rm, double dm, double amprms[21], double *ra, double *da
)
```

Arguments:

rm = double (Given)
Mean RA (radians).

dm = double (Given)
Mean Dec (radians).

amprms = double[21] (Given)
Star-independent mean-to-apparent parameters (see palMappa): (0-3) not used (4-6) heliocentric direction of the Earth (unit vector) (7) not used (8-10) abv: barycentric Earth velocity in units of c (11) sqrt(1-v^2) where v=modulus(abv) (12-20) precession/nutation (3,3) matrix

ra = double * (Returned)
Apparent RA (radians).

da = double * (Returned)
Apparent Dec (radians).

Notes:

- The reference systems and timescales used are IAU 2006.
- The mean place rm, dm and the vectors amprms[1-3] and amprms[4-6] are referred to the mean equinox and equator of the epoch specified when generating the precession/nutation matrix amprms[12-20]. In the call to palMappa (q.v.) normally used to populate amprms, this epoch is the first argument (eq).

- The vector amprms(4-6) is referred to the mean equinox and equator of epoch eq.
- Strictly speaking, the routine is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about 920 arcsec of the centre of the Sun's disc. The term has a maximum value of about 1.85 arcsec at this radius, and decreases to zero as the centre of the disc is approached.

palNut
Form the matrix of nutation

Description:

Form the matrix of nutation for a given date using the IAU 2006 nutation model and palDeuler.

Invocation:

```
void palNut( double date, double rmatn[3][3] );
```

Arguments:

date = double (Given)

TT as modified Julian date (JD-2400000.5)

rmatn = double [3][3] (Returned)

Nutation matrix in the sense v(true)=rmatn $*$ v(mean) where v(true) is the star vector relative to the true equator and equinox of date and v(mean) is the star vector relative to the mean equator and equinox of date.

Notes:

- Uses eraNut06a via palNutc
- The distinction between TDB and TT is negligible. For all but the most critical applications UTC is adequate.

palNutc
Calculate nutation longitude & obliquoty components

Description:

Calculates the longitude ∗ obliquity components and mean obliquity using the SOFA/ERFA library.

Invocation:

```
void palNutc( double date, double * dpsi, double *deps, double *eps0 );
```

Arguments:

date = double (Given)
TT as modified Julian date (JD-2400000.5)

dpsi = double ∗ (Returned)
Nutation in longitude

deps = double ∗ (Returned)
Nutation in obliquity

eps0 = double ∗ (Returned)
Mean obliquity.

Notes:

- Calls eraObl06 and eraNut06a and therefore uses the IAU 206 precession/nutation model.
- Note the change from SLA/F regarding the date. TT is used rather than TDB.

palOap
Observed to apparent place

Description:

Observed to apparent place.

Invocation:

```
void palOap ( const char *type, double ob1, double ob2, double date, double dut,
double elongm, double phim, double hm, double xp, double yp, double tdk, double
pmb, double rh, double wl, double tlr, double *rap, double *dap );
```

Arguments:

type = const char * (Given)
Type of coordinates - 'R', 'H' or 'A' (see below)

ob1 = double (Given)
Observed Az, HA or RA (radians; Az is N=0;E=90)

ob2 = double (Given)
Observed ZD or Dec (radians)

date = double (Given)
UTC date/time (Modified Julian Date, JD-2400000.5)

dut = double (Given)
delta UT: UT1-UTC (UTC seconds)

elongm = double (Given)
Mean longitude of the observer (radians, east +ve)

phim = double (Given)
Mean geodetic latitude of the observer (radians)

hm = double (Given)
Observer's height above sea level (metres)

xp = double (Given)
Polar motion x-coordinates (radians)

yp = double (Given)
Polar motion y-coordinates (radians)

tdk = double (Given)
Local ambient temperature (K; std=273.15)

pmb = double (Given)
Local atmospheric pressure (mb; std=1013.25)

rh = double (Given)
Local relative humidity (in the range 0.0-1.0)

wl = double (Given)
Effective wavelength (micron, e.g. 0.55)

tlr = double (Given)
Tropospheric laps rate (K/metre, e.g. 0.0065)

rap = double * (Given)
Geocentric apparent right ascension

dap = double * (Given)
Geocentric apparent declination

Notes:

- Only the first character of the TYPE argument is significant. 'R' or 'r' indicates that OBS1 and OBS2 are the observed right ascension and declination; 'H' or 'h' indicates that they are hour angle (west +ve) and declination; anything else ('A' or 'a' is recommended) indicates that OBS1 and OBS2 are azimuth (north zero, east 90 deg) and zenith distance. (Zenith distance is used rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.)
- The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted apparent RA,Dec should be within about 0.1 arcsec for a zenith distance of less than 70 degrees. Even at a topocentric zenith distance of 90 degrees, the accuracy in elevation should be better than 1 arcmin; useful results are available for a further 3 degrees, beyond which the palRefro routine returns a fixed value of the refraction. The complementary routines palAop (or palAopqk) and palOap (or palOapqk) are self-consistent to better than 1 micro- arcsecond all over the celestial sphere.
- It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
- "Observed" Az,El means the position that would be seen by a perfect theodolite located at the observer. This is related to the observed HA,Dec via the standard rotation, using the geodetic latitude (corrected for polar motion), while the observed HA and RA are related simply through the local apparent ST. "Observed" RA,Dec or HA,Dec thus means the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (n.b. not to the refracted pole). By removing from the observed place the effects of atmospheric refraction and diurnal aberration, the geocentric apparent RA,Dec is obtained.
- Frequently, mean rather than apparent RA,Dec will be required, in which case further transformations will be necessary. The palAmp etc routines will convert the apparent RA,Dec produced by the present routine into an "FK5" (J2000) mean place, by allowing for the Sun's gravitational lens effect, annual aberration, nutation and precession. Should "FK4" (1950) coordinates be needed, the routines palFk524 etc will also need to be applied.

- To convert to apparent RA,Dec the coordinates read from a real telescope, corrections would have to be applied for encoder zero points, gear and encoder errors, tube flexure, the position of the rotator axis and the pointing axis relative to it, non-perpendicularity between the mounting axes, and finally for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures). Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
- This routine takes time to execute, due mainly to the rigorous integration used to evaluate the refraction. For processing multiple stars for one location and time, call palAoppa once followed by one call per star to palOapqk. Where a range of times within a limited period of a few hours is involved, and the highest precision is not required, call palAoppa once, followed by a call to palAoppat each time the time changes, followed by one call per star to palOapqk.
- The DATE argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the delta UT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap delta UT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the delta UTC changes by one second to its post-leap new value.
- The delta UT (UT1-UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep delta UT within +/- 0.9 seconds.
- IMPORTANT – TAKE CARE WITH THE LONGITUDE SIGN CONVENTION. The longitude required by the present routine is east-positive, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the palOBS routine are west-positive, following astronomical usage, and must be reversed in sign before use in the present routine.
- The polar coordinates XP,YP can be obtained from IERS circulars and equivalent publications. The maximum amplitude is about 0.3 arcseconds. If XP,YP values are unavailable, use XP=YP=0D0. See page B60 of the 1988 Astronomical Almanac for a definition of the two angles.
- The height above sea level of the observing station, HM, can be obtained from the Astronomical Almanac (Section J in the 1988 edition), or via the routine palOBS. If P, the pressure in millibars, is available, an adequate estimate of HM can be obtained from the expression

HM $\sim$ -29.3$*$TSL$*$LOG(P/1013.25).

where TSL is the approximate sea-level air temperature in K (see Astrophysical Quantities, C.W.Allen, 3rd edition, section 52). Similarly, if the pressure P is not known, it can be estimated from the height of the observing station, HM, as follows:

P $\sim$ 1013.25$*$EXP(-HM/(29.3$*$TSL)).

Note, however, that the refraction is nearly proportional to the pressure and that an accurate P value is important for precise work.

- The azimuths etc. used by the present routine are with respect to the celestial pole. Corrections from the terrestrial pole can be computed using palPolmo.

palOapqk
Quick observed to apparent place

Description:

type = const char * (Given) Type of coordinates - ' R' , ' H' or ' A' (see below) ob1 = double (Given) Observed Az, HA or RA (radians; Az is N=0;E=90) ob2 = double (Given) Observed ZD or Dec (radians) aoprms = const double [14] (Given) Star-independent apparent-to-observed parameters. See palAopqk for details. rap = double * (Given) Geocentric apparent right ascension dap = double * (Given) Geocentric apparent declination

Invocation:

```
void palOapqk ( const char *type, double ob1, double ob2, const double aoprms[14],
double *rap, double *dap );
```

Arguments:

Quick observed to apparent place.

Notes:

- Only the first character of the TYPE argument is significant. ' R' or ' r' indicates that OBS1 and OBS2 are the observed right ascension and declination; ' H' or ' h' indicates that they are hour angle (west +ve) and declination; anything else (' A' or ' a' is recommended) indicates that OBS1 and OBS2 are azimuth (north zero, east 90 deg) and zenith distance. (Zenith distance is used rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.)
- The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted apparent RA,Dec should be within about 0.1 arcsec for a zenith distance of less than 70 degrees. Even at a topocentric zenith distance of 90 degrees, the accuracy in elevation should be better than 1 arcmin; useful results are available for a further 3 degrees, beyond which the palREFRO routine returns a fixed value of the refraction. The complementary routines palAop (or palAopqk) and palOap (or palOapqk) are self-consistent to better than 1 micro- arcsecond all over the celestial sphere.
- It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
- " Observed" Az,El means the position that would be seen by a perfect theodolite located at the observer. This is related to the observed HA,Dec via the standard rotation, using the geodetic latitude (corrected for polar motion), while the observed HA and RA are related simply through the local apparent ST. " Observed" RA,Dec or HA,Dec thus means the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth' s axis of rotation (n.b. not to

the refracted pole). By removing from the observed place the effects of atmospheric refraction and diurnal aberration, the geocentric apparent RA,Dec is obtained.

- Frequently, mean rather than apparent RA,Dec will be required, in which case further transformations will be necessary. The palAmp etc routines will convert the apparent RA,Dec produced by the present routine into an " FK5" (J2000) mean place, by allowing for the Sun' s gravitational lens effect, annual aberration, nutation and precession. Should " FK4" (1950) coordinates be needed, the routines palFk524 etc will also need to be applied.
- To convert to apparent RA,Dec the coordinates read from a real telescope, corrections would have to be applied for encoder zero points, gear and encoder errors, tube flexure, the position of the rotator axis and the pointing axis relative to it, non-perpendicularity between the mounting axes, and finally for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures). Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
- The star-independent apparent-to-observed-place parameters in AOPRMS may be computed by means of the palAoppa routine. If nothing has changed significantly except the time, the palAoppat routine may be used to perform the requisite partial recomputation of AOPRMS.
- The azimuths etc used by the present routine are with respect to the celestial pole. Corrections from the terrestrial pole can be computed using palPolmo.

palObs
Parameters of selected ground-based observing stations

Description:

Station numbers, identifiers, names and other details are subject to change and should not be hardwired into application programs.

All characters in " c" up to the first space are checked; thus an abbreviated ID will return the parameters for the first station in the list which matches the abbreviation supplied, and no station in the list will ever contain embedded spaces. " c" must not have leading spaces.

IMPORTANT – BEWARE OF THE LONGITUDE SIGN CONVENTION. The longitude returned by palOBS (and SLA_OBS) is west-positive in accordance with astronomical usage. However, this sign convention is left-handed and is the opposite of the one used by geographers; elsewhere in PAL the preferable east-positive convention is used. In particular, note that for use in palAop, palAoppa and palOap the sign of the longitude must be reversed.

Users are urged to inform the author of any improvements they would like to see made. For example:

typographical corrections more accurate parameters better station identifiers or names additional stations

Invocation:

```
int palObs( size_t n, const char * c, char * ident, size_t identlen, char * name,
size_t namelen, double * w, double * p, double * h );
```

Arguments:

n = size_t (Given)
Number specifying the observing station. If 0 the identifier in " c" is used to determine the observing station to use.

c = const char * (Given)
Identifier specifying the observing station for which the parameters should be returned. Only used if n is 0. Can be NULL for n>0. Case insensitive.

ident = char * (Returned)
Identifier of the observing station selected. Will be identical to " c" if n==0. Unchanged if " n" or " c" do not match an observing station. Should be at least 11 characters (including the trailing nul).

identlen = size_t (Given)
Size of the buffer " ident" including trailing nul.

name = char * (Returned)
Full name of the specified observing station. Contains " ?" if " n" or " c" did not correspond to a valid station. Should be at least 41 characters (including the trailing nul).

w = double ∗ (Returned)
Longitude (radians, West +ve). Unchanged if observing station could not be identified.

p = double ∗ (Returned)
Geodetic latitude (radians, North +ve). Unchanged if observing station could not be identified.

h = double ∗ (Returned)
Height above sea level (metres). Unchanged if observing station could not be identified.

Returned Value:

palObs = int
0 if an observing station was returned. -1 if no match was found.

Notes:

- Differs from the SLA interface in that the output short name is not the same variable as the input short name. This simplifies consting. Additionally the size of the output buffers are now specified in the API and a status integer is returned.

palPa
HA, Dec to Parallactic Angle

Description:

Converts HA, Dec to Parallactic Angle.

Invocation:

```
double palPa( double ha, double dec, double phi );
```

Arguments:

ha = double (Given)
Hour angle in radians (Geocentric apparent)

dec = double (Given)
Declination in radians (Geocentric apparent)

phi = double (Given)
Observatory latitude in radians (geodetic)

Returned Value:

palPa = double
Parallactic angle in the range -pi to +pi.

Notes:

- The parallactic angle at a point in the sky is the position angle of the vertical, i.e. the angle between the direction to the pole and to the zenith. In precise applications care must be taken only to use geocentric apparent HA,Dec and to consider separately the effects of atmospheric refraction and telescope mount errors.
- At the pole a zero result is returned.

palPcd
Apply pincushion/barrel distortion to a tangent-plane [x,y]

Description:

Applies pincushion and barrel distortion to a tangent plane coordinate.

Invocation:

```
palPcd( double disco, double * x, double * y );
```

Arguments:

disco = double (Given)

Pincushion/barrel distortion coefficient.

x = double * (Given & Returned)

On input the tangent-plane X coordinate, on output the distorted X coordinate.

y = double * (Given & Returned)

On input the tangent-plane Y coordinate, on output the distorted Y coordinate.

Notes:

- The distortion is of the form RP = R*(1 + C*R**2), where R is the radial distance from the tangent point, C is the DISCO argument, and RP is the radial distance in the presence of the distortion.
- For pincushion distortion, C is +ve; for barrel distortion, C is -ve.
- For X,Y in units of one projection radius (in the case of a photographic plate, the focal length), the following DISCO values apply:

Geometry DISCO

astrograph 0.0 Schmidt -0.3333 AAT PF doublet +147.069 AAT PF triplet +178.585 AAT f/8 +21.20 JKT f/8 +13.32

See Also :

- There is a companion routine, palUnpcd, which performs the inverse operation.

palPertel
Update elements by applying planetary perturbations

Description:

Update the osculating orbital elements of an asteroid or comet by applying planetary perturbations.

Invocation:

```
void palPertel (int jform, double date0, double date1, double epoch0, double
orbi0, double anode0, double perih0, double aorq0, double e0, double am0, double
*epoch1, double *orbi1, double *anode1, double *perih1, double *aorq1, double
*e1, double *am1, int *jstat );
```

Arguments:

jform = int (Given)
Element set actually returned (1-3; Note 6)

date0 = double (Given)
Date of osculation (TT MJD) for the given elements.

date1 = double (Given)
Date of osculation (TT MJD) for the updated elements.

epoch0 = double (Given)
Epoch of elements (TT MJD)

orbi0 = double (Given)
inclination (radians)

anode0 = double (Given)
longitude of the ascending node (radians)

perih0 = double (Given)
longitude or argument of perihelion (radians)

aorq0 = double (Given)
mean distance or perihelion distance (AU)

e0 = double (Given)
eccentricity

am0 = double (Given)
mean anomaly (radians, JFORM=2 only)

epoch1 = double * (Returned)
Epoch of elements (TT MJD)

orbi1 = double * (Returned)
inclination (radians)

anode1 = double ∗ (Returned)
longitude of the ascending node (radians)

perih1 = double ∗ (Returned)
longitude or argument of perihelion (radians)

aorq1 = double ∗ (Returned)
mean distance or perihelion distance (AU)

e1 = double ∗ (Returned)
eccentricity

am1 = double ∗ (Returned)
mean anomaly (radians, JFORM=2 only)

jstat = int ∗ (Returned)
status:

- +102 = warning, distant epoch
- +101 = warning, large timespan (> 100 years)
- +1 to +10 = coincident with planet (Note 6)
- 0 = OK
- -1 = illegal JFORM
- -2 = illegal E0
- -3 = illegal AORQ0
- -4 = internal error
- -5 = numerical error

Notes:

- Two different element-format options are available:

Option JFORM=2, suitable for minor planets:

EPOCH = epoch of elements (TT MJD) ORBI = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = mean distance, a (AU) E = eccentricity, e AM = mean anomaly M (radians)

Option JFORM=3, suitable for comets:

EPOCH = epoch of perihelion (TT MJD) ORBI = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = perihelion distance, q (AU) E = eccentricity, e

- DATE0, DATE1, EPOCH0 and EPOCH1 are all instants of time in the TT timescale (formerly Ephemeris Time, ET), expressed as Modified Julian Dates (JD-2400000.5).

DATE0 is the instant at which the given (i.e. unperturbed) osculating elements are correct.

DATE1 is the specified instant at which the updated osculating elements are correct.

EPOCH0 and EPOCH1 will be the same as DATE0 and DATE1 (respectively) for the JFORM=2 case, normally used for minor planets. For the JFORM=3 case, the two epochs will refer to perihelion passage and so will not, in general, be the same as DATE0 and/or DATE1 though they may be similar to one another.

- The elements are with respect to the J2000 ecliptic and equinox.
- Unused elements (AM0 and AM1 for JFORM=3) are not accessed.
- See the palPertue routine for details of the algorithm used.
- This routine is not intended to be used for major planets, which is why JFORM=1 is not available and why there is no opportunity to specify either the longitude of perihelion or the daily motion. However, if JFORM=2 elements are somehow obtained for a major planet and supplied to the routine, sensible results will, in fact, be produced. This happens because the palPertue routine that is called to perform the calculations checks the separation between the body and each of the planets and interprets a suspiciously small value (0.001 AU) as an attempt to apply it to the planet concerned. If this condition is detected, the contribution from that planet is ignored, and the status is set to the planet number (1-10 = Mercury, Venus, EMB, Mars, Jupiter, Saturn, Uranus, Neptune, Earth, Moon) as a warning.

See Also :

- Sterne, Theodore E., " An Introduction to Celestial Mechanics" , Interscience Publishers Inc., 1960. Section 6.7, p199.

palPertue
Update the universal elements by applying planetary perturbations

Description:

Update the universal elements of an asteroid or comet by applying planetary perturbations.

Invocation:

```
void palPertue( double date, double u[13], int *jstat );
```

Arguments:

date = double (Given)

Final epoch (TT MJD) for the update elements.

u = const double [13] (Given & Returned)

Universal orbital elements (Note 1) (0) combined mass (M+m) (1) total energy of the orbit (alpha) (2) reference (osculating) epoch (t0) (3-5) position at reference epoch (r0) (6-8) velocity at reference epoch (v0) (9) heliocentric distance at reference epoch (10) r0.v0 (11) date (t) (12) universal eccentric anomaly (psi) of date, approx

jstat = int * (Returned)

status: +102 = warning, distant epoch +101 = warning, large timespan (> 100 years) +1 to +10 = coincident with major planet (Note 5) 0 = OK

- 1 = numerical error

Notes:

- The " universal" elements are those which define the orbit for the purposes of the method of universal variables (see reference 2). They consist of the combined mass of the two bodies, an epoch, and the position and velocity vectors (arbitrary reference frame) at that epoch. The parameter set used here includes also various quantities that can, in fact, be derived from the other information. This approach is taken to avoiding unnecessary computation and loss of accuracy. The supplementary quantities are (i) alpha, which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of psi, the " universal eccentric anomaly" at a given date and (v) that date.
- The universal elements are with respect to the J2000 equator and equinox.
- The epochs DATE, U(3) and U(12) are all Modified Julian Dates (JD-2400000.5).
- The algorithm is a simplified form of Encke's method. It takes as a basis the unperturbed motion of the body, and numerically integrates the perturbing accelerations from the major planets. The expression used is essentially Sterne's 6.7-2 (reference 1). Everhart and Pitkin (reference 2) suggest rectifying the orbit at each integration step by propagating the new perturbed position and velocity as the new universal

variables. In the present routine the orbit is rectified less frequently than this, in order to gain a slight speed advantage. However, the rectification is done directly in terms of position and velocity, as suggested by Everhart and Pitkin, bypassing the use of conventional orbital elements.

The f(q) part of the full Encke method is not used. The purpose of this part is to avoid subtracting two nearly equal quantities when calculating the " indirect member" , which takes account of the small change in the Sun' s attraction due to the slightly displaced position of the perturbed body. A simpler, direct calculation in double precision proves to be faster and not significantly less accurate.

Apart from employing a variable timestep, and occasionally " rectifying the orbit" to keep the indirect member small, the integration is done in a fairly straightforward way. The acceleration estimated for the middle of the timestep is assumed to apply throughout that timestep; it is also used in the extrapolation of the perturbations to the middle of the next timestep, to predict the new disturbed position. There is no iteration within a timestep.

Measures are taken to reach a compromise between execution time and accuracy. The starting-point is the goal of achieving arcsecond accuracy for ordinary minor planets over a ten-year timespan. This goal dictates how large the timesteps can be, which in turn dictates how frequently the unperturbed motion has to be recalculated from the osculating elements.

Within predetermined limits, the timestep for the numerical integration is varied in length in inverse proportion to the magnitude of the net acceleration on the body from the major planets.

The numerical integration requires estimates of the major-planet motions. Approximate positions for the major planets (Pluto alone is omitted) are obtained from the routine palPlanet. Two levels of interpolation are used, to enhance speed without significantly degrading accuracy. At a low frequency, the routine palPlanet is called to generate updated position+velocity " state vectors" . The only task remaining to be carried out at the full frequency (i.e. at each integration step) is to use the state vectors to extrapolate the planetary positions. In place of a strictly linear extrapolation, some allowance is made for the curvature of the orbit by scaling back the radius vector as the linear extrapolation goes off at a tangent.

Various other approximations are made. For example, perturbations by Pluto and the minor planets are neglected and relativistic effects are not taken into account.

In the interests of simplicity, the background calculations for the major planets are carried out en masse. The mean elements and state vectors for all the planets are refreshed at the same time, without regard for orbit curvature, mass or proximity.

The Earth-Moon system is treated as a single body when the body is distant but as separate bodies when closer to the EMB than the parameter RNE, which incurs a time penalty but improves accuracy for near-Earth objects.

- This routine is not intended to be used for major planets. However, if major-planet elements are supplied, sensible results will, in fact, be produced. This happens because the routine checks the separation between the body and each of the planets and interprets a suspiciously small value (0.001 AU) as an attempt to apply the

routine to the planet concerned. If this condition is detected, the contribution from that planet is ignored, and the status is set to the planet number (1-10 = Mercury, Venus, EMB, Mars, Jupiter, Saturn, Uranus, Neptune, Earth, Moon) as a warning.

See Also :

- Sterne, Theodore E., " An Introduction to Celestial Mechanics" , Interscience Publishers Inc., 1960. Section 6.7, p199.
- Everhart, E. & Pitkin, E.T., Am.J.Phys. 51, 712, 1983.

palPlanel
Transform conventional elements into position and velocity

Description:

Heliocentric position and velocity of a planet, asteroid or comet, starting from orbital elements.

Invocation:

```
void palPlanel ( double date, int jform, double epoch, double orbinc, double
anode, double perih, double aorq, double e, double aorl, double dm, double pv[6],
int *jstat );
```

Arguments:

date = double (Given)
Epoch (TT MJD) of osculation (Note 1)

jform = int (Given)
Element set actually returned (1-3; Note 3)

epoch = double (Given)
Epoch of elements (TT MJD) (Note 4)

orbinc = double (Given)
inclination (radians)

anode = double (Given)
longitude of the ascending node (radians)

perih = double (Given)
longitude or argument of perihelion (radians)

aorq = double (Given)
mean distance or perihelion distance (AU)

e = double (Given)
eccentricity

aorl = double (Given)
mean anomaly or longitude (radians, JFORM=1,2 only)

dm = double (Given)
daily motion (radians, JFORM=1 only)

u = double [13] (Returned)
Universal orbital elements (Note 1) (0) combined mass (M+m) (1) total energy of the orbit (alpha) (2) reference (osculating) epoch (t0) (3-5) position at reference epoch (r0) (6-8) velocity at reference epoch (v0) (9) heliocentric distance at reference epoch (10) r0.v0 (11) date (t) (12) universal eccentric anomaly (psi) of date, approx

jstat = int * (Returned)
status: 0 = OK

- -1 = illegal JFORM
- -2 = illegal E
- -3 = illegal AORQ
- -4 = illegal DM
- -5 = numerical error

Notes:

- DATE is the instant for which the prediction is required. It is in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD-2400000.5).
- The elements are with respect to the J2000 ecliptic and equinox.
- A choice of three different element-set options is available:

Option JFORM = 1, suitable for the major planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = longitude of perihelion, curly pi (radians) AORQ = mean distance, a (AU) E = eccentricity, e (range 0 to <1) AORL = mean longitude L (radians) DM = daily motion (radians)

Option JFORM = 2, suitable for minor planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = mean distance, a (AU) E = eccentricity, e (range 0 to <1) AORL = mean anomaly M (radians)

Option JFORM = 3, suitable for comets:

EPOCH = epoch of elements and perihelion (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = perihelion distance, q (AU) E = eccentricity, e (range 0 to 10)

Unused arguments (DM for JFORM=2, AORL and DM for JFORM=3) are not accessed.

- Each of the three element sets defines an unperturbed heliocentric orbit. For a given epoch of observation, the position of the body in its orbit can be predicted from these elements, which are called " osculating elements" , using standard two-body analytical solutions. However, due to planetary perturbations, a given set of osculating elements remains usable for only as long as the unperturbed orbit that it describes is an adequate approximation to reality. Attached to such a set of elements is a date called the " osculating epoch" , at which the elements are, momentarily, a perfect representation of the instantaneous position and velocity of the body.

Therefore, for any given problem there are up to three different epochs in play, and it is vital to distinguish clearly between them:

. The epoch of observation: the moment in time for which the position of the body is to be predicted.

. The epoch defining the position of the body: the moment in time at which, in the absence of purturbations, the specified position (mean longitude, mean anomaly, or perihelion) is reached.

. The osculating epoch: the moment in time at which the given elements are correct.

For the major-planet and minor-planet cases it is usual to make the epoch that defines the position of the body the same as the epoch of osculation. Thus, only two different epochs are involved: the epoch of the elements and the epoch of observation.

For comets, the epoch of perihelion fixes the position in the orbit and in general a different epoch of osculation will be chosen. Thus, all three types of epoch are involved.

For the present routine:

. The epoch of observation is the argument DATE.

. The epoch defining the position of the body is the argument EPOCH.

. The osculating epoch is not used and is assumed to be close enough to the epoch of observation to deliver adequate accuracy. If not, a preliminary call to palPertel may be used to update the element-set (and its associated osculating epoch) by applying planetary perturbations.

- The reference frame for the result is with respect to the mean equator and equinox of epoch J2000.
- The algorithm was originally adapted from the EPHSLA program of D.H.P.Jones (private communication, 1996). The method is based on Stumpff's Universal Variables.

See Also :

Everhart, E. & Pitkin, E.T., Am.J.Phys. 51, 712, 1983.

palPlanet
Approximate heliocentric position and velocity of major planet

Description:

Calculates the approximate heliocentric position and velocity of the specified major planet.

Invocation:

```
void palPlanet ( double date, int np, double pv[6], int *j );
```

Arguments:

date = double (Given)

TDB Modified Julian Date (JD-2400000.5).

np = int (Given)

planet (1=Mercury, 2=Venus, 3=EMB, 4=Mars, 5=Jupiter, 6=Saturn, 7=Uranus, 8=Neptune)

pv = double [6] (Returned)

heliocentric x,y,z,xdot,ydot,zdot, J2000, equatorial triad in units AU and AU/s.

j = int * (Returned)

- -2 = solution didn't converge.
- -1 = illegal np (1-8)
- 0 = OK
- +1 = warning: year outside 1000-3000

Notes:

- See SOFA/ERFA eraPlan94 for details
- Note that Pluto is supported in SLA/F but not in this routine
- Status -2 is equivalent to eraPlan94 status +2.
- Note that velocity units here match the SLA/F documentation.

palPlante
Topocentric RA,Dec of a Solar-System object from heliocentric orbital elements

Description:

Topocentric apparent RA,Dec of a Solar-System object whose heliocentric orbital elements are known.

Invocation:

```
void palPlante ( double date, double elong, double phi, int jform, double epoch,
double orbinc, double anode, double perih, double aorq, double e, double aorl,
double dm, double *ra, double *dec, double *r, int *jstat );
```

Arguments:

date = double (Given)
TT MJD of observation (JD-2400000.5)

elong = double (Given)
Observer's east longitude (radians)

phi = double (Given)
Observer's geodetic latitude (radians)

jform = int (Given)
Element set actually returned (1-3; Note 6)

epoch = double (Given)
Epoch of elements (TT MJD)

orbinc = double (Given)
inclination (radians)

anode = double (Given)
longitude of the ascending node (radians)

perih = double (Given)
longitude or argument of perihelion (radians)

aorq = double (Given)
mean distance or perihelion distance (AU)

e = double (Given)
eccentricity

aorl = double (Given)
mean anomaly or longitude (radians, JFORM=1,2 only)

dm = double (Given)
daily motion (radians, JFORM=1 only)

ra = double ∗ (Returned)
Topocentric apparent RA (radians)

dec = double ∗ (Returned)
Topocentric apparent Dec (radians)

r = double ∗ (Returned)
Distance from observer (AU)

jstat = int ∗ (Returned)
status: 0 = OK

- -1 = illegal jform
- -2 = illegal e
- -3 = illegal aorq
- -4 = illegal dm
- -5 = numerical error

Notes:

- DATE is the instant for which the prediction is required. It is in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD-2400000.5).
- The longitude and latitude allow correction for geocentric parallax. This is usually a small effect, but can become important for near-Earth asteroids. Geocentric positions can be generated by appropriate use of routines palEpv (or palEvp) and palUe2pv.
- The elements are with respect to the J2000 ecliptic and equinox.
- A choice of three different element-set options is available:

Option JFORM = 1, suitable for the major planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = longitude of perihelion, curly pi (radians) AORQ = mean distance, a (AU) E = eccentricity, e (range 0 to <1) AORL = mean longitude L (radians) DM = daily motion (radians)

Option JFORM = 2, suitable for minor planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = mean distance, a (AU) E = eccentricity, e (range 0 to <1) AORL = mean anomaly M (radians)

Option JFORM = 3, suitable for comets:

EPOCH = epoch of elements and perihelion (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = perihelion distance, q (AU) E = eccentricity, e (range 0 to 10)

Unused arguments (DM for JFORM=2, AORL and DM for JFORM=3) are not accessed.

- Each of the three element sets defines an unperturbed heliocentric orbit. For a given epoch of observation, the position of the body in its orbit can be predicted from these elements, which are called " osculating elements" , using standard two-body analytical solutions. However, due to planetary perturbations, a given set of osculating elements remains usable for only as long as the unperturbed orbit that it describes is an adequate approximation to reality. Attached to such a set of elements is a date called the " osculating epoch" , at which the elements are, momentarily, a perfect representation of the instantaneous position and velocity of the body.

Therefore, for any given problem there are up to three different epochs in play, and it is vital to distinguish clearly between them:

. The epoch of observation: the moment in time for which the position of the body is to be predicted.

. The epoch defining the position of the body: the moment in time at which, in the absence of purturbations, the specified position (mean longitude, mean anomaly, or perihelion) is reached.

. The osculating epoch: the moment in time at which the given elements are correct.

For the major-planet and minor-planet cases it is usual to make the epoch that defines the position of the body the same as the epoch of osculation. Thus, only two different epochs are involved: the epoch of the elements and the epoch of observation.

For comets, the epoch of perihelion fixes the position in the orbit and in general a different epoch of osculation will be chosen. Thus, all three types of epoch are involved.

For the present routine:

. The epoch of observation is the argument DATE.

. The epoch defining the position of the body is the argument EPOCH.

. The osculating epoch is not used and is assumed to be close enough to the epoch of observation to deliver adequate accuracy. If not, a preliminary call to palPertel may be used to update the element-set (and its associated osculating epoch) by applying planetary perturbations.

- Two important sources for orbital elements are Horizons, operated by the Jet Propulsion Laboratory, Pasadena, and the Minor Planet Center, operated by the Center for Astrophysics, Harvard.

The JPL Horizons elements (heliocentric, J2000 ecliptic and equinox) correspond to PAL/SLALIB arguments as follows.

Major planets:

JFORM = 1 EPOCH = JDCT-2400000.5 ORBINC = IN (in radians) ANODE = OM (in radians) PERIH = OM+W (in radians) AORQ = A E = EC AORL = MA+OM+W (in radians) DM = N (in radians)

Epoch of osculation = JDCT-2400000.5

Minor planets:

JFORM = 2 EPOCH = JDCT-2400000.5 ORBINC = IN (in radians) ANODE = OM (in radians) PERIH = W (in radians) AORQ = A E = EC AORL = MA (in radians)

Epoch of osculation = JDCT-2400000.5

Comets:

JFORM = 3 EPOCH = Tp-2400000.5 ORBINC = IN (in radians) ANODE = OM (in radians) PERIH = W (in radians) AORQ = QR E = EC

Epoch of osculation = JDCT-2400000.5

The MPC elements correspond to SLALIB arguments as follows.

Minor planets:

JFORM = 2 EPOCH = Epoch-2400000.5 ORBINC = Incl. (in radians) ANODE = Node (in radians) PERIH = Perih. (in radians) AORQ = a E = e AORL = M (in radians)

Epoch of osculation = Epoch-2400000.5

Comets:

JFORM = 3 EPOCH = T-2400000.5 ORBINC = Incl. (in radians) ANODE = Node. (in radians) PERIH = Perih. (in radians) AORQ = q E = e

Epoch of osculation = Epoch-2400000.5

palPlantu
Topocentric RA,Dec of a Solar-System object from universal elements

Description:

Topocentric apparent RA,Dec of a Solar-System object whose heliocentric universal elements are known.

Invocation:

```
void palPlantu ( double date, double elong, double phi, const double u[13], double
*ra, double *dec, double *r, int *jstat ) {
```

Arguments:

date = double (Given)
TT MJD of observation (JD-2400000.5)

elong = double (Given)
Observer's east longitude (radians)

phi = double (Given)
Observer's geodetic latitude (radians)

u = const double [13] (Given)
Universal orbital elements

- (0) combined mass (M+m)
- (1) total energy of the orbit (alpha)
- (2) reference (osculating) epoch (t0)
- (3-5) position at reference epoch (r0)
- (6-8) velocity at reference epoch (v0)
- (9) heliocentric distance at reference epoch
- (10) r0.v0
- (11) date (t)
- (12) universal eccentric anomaly (psi) of date, approx

ra = double * (Returned)
Topocentric apparent RA (radians)

dec = double * (Returned)
Topocentric apparent Dec (radians)

r = double * (Returned)
Distance from observer (AU)

jstat = int * (Returned)
status: 0 = OK

- -1 = radius vector zero
- -2 = failed to converge

Notes:

- DATE is the instant for which the prediction is required. It is in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD-2400000.5).
- The longitude and latitude allow correction for geocentric parallax. This is usually a small effect, but can become important for near-Earth asteroids. Geocentric positions can be generated by appropriate use of routines palEpv (or palEvp) and palUe2pv.
- The " universal" elements are those which define the orbit for the purposes of the method of universal variables (see reference 2). They consist of the combined mass of the two bodies, an epoch, and the position and velocity vectors (arbitrary reference frame) at that epoch. The parameter set used here includes also various quantities that can, in fact, be derived from the other information. This approach is taken to avoiding unnecessary computation and loss of accuracy. The supplementary quantities are (i) alpha, which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of psi, the " universal eccentric anomaly" at a given date and (v) that date.
- The universal elements are with respect to the J2000 equator and equinox.

See Also :

- Sterne, Theodore E., " An Introduction to Celestial Mechanics" , Interscience Publishers Inc., 1960. Section 6.7, p199.
- Everhart, E. & Pitkin, E.T., Am.J.Phys. 51, 712, 1983.

palPm
Apply corrections for proper motion a star RA,Dec

Description:

Apply corrections for proper motion to a star RA,Dec using the SOFA/ERFA routine eraStarpm.

Invocation:

```
void palPm ( double r0, double d0, double pr, double pd, double px, double rv,
double ep0, double ep1, double *r1, double *d1 );
```

Arguments:

r0 = double (Given)
RA at epoch ep0 (radians)

d0 = double (Given)
Dec at epoch ep0 (radians)

pr = double (Given)
RA proper motion in radians per year.

pd = double (Given)
Dec proper motion in radians per year.

px = double (Given)
Parallax (arcsec)

rv = double (Given)
Radial velocity (km/sec +ve if receding)

ep0 = double (Given)
Start epoch in years, assumed to be Julian.

ep1 = double (Given)
End epoch in years, assumed to be Julian.

r1 = double * (Returned)
RA at epoch ep1 (radians)

d1 = double * (Returned)
Dec at epoch ep1 (radians)

Notes:

- Uses eraStarpm but ignores the status returns from that routine. In particular note that parallax should not be zero when the proper motions are non-zero. SLA/F allows parallax to be zero.
- Assumes all epochs are Julian epochs.

palPolmo
Correct for polar motion

Description:

Polar motion: correct site longitude and latitude for polar motion and calculate azimuth difference between celestial and terrestrial poles.

Invocation:

```
palPolmo ( double elongm, double phim, double xp, double yp, double *elong, double
*phi, double *daz );
```

Arguments:

elongm = double (Given)
Mean logitude of the observer (radians, east +ve)

phim = double (Given)
Mean geodetic latitude of the observer (radians)

xp = double (Given)
Polar motion x-coordinate (radians)

yp = double (Given)
Polar motion y-coordinate (radians)

elong = double * (Returned)
True longitude of the observer (radians, east +ve)

phi = double * (Returned)
True geodetic latitude of the observer (radians)

daz = double * (Returned)
Azimuth correction (terrestrial-celestial, radians)

Notes:

- " Mean" longitude and latitude are the (fixed) values for the site' s location with respect to the IERS terrestrial reference frame; the latitude is geodetic. TAKE CARE WITH THE LONGITUDE SIGN CONVENTION. The longitudes used by the present routine are east-positive, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the sla_OBS routine are west-positive, following astronomical usage, and must be reversed in sign before use in the present routine.
- XP and YP are the (changing) coordinates of the Celestial Ephemeris Pole with respect to the IERS Reference Pole. XP is positive along the meridian at longitude 0 degrees, and YP is positive along the meridian at longitude 270 degrees (i.e. 90 degrees west). Values for XP,YP can be obtained from IERS circulars and equivalent publications; the maximum amplitude observed so far is about 0.3 arcseconds.

- " True" longitude and latitude are the (moving) values for the site' s location with respect to the celestial ephemeris pole and the meridian which corresponds to the Greenwich apparent sidereal time. The true longitude and latitude link the terrestrial coordinates with the standard celestial models (for precession, nutation, sidereal time etc).
- The azimuths produced by sla_AOP and sla_AOPQK are with respect to due north as defined by the Celestial Ephemeris Pole, and can therefore be called " celestial azimuths" . However, a telescope fixed to the Earth measures azimuth essentially with respect to due north as defined by the IERS Reference Pole, and can therefore be called " terrestrial azimuth" . Uncorrected, this would manifest itself as a changing " azimuth zero-point error" . The value DAZ is the correction to be added to a celestial azimuth to produce a terrestrial azimuth.
- The present routine is rigorous. For most practical purposes, the following simplified formulae provide an adequate approximation:

elong = elongm+xp∗cos(elongm)-yp∗sin(elongm) phi = phim+(xp∗sin(elongm)+yp∗cos(elongm))∗tan(phi
daz = -sqrt(xp∗xp+yp∗yp)∗cos(elongm-atan2(xp,yp))/cos(phim)

An alternative formulation for DAZ is:

x = cos(elongm)∗cos(phim) y = sin(elongm)∗cos(phim) daz = atan2(-x∗yp-y∗xp,x∗x+y∗y)

- Reference: Seidelmann, P.K. (ed), 1992. " Explanatory Supplement to the Astronomical Almanac" , ISBN 0-935702-68-7, sections 3.27, 4.25, 4.52.

palPrebn
Generate the matrix of precession between two objects (old)

Description:

Generate the matrix of precession between two epochs, using the old, pre-IAU1976, Bessel-Newcomb model, using Kinoshita's formulation

Invocation:

```
void palPrebn ( double bep0, double bep1, double rmatp[3][3] );
```

Arguments:

bep0 = double (Given)
Beginning Besselian epoch.

bep1 = double (Given)
Ending Besselian epoch

rmatp = double[3][3] (Returned)
precession matrix in the sense V(BEP1) = RMATP $*$ V(BEP0)

See Also :

Kinoshita, H. (1975) 'Formulas for precession', SAO Special Report No. 364, Smithsonian Institution Astrophysical Observatory, Cambridge, Massachusetts.

palPrec
Form the matrix of precession between two epochs (IAU 2006)

Description:

The IAU 2006 precession matrix from ep0 to ep1 is found and returned. The matrix is in the sense V(EP1) = RMATP ∗ V(EP0). The epochs are TDB (loosely TT) Julian epochs.

Though the matrix method itself is rigorous, the precession angles are expressed through canonical polynomials which are valid only for a limited time span of a few hundred years around the current epoch.

Invocation:

```
palPrec( double ep0, double ep1, double rmatp[3][3] )
```

Arguments:

ep0 = double (Given)
Beginning epoch

ep1 = double (Given)
Ending epoch

rmatp = double[3][3] (Returned)
Precession matrix

palPreces
Precession - either FK4 or FK5 as required

Description:

Precess coordinates using the appropriate system and epochs.

Invocation:

```
void palPreces ( const char sys[3], double ep0, double ep1, double *ra, double
*dc );
```

Arguments:

sys = const char [3] (Given)
Precession to be applied: FK4 or FK5. Case insensitive.

ep0 = double (Given)
Starting epoch.

ep1 = double (Given)
Ending epoch

ra = double * (Given & Returned)
On input the RA mean equator & equinox at epoch ep0. On exit the RA mean equator & equinox of epoch ep1.

dec = double * (Given & Returned)
On input the dec mean equator & equinox at epoch ep0. On exit the dec mean equator & equinox of epoch ep1.

Notes:

- Uses palPrec for FK5 data and palPrebn for FK4 data.
- The epochs are Besselian if SYSTEM='FK4' and Julian if 'FK5'. For example, to precess coordinates in the old system from equinox 1900.0 to 1950.0 the call would be: palPreces(" FK4" , 1900.0, 1950.0, &ra, &dc);
- This routine will NOT correctly convert between the old and the new systems - for example conversion from B1950 to J2000. For these purposes see palFk425, palFk524, palFk45z and palFk54z.
- If an invalid SYSTEM is supplied, values of -99D0,-99D0 will be returned for both RA and DC.

palPrenut
Form the matrix of bias-precession-nutation (IAU 2006/2000A)

Description:

Form the matrix of bias-precession-nutation (IAU 2006/2000A). The epoch and date are TT (but TDB is usually close enough). The matrix is in the sense v(true) = rmatpn ∗ v(mean).

Invocation:

```
void palPrenut( double epoch, double date, double rmatpn[3][3] )
```

Arguments:

epoch = double (Returned)
Julian epoch for mean coordinates.

date = double (Returned)
Modified Julian Date (JD-2400000.5) for true coordinates.

rmatpn = double[3][3] (Returned)
combined NPB matrix

palPv2el
Position velocity to heliocentirc osculating elements

Description:

Heliocentric osculating elements obtained from instantaneous position and velocity.

Invocation:

```
void palPv2el ( const double pv[6], double date, double pmass, int jformr, int
*jform, double *epoch, double *orbinc, double *anode, double *perih, double *aorq,
double *e, double *aorl, double *dm, int *jstat );
```

Arguments:

pv = const double [6] (Given)
Heliocentric x,y,z,xdot,ydot,zdot of date, J2000 equatorial triad (AU,AU/s; Note 1)

date = double (Given)
Date (TT Modified Julian Date = JD-2400000.5)

pmass = double (Given)
Mass of the planet (Sun=1; Note 2)

jformr = int (Given)
Requested element set (1-3; Note 3)

jform = int * (Returned)
Element set actually returned (1-3; Note 4)

epoch = double * (Returned)
Epoch of elements (TT MJD)

orbinc = double * (Returned)
inclination (radians)

anode = double * (Returned)
longitude of the ascending node (radians)

perih = double * (Returned)
longitude or argument of perihelion (radians)

aorq = double * (Returned)
mean distance or perihelion distance (AU)

e = double * (Returned)
eccentricity

aorl = double * (Returned)
mean anomaly or longitude (radians, JFORM=1,2 only)

dm = double ∗ (Returned)
daily motion (radians, JFORM=1 only)

jstat = int ∗ (Returned)
status: 0 = OK

- -1 = illegal PMASS
- -2 = illegal JFORMR
- -3 = position/velocity out of range

Notes:

- The PV 6-vector is with respect to the mean equator and equinox of epoch J2000. The orbital elements produced are with respect to the J2000 ecliptic and mean equinox.
- The mass, PMASS, is important only for the larger planets. For most purposes (e.g. asteroids) use 0D0. Values less than zero are illegal.
- Three different element-format options are supported:

Option JFORM=1, suitable for the major planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = longitude of perihelion, curly pi (radians) AORQ = mean distance, a (AU) E = eccentricity, e AORL = mean longitude L (radians) DM = daily motion (radians)

Option JFORM=2, suitable for minor planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = mean distance, a (AU) E = eccentricity, e AORL = mean anomaly M (radians)

Option JFORM=3, suitable for comets:

EPOCH = epoch of perihelion (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = perihelion distance, q (AU) E = eccentricity, e

- It may not be possible to generate elements in the form requested through JFORMR. The caller is notified of the form of elements actually returned by means of the JFORM argument:

JFORMR JFORM meaning

1 1 OK - elements are in the requested format 1 2 never happens 1 3 orbit not elliptical

2 1 never happens 2 2 OK - elements are in the requested format 2 3 orbit not elliptical

3 1 never happens 3 2 never happens 3 3 OK - elements are in the requested format

- The arguments returned for each value of JFORM (cf Note 5: JFORM may not be the same as JFORMR) are as follows:

JFORM 1 2 3 EPOCH t0 t0 T ORBINC i i i ANODE Omega Omega Omega PERIH curly pi omega omega AORQ a a q E e e e AORL L M - DM n - -

where:

t0 is the epoch of the elements (MJD, TT) T " epoch of perihelion (MJD, TT) i " inclination (radians) Omega " longitude of the ascending node (radians) curly pi " longitude of perihelion (radians) omega " argument of perihelion (radians) a " mean distance (AU) q " perihelion distance (AU) e " eccentricity L " longitude (radians, 0-2pi) M " mean anomaly (radians, 0-2pi) n " daily motion (radians)

- means no value is set
- At very small inclinations, the longitude of the ascending node ANODE becomes indeterminate and under some circumstances may be set arbitrarily to zero. Similarly, if the orbit is close to circular, the true anomaly becomes indeterminate and under some circumstances may be set arbitrarily to zero. In such cases, the other elements are automatically adjusted to compensate, and so the elements remain a valid description of the orbit.
- The osculating epoch for the returned elements is the argument DATE.
- Reference: Sterne, Theodore E., " An Introduction to Celestial Mechanics" , Interscience Publishers, 1960

palPv2ue
Universal elements to position and velocity

Description:
Construct a universal element set based on an instantaneous position and velocity.

Invocation:

```
void palPv2ue( const double pv[6], double date, double pmass, double u[13], int
* jstat );
```

Arguments:

pv = double [6] (Given)
Heliocentric x,y,z,xdot,ydot,zdot of date, (AU,AU/s; Note 1)

date = double (Given)
Date (TT modified Julian Date = JD-2400000.5)

pmass = double (Given)
Mass of the planet (Sun=1; note 2)

u = double [13] (Returned)
Universal orbital elements (Note 3)

- (0) combined mass (M+m)
- (1) total energy of the orbit (alpha)
- (2) reference (osculating) epoch (t0)
- (3-5) position at reference epoch (r0)
- (6-8) velocity at reference epoch (v0)
- (9) heliocentric distance at reference epoch
- (10) r0.v0
- (11) date (t)
- (12) universal eccentric anomaly (psi) of date, approx

jstat = int * (Returned)
status: 0 = OK

- -1 = illegal PMASS
- -2 = too close to Sun
- -3 = too slow

Notes:

- The PV 6-vector can be with respect to any chosen inertial frame, and the resulting universal-element set will be with respect to the same frame. A common choice will be mean equator and ecliptic of epoch J2000.
- The mass, PMASS, is important only for the larger planets. For most purposes (e.g. asteroids) use 0D0. Values less than zero are illegal.
- The " universal" elements are those which define the orbit for the purposes of the method of universal variables (see reference). They consist of the combined mass of the two bodies, an epoch, and the position and velocity vectors (arbitrary reference frame) at that epoch. The parameter set used here includes also various quantities that can, in fact, be derived from the other information. This approach is taken to avoiding unnecessary computation and loss of accuracy. The supplementary quantities are (i) alpha, which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of psi, the " universal eccentric anomaly" at a given date and (v) that date.
- Reference: Everhart, E. & Pitkin, E.T., Am.J.Phys. 51, 712, 1983.

palPvobs
Position and velocity of an observing station

Description:

Returns the position and velocity of an observing station.

Invocation:

```
palPvobs( double p, double h, double stl, double pv[6] )
```

Arguments:

p = double (Given)
Latitude (geodetic, radians).

h = double (Given)
Height above reference spheroid (geodetic, metres).

stl = double (Given)
Local apparent sidereal time (radians).

pv = double[6] (Returned)
position/velocity 6-vector (AU, AU/s, true equator and equinox of date).

Notes:

- The WGS84 reference ellipsoid is used.

palPxy
Given arrays of " expected" and " measured" [x,y] coordinates, and a linear model relating them (as produced by palFitxy), compute the array of " predicted" coordinates and the RMS residuals

Description:

The model is supplied in the array coeffs. Naming the elements of coeffs as follows:

coeffs[0] = A
coeffs[1] = B
coeffs[2] = C
coeffs[3] = D
coeffs[4] = E
coeffs[5] = F

the model is applied thus:

$xp = A + B * xm + C * ym$
$yp = D + E * xm + F * ym$

The residuals are (xp - xe) and (yp - ye).

If np is less than or equal to zero, no coordinates are transformed, and the RMS residuals are all zero.

Invocation:

```
palPxy ( int np, double xye[][2], double xym[][2], double coeffs[6], double xyp[][2],
double *xrms, double *yrms, double *rrms )
```

Arguments:

np = int (Given)
number of samples

xye = double[np][2] (Given)
expected [x,y] for each sample

xym = double[np][2] (Given)
measured [x,y] for each sample

coeffs = double[6]
coefficients of model (see below)

xyp = double[np][2] (Returned)
predicted [x,y] for each sample

xrms = double (Returned)
RMS in x

yrms = double (Returned)
RMS in y

rrms = double (Returned)
total RMS (vector sum of xrms and yrms)

See also :

palFitxy, palInvf, palXy2xy and palDcmpf

palRanorm
Reduce angle to be within [0, 2∗pi) (single precision)

Description:

The result is " angle" expressed in the range 0 to 2∗pi.

Invocation:

```
float palRanorm ( float angle );
```

Arguments:

angle = float (Given)

Angle to be reduced (radians)

palRdplan
Approximate topocentric apparent RA,Dec of a planet

Description:

Approximate topocentric apparent RA,Dec of a planet, and its angular diameter.

Invocation:

```
void palRdplan( double date, int np, double elong, double phi, double * ra, double
* dec, double * diam );
```

Arguments:

date = double (Given)
MJD of observation (JD-2400000.5) in TDB. For all practical purposes TT can be used instead of TDB, and for many applications UT will do (except for the Moon).

np = int (Given)
Planet: 1 = Mercury 2 = Venus 3 = Moon 4 = Mars 5 = Jupiter 6 = Saturn 7 = Uranus 8 = Neptune else = Sun

elong = double (Given)
Observer's east longitude (radians)

phi = double (Given)
Observer's geodetic latitude (radians)

ra = double * (Returned)
RA (topocentric apparent, radians)

dec = double * (Returned)
Dec (topocentric apparent, radians)

diam = double * (Returned)
Angular diameter (equatorial, radians)

Notes:

- Unlike with slaRdplan, Pluto is not supported.
- The longitude and latitude allow correction for geocentric parallax. This is a major effect for the Moon, but in the context of the limited accuracy of the present routine its effect on planetary positions is small (negligible for the outer planets). Geocentric positions can be generated by appropriate use of the routines palDmoon and eraPlan94.

palRefco
Determine constants in atmospheric refraction model

Description:

Determine the constants A and B in the atmospheric refraction model dZ = A tan Z + B tan∗∗3 Z.

Z is the " observed" zenith distance (i.e. affected by refraction) and dZ is what to add to Z to give the " topocentric" (i.e. in vacuo) zenith distance.

Invocation:

```
void palRefco ( double hm, double tdk, double pmb, double rh, double wl, double
phi, double tlr, double eps, double *refa, double *refb );
```

Arguments:

hm = double (Given)
Height of the observer above sea level (metre)

tdk = double (Given)
Ambient temperature at the observer (K)

pmb = double (Given)
Pressure at the observer (millibar)

rh = double (Given)
Relative humidity at the observer (range 0-1)

wl = double (Given)
Effective wavelength of the source (micrometre)

phi = double (Given)
Latitude of the observer (radian, astronomical)

tlr = double (Given)
Temperature lapse rate in the troposphere (K/metre)

eps = double (Given)
Precision required to terminate iteration (radian)

refa = double ∗ (Returned)
tan Z coefficient (radian)

refb = double ∗ (Returned)
tan∗∗3 Z coefficient (radian)

Notes:

- Typical values for the TLR and EPS arguments might be 0.0065 and 1E-10 respectively.

- The radio refraction is chosen by specifying WL $>$ 100 micrometres.
- The routine is a slower but more accurate alternative to the palRefcoq routine. The constants it produces give perfect agreement with palRefro at zenith distances arctan(1) (45 deg) and arctan(4) (about 76 deg). It achieves 0.5 arcsec accuracy for ZD $<$ 80 deg, 0.01 arcsec accuracy for ZD $<$ 60 deg, and 0.001 arcsec accuracy for ZD $<$ 45 deg.

palRefro
Atmospheric refraction for radio and optical/IR wavelengths

Description:
Calculates the atmospheric refraction for radio and optical/IR wavelengths.

Invocation:

```
void palRefro( double zobs, double hm, double tdk, double pmb, double rh, double
wl, double phi, double tlr, double eps, double * ref ) {
```

Arguments:

zobs = double (Given)
Observed zenith distance of the source (radian)

hm = double (Given)
Height of the observer above sea level (metre)

tdk = double (Given)
Ambient temperature at the observer (K)

pmb = double (Given)
Pressure at the observer (millibar)

rh = double (Given)
Relative humidity at the observer (range 0-1)

wl = double (Given)
Effective wavelength of the source (micrometre)

phi = double (Given)
Latitude of the observer (radian, astronomical)

tlr = double (Given)
Temperature lapse rate in the troposphere (K/metre)

eps = double (Given)
Precision required to terminate iteration (radian)

ref = double * (Returned)
Refraction: in vacuao ZD minus observed ZD (radian)

Notes:

- A suggested value for the TLR argument is 0.0065. The refraction is significantly affected by TLR, and if studies of the local atmosphere have been carried out a better TLR value may be available. The sign of the supplied TLR value is ignored.

- A suggested value for the EPS argument is 1E-8. The result is usually at least two orders of magnitude more computationally precise than the supplied EPS value.
- The routine computes the refraction for zenith distances up to and a little beyond 90 deg using the method of Hohenkerk and Sinclair (NAO Technical Notes 59 and 63, subsequently adopted in the Explanatory Supplement, 1992 edition - see section 3.281).
- The code is a development of the optical/IR refraction subroutine AREF of C.Hohenkerk (HMNAO, September 1984), with extensions to support the radio case. Apart from merely cosmetic changes, the following modifications to the original HMNAO optical/IR refraction code have been made:

. The angle arguments have been changed to radians.

. Any value of ZOBS is allowed (see note 6, below).

. Other argument values have been limited to safe values.

. Murray's values for the gas constants have been used (Vectorial Astrometry, Adam Hilger, 1983).

. The numerical integration phase has been rearranged for extra clarity.

. A better model for Ps(T) has been adopted (taken from Gill, Atmosphere-Ocean Dynamics, Academic Press, 1982).

. More accurate expressions for Pwo have been adopted (again from Gill 1982).

. The formula for the water vapour pressure, given the saturation pressure and the relative humidity, is from Crane (1976), expression 2.5.5.

. Provision for radio wavelengths has been added using expressions devised by A.T.Sinclair, RGO (private communication 1989). The refractivity model currently used is from J.M.Rueger, " Refractive Index Formulae for Electronic Distance Measurement with Radio and Millimetre Waves" , in Unisurv Report S-68 (2002), School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia.

. The optical refractivity for dry air is from Resolution 3 of the International Association of Geodesy adopted at the XXIIth General Assembly in Birmingham, UK, 1999.

. Various small changes have been made to gain speed.

- The radio refraction is chosen by specifying WL $>$ 100 micrometres. Because the algorithm takes no account of the ionosphere, the accuracy deteriorates at low frequencies, below about 30 MHz.
- Before use, the value of ZOBS is expressed in the range +/- pi. If this ranged ZOBS is -ve, the result REF is computed from its absolute value before being made -ve to match. In addition, if it has an absolute value greater than 93 deg, a fixed REF value equal to the result for ZOBS = 93 deg is returned, appropriately signed.
- As in the original Hohenkerk and Sinclair algorithm, fixed values of the water vapour polytrope exponent, the height of the tropopause, and the height at which refraction is negligible are used.

- The radio refraction has been tested against work done by Iain Coulson, JACH, (private communication 1995) for the James Clerk Maxwell Telescope, Mauna Kea. For typical conditions, agreement at the 0.1 arcsec level is achieved for moderate ZD, worsening to perhaps 0.5-1.0 arcsec at ZD 80 deg. At hot and humid sea-level sites the accuracy will not be as good.
- It should be noted that the relative humidity RH is formally defined in terms of " mixing ratio" rather than pressures or densities as is often stated. It is the mass of water per unit mass of dry air divided by that for saturated air at the same temperature and pressure (see Gill 1982).
- The algorithm is designed for observers in the troposphere. The supplied temperature, pressure and lapse rate are assumed to be for a point in the troposphere and are used to define a model atmosphere with the tropopause at 11km altitude and a constant temperature above that. However, in practice, the refraction values returned for stratospheric observers, at altitudes up to 25km, are quite usable.

palRefv
Adjust an unrefracted Cartesian vector to include the effect of atmospheric refraction

Description:

Adjust an unrefracted Cartesian vector to include the effect of atmospheric refraction, using the simple A tan Z + B tan∗∗3 Z model.

Invocation:

```
void palRefv ( double vu[3], double refa, double refb, double vr[3] );
```

Arguments:

vu[3] = double (Given)
Unrefracted position of the source (Az/El 3-vector)

refa = double (Given)
tan Z coefficient (radian)

refb = double (Given)
tan∗∗3 Z coefficient (radian)

vr[3] = double (Returned)
Refracted position of the source (Az/El 3-vector)

Notes:

- This routine applies the adjustment for refraction in the opposite sense to the usual one - it takes an unrefracted (in vacuo) position and produces an observed (refracted) position, whereas the A tan Z + B tan∗∗3 Z model strictly applies to the case where an observed position is to have the refraction removed. The unrefracted to refracted case is harder, and requires an inverted form of the text-book refraction models; the algorithm used here is equivalent to one iteration of the Newton-Raphson method applied to the above formula.
- Though optimized for speed rather than precision, the present routine achieves consistency with the refracted-to-unrefracted A tan Z + B tan∗∗3 Z model at better than 1 microarcsecond within 30 degrees of the zenith and remains within 1 milliarcsecond to beyond ZD 70 degrees. The inherent accuracy of the model is, of course, far worse than this - see the documentation for palRefco for more information.
- At low elevations (below about 3 degrees) the refraction correction is held back to prevent arithmetic problems and wildly wrong results. For optical/IR wavelengths, over a wide range of observer heights and corresponding temperatures and pressures, the following levels of accuracy (arcsec, worst case) are achieved, relative to numerical integration through a model atmosphere:

ZD error

80 0.7 81 1.3 82 2.5 83 5 84 10 85 20 86 55 87 160 88 360 89 640 90 1100 91 1700 } relevant only to 92 2600 } high-elevation sites

The results for radio are slightly worse over most of the range, becoming significantly worse below ZD=88 and unusable beyond ZD=90.

- See also the routine palRefz, which performs the adjustment to the zenith distance rather than in Cartesian Az/El coordinates. The present routine is faster than palRefz and, except very low down, is equally accurate for all practical purposes. However, beyond about ZD 84 degrees palRefz should be used, and for the utmost accuracy iterative use of palRefro should be considered.

palRefz
Adjust unrefracted zenith distance

Description:

Adjust an unrefracted zenith distance to include the effect of atmospheric refraction, using the simple A tan Z + B tan∗∗3 Z model (plus special handling for large ZDs).

Invocation:

```
void palRefz ( double zu, double refa, double refb, double *zr );
```

Arguments:

zu = double (Given)
Unrefracted zenith distance of the source (radians)

refa = double (Given)
tan Z coefficient (radians)

refb = double (Given)
tan∗∗3 Z coefficient (radian)

zr = double ∗ (Returned)
Refracted zenith distance (radians)

Notes:

- This routine applies the adjustment for refraction in the opposite sense to the usual one - it takes an unrefracted (in vacuo) position and produces an observed (refracted) position, whereas the A tan Z + B tan∗∗3 Z model strictly applies to the case where an observed position is to have the refraction removed. The unrefracted to refracted case is harder, and requires an inverted form of the text-book refraction models; the formula used here is based on the Newton-Raphson method. For the utmost numerical consistency with the refracted to unrefracted model, two iterations are carried out, achieving agreement at the 1D-11 arcseconds level for a ZD of 80 degrees. The inherent accuracy of the model is, of course, far worse than this - see the documentation for palRefco for more information.
- At ZD 83 degrees, the rapidly-worsening A tan Z + B tan^3 Z model is abandoned and an empirical formula takes over. For optical/IR wavelengths, over a wide range of observer heights and corresponding temperatures and pressures, the following levels of accuracy (arcsec, worst case) are achieved, relative to numerical integration through a model atmosphere:

ZR error

80 0.7 81 1.3 82 2.4 83 4.7 84 6.2 85 6.4 86 8 87 10 88 15 89 30 90 60 91 150 } relevant only to 92 400 } high-elevation sites

For radio wavelengths the errors are typically 50% larger than the optical figures and by ZD 85 deg are twice as bad, worsening rapidly below that. To maintain 1 arcsec accuracy down to ZD=85 at the Green Bank site, Condon (2004) has suggested amplifying the amount of refraction predicted by palRefz below 10.8 deg elevation by the factor (1+0.00195∗(10.8-E_t)), where E_t is the unrefracted elevation in degrees.

The high-ZD model is scaled to match the normal model at the transition point; there is no glitch.

- Beyond 93 deg zenith distance, the refraction is held at its 93 deg value.
- See also the routine palRefv, which performs the adjustment in Cartesian Az/El coordinates, and with the emphasis on speed rather than numerical accuracy.

References :

Condon,J.J., Refraction Corrections for the GBT, PTCS/PN/35.2, NRAO Green Bank, 2004.

palRverot
Velocity component in a given direction due to Earth rotation

Description:

Calculate the velocity component in a given direction due to Earth rotation.

The simple algorithm used assumes a spherical Earth, of a radius chosen to give results accurate to about 0.0005 km/s for observing stations at typical latitudes and heights. For applications requiring greater precision, use the routine palPvobs.

Invocation:

```
double palRverot ( double phi, double ra, double da, double st );
```

Arguments:

phi = double (Given)
latitude of observing station (geodetic) (radians)

ra = double (Given)
apparent RA (radians)

da = double (Given)
apparent Dec (radians)

st = double (Given)
Local apparent sidereal time.

Returned Value:

palRverot = double
Component of Earth rotation in direction RA,DA (km/s). The result is +ve when the observatory is receding from the given point on the sky.

palRvgalc
Velocity component in a given direction due to the rotation of the Galaxy

Description:

This function returns the Component of dynamical LSR motion in the direction of R2000,D2000. The result is +ve when the dynamical LSR is receding from the given point on the sky.

Invocation:

```
double palRvgalc( double r2000, double d2000 )
```

Arguments:

r2000 = double (Given)
J2000.0 mean RA (radians)

d2000 = double (Given)
J2000.0 mean Dec (radians)

Returned Value:

Component of dynamical LSR motion in direction R2000,D2000 (km/s).

Notes:

- The Local Standard of Rest used here is a point in the vicinity of the Sun which is in a circular orbit around the Galactic centre. Sometimes called the " dynamical" LSR, it is not to be confused with a " kinematical" LSR, which is the mean standard of rest of star catalogues or stellar populations.

Reference :

- The orbital speed of 220 km/s used here comes from Kerr & Lynden-Bell (1986), MNRAS, 221, p1023.

palRvlg
Velocity component in a given direction due to Galactic rotation and motion of the local group

Description:

This function returns the velocity component in a given direction due to the combination of the rotation of the Galaxy and the motion of the Galaxy relative to the mean motion of the local group. The result is +ve when the Sun is receding from the given point on the sky.

Invocation:

```
double palRvlg( double r2000, double d2000 )
```

Arguments:

r2000 = double (Given)
J2000.0 mean RA (radians)

d2000 = double (Given)
J2000.0 mean Dec (radians)

Returned Value:

Component of SOLAR motion in direction R2000,D2000 (km/s).

Reference :

- IAU Trans 1976, 168, p201.

palRvlsrd
Velocity component in a given direction due to the Sun's motion with respect to the dynamical Local Standard of Rest

Description:

This function returns the velocity component in a given direction due to the Sun's motion with respect to the dynamical Local Standard of Rest. The result is +ve when the Sun is receding from the given point on the sky.

Invocation:

```
double palRvlsrd( double r2000, double d2000 )
```

Arguments:

r2000 = double (Given)
J2000.0 mean RA (radians)

d2000 = double (Given)
J2000.0 mean Dec (radians)

Returned Value:

Component of " peculiar" solar motion in direction R2000,D2000 (km/s).

Notes:

- The Local Standard of Rest used here is the " dynamical" LSR, a point in the vicinity of the Sun which is in a circular orbit around the Galactic centre. The Sun's motion with respect to the dynamical LSR is called the " peculiar" solar motion.
- There is another type of LSR, called a " kinematical" LSR. A kinematical LSR is the mean standard of rest of specified star catalogues or stellar populations, and several slightly different kinematical LSRs are in use. The Sun's motion with respect to an agreed kinematical LSR is known as the " standard" solar motion. To obtain a radial velocity correction with respect to an adopted kinematical LSR use the routine palRvlsrk.

Reference :

- Delhaye (1965), in " Stars and Stellar Systems" , vol 5, p73.

palRvlsrk
Velocity component in a given direction due to the Sun's motion with respect to an adopted kinematic Local Standard of Rest

Description:

This function returns the velocity component in a given direction due to the Sun's motion with respect to an adopted kinematic Local Standard of Rest. The result is +ve when the Sun is receding from the given point on the sky.

Invocation:

```
double palRvlsrk( double r2000, double d2000 )
```

Arguments:

r2000 = double (Given)
J2000.0 mean RA (radians)

d2000 = double (Given)
J2000.0 mean Dec (radians)

Returned Value:

Component of " standard" solar motion in direction R2000,D2000 (km/s).

Notes:

- The Local Standard of Rest used here is one of several " kinematical" LSRs in common use. A kinematical LSR is the mean standard of rest of specified star catalogues or stellar populations. The Sun's motion with respect to a kinematical LSR is known as the " standard" solar motion.
- There is another sort of LSR, the " dynamical" LSR, which is a point in the vicinity of the Sun which is in a circular orbit around the Galactic centre. The Sun's motion with respect to the dynamical LSR is called the " peculiar" solar motion. To obtain a radial velocity correction with respect to the dynamical LSR use the routine palRvlsrd.

Reference :

- Delhaye (1965), in " Stars and Stellar Systems" , vol 5, p73.

palSubet
Remove the E-terms from a pre IAU 1976 catalogue RA,Dec

Description:

Remove the E-terms (elliptic component of annual aberration) from a pre IAU 1976 catalogue RA,Dec to give a mean place.

Invocation:

```
void palSubet ( double rc, double dc, double eq, double *rm, double *dm );
```

Arguments:

rc = double (Given)
RA with E-terms included (radians)

dc = double (Given)
Dec with E-terms included (radians)

eq = double (Given)
Besselian epoch of mean equator and equinox

rm = double * (Returned)
RA without E-terms (radians)

dm = double * (Returned)
Dec without E-terms (radians)

Notes:

Most star positions from pre-1984 optical catalogues (or derived from astrometry using such stars) embody the E-terms. This routine converts such a position to a formal mean place (allowing, for example, comparison with a pulsar timing position).

See Also :

Explanatory Supplement to the Astronomical Ephemeris, section 2D, page 48.

palSupgal
Convert from supergalactic to galactic coordinates

Description:

Transformation from de Vaucouleurs supergalactic coordinates to IAU 1958 galactic coordinates

Invocation:

```
void palSupgal ( double dsl, double dsb, double *dl, double *db );
```

Arguments:

dsl = double (Given)
Supergalactic longitude.

dsb = double (Given)
Supergalactic latitude.

dl = double * (Returned)
Galactic longitude.

db = double * (Returned)
Galactic latitude.

See Also :

- de Vaucouleurs, de Vaucouleurs, & Corwin, Second Reference Catalogue of Bright Galaxies, U. Texas, page 8.
- Systems & Applied Sciences Corp., Documentation for the machine-readable version of the above catalogue, Contract NAS 5-26490.

(These two references give different values for the galactic longitude of the supergalactic origin. Both are wrong; the correct value is L2=137.37.)

palUe2el
Universal elements to heliocentric osculating elements

Description:

Transform universal elements into conventional heliocentric osculating elements.

Invocation:

```
void palUe2el ( const double u[13], int jformr, int *jform, double *epoch, double
*orbinc, double *anode, double *perih, double *aorq, double *e, double *aorl,
double *dm, int *jstat );
```

Arguments:

u = const double [13] (Given)

Universal orbital elements (Note 1) (0) combined mass (M+m) (1) total energy of the orbit (alpha) (2) reference (osculating) epoch (t0) (3-5) position at reference epoch (r0) (6-8) velocity at reference epoch (v0) (9) heliocentric distance at reference epoch (10) r0.v0 (11) date (t) (12) universal eccentric anomaly (psi) of date, approx

jformr = int (Given)

Requested element set (1-3; Note 3)

jform = int * (Returned)

Element set actually returned (1-3; Note 4)

epoch = double * (Returned)

Epoch of elements (TT MJD)

orbinc = double * (Returned)

inclination (radians)

anode = double * (Returned)

longitude of the ascending node (radians)

perih = double * (Returned)

longitude or argument of perihelion (radians)

aorq = double * (Returned)

mean distance or perihelion distance (AU)

e = double * (Returned)

eccentricity

aorl = double * (Returned)

mean anomaly or longitude (radians, JFORM=1,2 only)

dm = double * (Returned)

daily motion (radians, JFORM=1 only)

jstat = int ∗ (Returned)
status: 0 = OK

- 1 = illegal combined mass
- 2 = illegal JFORMR
- 3 = position/velocity out of range

Notes:

- The " universal" elements are those which define the orbit for the purposes of the method of universal variables (see reference 2). They consist of the combined mass of the two bodies, an epoch, and the position and velocity vectors (arbitrary reference frame) at that epoch. The parameter set used here includes also various quantities that can, in fact, be derived from the other information. This approach is taken to avoiding unnecessary computation and loss of accuracy. The supplementary quantities are (i) alpha, which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of psi, the " universal eccentric anomaly" at a given date and (v) that date.
- The universal elements are with respect to the mean equator and equinox of epoch J2000. The orbital elements produced are with respect to the J2000 ecliptic and mean equinox.
- Three different element-format options are supported:

Option JFORM=1, suitable for the major planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = longitude of perihelion, curly pi (radians) AORQ = mean distance, a (AU) E = eccentricity, e AORL = mean longitude L (radians) DM = daily motion (radians)

Option JFORM=2, suitable for minor planets:

EPOCH = epoch of elements (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = mean distance, a (AU) E = eccentricity, e AORL = mean anomaly M (radians)

Option JFORM=3, suitable for comets:

EPOCH = epoch of perihelion (TT MJD) ORBINC = inclination i (radians) ANODE = longitude of the ascending node, big omega (radians) PERIH = argument of perihelion, little omega (radians) AORQ = perihelion distance, q (AU) E = eccentricity, e

- It may not be possible to generate elements in the form requested through JFORMR. The caller is notified of the form of elements actually returned by means of the JFORM argument:

JFORMR JFORM meaning

1 1 OK - elements are in the requested format 1 2 never happens 1 3 orbit not elliptical

2 1 never happens 2 2 OK - elements are in the requested format 2 3 orbit not elliptical

3 1 never happens 3 2 never happens 3 3 OK - elements are in the requested format

- The arguments returned for each value of JFORM (cf Note 6: JFORM may not be the same as JFORMR) are as follows:

JFORM 1 2 3 EPOCH t0 t0 T ORBINC i i i ANODE Omega Omega Omega PERIH curly pi omega omega AORQ a a q E e e e AORL L M - DM n - -

where:

t0 is the epoch of the elements (MJD, TT) T " epoch of perihelion (MJD, TT) i " inclination (radians) Omega " longitude of the ascending node (radians) curly pi " longitude of perihelion (radians) omega " argument of perihelion (radians) a " mean distance (AU) q " perihelion distance (AU) e " eccentricity L " longitude (radians, 0-2pi) M " mean anomaly (radians, 0-2pi) n " daily motion (radians)

- means no value is set
- At very small inclinations, the longitude of the ascending node ANODE becomes indeterminate and under some circumstances may be set arbitrarily to zero. Similarly, if the orbit is close to circular, the true anomaly becomes indeterminate and under some circumstances may be set arbitrarily to zero. In such cases, the other elements are automatically adjusted to compensate, and so the elements remain a valid description of the orbit.

See Also :

- Sterne, Theodore E., " An Introduction to Celestial Mechanics" , Interscience Publishers Inc., 1960. Section 6.7, p199.
- Everhart, E. & Pitkin, E.T., Am.J.Phys. 51, 712, 1983.

palUe2pv
Heliocentric position and velocity of a planet, asteroid or comet, from universal elements

Description:

Heliocentric position and velocity of a planet, asteroid or comet, starting from orbital elements in the " universal variables" form.

Invocation:

```
void palUe2pv( double date, double u[13], double pv[6], int *jstat );
```

Arguments:

date = double (Given)
TT Modified Julian date (JD-2400000.5).

u = double [13] (Given & Returned)
Universal orbital elements (updated, see note 1) given (0) combined mass (M+m) " (1) total energy of the orbit (alpha) " (2) reference (osculating) epoch (t0) " (3-5) position at reference epoch (r0) " (6-8) velocity at reference epoch (v0) " (9) heliocentric distance at reference epoch " (10) r0.v0 returned (11) date (t) " (12) universal eccentric anomaly (psi) of date

pv = double [6] (Returned)
Position (AU) and velocity (AU/s)

jstat = int * (Returned)
status: 0 = OK

- 1 = radius vector zero
- 2 = failed to converge

Notes:

- The " universal" elements are those which define the orbit for the purposes of the method of universal variables (see reference). They consist of the combined mass of the two bodies, an epoch, and the position and velocity vectors (arbitrary reference frame) at that epoch. The parameter set used here includes also various quantities that can, in fact, be derived from the other information. This approach is taken to avoiding unnecessary computation and loss of accuracy. The supplementary quantities are (i) alpha, which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of psi, the " universal eccentric anomaly" at a given date and (v) that date.

- The companion routine is palEl2ue. This takes the conventional orbital elements and transforms them into the set of numbers needed by the present routine. A single prediction requires one one call to palEl2ue followed by one call to the present routine; for convenience, the two calls are packaged as the routine palPlanel. Multiple predictions may be made by again calling palEl2ue once, but then calling the present routine multiple times, which is faster than multiple calls to palPlanel.
- It is not obligatory to use palEl2ue to obtain the parameters. However, it should be noted that because palEl2ue performs its own validation, no checks on the contents of the array U are made by the present routine.
- DATE is the instant for which the prediction is required. It is in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD-2400000.5).
- The universal elements supplied in the array U are in canonical units (solar masses, AU and canonical days). The position and velocity are not sensitive to the choice of reference frame. The palEl2ue routine in fact produces coordinates with respect to the J2000 equator and equinox.
- The algorithm was originally adapted from the EPHSLA program of D.H.P.Jones (private communication, 1996). The method is based on Stumpff's Universal Variables.
- Reference: Everhart, E. & Pitkin, E.T., Am.J.Phys. 51, 712, 1983.

palUnpcd
Remove pincushion/barrel distortion

Description:

Remove pincushion/barrel distortion from a distorted [x,y] to give tangent-plane [x,y].

Invocation:

```
palUnpcd( double disco, double * x, double * y );
```

Arguments:

disco = double (Given)

Pincushion/barrel distortion coefficient.

x = double * (Given & Returned)

On input the distorted X coordinate, on output the tangent-plane X coordinate.

y = double * (Given & Returned)

On input the distorted Y coordinate, on output the tangent-plane Y coordinate.

Notes:

- The distortion is of the form RP = R*(1+C*R^2), where R is the radial distance from the tangent point, C is the DISCO argument, and RP is the radial distance in the presence of the distortion.
- For pincushion distortion, C is +ve; for barrel distortion, C is -ve.
- For X,Y in " radians" - units of one projection radius, which in the case of a photograph is the focal length of the camera - the following DISCO values apply:

Geometry DISCO

astrograph 0.0 Schmidt -0.3333 AAT PF doublet +147.069 AAT PF triplet +178.585 AAT f/8 +21.20 JKT f/8 +13.32

- The present routine is a rigorous inverse of the companion routine palPcd. The expression for RP in Note 1 is rewritten in the form x^3+a*x+b=0 and solved by standard techniques.
- Cases where the cubic has multiple real roots can sometimes occur, corresponding to extreme instances of barrel distortion where up to three different undistorted [X,Y]s all produce the same distorted [X,Y]. However, only one solution is returned, the one that produces the smallest change in [X,Y].

See Also :

palPcd

palVers
Obtain PAL version number

Description:

Retrieve the PAL version number as a string in the form " A.B.C" and as an integer (major∗1e6+minor∗1e3+release).

Invocation:

```
int palVers( char *verstring, size_t verlen );
```

Arguments:

verstring = char ∗ (Returned)

Buffer to receive version string of the form " A.B.C" . Can be NULL.

verlen = size_t (Given)

Allocated size of " verstring" including nul. Version string will be truncated if it does not fit in buffer.

Returned Value:

vernum = int (Returned)

Version number as an integer.

Notes:

- Note that this API does not match the slaVers API.

palXy2xy
Transform one [x,y] into another using a linear model of the type produced by the palFitxy routine

Description:

The model relates two sets of [x,y] coordinates as follows. Naming the elements of coeffs:

coeffs[0] = A
coeffs[1] = B
coeffs[2] = C
coeffs[3] = D
coeffs[4] = E
coeffs[5] = F

the present routine performs the transformation:

x2 = A + B ∗ x1 + C ∗ y1
y2 = D + E ∗ x1 + F ∗ y1

Invocation:

```
palXy2xy ( double x1, double y1, double coeffs[6], double *x2, double *y2)
```

Arguments:

x1 = double (Given)
x-coordinate

y1 = double (Given)
y-coordinate

coeffs = double[6] (Given)
transformation coefficients (see note)

x2 = double (Returned)
x-coordinate

y2 = double (Returned)
y-coordinate

See also :

palFitxy, palPxy, palInvf and palDcmpf

palCldj
Gregorian Calendar to Modified Julian Date

Description:
Gregorian calendar to Modified Julian Date.

Invocation:

```
palCldj( int iy, int im, int id, double *djm, int *j );
```

Arguments:

iy = int (Given)
Year in Gregorian calendar

im = int (Given)
Month in Gregorian calendar

id = int (Given)
Day in Gregorian calendar

djm = double * (Returned)
Modified Julian Date (JD-2400000.5) for 0 hrs

j = int * (Returned)
status: 0 = OK, 1 = bad year (MJD not computed), 2 = bad month (MJD not computed), 3 = bad day (MJD computed).

Notes:

- Uses eraCal2jd(). See SOFA/ERFA documentation for details.

palDbear
Bearing (position angle) of one point on a sphere relative to another

Description:
Bearing (position angle) of one point in a sphere relative to another.

Invocation:

```
pa = palDbear( double a1, double b1, double a2, double b2 );
```

Arguments:

a1 = double (Given)
Longitude of point 1 (e.g. RA) in radians.

b1 = double (Given)
Latitude of point 1 (e.g. Dec) in radians.

a2 = double (Given)
Longitude of point 2 in radians.

b2 = double (Given)
Latitude of point 2 in radians.

Returned Value:

The result is the bearing (position angle), in radians, of point

A2,B2 as seen from point A1,B1. It is in the range +/- pi. If

A2,B2 is due east of A1,B1 the bearing is +pi/2. Zero is returned

if the two points are coincident.

Notes:

- Uses eraPas(). See SOFA/ERFA documentation for details.

palDaf2r
Convert degrees, arcminutes, arcseconds to radians

Description:

Convert degrees, arcminutes, arcseconds to radians.

Invocation:

```
palDaf2r( int ideg, int iamin, double asec, double *rad, int *j );
```

Arguments:

ideg = int (Given)
Degrees.

iamin = int (Given)
Arcminutes.

iasec = double (Given)
Arcseconds.

rad = double * (Returned)
Angle in radians.

j = int * (Returned)
Status: 0 = OK, 1 = " ideg" out of range 0-359, 2 = " iamin" outside of range 0-59, 2 = " asec" outside range 0-59.99999

Notes:

- Uses eraAf2a(). See SOFA/ERFA documentation for details.

palDav2m
Form the rotation matrix corresponding to a given axial vector

Description:

A rotation matrix describes a rotation about some arbitrary axis, called the Euler axis. The " axial vector" supplied to this routine has the same direction as the Euler axis, and its magnitude is the amount of rotation in radians.

Invocation:

```
palDav2m( double axvec[3], double rmat[3][3] );
```

Arguments:

axvec = double [3] (Given)
Axial vector (radians)

rmat = double [3][3] (Returned)
Rotation matrix.

Notes:

- Uses eraRv2m(). See SOFA/ERFA documentation for details.

palDcc2s
Cartesian to spherical coordinates

Description:

The spherical coordinates are longitude (+ve anticlockwise looking from the +ve latitude pole) and latitude. The Cartesian coordinates are right handed, with the x axis at zero longitude and latitude, and the z axis at the +ve latitude pole.

Invocation:

```
palDcc2s( double v[3], double *a, double *b );
```

Arguments:

v = double [3] (Given)
x, y, z vector.

a = double * (Returned)
Spherical coordinate (radians)

b = double * (Returned)
Spherical coordinate (radians)

Notes:

- Uses eraC2s(). See SOFA/ERFA documentation for details.

palDcs2c
Spherical coordinates to direction cosines

Description:

The spherical coordinates are longitude (+ve anticlockwise looking from the +ve latitude pole) and latitude. The Cartesian coordinates are right handed, with the x axis at zero longitude and latitude, and the z axis at the +ve latitude pole.

Invocation:

```
palDcs2c( double a, double b, double v[3] );
```

Arguments:

a = double (Given)
Spherical coordinate in radians (ra, long etc).

b = double (Given)
Spherical coordinate in radians (dec, lat etc).

v = double [3] (Returned)
x, y, z vector

Notes:

- Uses eraS2c(). See SOFA/ERFA documentation for details.

palDd2tf
Convert an interval in days into hours, minutes, seconds

Description:
Convert and interval in days into hours, minutes, seconds.

Invocation:

```
palDd2tf( int ndp, double days, char *sign, int ihmsf[4] );
```

Arguments:

ndp = int (Given)
Number of decimal places of seconds

days = double (Given)
Interval in days

sign = char * (Returned)
' +' or ' -' (single character, not string)

ihmsf = int [4] (Returned)
Hours, minutes, seconds, fraction

Notes:

- Uses eraD2tf(). See SOFA/ERFA documentation for details.

palDimxv
Perform the 3-D backward unitary transformation

Description:
Perform the 3-D backward unitary transformation.

Invocation:

```
palDimxv( double dm[3][3], double va[3], double vb[3] );
```

Arguments:

dm = double [3][3] (Given)
Matrix

va = double [3] (Given)
vector

vb = double [3] (Returned)
Result vector

Notes:

- Uses eraTrxp(). See SOFA/ERFA documentation for details.

palDm2av
From a rotation matrix, determine the corresponding axial vector

Description:

A rotation matrix describes a rotation about some arbitrary axis, called the Euler axis. The " axial vector" returned by this routine has the same direction as the Euler axis, and its magnitude is the amount of rotation in radians. (The magnitude and direction can be separated by means of the routine palDvn.)

Invocation:

```
palDm2av( double rmat[3][3], double axvec[3] );
```

Arguments:

rmat = double [3][3] (Given)
Rotation matrix

axvec = double [3] (Returned)
Axial vector (radians)

Notes:

- Uses eraRm2v(). See SOFA/ERFA documentation for details.

palDjcl
Modified Julian Date to Gregorian year, month, day and fraction of day

Description:
Modified Julian Date to Gregorian year, month, day and fraction of day.

Invocation:

```
palDjcl( double djm, int *iy, int *im, int *id, double *fd, int *j );
```

Arguments:

djm = double (Given)
modified Julian Date (JD-2400000.5)

iy = int * (Returned)
year

im = int * (Returned)
month

id = int * (Returned)
day

fd = double * (Returned)
Fraction of day.

Notes:

- Uses eraJd2cal(). See SOFA/ERFA documentation for details.

palDmxm
Product of two 3x3 matrices

Description:
Product of two 3x3 matrices.

Invocation:

```
palDmxm( double a[3][3], double b[3][3], double c[3][3] );
```

Arguments:

a = double [3][3] (Given)
Matrix

b = double [3][3] (Given)
Matrix

c = double [3][3] (Returned)
Matrix result

Notes:

- Uses eraRxr(). See SOFA/ERFA documentation for details.

palDmxv
Performs the 3-D forward unitary transformation

Description:
Performs the 3-D forward unitary transformation.

Invocation:

```
palDmxv( double dm[3][3], double va[3], double vb[3] );
```

Arguments:

dm = double [3][3] (Given)
matrix

va = double [3] (Given)
vector

dp = double [3] (Returned)
result vector

Notes:

- Uses eraRxp(). See SOFA/ERFA documentation for details.

palDpav
Position angle of one celestial direction with respect to another

Description:
Position angle of one celestial direction with respect to another.

Invocation:

```
pa = palDpav( double v1[3], double v2[3] );
```

Arguments:

v1 = double [3] (Given)
direction cosines of one point.

v2 = double [3] (Given)
direction cosines of the other point.

Returned Value:

The result is the bearing (position angle), in radians, of point

V2 with respect to point V1. It is in the range +/- pi. The

sense is such that if V2 is a small distance east of V1, the

bearing is about +pi/2. Zero is returned if the two points

are coincident.

Notes:

- The coordinate frames correspond to RA,Dec, Long,Lat etc.
- Uses eraPap(). See SOFA/ERFA documentation for details.

palDr2af
Convert an angle in radians to degrees, arcminutes, arcseconds

Description:
Convert an angle in radians to degrees, arcminutes, arcseconds.

Invocation:

```
palDr2af( int ndp, double angle, char *sign, int idmsf[4] );
```

Arguments:

ndp = int (Given)
number of decimal places of arcseconds

angle = double (Given)
angle in radians

sign = char * (Returned)
' +' or ' -' (single character)

idmsf = int [4] (Returned)
Degrees, arcminutes, arcseconds, fraction

Notes:

- Uses eraA2af(). See SOFA/ERFA documentation for details.

palDr2tf
Convert an angle in radians to hours, minutes, seconds

Description:
Convert an angle in radians to hours, minutes, seconds.

Invocation:

```
palDr2tf ( int ndp, double angle, char *sign, int ihmsf[4] );
```

Arguments:

ndp = int (Given)
number of decimal places of arcseconds

angle = double (Given)
angle in radians

sign = char * (Returned)
' +' or ' -' (single character)

idmsf = int [4] (Returned)
Hours, minutes, seconds, fraction

Notes:

- Uses eraA2tf(). See SOFA/ERFA documentation for details.

palDranrm
Normalize angle into range 0-2 pi

Description:
Normalize angle into range 0-2 pi.

Invocation:

```
norm = palDranrm( double angle );
```

Arguments:

angle = double (Given)
angle in radians

Returned Value:

Angle expressed in the range 0-2 pi

Notes:

- Uses eraAnp(). See SOFA/ERFA documentation for details.

palDsep
Angle between two points on a sphere

Description:

Angle between two points on a sphere.

Invocation:

```
ang = palDsep( double a1, double b1, double a2, double b2 );
```

Arguments:

a1 = double (Given)
Spherical coordinate of one point (radians)

b1 = double (Given)
Spherical coordinate of one point (radians)

a2 = double (Given)
Spherical coordinate of other point (radians)

b2 = double (Given)
Spherical coordinate of other point (radians)

Returned Value:

Angle, in radians, between the two points. Always positive.

Notes:

- The spherical coordinates are [RA,Dec], [Long,Lat] etc, in radians.
- Uses eraSeps(). See SOFA/ERFA documentation for details.

palDsepv
Angle between two vectors

Description:
Angle between two vectors.

Invocation:

```
ang = palDsepv( double v1[3], double v2[3] );
```

Arguments:

v1 = double [3] (Given)
First vector

v2 = double [3] (Given)
Second vector

Returned Value:

Angle, in radians, between the two points. Always positive.

Notes:

- Uses eraSepp(). See SOFA/ERFA documentation for details.

palDtf2d
Convert hours, minutes, seconds to days

Description:
Convert hours, minutes, seconds to days.

Invocation:

```
palDtf2d( int ihour, int imin, double sec, double *days, int *j );
```

Arguments:

ihour = int (Given)
Hours

imin = int (Given)
Minutes

sec = double (Given)
Seconds

days = double * (Returned)
Interval in days

j = int * (Returned)
status: 0 = ok, 1 = ihour outside range 0-23, 2 = imin outside range 0-59, 3 = sec outside range 0-59.999...

Notes:

- Uses eraTf2d(). See SOFA/ERFA documentation for details.

palDtf2r
Convert hours, minutes, seconds to radians

Description:

Convert hours, minutes, seconds to radians.

Invocation:

```
palDtf2r( int ihour, int imin, double sec, double *rad, int *j );
```

Arguments:

ihour = int (Given)
Hours

imin = int (Given)
Minutes

sec = double (Given)
Seconds

days = double * (Returned)
Angle in radians

j = int * (Returned)
status: 0 = ok, 1 = ihour outside range 0-23, 2 = imin outside range 0-59, 3 = sec outside range 0-59.999...

Notes:

- Uses eraTf2a(). See SOFA/ERFA documentation for details.

palDvdv
Scalar product of two 3-vectors

Description:

Scalar product of two 3-vectors.

Invocation:

```
prod = palDvdv ( double va[3], double vb[3] );
```

Arguments:

va = double [3] (Given)
First vector

vb = double [3] (Given)
Second vector

Returned Value:

Scalar product va.vb

Notes:

- Uses eraPdp(). See SOFA/ERFA documentation for details.

palDvn
Normalizes a 3-vector also giving the modulus

Description:

Normalizes a 3-vector also giving the modulus.

Invocation:

```
palDvn( double v[3], double uv[3], double *vm );
```

Arguments:

v = double [3] (Given)
vector

uv = double [3] (Returned)
unit vector in direction of " v"

vm = double * (Returned)
modulus of " v"

Notes:

- Uses eraPn(). See SOFA/ERFA documentation for details.

palDvxv
Vector product of two 3-vectors

Description:
Vector product of two 3-vectors.

Invocation:

```
palDvxv( double va[3], double vb[3], double vc[3] );
```

Arguments:

va = double [3] (Given)
First vector

vb = double [3] (Given)
Second vector

vc = double [3] (Returned)
Result vector

Notes:

- Uses eraPxp(). See SOFA/ERFA documentation for details.

palEpb
Conversion of modified Julian Data to Besselian Epoch

Description:

Conversion of modified Julian Data to Besselian Epoch.

Invocation:

```
epb = palEpb ( double date );
```

Arguments:

date = double (Given)

Modified Julian Date (JD - 2400000.5)

Returned Value:

Besselian epoch.

Notes:

- Uses eraEpb(). See SOFA/ERFA documentation for details.

palEpb2d
Conversion of Besselian Epoch to Modified Julian Date

Description:

Conversion of Besselian Epoch to Modified Julian Date.

Invocation:

```
mjd = palEpb2d ( double epb );
```

Arguments:

epb = double (Given)
Besselian Epoch

Returned Value:

Modified Julian Date (JD - 2400000.5)

Notes:

- Uses eraEpb2jd(). See SOFA/ERFA documentation for details.

palEpj
Conversion of Modified Julian Date to Julian Epoch

Description:

Conversion of Modified Julian Date to Julian Epoch.

Invocation:

```
epj = palEpj ( double date );
```

Arguments:

date = double (Given)

Modified Julian Date (JD - 2400000.5)

Returned Value:

The Julian Epoch.

Notes:

- Uses eraEpj(). See SOFA/ERFA documentation for details.

palEpj2d
Conversion of Julian Epoch to Modified Julian Date

Description:
Conversion of Julian Epoch to Modified Julian Date.

Invocation:

```
mjd = palEpj2d ( double epj );
```

Arguments:

epj = double (Given)
Julian Epoch.

Returned Value:

Modified Julian Date (JD - 2400000.5)

Notes:

- Uses eraEpj2d(). See SOFA/ERFA documentation for details.

palEqeqx
Equation of the equinoxes (IAU 2000/2006)

Description:

Equation of the equinoxes (IAU 2000/2006).

Invocation:

```
palEqeqx( double date );
```

Arguments:

date = double (Given)

TT as Modified Julian Date (JD-400000.5)

Notes:

- Uses eraEe06a(). See SOFA/ERFA documentation for details.

palFk5hz
Transform an FK5 (J2000) star position into the frame of the Hipparcos catalogue

Description:

Transform an FK5 (J2000) star position into the frame of the Hipparcos catalogue.

Invocation:

```
palFk5hz ( double r5, double d5, double epoch, double *rh, double *dh );
```

Arguments:

r5 = double (Given)
FK5 RA (radians), equinox J2000, epoch " epoch"

d5 = double (Given)
FK5 dec (radians), equinox J2000, epoch " epoch"

epoch = double (Given)
Julian epoch

rh = double * (Returned)
RA (radians)

dh = double * (Returned)
Dec (radians)

Notes:

- Assumes zero Hipparcos proper motion.
- Uses eraEpj2jd() and eraFk5hz. See SOFA/ERFA documentation for details.

palGmst
Greenwich mean sidereal time (consistent with IAU 2006 precession)

Description:

Greenwich mean sidereal time (consistent with IAU 2006 precession).

Invocation:

```
mst = palGmst ( double ut1 );
```

Arguments:

ut1 = double (Given)

Universal time (UT1) expressed as modified Julian Date (JD-2400000.5)

Returned Value:

Greenwich mean sidereal time

Notes:

- Uses eraGmst06(). See SOFA/ERFA documentation for details.

palGmsta
Greenwich mean sidereal time (consistent with IAU 2006 precession)

Description:

Greenwich mean sidereal time (consistent with IAU 2006 precession).

Invocation:

```
mst = palGmsta ( double date, double ut1 );
```

Arguments:

date = double (Given)
UT1 date (MJD: integer part of JD-2400000.5)

ut1 = double (Given)
UT1 time (fraction of a day)

Returned Value:

Greenwich mean sidereal time (in range 0 to 2 pi)

Notes:

- For best accuracy use eraGmst06() directly.
- Uses eraGmst06(). See SOFA/ERFA documentation for details.

palHfk5z
Hipparcos star position to FK5 J2000

Description:

Transform a Hipparcos star position into FK5 J2000, assuming zero Hipparcos proper motion.

Invocation:

```
palHfk5z( double rh, double dh, double epoch, double *r5, double *d5, double
*dr5, double *dd5 );
```

Arguments:

rh = double (Given)
Hipparcos RA (radians)

dh = double (Given)
Hipparcos Dec (radians)

epoch = double (Given)
Julian epoch (TDB)

r5 = double * (Returned)
RA (radians, FK5, equinox J2000, epoch " epoch")

d5 = double * (Returned)
Dec (radians, FK5, equinox J2000, epoch " epoch")

Notes:

- Uses eraEpj2jd and eraHfk5z(). See SOFA/ERFA documentation for details.

palRefcoq
Determine the constants A and B in the atmospheric refraction model

Description:

Determine the constants A and B in the atmospheric refraction model dZ = A tan Z + B tan∗∗3 Z. This is a fast alternative to the palRefco routine.

Z is the " observed" zenith distance (i.e. affected by refraction) and dZ is what to add to Z to give the " topocentric" (i.e. in vacuo) zenith distance.

Invocation:

```
palRefcoq( double tdk, double pmb, double rh, double wl, double *refa, double
*refb );
```

Arguments:

tdk = double (Given)
Ambient temperature at the observer (K)

pmb = double (Given)
Pressure at the observer (millibar)

rh = double (Given)
Relative humidity at the observer (range 0-1)

wl = double (Given)
Effective wavelength of the source (micrometre). Radio refraction is chosen by specifying wl $>$ 100 micrometres.

refa = double ∗ (Returned)
tan Z coefficient (radian)

refb = double ∗ (Returned)
tan∗∗3 Z coefficient (radian)

Notes:

- Uses eraRefco(). See SOFA/ERFA documentation for details.
- Note that the SOFA/ERFA routine uses different order of of arguments and uses deg C rather than K.