Joint Astronomy Centre
Science & Technology Facilities Council
Starlink Project
Starlink User Note 267.1

SUN/267.1

Tim Jenness
2012 March 23

PAL — Positional Astronomy Library

0.1.0

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 library. It is provided with a GPL license.

1 Introduction

This library provides a C library designed as a API-compatible replacement for the C SLALIB library () and uses a GPL licence so is freely redistributable. Where possible the functions call equivalent SOFA routines 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.

A Function Descriptions

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*

A.2 More complex functions

These functions do not have a simple equivalent in SOFA so are reimplemented either completely standalone or using multiple SOFA functions.

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

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 palAirmas

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.

---

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

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 palAmpqk

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) not used (7) not used (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).

---

palCaldj Gregorian Calendar to Modified Julian Date palCaldj

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 iauCal2jd for other values.

Notes:

- Uses iauCal2jd
- Unlike iauCal2jd 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 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 palDafin

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

---

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

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 palDeuler

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 palDfltin

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).

---

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

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 palDjcal

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 iauJd2cal for other values.

Notes:

- Uses iauJd2cal

palDmat Matrix inversion & solution of simultaneous equations palDmat

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)
The vector of N unknowns. 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.

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 palDs2tp

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

---

palDtt Return offset between UTC and TT palDtt

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 iauDat function.
- This routine matches the slaDat interface which differs from the iauDat interface. Consider coding directly to the SOFA interface.
- See iauDat for a description of error conditions when calling this function with a time outside of the UTC range.
- The status argument from iauDat 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 palDmoon

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 palDrange

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 palDt

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 palDtp2s

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 palDtps2c

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 — palDtt

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 iauDat for a description of error conditions when calling this function with a time outside of the UTC range. This behaviour differs from slaDtt.

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

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 — palEl2ue

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' palEpco

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 — palEpv

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 iauEpv00 for details on accuracy
- Note that the status argument from iauEpv00 is ignored

---

palEtrms — Compute the E-terms vector — palEtrms

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 palEqecl

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 palEqgal

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 palEvp

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 iauEpv00.

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)

---

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

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 palFk524

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 — palFk54z

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 palGaleq

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 palGalsup

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 palGe50

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 palGeoc

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 iauGd2gc

---

palIntin Convert free-format input into an integer palIntin

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.

---

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

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.

---

palMappa Compute parameters needed by palAmpqk and palMapqk palMappa

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) (grav rad Sun)∗2/(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 palMapqk

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.

palMapqkz Quick mean to apparent place palMapqkz

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.

The reference systems and timescales used are IAU 2006.

Strictly speaking, the function is not valid for solar-system sources, though the error will usually be extremely small.

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) not used (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).

---

palNut — Form the matrix of nutation — palNut

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 iauNut06a 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 palNutc

Description: Calculates the longitude ∗ obliquity components and mean obliquity using the SOFA 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 iauObl06 and iauNut06a and therefore uses the IAU 206 precession/nutation model.
- Note the change from SLA/F regarding the date. TT is used rather than TDB.

palObs Parameters of selected ground-based observing stations palObs

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 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 — palPa

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.

---

palPertel — Update elements by applying planetary perturbations — palPertel

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 sla_PERTUE 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 palPertue

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 palPlanel

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 sla_PERTEL 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 — palPlanet

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 iauPlan94 for details
- Note that Pluto is supported in SLA/F but not in this routine
- Status -2 is equivalent to iauPlan94 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 palPlante

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 sla_PERTEL 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 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 palPlantu

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 palPm

Description: Apply corrections for proper motion to a star RA,Dec using the SOFA routine iauStarpm.

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 iauStarpm 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.

---

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

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) palPrec

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 palPreces

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) palPrenut

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 palPv2el

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 palPv2ue

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 palPvobs

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.

---

palRdplan Approximate topocentric apparent RA,Dec of a planet palRdplan

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 iauPlan94.

---

palRverot Velocity component in a given direction due to Earth rotation palRverot

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)

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 palRvgalc

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 palRvlg

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 palRvlsrd

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 sla_RVLSRK.

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 — palRvlsrk

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 sla_RVLSRD.

Reference:

- Delhaye (1965), in "Stars and Stellar Systems", vol 5, p73.

palSubet — Remove the E-terms from a pre IAU 1976 catalogue RA,Dec — palSubet

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 palSupgal

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 palUe2el

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 palUe2pv

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

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 sla_PLANEL. 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. in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD-2400000.5). 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.