

SLALIB

SLALIB/C Programmer's Manual


Patrick Wallace

Software version 9.0616

Document revision 3.79

2nd September 2009

Published by:

Tpoint  Software

www.tpsoft.demon.co.uk

TCSpkTM and TPOINTTM are trademarks.

Information in this document is subject to change without notice.

© Copyright 2009 Tpoint Software, UK, and P.T. Wallace. All Rights Reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws. Transmission of text or content to third parties, including disclosure via the Internet, is prohibited.

Contents

Preface	iv
Acknowledgements	iv
1 INTRODUCTION	1
1.1 Purpose	1
1.2 Example Application	1
1.3 Scope	2
1.4 Objectives	3
1.5 New Functions	4
2 COMPILING AND LINKING	5
3 EXPLANATION AND EXAMPLES	6
3.1 Spherical Trigonometry	6
3.1.1 Formatting angles	8
3.2 Vectors and Matrices	10
3.2.1 Using vectors	11
3.3 Celestial Coordinate Systems	13
3.4 Celestial coordinates, old and new	14
3.4.1 Frame bias	18
3.4.2 Equation of the origins	18
3.5 Classical Precession and Nutation	19
3.5.1 Precession-nutation and the CIO	20
3.6 SLALIB Support for Classical Precession	21
3.6.1 Mean places, and epoch	22
3.7 SLALIB Support for Classical Nutation	23
3.8 High Precision Precession-Nutation	23
3.9 Proper Motion	25
3.10 Parallax and Radial Velocity	26
3.11 Aberration	27
3.12 Different Sorts of Mean Place	28
3.13 Mean Place Transformations	29
3.14 Mean Place to Apparent Place	31
3.15 Apparent Place to Observed Place	33
3.15.1 Refraction	34
3.15.2 Efficiency considerations	35
3.16 The Hipparcos Catalogue and the ICRS	36
3.17 CIO based ICRS-CIRS-Observed Transformations	36
3.18 Time-scales	40
3.18.1 Atomic Time: TAI	41
3.18.2 Universal Time: UT, UTC	43
3.18.3 Sidereal Time: GMST, LAST <i>etc.</i>	44
3.18.4 Dynamical Time: TT, TDB	44
3.19 Calendars	46
3.20 Geocentric Coordinates	47
3.21 Ephemerides	48
3.22 Radial Velocity and Light-Time Corrections	61

3.23 Focal-Plane Astrometry	62
3.24 Numerical Methods	63
4 SUBPROGRAM SPECIFICATIONS	68
slaAddet: add E-terms of aberration.	68
slaAfin: sexagesimal character string to angle.	68
slaAirmas: air mass.	70
slaAltaz: velocities <i>etc.</i> for altazimuth mount.	71
slaAmp: apparent to mean.	72
slaAmpqk: quick apparent to mean.	73
slaAop: apparent to observed.	74
slaAoppa: apparent-to-observed parameters.	77
slaAoppat: update apparent-to-observed parameters.	79
slaAopqk: quick apparent to observed.	79
slaAtmdsp: atmospheric dispersion.	81
slaAv2m: rotation matrix from axial vector.	83
slaBear: direction between points on a sphere.	83
slaC2i: ICRS to CIRS.	84
slaC2ipa: ICRS to CIRS parameters.	85
slaC2ipad: make ICRS to CIRS parameters stop at GCRS.	86
slaC2ipas: ICRS to CIRS parameters, space observer.	87
slaC2iqk: quick ICRS to CIRS.	88
slaC2iqkz: quick ICRS to CIRS, no P.M. <i>etc.</i>	89
slaCaf2r: deg,arcmin,arcsec to radians.	90
slaCaldj: calendar date to MJD.	91
slaCalyd: calendar date to year,day.	91
slaCc2s: Cartesian to spherical.	92
slaCc62s: Cartesian 6-vector to spherical.	93
slaCd2tf: days to hour,min,sec.	93
slaCldj: calendar date to MJD.	94
slaClyd: calendar date to year,day.	95
slaCombn: next combination.	95
slaCr2af: radians to deg,arcmin,arcsec.	96
slaCr2tf: radians to hour,min,sec.	97
slaCs2c: spherical to Cartesian.	98
slaCs2c6: spherical position and velocity to Cartesian.	98
slaCtf2d: hour,min,sec to days.	99
slaCtf2r: hour,min,sec to radians.	99
slaDaf2r: deg,arcmin,arcsec to radians.	100
slaDafin: sexagesimal character string to angle.	100
slaDat: TAI–UTC.	102
slaDav2m: rotation matrix from axial vector.	103
slaDbear: direction between points on a sphere.	103
slaDbjin: decode string to B/J epoch.	104
slaDc62s: Cartesian 6-vector to spherical.	105
slaDcc2s: Cartesian to spherical.	105
slaDcmpf: interpret linear fit.	106
slaDcs2c: spherical to Cartesian.	107

slaDd2tf: days to hour,min,sec.	108
slaDe2h: $[h, \delta]$ to $[Az, El]$	108
slaDeuler: Euler angles to rotation matrix.	109
slaDfltin: decode a double precision number.	110
slaDh2e: $[Az, El]$ to $[h, \delta]$	112
slaDimxv: apply 3D reverse rotation.	112
slaDjcal: MJD to Gregorian for output.	113
slaDjcl: MJD to year,month,day,frac.	114
slaDm2av: rotation matrix to axial vector.	114
slaDmat: solve simultaneous equations.	115
slaDmoon: approximate Moon position and velocity.	116
slaDmxm: product of two 3×3 matrices.	117
slaDmxv: apply 3D rotation.	117
slaDpav: position-angle between two directions.	118
slaDr2af: radians to deg,min,sec,frac.	118
slaDr2tf: radians to hour,min,sec,frac.	119
slaDrange: put angle into range $\pm\pi$	120
slaDranrm: put angle into range $0-2\pi$	120
slaDs2c6: spherical position and velocity to Cartesian.	121
slaDs2tp: spherical to tangent plane.	121
slaDsep: angle between two points on a sphere.	122
slaDsepv: angle between two vectors.	122
slaDt: approximate ET minus UT.	123
slaDtf2d: hour,min,sec to days.	124
slaDtf2r: hour,min,sec to radians.	124
slaDtp2s: tangent plane to spherical.	125
slaDtp2v: tangent plane to $[x, y, z]$	125
slaDtps2c: plate centre from $[\xi, \eta]$ and $[\alpha, \delta]$	126
slaDtpv2c: plate centre from $[\xi, \eta]$ and $[x, y, z]$	127
slaDtt: TT minus UTC.	128
slaDv2tp: $[x, y, z]$ to tangent plane.	129
slaDvdv: scalar product.	129
slaDvn: normalize vector.	130
slaDvxv: vector product.	130
slaE2h: $[h, \delta]$ to $[Az, El]$	130
slaEarth: approximate Earth position and velocity.	131
slaEclep: ecliptic to equatorial.	132
slaEcmat: form $[\alpha, \delta] \rightarrow [\lambda, \beta]$ matrix.	132
slaEcor: radial velocity & light-time corrections to Sun.	133
slaEg50: B1950 $[\alpha, \delta]$ to $[l^{\text{II}}, b^{\text{II}}]$	134
slaEl2ue: conventional to universal elements.	134
slaEo: equation of the origins.	137
slaEors: EO from NPB matrix and s	137
slaEpb: MJD to Besselian epoch.	138
slaEpb2d: Besselian epoch to MJD.	138
slaEpc: convert epoch to B or J.	139
slaEpj: MJD to Julian epoch.	139
slaEpj2d: Julian epoch to MJD.	140

slaEpv: Earth position & velocity (high accuracy).	140
slaEqec1: J2000 $[\alpha, \delta]$ to $[\lambda, \beta]$.	141
slaEqeqx: equation of the equinoxes.	142
slaEqgal: J2000 $[\alpha, \delta]$ to $[l^II, b^II]$.	142
slaEra: Earth rotation angle.	143
slaEtrms: E-terms of aberration.	144
slaEuler: rotation matrix from Euler angles.	144
slaEvp: Earth position & velocity.	145
slaFitxy: fit linear model to two $[x, y]$ sets.	146
slaFk425: FK4 to FK5.	147
slaFk45z: FK4 to FK5, no proper motion or parallax.	149
slaFk524: FK5 to FK4.	150
slaFk52h: FK5 to Hipparcos.	151
slaFk54z: FK5 to FK4, no proper motion or parallax.	153
slaFk5hz: FK5 to Hipparcos, no proper motion.	154
slaFlotin: decode a single-precision real number.	155
slaFw2m: rotation matrix from F-W angles.	156
slaFw2xy: CIP X, Y given F-W NPB angles.	157
slaG2ixys: GCRS to CIRS matrix given X, Y, s .	158
slaGaleq: galactic to J2000 $[\alpha, \delta]$.	159
slaGalsup: galactic to supergalactic.	160
slaGe50: galactic to B1950 $[\alpha, \delta]$.	160
slaGeoc: geodetic to geocentric.	161
slaGmst: UT to GMST.	161
slaGmsta: UT to GMST (extra precision).	162
slaGst: GST (from ERA and EO).	163
slaH2e: $[Az, El]$ to $[h, \delta]$.	164
slaH2fk5: Hipparcos to FK5.	164
slaHfk5z: Hipparcos to FK5, no proper motion.	165
slaI2c: CIRS to ICRS.	167
slaI2cqk: quick CIRS to ICRS.	167
slaI2o: CIRS to observed place.	168
slaI2opa: CIRS-to-observed parameters.	171
slaI2opad: in CIRS-to-observed, disable diurnal aberration.	173
slaI2opat: update CIRS-to-observed parameters.	175
slaI2oqk: quick CIRS to observed place.	176
slaImxv: apply 3D reverse rotation.	178
slaInt2in: decode an integer number.	178
slaIntin: decode a long integer number.	180
slaInvf: invert linear model.	181
slaKbj: select epoch prefix.	182
slaM2av: rotation matrix to axial vector.	182
slaMap: mean to apparent.	183
slaMappa: mean to apparent parameters.	184
slaMapqk: quick mean to apparent.	185
slaMapqkz: quick mean to apparent, no P.M. <i>etc.</i>	187
slaMoon: approximate Moon position and velocity.	188
slaMxm: product of two 3×3 matrices.	188

slaMxv: apply 3D rotation.	189
slaNu00a: nutation, IAU 2000A.	189
slaNu: nutation components, IAU 2006.	191
slaNut: nutation matrix.	191
slaNutc: nutation components.	192
slaNutc80: nutation components, IAU 1980.	193
sla02i: observed place to CIRS.	194
sla02iqk: quick observed place to CIRS.	196
sla0ap: observed to apparent.	198
sla0apqk: quick observed to apparent.	201
slaObs: observatory parameters.	203
slaPa: $[h, \delta]$ to parallactic angle.	206
slaPav: position-angle between two directions.	206
slaPcd: apply radial distortion.	207
slaPda2h: hour angle for a given azimuth.	208
slaPdq2h: hour angle for a given parallactic angle.	208
slaPermut: next permutation.	209
slaPertel: perturbed orbital elements.	210
slaPertue: perturbed universal elements.	212
slaPfw: IAU 2006 precession angles.	215
slaPlanel: planet position from elements.	216
slaPlanet: planetary ephemerides, approximate.	218
slaPlante: $[\alpha, \delta]$ of planet from elements.	220
slaPlantu: $[\alpha, \delta]$ from universal elements.	223
slaPm: proper motion.	224
slaPncio: CIO based NPB matrix.	225
slaPneqx: equinox based NPB matrix.	226
slaPolmo: polar motion.	226
slaPomom: polar motion matrix.	228
slaPrebn: precession matrix (FK4).	229
slaPrec: precession matrix (IAU 1976).	229
slaPreces: precession.	230
slaPrecl: precession matrix (more accurate).	231
slaPrenut: precession-nutation matrix.	232
slaPv2el: orbital elements from position & velocity.	232
slaPv2ue: position and velocity to universal elements.	235
slaPvobs: observatory position & velocity.	236
slaPxy: apply linear model.	237
slaRange: put angle into range $\pm\pi$	237
slaRanorm: put angle into range $0-2\pi$	238
slaRcc: barycentric coordinate time.	238
slaRdplan: apparent $[\alpha, \delta]$ of planet.	240
slaRefco: refraction constants.	241
slaRefcoq: refraction constants (fast).	242
slaRefro: refraction.	245
slaRefv: apply refraction to vector.	248
slaRefz: apply refraction to ZD.	249
slaRverot: radial velocity correction to Earth centre.	251

slaRvgalc: radial velocity correction to galactic centre.	251
slaRvlg: radial velocity correction to local group.	252
slaRvlsrd: radial velocity correction to dynamical LSR.	253
slaRvlsrk: radial velocity correction to kinematical LSR.	253
slaS: the CIO locator s	254
slaS2tp: spherical to tangent plane.	255
slaSep: angle between two points on sphere.	256
slaSepv: angle between two vectors.	256
slaSmat: solve simultaneous equations.	257
slaSp: the CIO locator s'	258
slaSubet: remove E-terms.	258
slaSupgal: supergalactic to galactic.	259
slaSvd: singular value decomposition.	260
slaSvdcov: covariance matrix from SVD.	261
slaSvdsol: solution vector from SVD.	262
slaTp2s: tangent plane to spherical.	263
slaTp2v: tangent plane to $[x, y, z]$	264
slaTps2c: plate centre from $[\xi, \eta]$ and $[\alpha, \delta]$	265
slaTpv2c: plate centre from $[\xi, \eta]$ and $[x, y, z]$	266
slaUe2el: universal to conventional elements.	266
slaUe2pv: position & velocity from universal elements.	270
slaUnpcd: remove radial distortion.	271
slaV2tp: $[x, y, z]$ to tangent plane.	272
slaVdv: scalar product.	273
slaVn: normalize vector.	273
slaVxv: vector product.	274
slaXy2xy: apply linear model to an $[x, y]$	274
slaZd: $[h, \delta]$ to zenith distance.	275
5 SUMMARY OF CALLS	276
String Decoding	276
Sexagesimal Conversions	276
Angles, Vectors and Rotation Matrices	277
Calendars	278
Time-Scales	279
Earth Rotation and Sidereal Time	279
Precession and Nutation	280
Proper Motion	281
FK4/FK5/Hipparcos Transformations	281
Elliptic Aberration	282
Geographical and Geocentric Coordinates	282
ICRS-CIRS-Observed Transformations	282
Mean-Apparent-Observed Transformations	283
Azimuth and Elevation	284
Refraction and Air Mass	284
Ecliptic Coordinates	285
Galactic Coordinates	285
Supergalactic Coordinates	285

Ephemerides	286
Astrometry	287
Numerical Methods	288

Preface

SLALIB is a positional-astronomy subroutine library that is descended from a package of Fortran 66 routines that the author wrote for the Anglo-Australian Observatory 16-bit Interdata minicomputers in the mid-1970s. These routines were mainly for controlling the 3.9m AAT and for pulsar observing. By the early 1980s, the Digital Equipment Corporation VAX/VMS computer had become the standard in astronomy, and a much more comprehensive and thorough positional astronomy package could be contemplated, based on the AAO library but able to exploit for the first time essentially unlimited memory and fast double-precision floating point. Work in this first SLALIB was carried out at the Rutherford Appleton Laboratory, UK, at a time when the adoption of the IAU 1976 resolutions meant that astronomers were having to cope with a mixture of reference systems, time-scales and nomenclature, and a set of trustworthy readymade subroutines was likely to be popular.

The first C versions of SLALIB routines were hand-coded transcriptions of the Starlink Fortran versions, carried out by users of the library in order to satisfy local needs. This provided the stimulus for the present author to carry out a complete *ab initio* private rewrite to support certain classes of non-Starlink user, in particular groups needing telescope control capabilities; this rewrite included the development of a comprehensive validation suite that was subsequently translated back into Fortran for Starlink use.

Since the mid-1990s, the C SLALIB has become the principal variant in use world-wide, though the Fortran version was kept up to date and continued to be distributed by the Starlink project until its closure in June 2005. Permission for Tpoint Software to develop SLALIB/C commercially was granted by RAL in 1996.

Acknowledgements

Preparatory work on the original Fortran SLALIB was done by Althea Wilkinson of Manchester University, under the auspices of the UK Starlink Project. Subsequently, Andrew Murray, Catherine Hohenkerk, Andrew Sinclair, Bernard Yallop and Brian Emerson of Her Majesty's Nautical Almanac Office were consulted on many occasions and provided indispensable advice.

Early transcriptions into C were carried out by Steve Eaton (University of Leeds) in the course of MSc work in the late 1980s. These were subsequently enhanced by John Straede (AAO) and Martin Shepherd (Caltech). Useful comments on the author's own version came from Bob Payne (NRAO) and Jeremy Bailey (AAO).

A number of enhancements to SLALIB were at the suggestion of Russell Owen, University of Washington, the late Phil Hill, St Andrews University, Bill Vacca, JILA, Boulder and Ron Madgalena, NRAO and Mark Calabretta, CSIRO Radiophysics, Sydney. I am especially grateful to Derek Jones (RGO) for introducing me to the "universal variables" method of calculating orbits.

1 INTRODUCTION

1.1 Purpose

SLALIB/C¹ is a library of ANSI C functions that make accurate and reliable positional-astronomy applications easier to write. Most SLALIB functions are concerned with astronomical position and time, but a number have wider trigonometrical, numerical or general applications. The Tpoint Software applications TPOINT and TCSpk make extensive use of SLALIB/C functions, and the library plays a key role in pointing and tracking large telescopes and radio antennas around the world.

1.2 Example Application

Here is a simple example of an application program written using SLALIB/C calls:

```

#include <stdio.h>
#include <string.h>
#include <slalib.h>

/*
** Read a B1950 position from stdin and reply on stdout
** with the J2000 equivalent. Enter "q" to quit.
**
*/

int main ( )
{
    char b[80], s;
    int bad, i, j, i4[4];
    double r4, d4, r5, d5;

    /* Loop indefinitely. */
    for ( ; ; ) {

        /* Read a string, either h m s d ' ", or q to exit. */
        gets ( b );

        /* Exit if "q" entered. */
        if ( b[0] == (char) 'q' ) return 0;

        /* Preset the status to fail. */
        bad = 1;

```

¹The name SLALIB isn't an acronym; it just stands for "Subprogram Library A".

```

/* Decode the FK4 right ascension. */
i = 1;
slaDafin ( b, &i, &r4, &j );
if ( !j ) {
    r4 *= 15.0;

/* Decode the FK4 declination. */
slaDafin ( b, &i, &d4, &j );
if ( !j ) {

/* FK4 to FK5. */
slaFk45z ( r4, d4, 1950.0, &r5, &d5 );

/* Format and output the result. */
slaDr2tf ( 2, r5, &s, i4 );
printf ( "%2.2d %2.2d %2.2d.%2.2d",
        i4[0],i4[1],i4[2],i4[3] );
slaDr2af ( 1, d5, &s, i4 );
printf ( "   %c%2.2d %2.2d %2.2d.%1d\n",
        s,i4[0],i4[1],i4[2],i4[3] );

/* Success. */
bad = 0;
}
}

/* Warn if malformed input. */
if ( bad ) printf ( "?\n" );

/* Next RA,Dec. */
}
}

```

In this example, SLALIB not only provides the complicated FK4 to FK5 transformation but also simplifies the tedious and error-prone tasks of decoding and formatting angles expressed as hours, minutes *etc.* The example incorporates range checking, and avoids the notorious “minus zero” problem (an often-perpetrated bug where declinations between 0° and -1° lose their minus sign). With a little extra elaboration and a few more calls to SLALIB, defaulting can be provided (enabling unused fields to be replaced with commas to avoid retyping), proper motions can be handled, different epochs can be specified, and so on.

1.3 Scope

SLALIB contains 216 functions covering the following topics:

- string decoding, sexagesimal conversions
- angles, vectors & rotation matrices,
- calendars and time-scales,
- Earth rotation and sidereal time,
- precession & nutation,
- proper motion,
- FK4/FK5/Hipparcos, elliptic aberration,
- geocentric coordinates,
- ICRS-CIRS-observed transformations,
- mean-apparent-observed transformations
- azimuth & elevation²,
- refraction & air mass,
- ecliptic, galactic, supergalactic coordinates,
- ephemerides,
- astrometry,
- numerical methods.

1.4 Objectives

SLALIB was designed to give application programmers a basic set of positional-astronomy tools which are accurate and easy to use. To this end, the library is:

- Available as source code, and documented.
- Supported, maintained and actively developed.
- Portable – coded in ANSI standard C, and consequently usable on essentially all computers and operating systems. Arguments follow the usual conventions: by value (except for arrays and strings of course) for given arguments and by pointer for returned arguments. All the C functions are re-entrant: SLALIB/C can be run in a multi-threaded application and under such operating systems as VxWorks.
- Thoroughly commented, so that the source code can be used as a learning aid.

²The correct astronomical term for distance above the horizontal is *altitude*, but the present document follows the now well-established convention of using the word *elevation* instead. Although the latter is really an artillery term, it does have the advantage for computer applications of not sharing an initial with “azimuth”.

- Stable – changes are rare (and almost always improvements to internal documentation only) and calling conventions are never changed.
- Trustworthy – great care has gone into testing SLALIB, both by comparison with published data and by checks for internal consistency.
- As far as possible, rigorous – corners are not cut, even where the practical consequences would, as a rule, be negligible.
- Comprehensive, without including too many esoteric features required only by specialists.
- Practical – almost all the functions have been written to satisfy real needs encountered during the development of real-life applications.
- Environment-independent – the package is completely free of pauses, stops, I/O *etc.*
- Self-contained – SLALIB calls no other libraries.

A few *caveats*:

- SLALIB does not pretend to be canonical. It is in essence an anthology, and the adopted algorithms are liable to change as more up-to-date ones become available.³
- The functions are by no means orthogonal – there are several cases of different ones doing similar things, and many examples where sequences of SLALIB calls have simply been packaged, all to make applications less trouble to write.
- There are gaps – for example there are no functions for calculating physical ephemerides of Solar-System bodies, nor orbital ephemerides of Earth satellites.
- SLALIB is not homogeneous, though important subsets (for example the FK4/FK5 functions) are.
- The library is not foolproof. You have to know what you are trying to do (*e.g.* by reading textbooks on positional astronomy), and it is the caller’s responsibility to supply sensible arguments (although enough internal validation is done to avoid arithmetic errors).
- Without being written in a wasteful manner, SLALIB is nonetheless optimized for maintainability rather than speed. In addition, there are many places where considerable simplification would be possible if some specified amount of accuracy could be sacrificed; such compromises are left to the individual programmer and are not allowed to limit SLALIB’s value as a source of comparison results.

1.5 New Functions

In developing a package like SLALIB it is always difficult to know how far to go. Is it sufficient to provide the primitive operations, or should more complicated functions be packaged? Is it worth encroaching on specialist areas, where individual experts have all written their own

³For IAU-approved algorithms, see www.iau-sofa.rl.ac.uk.

software already? To what extent should CPU efficiency be an issue? How much support of different numerical precisions is required? And so on.

In practice, almost all the functions in SLALIB are there because they were needed for some specific application, and this is likely to remain the pattern for any enhancements in the future.

2 COMPILING AND LINKING

Code modules using SLALIB/C must `#include` the header file `<slalib.h>`.

To compile a program on Unix systems (Linux, Sun, HP-UX, MacOS-X, DEC Alpha *etc.*) with the GNU gcc compiler installed:

```
% gcc progname.c -I./include -L./lib -lsla -lm -o progname
```

(It may be necessary to point at different -I and -L directories.)

3 EXPLANATION AND EXAMPLES

To guide the writer of positional-astronomy applications software, this introductory chapter puts the SLALIB functions into the context of astronomical phenomena and techniques, and presents a few “cookbook” examples of the SLALIB calls in action. The astronomical content of the chapter is not, of course, intended to be a substitute for specialist text-books on positional astronomy, but may help bridge the gap between such books and the SLALIB functions. For further reading, the following cover a wide range of material and styles:

- *Explanatory Supplement to the Astronomical Almanac*, ed. P. Kenneth Seidelmann (1992), University Science Books.
- *Vectorial Astrometry*, C. A. Murray (1983), Adam Hilger.
- *Spherical Astronomy*, Robin M. Green (1985), Cambridge University Press.
- *Spacecraft Attitude Determination and Control*, ed. James R. Wertz (1986), Reidel.
- *Practical Astronomy with your Calculator*, Peter Duffett-Smith (1981), Cambridge University Press.

Also of considerable value, though out of date in places, are:

- *Explanatory Supplement to the Astronomical Ephemeris and the American Ephemeris and Nautical Almanac*, RGO/USNO (1974), HMSO.
- *Textbook on Spherical Astronomy*, W. M. Smart (1977), Cambridge University Press.

For full details of the latest standard algorithms, see:

- *IERS Conventions (2003)*, Dennis D. McCarthy & Gérard Petit (eds.), Verlag des Bundesamts für Kartographie und Geodäsie, Frankfurt am Main (2004).
- *IAU Standards Of Fundamental Astronomy*, <http://www.iau-sofa.rl.ac.uk/>.

Only brief details of individual SLALIB functions are given here, and readers will find it useful to refer to the subprogram specifications elsewhere in this document. The source code for the SLALIB functions is also intended to be used as documentation.

3.1 Spherical Trigonometry

Celestial phenomena occur at such vast distances from the observer that for most practical purposes there is no need to work in 3D; only the direction of a source matters, not how far away it is. Things can therefore be viewed as if they were happening on the inside of sphere with the observer at the centre – the *celestial sphere*. Problems involving positions and orientations

in the sky can then be solved by using the formulae of *spherical trigonometry*, which apply to *spherical triangles*, the sides of which are *great circles*.

Positions on the celestial sphere may be specified by using a spherical polar coordinate system, defined in terms of some fundamental plane and a direction in that plane chosen to represent zero longitude. Mathematicians usually work with the co-latitude, with zero at the principal pole, whereas most astronomical coordinate systems use latitude, reckoned plus and minus from the equator. Astronomical coordinate systems may be either right-handed (*e.g.* right ascension and declination $[\alpha, \delta]$, galactic longitude and latitude $[l^{\text{II}}, b^{\text{II}}]$) or left-handed (*e.g.* hour angle and declination $[h, \delta]$). In some cases different conventions have been used in the past, a fruitful source of mistakes. Azimuth and geographical longitude are examples; azimuth is now generally reckoned north through east (making a left-handed system); geographical longitude is now usually taken to increase eastwards (a right-handed system) but astronomers used to employ a west-positive convention. In reports and program comments it is wise to spell out what convention is being used, if there is any possibility of confusion.

When applying spherical trigonometry formulae, attention must be paid to rounding errors (for example it is a bad idea to find a small angle through its cosine) and to the possibility of problems close to poles. Also, if a formulation relies on inspection to establish the quadrant of the result, it is a sure sign that a vector-related method will be preferable.

As well as providing many functions which work in terms of specific spherical coordinates such as $[\alpha, \delta]$, SLALIB provides two functions which operate directly on generic spherical coordinates: `slaSep` computes the separation between two points (the distance along a great circle) and `slaBear` computes the bearing (or *position angle*) of one point seen from the other. The functions `slaDsep` and `slaDbear` are double precision equivalents.⁴ As a simple demonstration of SLALIB/C, we will use these facilities to estimate the distance from London to Sydney and the initial compass heading:

```
#include <stdio.h>
#include <slalib.h>
#define R2D 57.2957795      /* radians to degrees */
#define RKM 6375.0         /* Earth radius in km */
int main ( )
{
    /* Longitudes and latitudes (radians) for London and Sydney. */
    double al = -0.2/R2D, bl = 51.5/R2D,
           as = 151.2/R2D, bs = -33.9/R2D;

    /* Great-circle distance and initial heading. */
    printf ( "%5.0f km,%4.0f deg\n", slaDsep(al,bl,as,bs)*RKM,
            slaDbear(al,bl,as,bs)*R2D );

    return 0;
}
```

⁴The provision in some cases of `float` precision functions in addition to the more usual `double` is to some extent a product of SLALIB's long history and its Fortran background. However, the `float` versions sometimes have a role in applications with very tight memory and CPU constraints such as spaceborne instrumentation.

(The result is 17011 km, 61°.)

The functions `slaSepv`, `slaDsepv`, `slaPav`, `slaDpav` are equivalents of `slaSep`, `slaDsep`, `slaBear` and `slaDbear` but starting from vectors instead of spherical coordinates.

3.1.1 Formatting angles

SLALIB has functions for decoding decimal numbers from character form and for converting angles to and from sexagesimal form (hours, minutes, seconds or degrees, arcminutes, arcseconds). These apparently straightforward operations contain hidden traps which the SLALIB functions avoid.

There are five functions for decoding numbers from a character string, such as might be entered using a keyboard. They all work in the same style, and successive calls can work their way along a single string, decoding a sequence of numbers of assorted types. Number fields can be separated by spaces or commas, and can be defaulted to previous values or to preset defaults. Four of the functions decode single numbers: `slaInt2in` (integer), `slaIntin` (long integer), `slaFlotin` (single precision floating point) and `slaDfltin` (double precision). A minus sign can be detected even when the number is zero; this avoids the frequently-encountered “minus zero” bug, where declinations *etc.* in the range 0° to −1° mysteriously migrate to the range 0° to +1°.

Here is an example C program where we wish to read up to ten double precision numbers, defaulting to the previous values each time. The status is reported, showing among other things whether the field was defaulted or was negative even when zero. (Fields can be defaulted by omission or, individually, by using a comma.)

```
#include <stdio.h>
#include <slalib.h>

/* Read a string and decode it into doubles; "q" to quit. */

#define N 10      /* maximum number of fields */
int main ( )
{
    char b[100];
    int n, i, j;
    double d[N];

    /* Initial default values. */
    for ( n = 0; n < N; d[n++] = (double) n );

    /* Read strings until q entered. */
    while ( 1 ) {
        gets ( b );
        if ( b[0] == (char) 'q' ) return 0;
    }
}
```

```

/* Decode and report N numbers. */
i = 1;
for ( n = 0; n < N; n++ ) {
    slaDfltln ( b, &i, &d[n], &j );
    printf ( "value = %.20g, status = %d\n", d[n], j );
}
}
}

```

Two additional functions decode a 3-field sexagesimal number: `slaAfin` (degrees, arcminutes, arcseconds to single precision radians) and `slaDafin` (the same but double precision). They also work using other units such as hours *etc.* if you multiply the result by the appropriate factor. An example C program which uses `slaDafin` was given earlier, in Section 1.2.

SLALIB provides four functions for expressing an angle in radians in a preferred range. The function `slaRange` expresses an angle in the range $\pm\pi$; `slaRanorm` expresses an angle in the range $0-2\pi$. The functions `slaDrange` and `slaDranrm` are double precision versions.

Several functions (`slaCtf2d`, `slaCr2af` *etc.*) are provided to convert angles to and from sexagesimal form (hours, minute, seconds or degrees, arcminutes and arcseconds). They avoid the common “inconsistent rounding” bug (as well as the “minus zero” bug), which produces angles like $24^{\text{h}} 59^{\text{m}} 59^{\text{s}}.999$. Here is a program which displays an hour angle stored in radians, using two different resolutions:

```

#include <stdio.h>
#include <slalib.h>
int main ( )
{
    double ha;
    char sign;
    int ihmsf[4];

    ha = -0.261799315;
    slaDr2tf ( 3, ha, &sign, ihmsf );
    printf ( "%c%2.2d %2.2d %2.2d.%3.3d\n",
            sign, ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );
    slaDr2tf ( 2, ha, &sign, ihmsf );
    printf ( "%c%2.2d %2.2d %2.2d.%2.2d\n",
            sign, ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );
    return 0;
}

```

The output is:

```
-00 59 59.999
-01 00 00.00
```

3.2 Vectors and Matrices

As an alternative to employing a spherical polar coordinate system, the direction of an object can be defined in terms of the sum of any three vectors as long as they are different and not coplanar. In practice, three vectors at right angles are chosen, forming a system of *Cartesian coordinates*. The x - and y -axes lie in the fundamental plane (*e.g.* the equator in the case of $[\alpha, \delta]$), with the x -axis pointing to zero longitude. The z -axis is normal to the fundamental plane and points towards the positive pole. The y -axis can lie in either of the two possible directions, depending on whether the coordinate system is right-handed or left-handed. The three axes are sometimes called a *triad*. For most applications involving arbitrarily distant objects such as stars, the vector which defines the direction concerned is constrained to have unit length. The x -, y - and z -components can be regarded as the scalar (dot) product of this vector onto the three axes of the triad in turn. Because the vector is a unit vector, each of the three dot-products is simply the cosine of the angle between the unit vector and the axis concerned, and the x -, y - and z -components are sometimes called *direction cosines*.

For some applications, including those involving objects within the Solar System, unit vectors are inappropriate, and it is necessary to use vectors scaled in length-units such as AU, km *etc.* In these cases the origin of the coordinate system might not be the observer, but instead might be the Sun, the Solar-System barycentre, the centre of the Earth *etc.* But whatever the application, the final direction in which the observer sees the object can be expressed as direction cosines.

But where has this got us? Instead of two numbers—a longitude and a latitude—we now have three numbers to look after – the x -, y - and z -components, whose quadratic sum we have somehow to contrive to be unity. And, in addition to this apparent redundancy, most people find it harder to visualize problems in terms of $[x, y, z]$ than in $[\theta, \phi]$. Despite these objections, the vector approach turns out to have significant advantages over the spherical trigonometry approach:

- Vector formulae tend to be much more succinct; one vector operation is the equivalent of strings of sines and cosines.
- The formulae are as a rule rigorous, even at the poles.
- Precision is maintained all over the celestial sphere. When one Cartesian component is nearly unity and therefore insensitive to direction, the others become small and therefore more precise.
- Formulations usually deliver the quadrant of the result without the need for any inspection (except within the library function `atan2`).

A number of important transformations in positional astronomy turn out to be nothing more than changes of coordinate system, something which is especially convenient if the vector approach is used. A direction with respect to one triad can be expressed relative to another triad simply by multiplying the $[x, y, z]$ column vector by the appropriate 3×3 orthogonal matrix (a tensor of Rank 2, or *dyadic*). The three rows of this *rotation matrix* are the vectors in the old coordinate system of the three new axes, and the transformation amounts to obtaining the dot-product of the direction-vector with each of the three new axes. Precession, nutation, $[h, \delta]$ to $[Az, El]$, $[\alpha, \delta]$ to $[l^II, b^II]$ and so on are typical examples of the technique. A useful property of the rotation matrices is that they can be inverted simply by taking the transpose.

The elements of these vectors and matrices are assorted combinations of the sines and cosines of the various angles involved (hour angle, declination and so on, depending on which transformation is being applied). If you write out the matrix multiplications in full you get expressions which are essentially the same as the equivalent spherical trigonometry formulae. Indeed, many of the standard formulae of spherical trigonometry are most easily derived by expressing the problem initially in terms of vectors.

3.2.1 Using vectors

SLALIB provides transformations between spherical and vector form (`slaCs2c`, `slaCc2s` etc.), plus an assortment of standard vector and matrix operations (`slaVdv`, `slaMxv` etc.). There are also functions (`slaEuler` etc.) for creating a rotation matrix from three *Euler angles* (successive rotations about specified Cartesian axes). Instead of Euler angles, a rotation matrix can be expressed as an *axial vector*⁵ (the pole of the rotation, scaled by the amount of rotation), and functions are provided for this (`slaAv2m`, `slaM2av` etc.).

Here is a fragment of C code where spherical coordinates $[p, q]$ undergo a coordinate transformation; the transformation consists of a rotation of the coordinate system through angles a , b and c radians about the z , y and z axes respectively (*i.e.* $\mathbf{R} = R_3(c) \cdot R_2(b) \cdot R_3(a)$):

```

:
double p, q, a, b, c, r[3][3], v[3];
:
/* Create rotation matrix. */
slaEuler ( "zyz", a, b, c, r );

/* Transform position (p,q) from spherical to Cartesian. */
slaCs2c ( p, q, v );

/* Apply the rotation. */
slaMxv ( e, v, v );

```

⁵The name “axial vector” is specific to SLALIB; the IAU SOFA software calls them “rotation vectors” or “r-vectors”. Indeed there seems to be little mention of this concise 3-number parametrization in the literature. The reason is probably that the closely-related “Euler symmetric parameters” parametrization offers some of the advantages and is computationally exceptionally efficient: many applications involving complicated sequences of rotations can be done without computing trig functions. However, the axial vector approach has advantages of its own, including intuitive appeal and the ability to express multi-turn rotations.

```

/* Back to spherical. */
slaCc2s ( v, &p, &q );
:

```

Small adjustments to the direction of a position vector are often most conveniently described in terms of $[\Delta x, \Delta y, \Delta z]$. Adding the correction vector needs careful handling if the position vector is to remain of length unity, an advisable precaution which ensures that the $[x, y, z]$ components are always available to mean the cosines of the angles between the vector and the axis concerned. Two types of shifts are commonly used, the first where a small vector of arbitrary direction is added to the unit vector, and the second where there is a displacement in the latitude coordinate (declination, elevation *etc.*) alone.

For a shift produced by adding a small $[x, y, z]$ vector \mathbf{d} to a unit vector \mathbf{v}_1 , the resulting vector \mathbf{v}_2 has direction $< \mathbf{v}_1 + \mathbf{d} >$ but is no longer of unit length. A better approximation is available if the result is multiplied by a scaling factor of $(1 - \mathbf{d} \cdot \mathbf{v}_1)$, where the dot means scalar product. In C:

```

f = (1.0-(dx*v1x+dy*v1y+dz*v1z));
v2x = f*(v1x+dx);
v2y = f*(v1y+dy);
v2z = f*(v1z+dz);

```

The correction for diurnal aberration (discussed later) is an example of this form of shift.

As an example of the second kind of displacement we will apply a small change in elevation ΔE to an $[Az, El]$ direction vector. The direction of the result can be obtained by making the allowable approximation $\tan \Delta E \approx \Delta E$ and adding a adjustment vector of length ΔE normal to the direction vector in the vertical plane containing the direction vector. The z -component of the adjustment vector is $\Delta E \cos E$, and the horizontal component is $\Delta E \sin E$ which has then to be resolved into x and y in proportion to their current sizes. To approximate a unit vector more closely, a correction factor of $\cos \Delta E$ can then be applied, which is nearly $(1 - \Delta E^2/2)$ for small ΔE . Expressed in C code, for initial vector $v1x, v1y, v1z$, change in elevation del (positive \equiv upwards), and result vector $v2x, v2y, v2z$:

```

cosdel = 1.0-del*del/2.0;
r1 = sqrt(v1x*v1x+v1y*v1y);
f = cosdel*(r1-del*v1z)/r1;
v2x = f*v1x;
v2y = f*v1y;
v2z = cosdel*(v1z+del*r1);

```

Table 1: Spherical coordinate systems supported by SLALIB.

<i>system</i>	<i>symbols</i>	<i>longitude</i>	<i>latitude</i>	<i>x-y plane</i>	<i>longitude zero</i>	<i>RH/LH</i>
horizon	—	azimuth	elevation	horizontal	north	L
equatorial	α, δ	R.A.	Dec.	equator	equinox or CIO	R
ICRS	α, δ	R.A.	Dec.	ICRS equator	ICRS origin	R
local equ.	h, δ	H.A.	Dec.	equator	meridian	L
ecliptic	λ, β	ecl. long.	ecl. lat.	ecliptic	equinox	R
galactic	l'', b''	gal. long.	gal. lat.	gal. equator	gal. centre	R
supergalactic	SGL, SGB	SG long.	SG lat.	SG equator	node w. gal. equ.	R

An example of this type of shift is the correction for atmospheric refraction (discussed later). Depending upon the relationship between ΔE and E , special handling at the pole (the zenith for our example) may be required.

SLALIB includes functions for the case where both a position and a velocity are involved. The functions `slaCs2c6` and `slaCc62s` convert from $[\theta, \phi, \dot{\theta}, \dot{\phi}]$ to $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ and back; `slaDs2c6` and `slaDc62s` are double precision equivalents.

3.3 Celestial Coordinate Systems

SLALIB has functions to perform transformations of celestial positions between different spherical coordinate systems, including those shown in Table 1.

Transformations between $[h, \delta]$ and $[Az, El]$ can be performed by calling `slaE2h` and `slaH2e`, or, in double precision, `slaDe2h` and `slaDh2e`. There is also a function for obtaining zenith distance alone for a given $[h, \delta]$, `slaZd`, and one for determining the parallactic angle, `slaPa`. Three functions are included which relate to altazimuth telescope mountings. For a given $[h, \delta]$ and latitude, `slaAltaz` returns the azimuth, elevation and parallactic angle, plus velocities and accelerations for sidereal tracking. The functions `slaPda2h` and `slaPdQ2h` predict at what hour angle a given azimuth or parallactic angle will be reached.⁶

The functions `slaEqec1` and `slaEc1eq` transform between ecliptic coordinates $[\lambda, \beta]$ and equinox-based equatorial coordinates $[\alpha, \delta]$; there is also a function for generating the equatorial to ecliptic rotation matrix for a given date: `slaEcmat` (using the IAU 1980 ecliptic—the latest precession-nutation methods do not use the ecliptic for high-precision purposes, and in any case the ecliptic does not pass through the ICRS origin).

For transformation between galactic coordinates $[l'', b'']$ and equatorial coordinates $[\alpha, \delta]$ there are two sets of functions, depending on whether the $[\alpha, \delta]$ is old-style, B1950, or new-style, J2000;

⁶Both “elevation” and “parallactic angle” are somewhat contentious terms, though firmly entrenched. “Elevation” is borrowed from gunnery; the correct astronomical term is “altitude”. “Parallactic angle”, which now means the angle between the direction to the zenith and the direction to equatorial north and should really be “vertical angle”, seems originally to have meant instead the angle between the direction to ecliptic north and the direction to equatorial north.

`slaEg50` and `slaGe50` are $[\alpha, \delta]$ to $[l^H, b^H]$ and *vice versa* for the B1950 case, while `slaEqgal` and `slaGaleq` are the J2000 equivalents. (J2000 and ICRS are close enough for this and many other applications. There is no SLALIB support for high-precision use of galactic coordinates, which would be inappropriate.)

Finally, the functions `slaGalsup` and `slaSupgal` transform $[l^H, b^H]$ to de Vaucouleurs supergalactic longitude and latitude and *vice versa*.

It should be appreciated that the table, above, constitutes a gross oversimplification. Apparently simple concepts such as equator, equinox *etc.* are apt to be very hard to pin down precisely (polar motion, geometrical or kinematic...) and some have several interpretations, all subtly different. The various coordinate systems move in complicated ways with respect to one another or to the stars (themselves in motion). And in some instances the coordinate system is slightly distorted, so that the ordinary rules of spherical trigonometry no longer strictly apply.

These *caveats* apply particularly to the bewildering variety of different $[\alpha, \delta]$ systems that are in use. Figures 1 and 2 show how some of these systems are related, to one another and to the direction in which a celestial source actually appears in the sky. Figure 1 is for the old system of *mean places*, found in star catalogues and papers prior to the introduction of the International Celestial Reference System (ICRS); Figure 2 begins with ICRS coordinates and represents modern practices. At the bottom of each Figure is the *observed* $[Az, El]$, where a perfect theodolite would be pointed to see the source; and in the bodies of the two diagrams are the intermediate processing steps and coordinate systems.

To help understand these diagrams, and the SLALIB functions that can be used to carry out the various calculations, we will look at the coordinate systems involved, and the astronomical phenomena that affect them. But before doing so, it is necessary to distinguish between the two fundamentally different ways in which celestial coordinates can be managed.

3.4 Celestial coordinates, old and new

Most existing text-books use reference systems based on the celestial equator and the ecliptic, the projection of, respectively, the Earth's equator and orbit onto the celestial sphere. The points where these two great circles intersect are the equinoxes, one of which is used as a convenient zero point for stellar longitudes.

This simple geometrical picture is familiar, comfortable and works well for ordinary low-precision applications such as describing astronomical phenomena. However, when applied to more precise problems, complications arise. The Earth's pole (and hence equator) is in constant motion because of precession, nutation and polar motion; the ecliptic is a somewhat ill-defined concept and is itself constantly changing because of planetary perturbations. Using these moving planes in high-precision applications such as VLBI or pulsar timing became extremely complicated, and by the end of the 20th century the recommended procedures had changed:

1. The International Celestial Reference System was introduced, fixing the coordinate axes with reference to very distant objects. Although ICRS coordinates are close to FK5 J2000 mean places, this alignment was made merely for convenience. The ICRS is completely

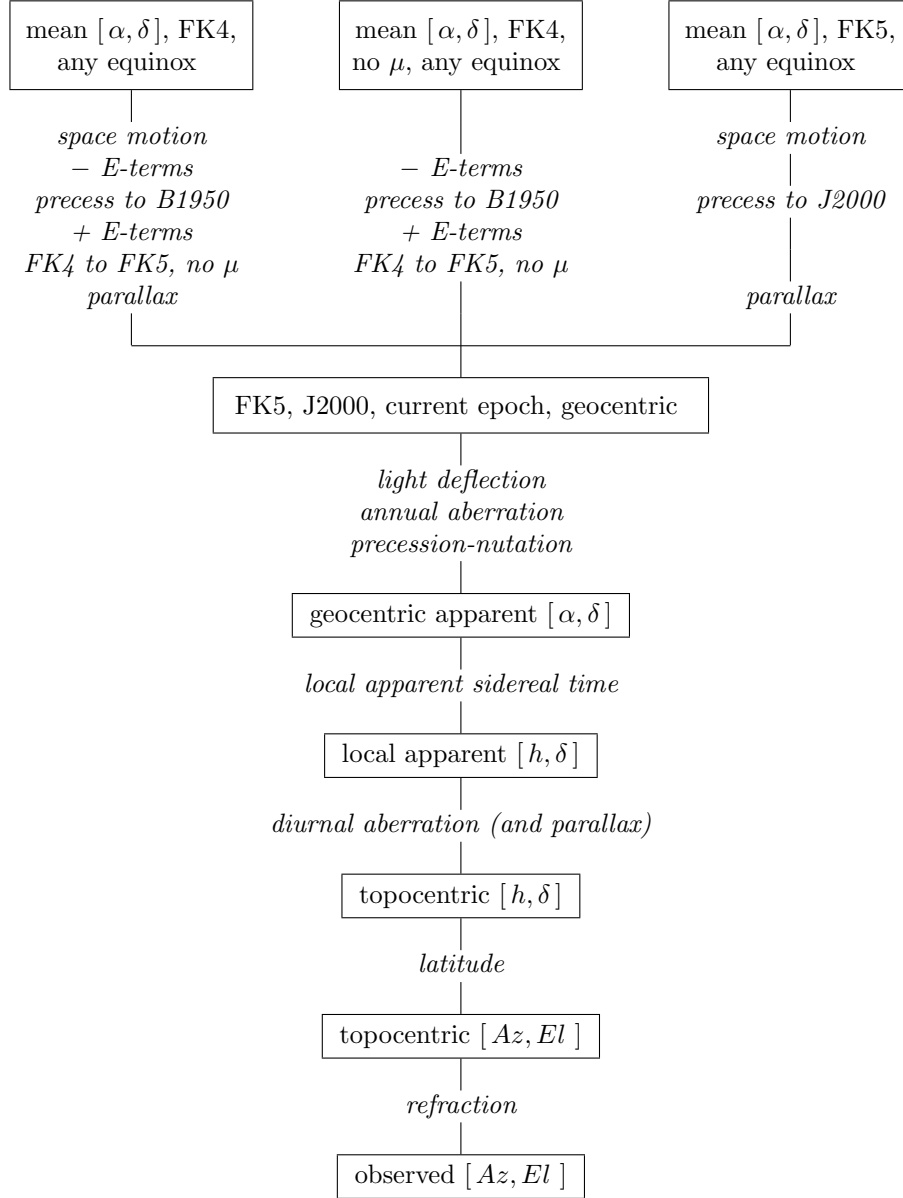


Figure 1: **Relationship between celestial coordinates, pre-ICRS**

Prior to the introduction of the International Celestial Reference System, star positions were published or catalogued using one of the mean $[\alpha, \delta]$ systems shown at the top, each with its own specific precession and aberration models. The “FK4” systems were used before about 1980 and were usually equinox B1950. The “FK5” system, equinox J2000 (to which the ICRS is nominally aligned), then took over. The figure relates a star’s mean $[\alpha, \delta]$ to the actual line-of-sight to the star. Note that for the conventional choices of equinox, namely B1950 or J2000, all of the precession and E-terms corrections are superfluous.

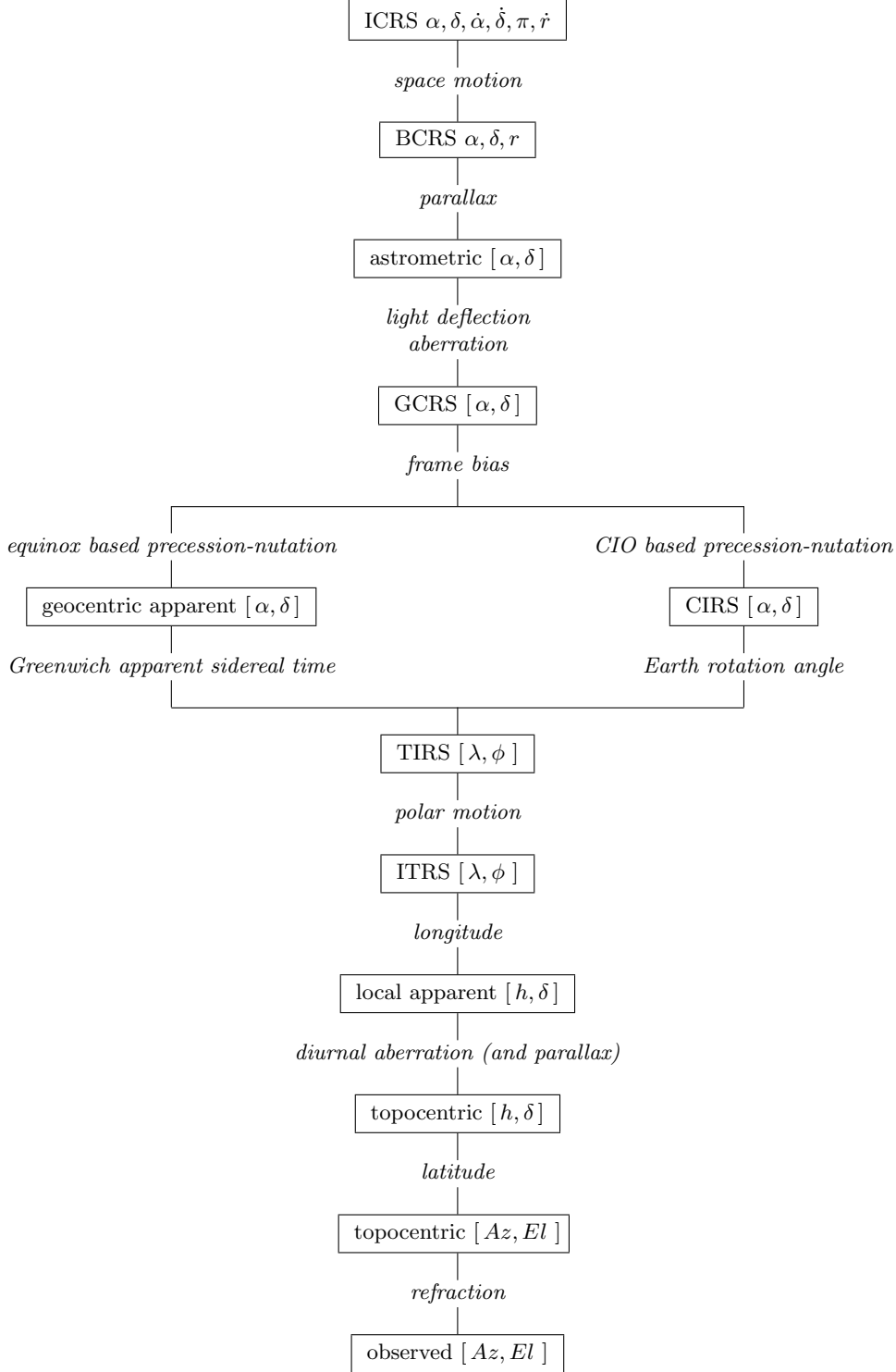


Figure 2: **Relationship between celestial coordinates, post-IAU-2000**

Since the introduction of the ICRS, the moving equator and ecliptic no longer provide the basis of high-precision astrometry, and mean places have fallen out of use. ICRS was nominally aligned to mean J2000, with a residual *frame bias* of about 23mas that can be ignored for many applications. In addition, a new zero-point for right ascension of date, the CIO, has replaced the equinox, breaking the final link with the ecliptic as far as the user is concerned. The figure relates a star's ICRS $[\alpha, \delta]$ to the actual line-of-sight to the star. The right-hand branch shows the modern CIO/ERA method of reckoning the Earth rotation; the classical equinox/GST method, shown in the left-hand branch, remains an option for supporting older applications.

independent of the rotation and motion of the Earth: a given star has ICRS coordinates that are essentially fixed, changing only as a consequence of the star's own motion through space. In addition to the ICRS itself, new barycentric (BCRS) and geocentric (GCRS) celestial reference systems were introduced, to describe the relationship between different choices of space-time coordinates. These different systems are intimately related to the changing apparent direction of the light ray from the star as seen by different observers. As can be seen from Figure 2, BCRS coordinates apply to the star position once space motion has been taken into account; the apparent direction of the star as seen by a terrestrial observer is affected by parallax, light deflection by the Sun (and planets) and aberration, the result being expressed in GCRS coordinates. In effect, ICRS, BCRS and GCRS share the same orientation, and it is only once precession-nutation is considered that axes with a different orientation arise.

2. The ecliptic was retired as a reference plane for high-precision work. One difficulty had been a certain degree of confusion about exactly what “ecliptic” meant. Classically, the basis was the geometric path of the Sun as seen from the Earth-Moon barycentre, but it had become more usual instead to start from the EMB orbital angular momentum vector. It turns out these give the same result only if the orbital angular momentum vector is referred to the moving ecliptic itself; the angular momentum vector in inertial space leads to a slightly different plane, moving the equinox by about $0''.1$. Other difficulties with the ecliptic included exactly how to average out the orbital perturbations, uncertainty about whether the plane should go through the Sun or the solar-system barycentre, and so on.
3. Greenwich sidereal time (GST), which is a complicated function of both Earth rotation (UT1) and time (not quite the same thing), was replaced by *Earth rotation angle* (ERA), which is a straightforward linear transformation of UT1 alone and which represents the Earth's rotation in inertial space.
4. The corollary of the change from GST to ERA is that the zero point of apparent right ascension has changed. The new α zero-point is called the *celestial intermediate origin* (CIO). The CIO is kinematically, not geometrically, defined, with the result that it is impossible to draw a diagram that explains where the CIO is on a given date. However, the concept (the CIO is what is known as a “non-rotating origin”) is very easy to grasp: the CIO is simply a point on the equator that moves as little as possible. It necessarily bobs north and south as the equator of date moves, but does not move along the equator; from moment to moment its motion is always at right-angles to the equator. The CIO was placed close to the GCRS prime meridian at epoch J2000, and throughout the 20th and 21st centuries the GCRS right ascension of the CIO remains smaller than $0''.1$. In fact for some low-precision applications it is acceptable to regard the CIO as the point on the moving equator where $\alpha_{\text{ICRS}} = 0$.

The change from equinox/GST to CIO/ERA, which brought about a clean separation between Earth rotation and precession-nutation, will take some years to come into wide use. Although modern textbooks describe the ICRS, most pre-date the introduction of the CIO, making it difficult to teach the new methods – even though they are in many ways simpler, with less to go wrong. There is in any case little incentive now that astronomers no longer need to prepare apparent places for their observing runs, a chore left to the telescope control computer. Furthermore, the latest and most precise precession-nutation and Earth rotation theories can be presented using either framework, with equal accuracy and with hardly any difference in the

total amount of computation required. Given these factors, and taking into account the large number of equinox/GST based applications that exist, the SLALIB policy is to support the old and new methods equally—with the recommendation that for all new applications the highest precision CIO-based methods should be adopted.

Quantitatively, the difference between old and new can be summarized as (i) *frame bias*, the relative orientation of ICRS and mean J2000 and (ii) the *equation of the origins*, the distance between the CIO and equinox. These topics are covered in the next two sections.

3.4.1 Frame bias

The International Celestial Reference System was nominally oriented to match the J2000 dynamical system. Subsequent observations located the J2000 mean pole at ICRS $[X, Y]$ coordinates $[-0''.016617, -0''.0068192]^7$, with the J2000 mean equinox lying $0''.0146$ west of the ICRS meridian.

Strictly speaking, this “frame bias” should play no part in the classical transformations pictured in Figure 1. However, because many existing applications use the corresponding SLALIB functions to operate upon ICRS coordinates (as opposed to mean J2000 coordinates), the pragmatic decision was made to include frame bias in SLALIB’s classical nutation model. This small departure from rigour is acceptable because the angles involved are small compared with the accuracies of the star catalogues and precession-nutation models that were in use prior to the introduction of the ICRS.

3.4.2 Equation of the origins

The expression for calculating hour angle is the same in either the old or new system:

$$h = \theta - \alpha,$$

where θ is now Earth rotation angle but was formerly Greenwich apparent sidereal time and α is referred to the celestial intermediate origin but was formerly referred to the true equinox of date. The difference ERA–GST, or equivalently the difference $\alpha_{\text{CIO}} - \alpha_{\text{equinox}}$, is called the “equation of the origins” (EO). The EO is dominated by the precession in right ascension, changing at about $46''$ per year, but includes also the nutation in right ascension (otherwise known as the equation of the equinoxes) and a number of small correction terms.

The CIO itself is located using a quantity called $s(t)$, which is the difference between two right ascensions of the same point. The point is the ascending node of the (moving) CIP equator on the (fixed) ICRS equator; the two right ascensions are with respect to the ICRS and the CIP/CIO respectively. The quantity s is defined by an integral that depends on the path of the precessing-nutating CIP, but is normally estimated using a series. The series used in the SLALIB function `slaS` is in fact for $s + XY/2$, where X and Y are the CIP coordinates of

⁷When components of a unit vector are quoted in angular measure this should be taken to mean the radian equivalent.

date. The value of s is well under $0''.1$ throughout the 20th and 21st centuries, and for some applications zero can be used.

(There is an analogous quantity s' that applies to terrestrial coordinates and depends on the history of polar motion. It is extremely small, under 0.1 mas even after two centuries. The function `slaSp` implements a standard model.)

3.5 Classical Precession and Nutation

Right ascension and declination, $[\alpha, \delta]$, are the names of the longitude and latitude in a spherical polar coordinate system based on the Earth's axis of rotation. The (classical) zero point of α is the intersection of the *celestial equator* and the *ecliptic* (the apparent path of the Sun through the year) where the Sun moves into the northern hemisphere. This point is called the *first point of Aries*, the *vernal equinox* (with apologies to southern-hemisphere readers) or simply the *equinox*.

This simple picture is unfortunately complicated by the fact that neither the equator nor the ecliptic is fixed with respect to the distant background. Both are affected by the gravitational interactions between the Earth and the other solar-system bodies. These cause both long-term and short-term changes in a star's $[\alpha, \delta]$ coordinates. The long-term and short-term changes are classically referred to as “precession” and “nutation” respectively.

Precession is further divided into *luni-solar precession*, which comes from the motion of the equator, and *planetary precession*, which comes from the motion of the ecliptic, the combination being called *general precession*. These terms are falling out of use, in favour simply of “precession of the equator” and “precession of the ecliptic”.

The character of the precession is that the Earth's rotation axis sweeps out a cone centred on the ecliptic pole, completing one revolution in about 26,000 years. The motion is caused by the torque exerted on the distorted and spinning Earth by the Sun and (especially) the Moon.

Consider the effect of the Sun alone, at or near the northern summer solstice. The Sun ‘sees’ the top/north pole of the Earth tilted towards it (by about $23^\circ 5'$, the *obliquity of the ecliptic*), and sees the nearer part of the Earth's equatorial bulge below centre and the further part above centre. Although the Earth is in free fall, the gravitational force on the nearer part of the equatorial bulge is greater than that on the further part, and so there is a net torque acting as if to eliminate the tilt. Six months later the same thing is happening in reverse, except that the torque is still trying to eliminate the tilt. In between (at the equinoxes) the torque shrinks to zero. A torque acting on a spinning body is gyroscopically translated into a precessional motion of the spin axis at right-angles to the torque, and this happens to the Earth. The motion varies during the year, going through two maxima, but always acts in the same direction. The Moon produces the same effect, adding a contribution to the precession which peaks twice per month. The Moon's proximity to the Earth more than compensates for its smaller mass and gravitational attraction, so that it in fact contributes most of the effect.

The complex interactions produce a precessional motion that is wobbly rather than completely smooth. However, the main 26,000-year component is on such a grand scale that it dwarfs the remaining terms, the biggest of which has an amplitude of only $11''$ and a period of about

18.6 years. This difference of scale makes it convenient to treat these two components of the motion separately.

Note that precession and nutation are simply different frequency components of the same physical effect, which is now called “precession-nutation”. There is a common misconception that precession is caused by the Sun and nutation is caused by the Moon. In fact the Moon is responsible for two-thirds of the precession, and, while it is true that much of the complex detail of the nutation is a reflection of the intricacies of the lunar orbit, there are nonetheless important solar terms in the nutation.

In addition to, and quite separate from, the precession-nutation effect, the orbit of the Earth-Moon system around the Sun is not fixed in orientation, a result of the attractions of the planets. This slow (about $0''.5$ per year) secular rotation of the ecliptic about a slowly-moving diameter is the “planetary precession” mentioned earlier and, along with the luni-solar precession is included in the general precession. The equator and ecliptic as affected by general precession are what define the various “mean” $[\alpha, \delta]$ reference systems of the pre-ICRS era.

The models for precession and nutation come from a combination of observation and theory, and are subject to continuous refinement. Nutation models in particular have reached a high degree of sophistication, taking into account such things as the non-rigidity of the Earth and the effects of the planets; the SF2001 nutation model used in SLALIB’s older equinox-based functions comprises 194 terms in each of ψ (longitude) and ϵ (obliquity), some as small as a few microarcseconds. The IAU 2000A nutation model, implemented in the function `slaNu00a` and used in the newer CIO-based SLALIB functions, is about seven times larger.

3.5.1 Precession-nutation and the CIO

Although in the classical picture precession-nutation is thought of as affecting both the equator and the equinox, the latter aspect is nothing more than a geometrical consequence of the chosen coordinate system. The essence of precession-nutation is that it is all about the motion of the pole (and hence the equator), and in the new CIO based approach it is both normal and sufficient to specify the instantaneous pole, which is called the *celestial intermediate pole* (CIP), simply as $[X, Y]$, the first two components of the CIP’s ICRS unit vector. Indeed, one attraction of the new method is that series for $X(t)$ and $Y(t)$ exist (though not used by SLALIB) that deliver the CIP as a function of time in a particularly straightforward way.

Even so, the ecliptic is never far away. The tidal forces that cause precession-nutation are from bodies that remain near the ecliptic plane, and descriptions of these effects are at their most compact when expressed in ecliptic-based coordinates. Hence even in the latest precession theories there is still an ecliptic, and the existing nutation series are with respect to the mean equator and ecliptic of date.

Moreover, the classical precession angles, when used to generate the 3×3 precession matrix, automatically deliver the equinox, whether it is wanted or not. It appears as the top row of the epoch-to-date precession-nutation matrix, which is the unit vector towards the equinox (just as the bottom row is the unit vector towards the pole; the middle row is simply the vector product pole \times equinox).

On the face of it, this makes direct use of the classical matrix attractive. Starting with existing precession and nutation models, it delivers traditional equinox-based right ascensions without further ado, ready to be used in conjunction with sidereal time. However, there is a hidden danger. Any change in the precession or nutation models requires a corresponding change to the formulation of sidereal time, and it is the user's responsibility to employ a matching set of precession, nutation and sidereal time models if hour angles are to be calculated reliably. For example, mixing old and new precession-nutation and sidereal time can introduce errors in h at the 0".05 level.

In the newer SLALIB functions, multiple sidereal time formulations are avoided by working internally with respect to the CIO, which is nearly independent of the adopted precession-nutation model. The distance between the CIO and the chosen equinox is the *equation of the origins* (see Section 3.4.2). The EO can be calculated from any supplied precession-nutation matrix, and if subtracted from the Earth rotation angle yields Greenwich apparent sidereal time appropriate for use with that particular matrix.

3.6 SLALIB Support for Classical Precession

This Section applies to the equinox based transformations shown in Figure 1. For the new CIO based methods shown in Figure 2, refer to Section 3.8.

For equinox-based use, SLALIB offers a choice of three precession models:

- The old Bessel-Newcomb, pre IAU 1976, “FK4” model, used for B1950 star positions and other pre-1984.0 purposes (`slaPrebn`).
- The newer Fricke-Lieske, IAU 1976, “FK5” model, used for J2000 star positions and other post-1984.0 purposes (`slaPrec`).
- A model published by Simon *et al.* which is more accurate than the IAU 1976 model, especially over long periods of time (`slaPrec1`).

(Their accuracies are compared, with each other and with much more elaborate new models, in Table 2.) In each case, the named SLALIB function generates the (3×3) *precession matrix* for a given start and finish epoch. For example, here is C code for generating the rotation matrix which describes the precession between the epochs J2000 and J1985.372 (IAU 1976 model):

```
double pmat[3][3];
:
slaPrec ( 2000.0, 1985.372, pmat );
```

It is instructive to examine the resulting matrix:

$$\begin{array}{rrr} +0.9999936402 & +0.0032709208 & +0.0014214694 \\ -0.0032709208 & +0.9999946505 & -0.0000023247 \\ -0.0014214694 & -0.0000023248 & +0.9999989897 \end{array}$$

Note that the diagonal elements are close to unity, and the other elements are small. This shows that over an interval as short as 15 years the precession will not move a position vector very far—in this case about $0^{\circ}2$.

For convenience, a direct $[\alpha, \delta]$ to $[\alpha, \delta]$ precession function is also provided (`slaPreces`), suitable for either the FK4 or the FK5 system (but not a mixture of the two).

3.6.1 Mean places, and epoch

From a classical standpoint, the main effect of the classical precession-nutation is an increase of about $50''/\text{year}$ in the ecliptic longitudes of the stars. It is therefore essential, when reporting the position of an astronomical target with respect to an equinox-based reference system, to qualify the coordinates with a date, or *epoch*. Specifying the epoch ties down the equator and equinox which define the $[\alpha, \delta]$ coordinate system that is being used.⁸ For simplicity, only the smooth and steady “precession” portion of the complete precession-nutation effect is included, thereby defining what is called the *mean* equator and equinox for the epoch concerned. We would say a star has a mean place of (for example) $12^{\text{h}} 07^{\text{m}} 58^{\text{s}}.09 - 19^{\circ} 44' 37''.1$ “with respect to the mean equator and equinox of epoch J2000”. The short way of saying this is “ $[\alpha, \delta]$ equinox J2000” (**not** “ $[\alpha, \delta]$ epoch J2000”, which means something different, to do with proper motion).

In fact the word “epoch” just means a (significant) moment in time, and can be supplied in a variety of forms, using different calendar systems and time-scales. For the purpose of specifying the epochs associated with the mean place of a star, two conventions exist. Both sorts of epoch superficially resemble years AD but are not tied to the civil (Gregorian) calendar; to distinguish them from ordinary calendar-years there is often a “.0” suffix (as in “1950.0”), although any other fractional part is perfectly legal (*e.g.* 1987.5).

The older system, *Besselian epoch*, was defined in such a way that its units are tropical years of about 365.2422 days and its time-scale is the obsolete *Ephemeris Time*. The start of the Besselian year is the moment when the ecliptic longitude of the mean Sun is 280° ; this happens near the start of the calendar year (which is why 280° was chosen).

The new system, *Julian epoch*, was adopted as part of the IAU 1976 revisions (about which more will be said in due course) and came formally into use at the beginning of 1984. It uses the Julian year of exactly 365.25 days; Julian epoch 2000 is defined to be 2000 January 1.5 in the TT time-scale, at the geocentre.

⁸An equinox is, however, not required for coordinates in the International Celestial Reference System. Such coordinates must be labelled simply “ICRS”, or the specific catalogue can be mentioned, such as “Hipparcos”; constructions such as “Hipparcos, J2000” are redundant and misleading.

For specifying mean places of stars, various standard epochs are in use, the most common ones being Besselian epoch 1950.0 and Julian epoch 2000.0. To distinguish the two systems, Besselian epochs are now prefixed “B” and Julian epochs are prefixed “J”. Epochs without an initial letter can be assumed to be Besselian if before 1984.0, otherwise Julian; furthermore, the phrase “at epoch” has specifically come to mean “at J2000.0”. These details are supported by the SLALIB functions `slaDbjin` (decodes numbers from a character string, accepting an optional leading B or J), `slaKbj` (decides whether B or J depending on prefix or range) and `slaEpc` (converts one epoch to match another).

SLALIB has four functions for converting Besselian and Julian epochs into other forms. The functions `slaEpb2d` and `slaEpj2d` convert Besselian and Julian epochs into MJD; the functions `slaEpb` and `slaEpj` do the reverse. For example, to express B1950 as a Julian epoch:

```
printf ( "J%10.5f\n", slaEpj(slaEpb2d(1950.0)) );
```

(The answer is J1949.99979.)

3.7 SLALIB Support for Classical Nutation

SLALIB provides two nutation models for equinox based use: the IAU 1980 model, implemented in the function `slaNutc80`, and a much more accurate newer theory, SF2001, implemented in the function `slaNutc`. Both return the components of nutation in longitude and latitude (and also provide the obliquity) from which a nutation matrix can be generated by calling `slaDeuler` (and from which the *equation of the equinoxes*, described later, can be found). Alternatively, the SF2001 nutation matrix can be generated in a single call by using `slaNut`. The SF2001 nutation theory includes components that correct for errors in the IAU 1976 precession and also for the frame bias (see Section 3.4.1), achieving a final accuracy better than 1 mas in the present era when starting from ICRS coordinates.

A rotation matrix for applying the entire equinox based precession-nutation transformation in one go can be generated by calling `slaPrenut`.

3.8 High Precision Precession-Nutation

The precession and nutation SLALIB functions used with the new methods (see Figure 2) are rather differently structured to those already described. This is a reflection of (i) the frame bias, which is new, (ii) the benefits of treating bias and precession and nutation as a single transformation and (iii) the need to provide highly consistent transformations even when multiple routes are taken.

As can be seen from Figure 2, precession-nutation is just part of the following complete transformation from celestial coordinates \mathbf{v}_{GCRS} to terrestrial coordinates \mathbf{v}_{ITRS} :

$$\mathbf{v}_{ITRS} = \mathbf{R}_{PM} \times R_3(\theta) \times \mathbf{R}_{NPB} \times \mathbf{v}_{GCRS} \quad (1)$$

where:

- the matrix \mathbf{R}_{NPB} represents the combined effects of frame bias and precession-nutation;
- $R_3(\theta)$ is Earth rotation, with θ either ERA or GST depending on whether \mathbf{R}_{NPB} is CIO or equinox based; and
- the matrix \mathbf{R}_{PM} takes account of polar motion.

The sequence of coordinate systems in Expression (1) is:

1. The Geocentric Celestial Reference System, which is effectively the same orientation as ICRS. The precession-nutation transformation then produces...
2. either Celestial Intermediate Reference System coordinates, if the matrix \mathbf{R}_{NPB} was CIO based, or classical apparent place, if the matrix \mathbf{R}_{NPB} was equinox based. The rotation through either ERA or GST, as appropriate, then gives...
3. Terrestrial Intermediate Reference System coordinates, which after polar motion has been applied and the small adjustment onto the terrestrial intermediate origin (TIO) has been made gives...
4. International Terrestrial Reference System coordinates: where on the Earth the star appears to be in the zenith, and easily transformed into $[h, \delta]$ for other places.

The fundamental models from which all the SLALIB high-precision precession-nutation and matching Earth rotation functions are derived are as follows:

- The IAU 2006 precession model, in the Fukushima-Williams four angle form.
- The IAU 2000A nutation model.
- The IAU 2006 adjustments to the IAU 2000A nutation model.
- The IAU 2006 model for the quantity $s + XY/2$, which locates the celestial intermediate origin on the CIP equator.
- The IAU 2000 model for the quantity s' , which locates the terrestrial intermediate origin on the CIP equator.
- The IAU 2000 expression for ERA(UT1).

The derived products are:

- Precession-nutation matrices, for transforming between GCRS and apparent place (either CIO based or equinox based).
- CIP coordinates X, Y .
- The equation of the origins, $EO = ERA - GST$.
- Greenwich (apparent) sidereal time, GST.

(The polar motion matrix \mathbf{R}_{PM} in Expression (1) depends on IERS tabulations of the pole coordinates. The SLALIB function `slaPomom` uses these pole coordinates, together with an approximation for the TIO locator s' (provided by `slaSp`), to construct the \mathbf{R}_{PM} matrix.)

For any given date, the complete matrix \mathbf{R}_{NPB} in Expression (1) can be obtained by calling the function `slaPncio`, for the CIO based version, or `slaPneqx`, for the equinox based version.

At a lower level, the function `slaPfw` returns the precession angles themselves (with frame bias included), while `slaNu` obtains the nutation components. Once these nutation components have been added to the precession angles from `slaPfw`, the equinox based \mathbf{R}_{NPB} matrix can be generated by calling `slaFw2m`. Alternatively, the CIP X, Y coordinates can be obtained by calling `slaFw2xy`, the CIO locator s computed using the function `slaS`, and the CIO based \mathbf{R}_{NPB} matrix formed using `slaG2ixys`. This latter route provides an opportunity to introduce corrections to X, Y , as published by the IERS. The resulting matrix transforms from GCRS to CIRS.

An important canonical component of these calculations is the IAU 2000A nutation model, implemented in the function `slaNu00a`.

Earth rotation angle, from which $R_3(\theta)$ in the CIO based form of Expression (1) can be applied, is available using `slaEra`, which is a function of UT1 alone. Or, for the case where the equinox based form of Expression (1) is being used, the requisite Greenwich apparent sidereal time can be obtained by calling `slaGst`, a function of time (*i.e.* TT) as well as UT1.

A key component of the GST prediction is the equation of the origins, which can be obtained as a function of time by calling `slaEo`. Alternatively, to obtain the EO corresponding to an independently calculated \mathbf{R}_{NPB} matrix and CIO locator, `slaEors` can be called. As well as its role in computing GST, the EO is useful when CIRS right ascension is to be computed from geocentric apparent right ascension or *vice versa*. The relevant expressions are $EO = ERA - GST$ and $EO = \alpha_{CIRS} - \alpha_{\text{apparent}}$.

Table 2 compares the accuracy of different sequences of SLALIB functions for computing $[h, \delta]$. The CIO based and equinox based high-precision methods agree at sub-mas level, which should be compared with the ~ 0.5 mas intrinsic accuracy of the models used. The older functions, based on the SF2001 nutation and the associated corrections to the IAU 1976 precession, deliver adequate accuracy for most applications and are somewhat faster to compute; these are used in the `slaMap` *etc.* functions.

3.9 Proper Motion

Stars in catalogues usually have, in addition to the $[\alpha, \delta]$ coordinates, a *proper motion* $[\mu_\alpha, \mu_\delta]$. This is an intrinsic motion of the star across the background. Very few stars have a proper motion which exceeds $1''/\text{year}$, and most are far below this level. A star observed as part of normal astronomy research will, as a rule, have a proper motion which is unknown.

Mean or ICRS $[\alpha, \delta]$ and the proper motion $[\dot{\alpha}, \dot{\delta}]$ are not sufficient: the epoch at which the $[\alpha, \delta]$ was or will be correct is also needed. Note the distinction between the epoch which specifies the coordinate system and the epoch at which the star passed through the given $[\alpha, \delta]$.

Table 2: Comparison between different SLALIB precession-nutation and Earth rotation functions: errors at three dates compared with reference method (IAU 2006, CIO-based). *n.b.* The units of the second line of the table are picoarcseconds; the numbers are merely rounding errors.

<i>functions</i>	<i>1950</i>	<i>2000</i>	<i>2010</i>	<i>2050</i>	<i>2100</i>	<i>2200</i>	<i>unit</i>
<code>slaPncio</code> , <code>slaEra</code> (reference)	-	-	-	-	-	-	
<code>slaPneqx</code> , <code>slaGst</code>	8	69	23	10	62	34	<i>pas</i>
<code>slaPrenut</code> , <code>slaGmst</code> , <code>slaEeqqx</code>	1.14	0.24	0.34	1.41	3.66	11.88	<i>mas</i>
<code>slaPrecl</code> , <code>slaNutc80</code> , <code>slaGmst</code> , <code>slaEeqqx</code>	32	50	37	51	67	108	<i>mas</i>
<code>slaPrec</code> , <code>slaNutc80</code> , <code>slaGmst</code> , <code>slaEeqqx</code>	111	50	64	187	338	643	<i>mas</i>

The full specification for a star is $[\alpha, \delta]$, proper motions, equinox (unless ICRS) and epoch, plus something to identify which set of models for the precession *etc.* is being used (and, in the case of nearby stars, parallax and radial velocity). For convenience, coordinates given in star catalogues are almost always adjusted to make the equinox and epoch the same—for example J2000 in the case of FK5. However, there are some notable exceptions to this rule, in particular the Hipparcos and Tycho 1 catalogues, which provide quasi-J2000-equinox coordinates with an epoch of 1991.25—this is a fruitful source of mistakes.

SLALIB provides one function to handle proper motion on its own, `slaPm`. Proper motion is also allowed for in various other functions as appropriate, for example `slaMap` and `slaFk425`. Note that in all SLALIB functions which involve proper motion the units are radians per year and the α component is in the form $\dot{\alpha}$ (*i.e.* big numbers near the poles). Some star catalogues have proper motion per century, and in most recent catalogues the α component is in the form $\dot{\alpha} \cos \delta$ (*i.e.* angle on the sky). If an RA proper motion is quoted using either arcseconds or milliarcseconds as the angular measure, it can assumed to be the latter variety.

3.10 Parallax and Radial Velocity

For the utmost accuracy and the nearest stars, allowance can be made for *annual parallax* and for the effects of perspective on the proper motion.

Parallax is appreciable only for nearby stars; even the nearest, Proxima Centauri, is displaced from its average position by less than an arcsecond as the Earth revolves in its orbit.

For stars with a known parallax, knowledge of the radial velocity allows the proper motion to be expressed as an actual space motion in three dimensions. The proper motion is, in fact, a snapshot of the transverse component of the space motion, and in the case of nearby stars will change with time due to perspective.⁹

SLALIB does not provide facilities for handling parallax and radial-velocity on their own, but their contribution is allowed for in such functions as `slaPm`, `slaMap` and `slaFk425`. Catalogue

⁹Even this space-motion treatment lacks rigor, because no account is taken of light-time effects. See Stumpff, P., 1985, *Astron. Astrophys.* **144**, 232-240 for a special-relativity-compliant Lorentz transformation approach.

mean places do not include the effects of parallax and are therefore *barycentric*; when pointing telescopes *etc.* it is usually most efficient to apply the slowly-changing parallax correction to the mean place of the target early on and to work with the *geocentric* mean place. This latter approach is implied in Figure 1.

3.11 Aberration

The finite speed of light combined with the motion of the observer around the Sun during the year causes apparent displacements of the positions of the stars. The effect is called the *annual aberration* (or “stellar” aberration). Its maximum size, about $20''.5$, occurs for stars 90° from the point towards which the Earth is headed as it orbits the Sun; a star exactly in line with the Earth’s motion is not displaced. To receive the light of a star, the telescope has to be offset slightly in the direction of the Earth’s motion. A familiar analogy is the need to tilt your umbrella forward when on the move, to avoid getting wet. This classical model is, in fact, misleading in the context of light as opposed to rain, but happens to give the same answer as a relativistic treatment to first order (better than 1 milliarcsecond).

Before the IAU 1976 resolutions, different values for the approximately $20''.5$ *aberration constant* were employed at different times, and this can complicate comparisons between different catalogues. Another complication comes from the so-called *E-terms of aberration*, that small part of the annual aberration correction that is a function of the eccentricity of the Earth’s orbit. The E-terms, maximum amplitude about $0''.3$, happen to be approximately constant for a given star, and so they used to be incorporated in the catalogue $[\alpha, \delta]$ to reduce the labour of converting to and from apparent place. The E-terms can be removed from a catalogue $[\alpha, \delta]$ by calling `slaSubet` or applied (for example to allow a pulsar timing-position to be plotted on a B1950 finding chart) by calling `slaAddet`; the E-terms vector itself can be obtained by calling `slaEtrms`. Star positions post IAU 1976 are free of these distortions, and to apply corrections for annual aberration involves the actual barycentric velocity of the Earth rather than the use of canonical circular-orbit models.

The annual aberration is the aberration correction for an imaginary observer at the Earth’s centre. The motion of a real observer around the Earth’s rotation axis in the course of the day makes a small extra contribution to the total aberration effect called the *diurnal aberration*. Its maximum amplitude is about $0''.3$.

No SLALIB function is provided for calculating the aberration on its own, though the required velocity vectors can be generated using `slaEvp` (or `slaEpv`) and `slaGeoc`. Annual and diurnal aberration are allowed for where required, for example in `slaMap` *etc.* and `slaAop` *etc.* Note that this sort of aberration is different from the *planetary aberration*, which is the apparent displacement of a solar-system body, with respect to the ephemeris position, as a consequence of the motion of *both* the Earth and the source. The planetary aberration can be computed either by correcting the position of the solar-system body for light-time, followed by the ordinary stellar aberration correction, or more directly by expressing the position and velocity of the source in the observer’s frame and correcting for light-time alone.

3.12 Different Sorts of Mean Place

A confusing aspect of the mean places used in the pre-ICRS era is that they are sensitive to the precise way they were determined. A mean place is not directly observable, even with fundamental instruments such as transit circles, and to produce one will involve working with respect to some existing star catalogue, for example the fundamental catalogues FK4 and FK5, and applying specific mathematical models of precession, nutation, aberration and so on. Note in particular that no star catalogue, even a fundamental catalogue such as FK4 or FK5, nor even the International Celestial Reference System itself, defines a coordinate system, strictly speaking; it is merely a list of star positions and proper motions. However, once the stars from a given catalogue are used as position calibrators, *e.g.* for transit-circle observations or for plate reductions, then a broader sense of there being a coordinate space naturally arises, and such phrases as “in the system of the FK4” can legitimately be employed. However, there is no formal link between the two concepts: no “standard least squares fit” between reality and the inevitably flawed catalogues. All such catalogues suffer at some level from systematic, zonal distortions of both the star positions and of the proper motions, and include measurement errors peculiar to individual stars.

Many of these complications are of little significance except to specialists. However, astronomers who need to refer to 20th-century observations cannot escape exposure to at least the two main varieties of mean place, loosely called FK4 and FK5, and should be aware of certain pitfalls. For most practical purposes the more recent system, FK5, is free of surprises and tolerates naive use well—it can even for many purposes be regarded as equivalent to ICRS. FK4, in contrast, contains two important traps:

- The FK4 system rotates at about 0".5 per century relative to distant galaxies. This is manifested as a systematic distortion in the proper motions of all FK4-derived catalogues, which will in turn pollute any astrometry done using those catalogues. For example, FK4-based astrometry of a QSO using plates taken decades apart will reveal a non-zero *fictitious proper motion*, and any FK4 star which happens to have zero proper motion is, in fact, slowly moving against the distant background. The FK4 system rotates because it was established before the nature of the Milky Way, and hence the existence of systematic motions of nearby stars, had been recognized.
- Star positions in the FK4 system are part-corrected for annual aberration (see above) and embody the so-called E-terms of aberration.

The change from the old FK4-based system to FK5 occurred at the beginning of 1984 as part of a package of resolutions made by the IAU in 1976, along with the adoption of J2000 as the reference epoch. Star positions in the newer, FK5, system are free from the E-terms, and the system is a much better approximation to an inertial frame—about five times better (and ICRS is hundreds of times better still).

It may occasionally be convenient to specify the FK4 fictitious proper motion directly. In FK4, the centennial proper motion of (for example) a QSO is:

$$\begin{aligned}\mu_\alpha &= -0^s.015869 + ((0^s.029032 \sin \alpha + 0^s.000340 \cos \alpha) \sin \delta - 0^s.000105 \cos \alpha - 0^s.000083 \sin \alpha) \sec \delta \\ \mu_\delta &= +0''.43549 \cos \alpha - 0''.00510 \sin \alpha + (0''.00158 \sin \alpha - 0''.00125 \cos \alpha) \sin \delta - 0''.00066 \cos \delta\end{aligned}$$

3.13 Mean Place Transformations

Figure 1 is based upon three varieties of mean $[\alpha, \delta]$ all of which are of practical significance to observing astronomers in the present era:

- Old style (FK4) with known proper motion in the FK4 system, and with parallax and radial velocity either known or assumed zero.
- Old style (FK4) with zero proper motion in FK5, and with parallax and radial velocity assumed zero.
- New style (FK5 or, loosely, ICRS) with proper motion, parallax and radial velocity either known or assumed zero.

The figure outlines the steps required to convert positions in any of these systems to a J2000 $[\alpha, \delta]$ for the current date, as might be required in a telescope-control program for example (using the older, equinox based, SLALIB functions; a newer set of functions perform similar actions but based on the CIO based approach). Most of the steps can be carried out by calling a single function; there are other SLALIB functions which offer set-piece end-to-end transformation functions for common cases. Note, however, that SLALIB does not set out to provide the capability for arbitrary transformations of star-catalogue data between all possible systems of mean $[\alpha, \delta]$. Only in the (common) cases of FK4, equinox and epoch B1950, to FK5, equinox and epoch J2000, and *vice versa* are proper motion, parallax and radial velocity transformed along with the star position itself, the focus of SLALIB support.

As an example of using the SLALIB functions to transform mean places, here is C code that implements the top-left path of Figure 1. An FK4 $[\alpha, \delta]$ of arbitrary equinox and epoch and with known proper motion and parallax is transformed into an FK5 J2000 $[\alpha, \delta]$ for the current epoch. As a test star we will use $\alpha = 16^{\text{h}} 09^{\text{m}} 55^{\text{s}}.13$, $\delta = -75^{\circ} 59' 27''.2$, equinox 1900, epoch 1963.087, $\mu_{\alpha} = -0''.0312/y$, $\mu_{\delta} = +0''.103/y$, parallax = $0''.062$, radial velocity = -34.22 km/s. The date of observation is 1994.35.

```

:
int j, i;
double r0, d0, eq0, ep0, pr, pd, px, rv, ep1, r1, d1, r2, d2,
      r3, d3, r4, d4, r5, d5, r6, d6, ep1d, ep1b,
      w[3], eb[3], pxx, v[3];

/* RA, Dec etc. of example star. */
slaDtf2r ( 16, 9, 55.13, &r0, &j );
slaDaf2r ( 75, 59, 27.2, &d0, &j );
d0 = -d0;
eq0 = 1900.0;
ep0 = 1963.087;
pr = -0.0312 * S2R;
pd = 0.103 * AS2R;
px = 0.062;
```

```

rv = -34.22;
ep1 = 1994.35;

/* Date of observation as MJD and Besselian epoch. */
ep1d = slaEpj2d ( ep1 );
ep1b = slaEpb ( ep1d );

/* Space motion to the current epoch. */
slaPm ( r0, d0, pr, pd, px, rv, ep0, ep1b, &r1, &d1 );

/* Remove E-terms of aberration for the original equinox. */
slaSubet ( r1, d1, eq0, &r2, &d2 );

/* Precess to B1950. */
r3 = r2;
d3 = d2;
slaPreces ( "FK4", eq0, 1950.0, &r3, &d3 );

/* Add E-terms for the standard equinox B1950. */
slaAddet ( r3, d3, 1950.0, &r4, &d4 );

/* Transform to J2000, no proper motion. */
slaFk45z ( r4, d4, ep1b, &r5, &d5 );

/* Parallax. */
slaEvp ( slaEpj2d ( ep1 ), 2000.0, w, eb, w, w );
pxr = px * AS2R;
slaDcs2c ( r5, d5, v );
for ( i = 0; i < 3; i++ ) {
    v[i] -= pxr * eb[i];
}
slaDcc2s ( v, &r6, &d6 );
:

```

It is interesting to look at how the $[\alpha, \delta]$ changes during the course of the calculation:

16 09 55.130 -75 59 27.20	<i>original equinox and epoch</i>
16 09 54.155 -75 59 23.98	<i>with space motion</i>
16 09 54.229 -75 59 24.18	<i>with old E-terms removed</i>
16 16 28.213 -76 06 54.57	<i>precessed to 1950.0</i>
16 16 28.138 -76 06 54.37	<i>with new E-terms</i>
16 23 07.901 -76 13 58.87	<i>J2000, current epoch</i>
16 23 07.907 -76 13 58.92	<i>including parallax</i>

Other remarks about the above (deliberately complicated) example:

- If the original equinox and epoch were B1950, as is quite likely, then it would be unnecessary to treat space motions and E-terms explicitly. Transformation to FK5 J2000 could be accomplished simply by calling `slaFk425`, after which a call to `slaPm` and the parallax code would complete the work.
- The rigorous treatment of the E-terms has only a small effect on the result. Such refinements are, nevertheless, worthwhile in order to facilitate comparisons and to increase the chances that star positions from different suppliers are compatible.
- The FK4 to FK5 transformations, `slaFk425` and `slaFk45z`, are not as is sometimes assumed simply 50 years of precession, though this indeed accounts for most of the change. The transformations also include adjustments to the equinox, a revised precession model, elimination of the E-terms, a change to the proper-motion time unit and so on. The reason there are two functions rather than just one is that the FK4 system rotates relative to the background, whereas the FK5 system is a much better approximation to an inertial frame, and zero proper motion in FK4 does not, therefore, mean zero proper motion in FK5. SLALIB also provides two functions, `slaFk524` and `slaFk54z`, to perform the inverse transformations.
- Some star catalogues (FK4 itself is one) were constructed using slightly different procedures for the polar regions compared with elsewhere. SLALIB ignores this inhomogeneity and always applies the standard transformations, irrespective of location on the celestial sphere.

3.14 Mean Place to Apparent Place

The *geocentric apparent place* of a source, or *apparent place* for short, is the $[\alpha, \delta]$ if viewed from the centre of the Earth, with respect to the true equator and equinox of date. Transformation of an FK5 mean $[\alpha, \delta]$, equinox J2000, current epoch, to apparent place involves the following effects:

- Light deflection: the gravitational lens effect of the sun.
- Annual aberration.
- Precession-nutation.

The *light deflection* is insignificant for most applications. Its value at the limb of the Sun is about $1''.74$; it falls off rapidly with distance from the Sun and at an elongation of 20° has shrunk to about $0''.02$. (SLALIB neglects light deflection from the planets, which can reach about 20 mas.)

As already described, the *annual aberration* is a function of the Earth's velocity relative to the solar system barycentre (available through the SLALIB functions `slaEvp` and `slaEpv`) and produces shifts of up to about $20''.5$.

The *precession-nutation*, from J2000 to the current epoch, is expressed by a rotation matrix which is available through the SLALIB function `slaPrenut`.

The whole mean-to-apparent transformation can be done using the SLALIB function `slaMap`. As a demonstration, here is a program which lists the *North Polar Distance* ($90^\circ - \delta$) of Polaris for the decade of closest approach to the pole:

```

#include <stdio.h>
#include <slalib.h>

#define PI 3.141592653589793238462643
#define D2R (PI/180.0)
#define PIBY2 (PI/2.0)
#define S2R (PI/(12.0*3600.0))
#define AS2R (PI/(180.0*3600.0))

int main ( )
{
    double rm, dm, pr, pd, date, ra, da;
    int j, iymdf[4];
    long ids, ide, id;

    /* Polaris J2000 RA,Dec and proper motion. */
    slaDtf2r ( 2, 31, 49.8131, &rm, &j );
    slaDaf2r ( 89, 15, 50.661, &dm, &j );
    pr = 0.217272 * S2R;
    pd = -0.01571 * AS2R;

    /* Report co-declination every 10 days from 2096 to 2105. */
    printf ( "Polaris north polar distance (deg) 2096-2105\n\n"
            "   date          NPD\n\n" );
    slaCldj ( 2096, 1, 1, &date, &j );
    ids = (long) ( date + 0.5 );
    slaCldj ( 2105, 12, 31, &date, &j );
    ide = (long) ( date + 0.5 );
    for ( id = ids; id <= ide; id += 10L ) {
        date = (double) id;
        slaDjcal ( 0, date, iymdf, &j );
        slaMap ( rm, dm, pr, pd, 0.0, 0.0, 2000.0, date, &ra, &da );
        printf ( "%4d %2.2d %2.2d %8.5f\n",
                iymdf[0], iymdf[1], iymdf[2], (PIBY2-da)/D2R );
    }

    return 0;
}

```

For cases where the transformation has to be repeated for different times or for more than one star, the straightforward `slaMap` approach is apt to be wasteful as both the Earth velocity and the precession-nutation matrix can be re-calculated relatively infrequently without ill effect. A more efficient method is to perform the target-independent calculations only when necessary, by calling `slaMappa`, and then to use either `slaMapqkz`, when only the $[\alpha, \delta]$ is known, or

`slaMapqk`, when full catalogue positions, including proper motion, parallax and radial velocity, are available. How frequently to call `slaMappa` depends on the accuracy objectives; once per night will deliver sub-arcsecond accuracy for example.

The functions `slaAmp` and `slaAmpqk` allow the reverse transformation, from apparent to mean place.

Similar functions, but using the newer CIO based methods, are described later, in Section 3.17.

3.15 Apparent Place to Observed Place

The *observed place* of a source is its position as seen by a perfect theodolite at the location of the observer. Transformation of an apparent $[\alpha, \delta]$ to observed place involves the following effects:

- $[\alpha, \delta]$ to $[h, \delta]$.
- Diurnal aberration.
- $[h, \delta]$ to $[Az, El]$.
- Refraction.

The transformation from apparent $[\alpha, \delta]$ to topocentric $[h, \delta]$ is made by allowing for *Earth rotation* through the *sidereal time*, θ :

$$h = \theta - \alpha$$

For this equation to work, α must be the apparent right ascension for the time of observation, and θ must be the *local apparent sidereal time*. The latter is obtained as follows:

1. From civil time obtain the coordinated universal time, UTC. (More later on this.)
2. Add the UT1–UTC (typically a few tenths of a second) to give the UT.
3. From the UT compute the Greenwich mean sidereal time (using `slaGmst`).
4. Add the observer’s (east) longitude, giving the local mean sidereal time.
5. Add the equation of the equinoxes (using `slaEqeqx`) to give local apparent sidereal time.

The *equation of the equinoxes* ($= \Delta\psi \cos \epsilon$ plus small terms) is the distance between the mean and true equinox at the date concerned and hence is nutation’s contribution to the sidereal time. Its value is typically a second (of time) or less.

Analogous (but somewhat simpler) computations take place when using the newer CIO based functions, based on intermediate rather than apparent α and on Earth rotation angle rather than sidereal time – see Section 3.17.

Note that for precise work *polar motion* should be taken into account. In the older equinox based SLALIB functions, this is done by introducing changes to the observer's longitude and latitude. This can be done with the function `slaPolmo`. The corrections are always less than about 0".3, and are futile unless the position of the observer's telescope is known to better than a few metres.¹⁰

Bulletins containing tables of observed and predicted UT1–UTC corrections and polar motion data are published every few weeks by the International Earth Rotation and reference systems Service (IERS).

The transformation from apparent $[h, \delta]$ to *topocentric* $[h, \delta]$ consists of allowing for *diurnal aberration* (and, in the case of solar-system objects, diurnal parallax). The diurnal aberration, maximum amplitude 0".2, was described earlier. There is no specific SLALIB function for computing the diurnal aberration, though the functions `slaAop` *etc.* include it, and the required velocity vector can be determined by calling `slaGeoc`.

The next stage is the major coordinate rotation from local equatorial coordinates $[h, \delta]$ into horizon coordinates. The SLALIB functions `slaE2h` *etc.* can be used for this. For high-precision applications the mean geodetic latitude should be corrected for polar motion.

3.15.1 Refraction

The final correction is for atmospheric refraction. This effect, which depends on local meteorological conditions¹¹ and the effective colour of the source/detector combination, increases the observed elevation of the source by a significant effect even at moderate zenith distances, and near the horizon by over 0.5°. The amount of refraction can be computed by calling the SLALIB function `slaRefro`, which performs a ray-trace through a model atmosphere starting from the observer and out into space; however, this requires as input the observed zenith distance, which is what we are trying to predict. For high precision it is therefore necessary to iterate, using the topocentric zenith distance as the initial estimate of the observed zenith distance.

The full `slaRefro` refraction calculation is onerous, and for zenith distances of less than, say, 75° the following model can be used instead:

$$\zeta_{vac} \approx \zeta_{obs} + A \tan \zeta_{obs} + B \tan^3 \zeta_{obs}$$

where ζ_{vac} is the topocentric zenith distance (*i.e. in vacuo*), ζ_{obs} is the observed zenith distance (*i.e. affected by refraction*), and A and B are constants, about 60" and $-0".06$ respectively for a

¹⁰Interpreting polar motion as changes to the observer's $[\lambda, \phi]$ is now considered old-fashioned, and has the disadvantage that it leaves out the effects on observed azimuth. The rigorous method, used in SLALIB's newer CIO based transformations, is to treat polar motion as a full 3-D rotation from the Terrestrial Intermediate Reference System into the International Terrestrial Reference System. The requisite matrix, \mathbf{R}_{PM} in Expression (1), earlier, can be computed using the function `slaPomom`.

¹¹The refraction down to zenith distances of about 70° depends almost entirely upon the conditions at the observer and is hardly affected by the conditions along the incoming beam. The plausibility of this convenient but perhaps surprising result can be seen by considering the simple case of an atmosphere that consists of plane-parallel homogeneous layers; the individual deviations suffered by the beam as it passes through the successive layers all cancel out and so the only refractive indices of consequence are those at the observer and in empty space.

sea-level site. The two constants can be calculated for a given set of conditions by calling either `slaRefco` or `slaRefcoq`.¹²

`slaRefco` works by calling `slaRefro` for two zenith distances and fitting A and B to match. The calculation is onerous, but delivers accurate results whatever the conditions. `slaRefcoq` uses a direct formulation of A and B and is much faster; it is slightly less accurate than `slaRefco` but more than adequate for most practical purposes.

Like the full refraction model, the two-term formulation works in the wrong direction for our purposes, predicting the *in vacuo* (topocentric) zenith distance given the refracted (observed) zenith distance, rather than *vice versa*. The obvious approach of interchanging ζ_{vac} and ζ_{obs} and reversing the signs, though approximately correct, gives avoidable errors which are just significant in some applications; for example about 0".2 at 70° zenith distance. A much better result can easily be obtained, by using one Newton-Raphson iteration as follows:

$$\zeta_{obs} \approx \zeta_{vac} - \frac{A \tan \zeta_{vac} + B \tan^3 \zeta_{vac}}{1 + (A + 3B \tan^2 \zeta_{vac}) \sec^2 \zeta_{vac}}$$

The effect of refraction can be applied to an unrefracted zenith distance by calling `slaRefz` or to an unrefracted $[x, y, z]$ by calling `slaRefv`. Over most of the sky these two functions deliver almost identical results, but beyond $\zeta = 83^\circ$ `slaRefv` becomes unacceptably inaccurate while `slaRefz` remains usable. (However `slaRefv` is significantly faster, which may be important in some applications.) SLALIB also provides a function for computing the airmass, the function `slaAirmas`.

The refraction “constants” returned by `slaRefco` and `slaRefcoq` are slightly affected by colour, especially at the blue end of the spectrum. Where values for more than one wavelength are needed, rather than calling `slaRefco` several times it is more efficient to call `slaRefco` just once, for a selected “base” wavelength, and then to call `slaAtmdsp` once for each wavelength of interest.

All the SLALIB refraction functions work for radio wavelengths as well as the optical/IR band. The radio refraction is very dependent on humidity, and an accurate value must be supplied. There is little wavelength dependence, however, except for close to oxygen and water bands in the region above about 100 GHz. The choice of optical/IR or radio is made by specifying a wavelength greater than 100 μm for the radio case. (The functions in fact accept either wavelength in μm or frequency in GHz, the latter signalled by specifying a negative value.)

3.15.2 Efficiency considerations

The complete apparent place to observed place transformation can be carried out by calling `slaAop`. For improved efficiency in cases of more than one star or a sequence of times, the target-independent calculations can be done once by calling `slaAoppa`, the sidereal time can be updated

¹²The A term is essentially the contribution from a flat atmosphere and the B term the correction for curvature. At zenith distances smaller than about 40°, the B term can be neglected. Much beyond $\zeta = 70^\circ$, the two-term formulation becomes insufficient, at about the same level at which the conditions in the atmosphere along the beam become increasingly important, making simple formulations based only on the ambient conditions untrustworthy.

by calling `slaAoppat`, and `slaAopqk` can then be used to perform the apparent-to-observed transformation. The reverse transformation is available through `sla0ap` and `sla0apqk`. (*n.b.* These functions use accurate but computationally-expensive refraction algorithms for zenith distances beyond about 76° . For many purposes, in-line code tailored to the accuracy requirements of the application will be preferable, for example ignoring polar motion, omitting diurnal aberration and using `slaRefz` to apply the refraction.)

3.16 The Hipparcos Catalogue and the ICRS

With effect from the beginning of 1998, the IAU adopted a new reference system to replace FK5 J2000. The new system, called the International Celestial Reference System (ICRS), differs profoundly from all predecessors in that the link with solar-system dynamics was broken; the ICRS axes are defined in terms of the coordinates of a set of extragalactic sources, not in terms of the mean equator and equinox at a given reference epoch. Although the ICRS and FK5 coordinates of any given object are almost the same, the orientation of the new system was essentially arbitrary, and the close match to FK5 J2000 (about 23 mas, it turned out) was contrived purely for reasons of continuity and convenience.

A distinction is made between the reference *system* (the ICRS) and *frame* (ICRF). The ICRS is the set of prescriptions and conventions together with the modelling required to define, at any time, a triad of axes. The ICRF is a practical realization, and currently consists of a catalogue of equatorial coordinates for 608 extragalactic radio sources observed by VLBI.

The best optical realization of the ICRS currently available is the Hipparcos catalogue. The extragalactic sources were not directly observable by the Hipparcos satellite and so the link from Hipparcos to ICRS was established through a variety of indirect techniques: VLBI and conventional interferometry of radio stars, photographic astrometry and so on. The Hipparcos system is aligned to the ICRS to within about 0.5 mas and 0.5 mas/year (at epoch 1991.25).

The Hipparcos catalogue includes all of the FK5 stars, which has enabled the orientation and spin of the latter to be studied. At epoch J2000, the misalignment of the FK5 system with respect to Hipparcos (and hence ICRS) are about 32 mas and 1 mas/year respectively. Consequently, for many practical purposes, including pointing telescopes, the IAU 1976-1982 conventions on reference systems and Earth orientation remain adequate and there is no need to change to Hipparcos coordinates, new precession-nutation models and so on. However, for the most exacting astrometric applications, SLALIB provides some support for Hipparcos coordinates in the form of four new functions: `slaFk52h` and `slaH2fk5`, which transform FK5 positions and proper motions to the Hipparcos system and *vice versa*, and `slaFk5hz` and `slaHfk5z`, where the transformations are for stars whose Hipparcos proper motion is zero.

Further information on the ICRS can be found in the paper by M. Feissel and F. Mignard, *Astron. Astrophys.* **331**, L33-L36 (1988).

3.17 CIO based ICRS-CIRS-Observed Transformations

SLALIB includes functions to implement all the transformations set out in Figure 2, and these are recommended for new applications in preference to the older functions: `slaMap`, `slaAop` *etc.*

The latter are, however, computationally less demanding than the newer functions (because of the much shorter nutation series) and give results of more than adequate accuracy. Consequently there may be little incentive to modernize existing applications.

The transformation from ICRS catalogue data to CIRS coordinates can be carried out using the function `slaC2i`. Where more than one such transformation is to be performed for the same time (or nearly the same time), it is much more efficient to precompute the transformation parameters using the function `slaC2ipa` (for a terrestrial observer) or `slaC2ipas` (for an observer with a specified geocentric position and velocity such as a satellite in Earth orbit) once and then to call the transformation function `slaC2iqk` (or the function `slaC2iqkz` for the case where proper motion and parallax corrections are not required) multiple times. Note that the `slaC2ipa` function returns the equation of the origins (EO) as well as the ICRS-to-CIRS parameters; in equinox based applications this can be used to compute the Greenwich sidereal time (subtract the EO from the Earth rotation angle) and apparent right ascension (subtract the EO from the CIRS right ascension).

Should GCRS coordinates be required, rather than CIRS, the procedure is to use the `slaC2ipa` to compute the ICRS-to-CIRS parameters, and then to call `slaCripad`. This disables the bias-precession-nutation portion of the transformation chain.

The reverse transformation, from CIRS to astrometric place, can be accomplished by means of the functions `slaI2c` (self-contained) or `slaI2cqk` (given precomputed ICRS-to-CIRS parameters obtained with `slaC2ipa`).

The transformation from CIRS coordinates to observed place is supported by a similarly arranged set of functions. To perform one such transformation, call `slaI2o`; to perform many such transformations for one set of circumstances, call `slaI2opa` once, to precompute the CIRS-to-observed parameters, and then `slaI2oqk` repeatedly. Often the circumstances are static (or nearly so) and the only factor that is changing rapidly is time; this is the case when controlling a telescope for example. Under these circumstances, a call to `slaI2opat` will update only the time-dependent part of the transformation. Finally, the function `slaI2opad` disables the diurnal aberration part of the transformation. This is especially useful in combination with the `slaC2ipas` function, which allows the observer's velocity to be specified, for use in the aberration part of the ICRS-to-GCRS transformation. For a terrestrial observer, the option therefore exists to take the diurnal velocity into account during the ICRS-to-GCRS phase rather than separately during the CIRS-to-observed phase; for a space observer, the spacecraft velocity can be taken into account during the ICRS-to-GCRS phase and the unwanted diurnal aberration correction disabled when computing the terrestrial coordinates of the sub-stellar point.

For the reverse transformation, observed place to CIRS, the functions `slaO2i` (self-contained) and `slaO2iqk` (using precomputed parameters) are available.

When working in topocentric place, rather than observed place, the refraction corrections can be disabled by specifying zero pressure in the calls to `slaI2o`, `slaI2opa` and `slaO2i`. It is a good plan to perform such an "airless" prediction as a preliminary, to check if the star is above the horizon, and only to carry out the full prediction, which includes the computationally expensive refraction calculation, if it is.

The following C program computes the observed azimuth and altitude (or elevation) for a given star seen from a given site at a given time. Two completely independent methods are demon-

strated – (i) using the newer, CIO-based, functions and (ii) using the older, equinox based, functions – and the results compared. Realistic star, site and Earth orientation data are used.

```

#include <stdio.h>
#include <slalib.h>

int main ( )
#define R2D 57.29577951308232      /* radians to degrees */
#define R2MAS 206264.806247096355e3 /* radians to mas */
{
    CIPars ciprms;
    double rc, dc, pr, pd, px, rv, date, xp, yp, dut, dx, dy,
           elong, phi, hm, tk, phpa, rh, wlfq, tlr, eo,
           ri, di, aob, zob, hob, dob, rob, ra, da,
           ra1, da1, aob1, zob1, elong1, phi1, daz;

    /* A star (phi Peg). */
    rc = 358.1220336/R2D;      /* RA (ICRS, radians) */
    dc = 19.12028568/R2D;     /* Dec (ICRS, radians) */
    pr = 57.21/cos(dc)/R2MAS; /* RA proper motion (radians/year) */
    pd = 62.03/R2MAS;         /* Dec proper motion (radians/year) */
    px = 0.01270;             /* parallax (arcsec) */
    rv = -14.1;               /* radial velocity (km/s) */

    /* Date (UTC, but also used as approximate TDB). */
    date = 53569.4;           /* 2005 July 18 09:36, as an MJD */

    /* Earth orientation parameters (from IERS Bulletin B). */
    xp = -19.60/R2MAS;        /* polar motion x (radians) */
    yp = -414.62/R2MAS;       /* polar motion y (radians) */
    dut = -0.606376;          /* UT1-UTC (s) */
    dx = 0.09/R2MAS;          /* nutation adjustment dX */
    dy = -0.12/R2MAS;         /* nutation adjustment dY */

    /* Local circumstances (Gemini North). */
    elong = -155.4690472/R2D; /* east longitude (radians) */
    phi = 19.82380278/R2D;    /* latitude (radians) */
    hm = 4213.4;              /* height a.s.l. (m) */
    tk = 278;                 /* temperature (K) */
    phpa = 605.0;             /* pressure (hPa = mb) */
    rh = 0.09;                /* relative humidity (0-1) */
    wlfq = 1.2;               /* observing wavelength (microns) */
    tlr = 0.0065;             /* lapse rate (K/m) */

    /* Compute the ICRS-to-CIRS parameters. */
    slaC2ipa ( date, dx, dy, &ciprms, &eo );

```



```

/* Perform the transformation to CIRS. */
slaC2iqk ( rc, dc, pr, pd, px, rv, &ciprms, &ri, &di );

/* Perform the transformation from CIRS to observed direction. */
slaI2o ( ri, di, date, dut, elong, phi, hm, xp, yp,
        tk, phpa, rh, wlfq, tlr, &aob, &zob, &hob, &dob, &rob );

/* Determine also the equinox-based apparent place. */
ra = slaDranrm ( ri - eo );
da = di;

/* Do the same transformations using the earlier classical functions. */
slaMap ( rc, dc, pr, pd, px, rv, 2000.0, date, &ra1, &da1 );
slaAop ( ra1, da1, date, dut, elong, phi, hm, xp, yp,
        tk, phpa, rh, wlfq, tlr, &aob1, &zob1, &hob, &dob, &rob );
slaPolmo ( elong, phi, xp, yp, &elong1, &phi1, &daz );
aob1 += daz;

/* Report. */
printf ( "          new          old          diff\n\n"
        "ICRS RA  %+15.9f\n", slaDranrm(rc)*R2D );
printf ( "      Dec %+15.9f\n\n", dc*R2D );
printf ( "CIRS RA  %+15.9f\n", slaDranrm(ri)*R2D );
printf ( "      Dec %+15.9f\n\n", di*R2D );
printf ( "appt RA  %+15.9f", slaDranrm(ra)*R2D );
printf ( "%+16.9f", slaDranrm(ra1)*R2D );
printf ( "%+9.3f\n", slaDrange(ra1-ra)*R2MAS );
printf ( "      Dec %+15.9f", da*R2D );
printf ( "%+16.9f", da1*R2D );
printf ( "%+9.3f\n\n", (da1-da)*R2MAS );
printf ( "obs  Az  %+15.9f", slaDranrm(aob)*R2D );
printf ( "%+16.9f", slaDranrm(aob1)*R2D );
printf ( "%+9.3f\n", slaDrange(aob1-aob)*R2MAS );
printf ( "      Alt %+15.9f", 90.0-zob*R2D );
printf ( "%+16.9f", 90.0-zob1*R2D );
printf ( "%+9.3f\n\n", (zob-zob1)*R2MAS );
printf ( "          deg          deg          mas\n" );

return 0;
}

```

The program produces the following output:

new	old	diff
-----	-----	------

ICRS	RA	+358.122033600		
	Dec	+19.120285680		
CIRS	RA	+358.123557503		
	Dec	+19.149967714		
appt	RA	+358.193464260	+358.193449769	-52.170
	Dec	+19.149967714	+19.149967781	+0.241
obs	Az	+76.523695169	+76.523695112	-0.204
	Alt	+21.469419881	+21.469419946	+0.233
		deg	deg	mas

Note in particular the following:

- The end result, namely the predicted azimuth and altitude, is essentially the same irrespective of whether the old or new SLALIB functions are used. The difference, a small fraction of 1 mas, comes mainly from the use of different precession-nutation models. (The real accuracy is unlikely to be better than, say, 0".2, limited by the refraction predictions.)
- The CIRS and apparent right ascensions are markedly different, whereas the two declinations are identical. This shows that they differ only in respect of the adopted α zero-point. (The CIRS α is just a few arcseconds from the original ICRS α , a reflection of the fact that the CIO is at present very close to the GCRS meridian.)
- The apparent right ascensions computed using the old and new functions disagree by a hefty 0".052. There is a compensating difference in the Greenwich sidereal time (not shown), so that the disagreement cancels out when hour angle is computed. The reason for the difference is that the former methods assumed that the J2000 equinox was at the GCRS origin, whereas the new methods place the J2000 equinox at the intersection of the ecliptic and the GCRS equator, which is correct. The difference is a reminder that it is dangerous to mix old and new: when using the older, equinox based, SLALIB functions, GST should be computed using `slaGmst` or `slaGmsta`, and `slaEeqqx`, whereas when developing equinox based applications but using the newer, CIO-based, SLALIB functions, GST should be computed using the `slaGst` function.

3.18 Time-scales

SLALIB provides transformations between several time-scales, and involves use of one or two others:

- TAI: International Atomic Time
- UTC: Coordinated Universal Time
- TT: Terrestrial Time

- TDB: Barycentric Dynamical Time.
- UT: Universal Time
- GMST: Greenwich mean sidereal time
- GAST (or GST): Greenwich apparent sidereal time.
- LAST: local apparent sidereal time

Strictly speaking, UT and the sidereal times are not *times* in the physics sense, but *angles* that describe Earth rotation.

Three obsolete time-scales should be mentioned here to avoid confusion.

- GMT: Greenwich Mean Time – can mean either UTC or UT.
- ET: Ephemeris Time – more or less the same as either TT or TDB.
- TDT: Terrestrial Dynamical Time – former name of TT.

Time-scales that have no SLALIB support at present:

- Any form of local civil time (BST, PDT *etc.*)
- TCG: geocentric coordinate time.
- TCB: barycentric coordinate time.
- GPS time: a common starting point for real-time applications, exactly 19 s behind TAI.

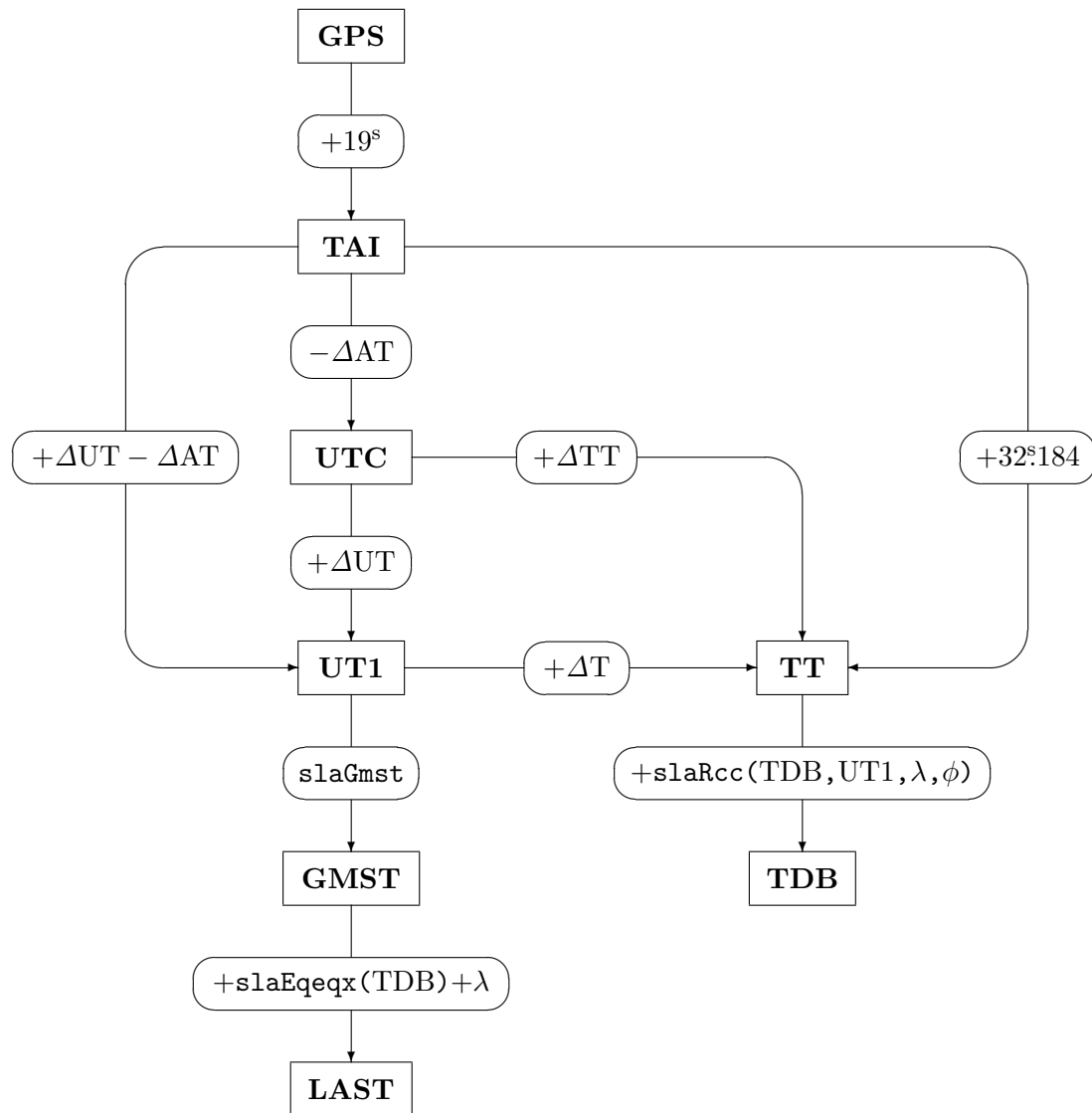
The relationships between the various time-scales and Earth rotation measures are shown in Figure 3.

3.18.1 Atomic Time: TAI

International Atomic Time, TAI, is a “laboratory” time-scale with no link to astronomical observations except in an historical sense. Its unit is the SI second, which is defined in terms of a specified number of wavelengths of the radiation produced by a certain electronic transition in the caesium 133 atom. It is realized through a changing population of high-precision atomic clocks held at standards institutes in various countries. There is an elaborate process of continuous intercomparison, leading to a weighted average of all the clocks involved.

Though TAI shares the same second as the more familiar UTC, the two time-scales are noticeably separated in epoch because of the build-up of leap seconds (see Section 3.18.2). At the time of writing, UTC lags over half a minute behind TAI.

For any given date, the difference TAI–UTC can be obtained by calling the SLALIB function `slaDat`. Note, however, that an up-to-date copy of the function must be used if the most recent leap seconds are required. For applications where this is critical, mechanisms independent of SLALIB and under local control must be set up; in such cases `slaDat` can be useful as an independent check, for test dates within the range of the available version. Up-to-date information on TAI–UTC is available from <ftp://maia.usno.navy.mil/ser7/tai-utc.dat>.

Figure 3: **Relationships Between Time-Scales**

The diagram shows how the various time-scales (and Earth rotation measures) supported by SLALIB are interrelated. GPS time is also shown, as this is a common input to real-time applications. In some cases the time-scales are offset from one another by specified amounts, usually designated by standard symbols such as “ ΔAT ”. In other cases the transformation involves canonical formulae, and here these are indicated by quoting the name of the appropriate SLALIB function, for example “`slaRcc`”. It is possible to reach some of the timescales via more than one route. An important example of this is the transformation from TAI to UT1, where there is the choice of going directly or via UTC. In a practical application (such as a telescope control system), the direct route has the important advantage that the offset varies in a continuous manner; the route via UTC introduces the complication of dealing with leap seconds.

3.18.2 Universal Time: UT, UTC

Universal Time, UT, or more specifically UT1, is in effect mean solar time and is really an expression of Earth rotation rather than a measure of time. Originally defined in terms of a point in the sky called “the fictitious mean Sun”, UT is now defined through its relationship with Earth rotation angle (formerly through sidereal time). Because the Earth’s rotation rate is slightly irregular and gradually decreasing,¹³ the UT second is not precisely matched to the SI second. This makes UT itself unsuitable for use as a time-scale.

That role is instead taken by *Coordinated Universal Time*, UTC, which is clock-based and is the foundation of civil timekeeping. Most time zones differ from UTC by an integer number of hours, though a few (*e.g.* parts of Canada and Australia) differ by $n + 0.5$ hours. Since its introduction, UTC has been kept roughly in step with UT by a variety of adjustments that are agreed in advance and then carried out in a coordinated manner by the providers of time services—hence the name. Though rate changes were used in the past, nowadays all such adjustments are made by occasionally inserting a whole second. This procedure is called a *leap second*. Because the day length is now slightly longer than 86400 SI seconds, a leap second amounts to stopping the UTC clock for a second to let the Earth catch up.

You need UT1 in order to point a telescope or antenna at a celestial target. To obtain it starting from UTC, you have to look up the value of UT1–UTC for the date concerned in tables published by the International Earth Rotation and reference systems Service (IERS); this quantity, kept in the range $\pm 0^s.9$ by means of leap seconds, is then added to the UTC. The quantity UT1–UTC, which typically changes by of order 1 ms per day, can be obtained only by observation (VLBI using extragalactic radio sources), though seasonal trends are well known and the IERS listings are able to predict some way into the future with adequate accuracy for pointing telescopes.

UTC leap seconds are introduced as necessary, usually at the end of December or June. Because on the average the solar day is slightly longer than the nominal 86,400 SI seconds, leap seconds are always positive; however, provision exists for negative leap seconds if needed. The form of a leap second can be seen from the following description of the end of June 1994:

			UTC	UT1–UTC	UT1
1994	June	30	23 59 58	–0.218	23 59 57.782
			23 59 59	–0.218	23 59 58.782
			23 59 60	–0.218	23 59 59.782
	July	1	00 00 00	+0.782	00 00 00.782
			00 00 01	+0.782	00 00 01.782

¹³The Earth is slowing down because of tidal effects. The SI second reflects the length-of-day in the mid-19th century, when the astronomical observations that established modern timekeeping were being made. Since then, the average length-of-day has increased by roughly 2 ms. Superimposed in this gradual slowdown are variations (seasonal and decadal) that are geophysical in origin, notably due to large scale movements of water and atmosphere. Because of conservation of angular momentum, as the Earth’s rotation-rate decreases, the Moon moves farther away. In 50 billion years the distance of the Moon will be at a maximum, 44% greater than now, at which stage day and month will both equal 47 present days.

Note that UTC has to be expressed as hours, minutes and seconds (or at least in seconds for a given date) if leap seconds are to be taken into account in the correct manner. It is improper to express a UTC as a Julian Date, for example, because there will be an ambiguity during a leap second (in the above example, 1994 June 30 23^h 59^m 60^s0 and 1994 July 1 00^h 00^m 00^s0 would *both* come out as MJD 49534.00000). Although in the vast majority of cases this won't matter, there are potential problems in on-line data acquisition systems and in applications involving taking the difference between two times. Note that although the functions `slaDat` and `slaDtt` expect UTC in the form of an MJD, the meaning here is really a whole-number *date* rather than a time. Though the functions will accept a fractional part and will almost always function correctly, on a day which ends with a leap second incorrect results would be obtained during the leap second itself because by then the MJD would have moved into the next day.

3.18.3 Sidereal Time: GMST, LAST *etc.*

Sidereal time is like the time of day but relative to the stars rather than to the Sun. After one sidereal day the stars come back to the same place in the sky, apart from sub-arcsecond precession effects. Because the Earth rotates faster relative to the stars than to the Sun by one day per year, the sidereal second is shorter than the solar second; the ratio is about 0.9973.

The *Greenwich mean sidereal time*, GMST, is linked to UT1 by a numerical formula which is implemented in the SLALIB functions `slaGmst` and `slaGmsta`. There are, of course, no leap seconds in GMST, but the sidereal second (measured in SI seconds) changes in length along with the UT1 second, and also varies over long periods of time because of slow changes in the Earth's orbit. This makes sidereal time unsuitable for everything except predicting the apparent directions of celestial sources, in other words as an angle rather than a time.

The *local apparent sidereal time*, LAST, is the apparent right ascension of the local meridian, from which the hour angle of any star can be determined knowing its right ascension. LAST can be obtained from the GMST by adding the east longitude and the *equation of the equinoxes*. The latter, already described, is an aspect of the nutation effect and can be predicted by calling the SLALIB function `slaEqeqx` or, neglecting certain very small terms, by calling `slaNutc` and using the expression $\Delta\psi \cos \epsilon$.

GAST, or plain GST, is GMST plus the equation of the equinoxes.

The celestial transformation methods introduced by the IAU in 2000 do not use sidereal time, its role having been taken over by Earth rotation angle. ERA is a simple linear transformation of UT1, involving neither the precession terms that disfigure the GMST(UT1) formula nor the complications of the equation of the equinoxes. The difference between the two, ERA–GST, is the equation of the origins, which can be computed by the functions `slaEo` and `slaEors`. Equivalently, $EO = \alpha_{\text{CIRS}} - \alpha_{\text{apparent}}$.

3.18.4 Dynamical Time: TT, TDB

Dynamical time (formerly Ephemeris Time, ET) is the independent variable in the theories which describe the motions of bodies in the solar system. When using published formulae or tables

that model the position of the Earth in its orbit, for example, or look up the Moon's position in a precomputed ephemeris, the date and time must be in terms of one of the dynamical time-scales. It is a common but understandable mistake to use UTC directly, in which case the results will be over a minute out (at the time of writing).

It is not hard to see why such time-scales are necessary. UTC would clearly be unsuitable as the argument of an ephemeris because of leap seconds. A solar-system ephemeris based on UT1 or sidereal time would somehow have to include the unpredictable variations of the Earth's rotation. TAI would work, but in principle the ephemeris and the ensemble of atomic clocks would eventually drift apart. In effect, the ephemeris *is* a clock, with the bodies of the solar system the hands from which the ephemeris time is read.

Only two of the dynamical time-scales are of any great importance to observational astronomers, TT and TDB.

Terrestrial Time, TT, is the theoretical time-scale of apparent geocentric ephemerides of solar system bodies. It applies to clocks at sea-level, and for practical purposes it is tied to Atomic Time TAI through the formula $TT = TAI + 32^s.184$. In practice, therefore, the units of TT are ordinary SI seconds, and the offset of $32^s.184$ with respect to TAI is fixed. The SLALIB function `slaDtt` returns TT–UTC for a given UTC (*n.b.* `slaDtt` calls `slaDat`, and the latter must be an up-to-date version if recent leap seconds are to be taken into account).

Barycentric Dynamical Time, TDB, is a *coordinate time*, suitable for labelling events that are most simply described in a context where the bodies of the solar system are absent. Applications include the emission of pulsar radiation and the motions of the solar-system bodies themselves. When the readings of the observer's TT clock are labelled using such a coordinate time, differences are seen because the clock is affected by its speed in the barycentric coordinate system and the gravitational potential in which it is immersed. Equivalently, observations of pulsars expressed in TT would display similar variations (quite apart from the familiar light-time effects).

TDB is defined in such a way that (for terrestrial observers) it keeps close to TT on the average, with the relativistic effects emerging as quasi-periodic differences of maximum amplitude rather less than 2 ms. This is negligible for many purposes, so that TT can act as a perfectly adequate surrogate for TDB in most cases, but unless taken into account would swamp long-term analysis of pulse arrival times from the millisecond pulsars.

Most of the variation between TDB and TT comes from the ellipticity of the Earth's orbit; the TT clock's speed and gravitational potential vary slightly during the course of the year, and as a consequence its rate as seen from an outside observer varies due to transverse Doppler effect and gravitational redshift. The main component is a sinusoidal variation of amplitude $0^s.0017$; higher harmonics, and terms caused by Moon and planets, lie two orders of magnitude below this dominant annual term. Diurnal (topocentric) terms, a function of UT, are $2\ \mu\text{s}$ or less.

The IAU 1976 resolution defined TDB by stipulating that TDB–TT consists of periodic terms only. This provided a good qualitative description, but turned out to contain hidden assumptions about the form of the solar-system ephemeris and hence lacked dynamical rigour. A later resolution, in 1991, introduced new coordinate time-scales, TCG and TCB, and identified TDB as a linear transformation of one of them (TCB) with a rate chosen not to drift from TT on

the average. Unfortunately even this improved definition has proved to contain ambiguities. The SLALIB `slaRcc` function implements TDB in the way that is most consistent with the 1976 definition and with existing practice. It provides a model of TDB–TT accurate to a few nanoseconds.

Unlike TDB, the IAU 1991 coordinate time-scales TCG and TCB (not supported by SLALIB functions at present) do not have their rates adjusted to track TT and consequently gain on TT and TDB, by about 0^s.02/year and 0^s.5/year respectively.

As already pointed out, the distinction between TT and TDB is of no practical importance for most purposes. For example when calling `slaPrenut` to generate a precession-nutation matrix, or when calling `slaEvp` or `slaEpv` to predict the Earth’s position and velocity, the time argument is strictly TDB, but TT is entirely adequate and will require much less computation.

The time-scale used by the JPL solar-system ephemerides is called T_{eph} and numerically is essentially the same as TDB.

Predictions of topocentric solar-system phenomena such as occultations and eclipses require solar time UT as well as dynamical time. TT/TDB/ET is all that is required in order to compute the geocentric circumstances, but if horizon coordinates or geocentric parallax are to be tackled UT is also needed. A rough estimate of $\Delta T = ET - UT$ is available via the function `slaDt`. For a given epoch (*e.g.* 1650) this returns an approximation to ΔT in seconds.

3.19 Calendars

The ordinary *Gregorian Calendar Date*, together with a time of day, can be used to express a point in time in any desired time-scale. For many purposes, however, a continuous count of days is more convenient, and for this purpose the system of *Julian Day Number* can be used. JD zero is located about 7000 years ago, well before the historical era, and is formally defined in terms of Greenwich noon; for example Julian Day Number 2449444 began at noon on 1994 April 1. *Julian Date* is the same system but with a fractional part appended; Julian Date 2449443.5 was the midnight on which 1994 April 1 commenced. Because of the unwieldy size of Julian Dates and the awkwardness of the half-day offset, it is accepted practice to remove the leading ‘24’ and the trailing ‘.5’, producing what is called the *Modified Julian Date*: $MJD = JD - 2400000.5$. SLALIB functions use MJD, as opposed to JD, throughout, largely to avoid loss of precision. 1994 April 1 commenced at MJD 49443.0.

Despite JD (and hence MJD) being defined in terms of (in effect) UT, the system can be used in conjunction with other time-scales such as TAI, TT and TDB (and even sidereal time through the concept of *Greenwich Sidereal Date*). However, it is improper to express a UTC as a JD or MJD because of leap seconds.

SLALIB has six functions for converting to and from dates in the Gregorian calendar. The functions `slaCldj` and `slaCaldj` both convert a calendar date into an MJD, the former interpreting years between 0 and 99 as 1st century and the latter as late 20th or early 21st century. The functions `slaDjcl` and `slaDjcal` both convert an MJD into calendar year, month, day and fraction of a day; the latter performs rounding to a specified precision. Some of SLALIB’s low-precision ephemeris functions (`slaEarth`, `slaMoon` and `slaEcor`) work in terms of year plus day-in-year

(where day 1 = January 1st, at least for the modern era). This form of date can be generated by calling `slaCalyd` (which defaults years 0-99 into 1950-2049) or `slaClyd` (which covers the full range from prehistoric times).

3.20 Geocentric Coordinates

The location of the observer on the Earth is significant in a number of ways. The most obvious, of course, is the effect of longitude and latitude on the observed $[Az, El]$ of a star. Less obvious is the need to allow for geocentric parallax when finding the Moon with a telescope (and when doing high-precision work involving the Sun or planets), and the need to correct observed radial velocities and apparent pulsar periods for the effects of the Earth's rotation.

The SLALIB function `slaObs` supplies details of groundbased observatories from an internal list. This is useful when writing applications that apply to more than one observatory; the user can enter a brief name, or browse through a list, and be spared the trouble of typing in the full latitude, longitude *etc.* The following C program reports the full name, longitude and latitude of a specified observatory (*n.b.* beware of the longitude sign convention, which is west positive for historical reasons):

```
#include <stdio.h>
#include <string.h>
#include <slalib.h>

#define R2D 57.295779513

int main ( )
{
    char ident[11], name[41];
    double w, p, h;

    strcpy ( ident, "AAT" );
    slaObs ( 0, ident, name, &w, &p, &h );
    printf ( "%s:  E%g, %g, %gm\n", name[0] == (char) '?' ? "?" : name,
              -w*R2D, p*R2D, h );
    return 0;
}
```

The output is:

```
Anglo-Australian 3.9m Telescope:  E149.066, -31.277, 1164m
```

The following C program lists all the supported observatories:

```

#include <stdio.h>
#include <slalib.h>

int main ( )
{
    char ident[11], name[41];
    int n;
    double w, p, h;

    n = 1;
    name[0] = (char) 0;
    while ( name[0] != (char) '?' ) {
        slaObs ( n, ident, name, &w, &p, &h );
        if ( name[0] != (char) '?' ) {
            printf ( "%3d    %-10s    %s\n", n, ident, name );
        }
        n++;
    }
    return 0;
}

```

The function `slaGeoc` converts a *geodetic latitude* (one referred to the local horizon) to a geocentric position, taking into account the Earth's oblateness and also the height above sea level of the observer. The results are expressed in vector form, namely as the distance of the observer from the spin axis and equator respectively. The *geocentric latitude* can be found by evaluating `atan2` of the two numbers. A full 3-D vector description of the position and velocity of the observer is available through the function `slaPvobs`. For a specified geodetic latitude, height above sea level, and local sidereal time, `slaPvobs` generates a 6-element vector containing the position and velocity with respect to the true equator and equinox of date (*i.e.* compatible with apparent $[\alpha, \delta]$). For some applications it will be necessary to convert to a mean $[\alpha, \delta]$ system (notably FK5, J2000) by multiplying elements 1-3 and 4-6 respectively with the appropriate precession matrix. (In theory an additional correction to the velocity vector is needed to allow for differential precession, but this correction is always negligible.)

See also the discussion of the function `slaRverot`, later.

3.21 Ephemerides

SLALIB includes functions for generating positions and velocities of Solar-System bodies. The accuracy objectives are modest, and the SLALIB facilities do not attempt to compete with precomputed ephemerides such as those provided by JPL, or with models containing thousands of terms. It is also worth noting that SLALIB's very accurate star coordinate transformation functions are not strictly applicable to solar-system cases, though they are adequate for most practical purposes.

Earth/Sun ephemerides can be generated using the functions `slaEvp` and `slaEpv`, each of which predict Earth position and velocity with respect to both the solar-system barycentre and the Sun. The two functions offer different trade-offs between accuracy and execution time. For most purposes, `slaEvp` is adequate: maximum velocity error is 0.42 metres per second; maximum heliocentric position error is 1600 km (equivalent to about 2" at 1 AU), with barycentric position errors about 4 times worse. The larger and slower `slaEpv` delivers 3σ results of 0.005 metres per second in velocity and 15 km in position, and is particularly useful when predicting apparent directions of near-Earth objects. (The Sun's position as seen from the Earth can, of course, be obtained simply by reversing the signs of the Cartesian components of the Earth:Sun vector.)

Geocentric Moon ephemerides are available from `slaDmoon`, which predicts the Moon's position and velocity with respect to the Earth's centre. Direction accuracy is usually better than 10 km (5") and distance accuracy a little worse.

Lower-precision but faster predictions for the Sun and Moon can be made by calling `slaEarth` and `slaMoon`. Both are `float` precision and accept dates in the form of year, day-in-year and fraction of day (starting from a calendar date you need to call `slaClyd` or `slaCalyd` to get the required year and day). The `slaEarth` function returns the heliocentric position and velocity of the Earth's centre for the mean equator and equinox of date. The accuracy is better than 20,000 km in position and 10 metres per second in speed. The position and velocity of the Moon with respect to the Earth's centre for the mean equator and ecliptic of date can be obtained by calling `slaMoon`. The positional accuracy is better than 30" in direction and 1000 km in distance.

Approximate ephemerides for all the major planets (also Sun, Moon and Pluto) can be generated by calling `slaPlanet` or `slaRdplan`. These functions offer arcminute accuracy (much better for the inner planets) and over a span of several millennia. (The accuracy for the dwarf planet Pluto is similar but for only ± 100 years.) The function `slaPlanet` produces heliocentric position and velocity in the form of equatorial $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ for the mean equator and equinox of J2000. The vectors produced by `slaPlanet` can be used in a variety of ways according to the requirements of the application concerned. The function `slaRdplan` uses `slaPlanet` and `slaDmoon` to deal with the common case of predicting a planet's apparent $[\alpha, \delta]$ and angular size as seen by a terrestrial observer.

Note that in predicting the position in the sky of a solar-system body it is necessary to allow for geocentric parallax. This correction is *essential* in the case of the Moon, where the observer's position on the Earth can affect the Moon's $[\alpha, \delta]$ by up to 1° . The calculation can most conveniently be done by calling `slaPvobs` and subtracting the resulting 6-vector from the one produced by `slaDmoon`, as is demonstrated by the following example:

```
#include <stdio.h>
#include <slalib.h>

int main ( )

/* Demonstrate the size of the geocentric parallax correction */
/* in the case of the Moon. The test example is for the AAT, */
/* before midnight, in summer, near first quarter.          */
```

```

{
    char name[41], s;
    int j, i, m[4];
    double slongw, slat, h, djutc, fdutc, djut1, djtt, stl,
           rmatn[3][3], pmm[6], pmt[6], rm, dm, pvo[6], tl;

    /* Get AAT longitude and latitude in radians and height in metres. */
    slaObs ( 0, "AAT", name, &slongw, &slat, &h );

    /* UTC (1992 January 13, 11 13 59) to MJD. */
    slaCldj ( 1992, 1, 13, &djutc, &j );
    slaDtf2d ( 11, 13, 59.0, &fdutc, &j );
    djutc += fdutc;

    /* UT1 (UT1-UTC value of -0.152 sec is from IERS Bulletin B). */
    djut1 = djutc + (-0.152) / 86400.0;

    /* TT. */
    djtt = djutc + slaDtt ( djutc ) / 86400.0;

    /* Local apparent sidereal time. */
    stl = slaGmst( djut1 ) - slongw + slaEeqqx ( djtt );

    /* Geocentric position/velocity of Moon (mean of date). */
    slaDmoon ( djtt, pmm );

    /* Nutation to true equinox of date. */
    slaNut ( djtt, rmatn );
    slaDmxv ( rmatn, pmm, pmt );
    slaDmxv ( rmatn, pmm+3, pmt+3 );

    /* Report geocentric HA,Dec. */
    slaDcc2s ( pmt, &rm, &dm );
    slaDr2tf ( 2, slaDranrm(stl-rm), &s, m );
    printf ( " geocentric:  %c%2.2d %2.2d %2.2d.%2.2d",
            s, m[0], m[1],m[2],m[3] );

    slaDr2af ( 1, dm, &s, m );
    printf ( "  %c%2.2d %2.2d %2.2d.%1d\n", s, m[0], m[1],m[2],m[3] );

    /* Geocentric position of observer (true equator and equinox of date). */
    slaPvobs ( slat, h, stl, pvo );

    /* Place origin at observer. */
    for ( i = 0; i < 6; i++ ) {
        pmt[i] -= pvo[i];
    }
}

```

```

/* Allow for planetary aberration. */
t1 = 499.004782 * sqrt ( pmt[0]*pmt[0]
                        + pmt[1]*pmt[1]
                        + pmt[2]*pmt[2] );
for ( i = 0; i < 3; i++ ) {
    pmt[i] -= t1 * pmt[i+3];
}

/* Report topocentric HA,Dec. */
slaDcc2s ( pmt, &rm, &dm );
slaDr2tf ( 2, slaDranrm(stl-rm), &s, m );
printf ( "topocentric:  %c%2.2d %2.2d %2.2d.%2.2d",
        s, m[0], m[1],m[2],m[3] );

slaDr2af ( 1, dm, &s, m );
printf ( "  %c%2.2d %2.2d %2.2d.%1d\n", s, m[0], m[1],m[2],m[3] );

return 0;
}

```

The output produced is as follows:

```

geocentric:  +03 06 55.55  +15 03 38.8
topocentric:  +03 09 23.76  +15 40 51.4

```

(An easier but less instructive method of estimating the topocentric apparent place of the Moon is to call the function `slaRdplan`.)

As an example of using `slaPlanet`, the following program estimates the geocentric separation between Venus and Jupiter during a close conjunction in 2 BC, which is a star-of-Bethlehem candidate:

```

#include <stdio.h>
#include <slalib.h>

int main ( )

/* Compute time and minimum geocentric apparent separation      */
/* between Venus and Jupiter during the close conjunction of 2 BC. */

{
    double sepmin, djd0, fd, djd, djdm, pv[6], rmatp[3][3],
        pvm[6], pve[6], t1, dx, dy, dz, rv, dv, rj, dj, sep;
    int ihour, imin, j, i, ihmin, immin;

```

```

/* Search for closest approach on the given day. */
djd0 = 1720859.5;
sepmin = 1e10;
for ( ihour = 20; ihour <= 22; ihour++ ) {
    for ( imin = 0; imin < 60; imin++ ) {
        slaDtf2d ( ihour, imin, 0.0, &fd, &j );

        /* Julian date and MJD. */
        djd = djd0 + fd;
        djdm = djd - 2400000.5;

        /* Earth to Moon (mean of date). */
        slaDmoon ( djdm, pv );

        /* Precess Moon position to J2000. */
        slaPrecl ( slaEpj ( djdm ), 2000.0, rmatp );
        slaDmxv ( rmatp, pv, pvm );

        /* Sun to Earth-Moon Barycentre (mean J2000). */
        slaPlanet ( djdm, 3, pve, &j );

        /* Correct from EMB to Earth. */
        for ( i = 0; i < 3; i++ ) {
            pve[i] -= 0.012150581 * pvm[i];
        }

        /* Sun to Venus. */
        slaPlanet ( djdm, 2, pv, &j );

        /* Earth to Venus. */
        for ( i = 0; i < 6; i++ ) {
            pv[i] -= pve[i];
        }

        /* Light time to Venus (sec). */
        dx = pv[0] - pve[0];
        dy = pv[1] - pve[1];
        dz = pv[2] - pve[2];
        tl = 499.004782 * sqrt ( dx*dx + dy*dy + dz*dz );

        /* Extrapolate backwards in time by that much. */
        for ( i = 0; i < 3; i++ ) {
            pv[i] -= tl * pv[i+3];
        }

        /* To RA,Dec. */
        slaDcc2s ( pv, &rv, &dv );

```

```

/* Same for Jupiter. */
slaPlanet ( djdm, 5, pv, &j );
for ( i = 0; i < 6; i++ ) {
    pv[i] -= pve[i];
}
dx = pv[0] - pve[0];
dy = pv[1] - pve[1];
dz = pv[2] - pve[2];
t1 = 499.004782 * sqrt ( dx*dx +dy*dy + dz*dz );
for ( i = 0; i < 3; i++ ) {
    pv[i] -= t1 * pv[i+3];
}
slaDcc2s ( pv, &rj, &dj );

/* Separation (arcsec) */
sep = slaDsep ( rv, dv, rj, dj );

/* Keep if smallest so far. */
if ( sep < sepmin ) {
    ihmin = ihour;
    immin = imin;
    sepmin = sep;
}
}

/* Report. */
printf ( "%2.2d:%2.2d %5.1f\n", ihmin, immin, 206264.8062 * sepmin );

return 0;
}

```

The output produced (the Ephemeris Time on the day in question, and the closest approach in arcseconds) is as follows:

```
21:16 33.3
```

For comparison, accurate JPL predictions give a separation 8" less than the above estimate, occurring 30^m earlier (see *Sky and Telescope*, April 1987, p 357).

The following program demonstrates `slaRdplan`.

```

#include <stdio.h>
#include <ctype.h>

```

```

#include <string.h>
#include <slalib.h>

#define R2AS 206264.80625          /* radians to arcsec */
#define D15B2P 2.3873241463784300365 /* 15/2pi          */

int main ( )

/* For a given date, time and geographical location, output */
/* a table of planetary positions and diameters.           */

{
    static char *pnames[] = { "Sun", "Mercury", "Venus", "Moon",
                              "Mars", "Jupiter", "Saturn",
                              "Uranus", "Neptune" };

    char b[100], s;
    int n, i, np, iy, j, im, id, m[4];
    double fd, djm, elong, phi, ra, dec, diam;

    /* Loop until "end" typed. */
    n = strlen ( b );
    for ( i = 0; i < n; i++ ) b[i] = (char) toupper ( b[i] );
    while ( strcmp ( b, "END" ) ) {

        /* Get date, time and observer's location */
        printf ( "Date? (Y,M,D, Gregorian)\n" );
        gets ( b );
        n = strlen ( b );
        for ( i = 0; i < n; i++ ) b[i] = (char) toupper ( b[i] );
        if ( strcmp ( b, "END" ) ) {
            i = 1;
            slaInt2in ( b, &i, &iy, &j );
            slaInt2in ( b, &i, &im, &j );
            slaInt2in ( b, &i, &id, &j );
            printf ( "Time? (H,M,S, dynamical)\n" );
            gets ( b );
            i = 1;
            slaDafin ( b, &i, &fd, &j );
            fd *= D15B2P;
            slaCldj ( iy, im, id, &djm, &j );
            djm += fd;
            printf ( "Longitude? (d,',\", east +ve)\n" );
            gets ( b );
            i = 1;
            slaDafin ( b, &i, &elong, &j );
            printf ( "Latitude? (d,',\", geodetic)\n" );
            gets ( b );

```



```

        i = 1;
        slaDafin ( b, &i, &phi, &j );
        printf ( "\n" );

    /* Loop planet by planet. */
    for ( np = 0; np <= 8; np++ ) {

        /* Get RA,Dec and diameter. */
        slaRdplan ( djm, np, elong, phi, &ra, &dec, &diam );

        /* One line of report. */
        slaDr2tf ( 2, ra, &s, m );
        printf ( "%-9s %2.2d %2.2d %2.2d.%2.2d",
                    pnames[np], m[0], m[1], m[2], m[3] );
        slaDr2af ( 1, dec, &s, m );
        printf ( "   %c%2.2d %2.2d %2.2d.%1d %7.1f\n",
                    s, m[0], m[1], m[2], m[3], R2AS*diam );

        /* Next planet. */
    }
    printf ( "\n" );
}

/* Next case. */
}

return 0;
}

```

Entering the following data (for 1927 June 29 at 5^h 25^m ET and the position of Preston, UK):

```

1927 6 29
5 25
-2 42
53 46

```

produces the following report:

Sun	06 28 14.03	+23 17 17.3	1887.8
Mercury	08 08 58.60	+19 20 57.1	9.3
Venus	09 38 53.61	+15 35 32.8	22.8
Moon	06 28 15.95	+23 17 21.3	1902.3
Mars	09 06 49.34	+17 52 26.6	4.0
Jupiter	00 11 12.08	-00 10 57.5	41.1
Saturn	16 01 43.35	-18 36 55.9	18.2
Uranus	00 13 33.54	+00 39 36.1	3.5
Neptune	09 49 35.76	+13 38 40.8	2.2

Inspection of the Sun and Moon data reveals that a total solar eclipse is in progress.

SLALIB also provides for the case where orbital elements (with respect to the J2000 equinox and ecliptic) are available. This allows predictions to be made for minor-planets and (if you ignore non-gravitational effects) comets. Furthermore, if major-planet elements for an epoch close to the date in question are available, more accurate predictions can be made than are offered by `slaRdplan` and `slaPlanet`.

The SLALIB planetary-prediction functions that work with orbital elements are `slaPlante` (the orbital-elements equivalent of `slaRdplan`), which predicts the topocentric $[\alpha, \delta]$, and `slaPlanel` (the orbital-elements equivalent of `slaPlanet`), which predicts the heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ with respect to the J2000 equinox and equator. In addition, the function `slaPv2el` does the inverse of `slaPlanel`, transforming $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ into *osculating elements*.

Osculating elements describe the unperturbed 2-body orbit. Depending on accuracy requirements, this unperturbed orbit is an adequate approximation to the actual orbit for a few weeks either side of the specified epoch, outside which perturbations due to the other bodies of the Solar System lead to increasing errors. Given a minor planet's osculating elements for a particular date, predictions for a date only 100 days earlier or later are likely to be in error by several arcseconds. These errors can be reduced if new elements are generated which take account of the perturbations of the major planets, and this is what the function `slaPertel` does. Once `slaPertel` has been called, to provide osculating elements close to the required date, the elements can be passed to `slaPlanel` or `slaPlante` in the normal way. Predictions of arcsecond accuracy over a span of a decade or more are available using this technique.

Three different combinations of orbital elements are provided for, matching the usual conventions for major planets, minor planets and comets respectively. The choice is made through the argument `jform`:

jform=1	jform=2	jform=3
t_0	t_0	T
i	i	i
Ω	Ω	Ω
ϖ	ω	ω
a	a	q
e	e	e
L	M	
n		

The symbols have the following meanings:

t_0	epoch of osculation
T	epoch of perihelion passage
i	inclination of the orbit
Ω	longitude of the ascending node
ϖ	longitude of perihelion ($\varpi = \Omega + \omega$)
ω	argument of perihelion
a	semi-major axis of the orbital ellipse

q	perihelion distance
e	orbital eccentricity
L	mean longitude ($L = \varpi + M$)
M	mean anomaly
n	mean motion

The mean motion, n , tells `slaPlane1` the mass of the planet. If it is not available, it should be calculated from $n^2 a^3 = k^2(1 + m)$, where $k = 0.01720209895$ and m is the mass of the planet ($M_\odot = 1$); a is in AU.

Note that 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 perturbations, the specified position—mean longitude, mean anomaly, or perihelion—is reached.
- The epoch of osculation: the moment in time at which the given elements precisely specify the body’s position and velocity.

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, for planets (major and minor) 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, for comets all three types of epoch are involved. How many of the three elements are present in a given SLALIB argument list depends on the function concerned.

Two important sources for orbital elements are the *Horizons* service, operated by the Jet Propulsion Laboratory, Pasadena, and the Minor Planet Center, operated by the Center for Astrophysics, Harvard. The JPL elements (heliocentric, J2000 ecliptic and equinox) and MPC elements correspond to SLALIB arguments as shown in the following table, where “(rad)” means conversion from degrees to radians, and “(MJD)” means “subtract 2400000.5”:

<i>SLALIB</i> argument	<i>JPL</i>			<i>MPC</i>	
	<i>major planet</i>	<i>minor planet</i>	<i>comet</i>	<i>minor planet</i>	<i>comet</i>
jform	1	2	3	2	3
epoch	JDCT (MJD)	JDCT (MJD)	Tp (MJD)	Epoch (MJD)	T (MJD)
orbinc	IN (rad)	IN (rad)	IN (rad)	Incl. (rad)	Incl. (rad)
anode	OM (rad)	OM (rad)	OM (rad)	Node (rad)	Node. (rad)
perih	OM+W (rad)	W (rad)	W (rad)	Perih. (rad)	Perih. (rad)
aorq	A	A	QR	a	q
e	EC	EC	EC	e	e
aorl	MA+OM+W (rad)	MA (rad)		M (rad)	
dm	N (rad)				
<i>epoch of osculation</i>	JDCT (MJD)	JDCT (MJD)	JDCT (MJD)	Epoch (MJD)	Epoch (MJD)

Conventional elements are not the only way of specifying an orbit. The $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ state vector is an equally valid specification, and the so-called *method of universal variables* allows orbital calculations to be made directly, bypassing angular quantities and avoiding Kepler's Equation. The universal-variables approach has various advantages, including better handling of near-parabolic cases and greater efficiency. SLALIB uses universal variables for its internal calculations and also offers a number of functions which applications can call.

The universal elements are the $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ and its epoch, plus the mass of the body. The SLALIB functions supplement these elements with certain redundant values in order to avoid unnecessary recomputation when the elements are next used.

The functions `slaEl2ue` and `slaUe2el` transform conventional elements into the universal form and *vice versa*. The function `slaPv2ue` takes an $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ and forms the set of universal elements; `slaUe2pv` takes a set of universal elements and predicts the $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ for a specified epoch. The function `slaPertue` provides updated universal elements, taking into account perturbations from the major planets. Starting with universal elements, the function `slaPlantu` (the universal elements equivalent of `slaPlante`) predicts topocentric $[\alpha, \delta]$.

The following C program demonstrates the use of the `slaPertel` and `slaPlante` functions for predicting the astrometric place of a minor planet seen from a specified place at a specified time. The prediction is carried out twice, first with elements that are up to date and then with elements several years old:

```
#include <stdio.h>
#include <slalib.h>
void predict ( double, double, double, int, int, int, double,
               double, double, double, double, double, double );

#define D2R 0.017453292519943 /* degrees to radians */

int main ( )

/* Two predictions of astrometric RA,Dec for a minor      */
/* planet, using both stale and fresh orbital elements. */

{
    int j;
    double el, phi, utc, fday, tt;

    /* UTC date and time: 2006 Apr 1 04:10:23. */
    slaCldj ( 2006, 4, 1, &utc, &j );
    slaDtf2d ( 4, 10, 23.0, &fday, &j );
    utc += fday;

    /* TT. */
```

```

    tt = utc + slaDtt ( utc ) / 86400.0;

/* Site: 50 45 N, 01 42 W. */
    el = -1.7*D2R;
    phi = 50.75*D2R;

/* Orbital elements for a minor planet, from the MPCORBcr.DAT file.

A) Fresh elements, close to the date of observation:

21903   12.7   0.15 K0636  16.72954   72.75790   29.18688   9.73664...
    0.0439154 0.18492020   3.0512642   0 MP0 27572   165   5 1991-2002...
    0.50 M-v 38h MPC           0000 (21903)Wallace           20020220

B) Stale elements, several years old:

21903   12.7   0.15 K0256 118.68597   72.04680   29.21207   9.73712...
    0.0448818 0.18492704   3.0511890   2 MP0 8901   118   4 1991-2001...
    0.51 M-v 38h Williams  0000 (21903) Wallace           20010203

*/

/* Prediction using up-to-date elements. */
    predict ( tt, el, phi, 2, 2006, 3, 6.0, 9.73664*D2R, 29.18688*D2R,
              72.75790*D2R, 3.0512642, 0.0439154, 16.72954*D2R );

/* Prediction using stale elements. */
    predict ( tt, el, phi, 2, 2002, 5, 6.0, 9.73712*D2R, 29.21207*D2R,
              72.04680*D2R, 3.0511890, 0.0448818, 118.68597*D2R );

    return 0;
}

void predict ( double tt, double el, double phi, int jform,
              int iyear, int imonth, double days,
              double orbinc, double anode, double perih,
              double aorq, double e, double am )
/*
** Perform the prediction and report the results.
**
** Given:
**   tt      d   date of observation (TT MJD)
**   el      d   longitude, east +ve (radians)
**   phi     d   latitude (radians)
**   jform   i   2 = minor planet, 3 = comet
**   iyear   i   year           } epoch of elements;
**   imonth  i   month          } for comets, also date
**   days    d   day + fraction } of perihelion passage

```

```

**      orbinc  d   inclination (radians)
**      anode   d   longitude of ascending node (radians)
**      perih   d   argument of perihelion (radians)
**      aorq    d   a, for minor planets, or q, for comets (AU)
**      e       d   eccentricity
**      am      d   mean anomaly
**
**      n.b. Validation has been omitted.  A real application would check
**            all returned status values.
*/

{
    int iday, j, m[4];
    double fday, epoch, ep, orbi, ano, ph, aq, ecc, anom,
           rt, dt, r, ra, da, rm, dm;
    char s;

    /* Transform epoch from calendar format to MJD. */
    iday = (int) days;
    fday = days - (double) iday;
    slaCldj ( iyear, imonth, iday, &epoch, &j );
    epoch += fday;

    /* Update the elements for planetary perturbations. */
    slaPertel ( jform, epoch, tt,
                epoch, orbinc, anode, perih, aorq, e, am,
                &ep, &orbi, &ano, &ph, &aq, &ecc, &anom,
                &j );

    /* Predict topocentric (i.e. no refraction) apparent place. */
    slaPlante ( tt, el, phi, jform, ep, orbi, ano, ph, aq, ecc, anom,
                0.0, &rt, &dt, &r, &j );

    /* To astrometric J2000 (UT=TT, J2000=ICRS, near enough). */
    slaOap ( "r", rt, dt, tt, 0.0, el, phi, 0.0, 0.0, 0.0,
             0.0, 0.0, 0.0, 0.0, 0.0, &ra, &da );
    slaAmp ( ra, da, tt, 2000.0, &rm, &dm );

    /* Report. */
    slaDr2tf ( 2, rm, &s, m );
    printf ( "J2000 astrometric: %2.2i %2.2i %2.2i.%2.2i  ",
            m[0], m[1], m[2], m[3] );

    slaDr2af ( 1, dm, &s, m );
    printf ( "%c%2.2i %2.2i %2.2i.%1.1i\n", s, m[0], m[1], m[2], m[3] );
}

```

The program produces the following output; the first line is the prediction using up to date elements, the second using stale elements:

```
J2000 astrometric: 07 12 20.27 +33 03 13.9
J2000 astrometric: 07 12 20.31 +33 03 13.9
```

The results from the two sets of elements are very similar, showing that in the second case `slaPertel` has successfully compensated for perturbations in the four-year interval between the orbital elements and the date of observation.

If the program is modified so that the call to `slaPertel` is omitted, the results using fresh elements (first line) are little affected, but the results using four-year-old elements (second line) are markedly worse:

```
J2000 astrometric: 07 12 20.16 +33 03 14.2
J2000 astrometric: 07 13 41.72 +33 01 58.8
```

3.22 Radial Velocity and Light-Time Corrections

When publishing high-resolution spectral observations it is necessary to refer them to a specified standard of rest. This involves knowing the component in the direction of the source of the velocity of the observer. SLALIB provides a number of functions for this purpose, allowing observations to be referred to the Earth's centre, the Sun, a Local Standard of Rest (either dynamical or kinematical), the centre of the Galaxy, and the mean motion of the Local Group.

The function `slaRverot` corrects for the diurnal rotation of the observer around the Earth's axis. This is always less than 0.5 km/s.

No specific function is provided to correct a radial velocity from geocentric to heliocentric, but this can easily be done by calling `slaEvp` as follows:

```
double rm, dm, v[3], dvb[3], dpb[3], dvh[3], dph[3], vcorb;
:
/* Star vector, J2000. */
slaDcs2c ( rm, dm, v );

/* Earth/Sun velocity and position, J2000. */
slaEvp ( tdb, 2000.0, dvb, dpb, dvh, dph );

/* Radial velocity correction due to Earth orbit (km/s). */
vcorb = - slaDvdv ( v, dvh ) * 149.597870e6;
:
```

The maximum value of this correction is the Earth’s orbital speed of about 30 km/s. A related function, `slaEcor`, computes the light-time correction with respect to the Sun. It would be used when reducing observations of a rapid variable-star for instance. For pulsar work the `slaEvp` function is not sufficiently accurate for phase predictions, being limited to about 25 ms. The alternative `slaEpv` function will deliver pulse arrival times accurate to 50 μ s, but is significantly slower.

To remove the intrinsic ~ 20 km/s motion of the Sun relative to other stars in the solar neighbourhood, a velocity correction to a *local standard of rest* (LSR) is required. There are opportunities for mistakes here. There are two sorts of LSR, *dynamical* and *kinematical*, and multiple definitions exist for the latter. The dynamical LSR is a point near the Sun which is in a circular orbit around the Galactic centre; the Sun has a “peculiar” motion relative to the dynamical LSR. A kinematical LSR is the mean standard of rest of specified star catalogues or stellar populations, and its precise definition depends on which catalogues or populations were used and how the analysis was carried out. The Sun’s motion with respect to a kinematical LSR is called the “standard” solar motion. Radial velocity corrections to the dynamical LSR are produced by the function `slaRvlsrd` and to the adopted kinematical LSR by `slaRvlsrk`. See the individual specifications for these functions for the precise definition of the LSR in each case.

For extragalactic sources, the centre of the Galaxy can be used as a standard of rest. The radial velocity correction from the dynamical LSR to the Galactic centre can be obtained by calling `slaRvgalc`. Its maximum value is 220 km/s.

For very distant sources it is appropriate to work relative to the mean motion of the Local Group. The function for computing the radial velocity correction in this case is `slaRvlg`. Note that in this case the correction is with respect to the dynamical LSR, not the Galactic centre as might be expected. This conforms to the IAU definition, and confers immunity from revisions of the Galactic rotation speed.

3.23 Focal-Plane Astrometry

The relationship between the position of a star image in the focal plane of a telescope and the star’s celestial coordinates is usually described in terms of the *tangent plane* or *gnomonic* projection. This is the projection produced by a pin-hole camera and is a good approximation to the projection geometry of a traditional large f -ratio astrographic refractor. SLALIB includes a group of functions which transform star positions between their observed places on the celestial sphere and their $[x, y]$ coordinates in the tangent plane. The spherical coordinate system does not have to be $[\alpha, \delta]$ but usually is. The so-called *standard coordinates* of a star are the tangent plane $[x, y]$, in radians, with respect to an origin at the tangent point, with the y -axis pointing north and the x -axis pointing east (in the direction of increasing α). The factor relating the standard coordinates to the actual $[x, y]$ coordinates in, say, millimetres is simply the focal length of the telescope.

Given the $[\alpha, \delta]$ of the *plate centre* (the tangent point) and the $[\alpha, \delta]$ of a star within the field, the standard coordinates can be determined by calling `slaS2tp` (single precision) or `slaDs2tp` (double precision). The reverse transformation, where the $[x, y]$ is known and we wish to find the $[\alpha, \delta]$, is carried out by calling `slaTp2s` or `slaDtp2s`. Occasionally we know the both the

$[x, y]$ and the $[\alpha, \delta]$ of a star and need to deduce the $[\alpha, \delta]$ of the tangent point; this can be done by calling `slaTps2c` or `slaDtps2c`. (All of these transformations apply not just to $[\alpha, \delta]$ but to other spherical coordinate systems, of course.) Equivalent (and faster) functions are provided which work directly in $[x, y, z]$ instead of spherical coordinates: `slaV2tp` and `slaDv2tp`, `slaTp2v` and `slaDtp2v`, `slaTpv2c` and `slaDtpv2c`.

Even at the best of times, the tangent plane projection is merely an approximation. Some telescopes and cameras exhibit considerable pincushion or barrel distortion and some have a curved focal surface. For example, neither Schmidt cameras nor (especially) large reflecting telescopes with wide-field corrector lenses are adequately modelled by tangent-plane geometry. In such cases, however, it is still possible to do most of the work using the (mathematically convenient) tangent-plane projection by inserting an extra step which applies or removes the distortion peculiar to the system concerned. A simple $r_1 = r_0(1 + kr_0^2)$ law works well in the majority of cases; r_0 is the radial distance in the tangent plane, r_1 is the radial distance after adding the distortion, and k is a constant which depends on the telescope; θ is unaffected. The function `slaPcd` applies the distortion to an $[x, y]$ and `slaUnpcd` removes it. For $[x, y]$ in radians, k values range from $-1/3$ for the tiny amount of barrel distortion in Schmidt geometry to several hundred for the serious pincushion distortion produced by wide-field correctors in big reflecting telescopes (the AAT prime focus triplet corrector is about $k = +178.6$).

SLALIB includes a group of functions which can be put together to build a simple plate-reduction program. The heart of the group is `slaFitxy`, which fits a linear model to relate two sets of $[x, y]$ coordinates, in the case of a plate reduction the measured positions of the images of a set of reference stars and the standard coordinates derived from their catalogue positions. The model is of the form:

$$\begin{aligned} x_p &= a + bx_m + cy_m \\ y_p &= d + ex_m + fy_m \end{aligned}$$

where the p subscript indicates “predicted” coordinates (the model’s approximation to the ideal “expected” coordinates) and the m subscript indicates “measured coordinates”. The six coefficients a, b, c, d, e, f can optionally be constrained to represent a “solid body rotation” free of any squash or shear distortions. Without this constraint the model can, to some extent, accommodate effects like refraction, allowing mean places to be used directly and avoiding the extra complications of a full mean-apparent-observed transformation for each star. Having obtained the linear model, `slaPxy` can be used to process the set of measured and expected coordinates, giving the predicted coordinates and determining the RMS residuals in x and y . The function `slaXy2xy` transforms one $[x, y]$ into another using the linear model. A model can be inverted by calling `slaInvf`, and decomposed into zero points, scales, x/y nonperpendicularity and orientation by calling `slaDcmpf`.

3.24 Numerical Methods

SLALIB contains a small number of simple, general-purpose numerical-methods functions.

Applications which perform a least-squares fit using a traditional normal-equations methods can accomplish the required matrix-inversion by calling either `slaSmat` (float precision) or `slaDmat`

(double). A generally better way to perform such fits is to use singular value decomposition. SLALIB provides a function to do the decomposition itself, `slaSvd`, and two functions to use the results: `slaSvdsol` generates the solution, and `slaSvdcov` produces the covariance matrix. A simple demonstration of the use of the SLALIB SVD functions is given below. It generates 500 simulated data points and fits them to a model which has four unknown coefficients. The arrays in the example are sized to accept up to 1000 points and 10 unknowns. The model is:

$$y = c_1 + c_2x + c_3\sin x + c_4\cos x$$

The test values for the four coefficients are $c_1 = +50.0$, $c_2 = -2.0$, $c_3 = -10.0$ and $c_4 = +25.0$. Gaussian noise, $\sigma = 2.0$, is added to each “observation”.¹⁴

```
#include <stdio.h>
#include <slalib.h>
double Gresid ( double );
double Random ( double );

int main ( )

/* Sizes of arrays, physical and logical */
#define MP 1000 /* maximum number of samples allowed for */
#define NP 10   /* maximum number of unknowns allowed for */
#define NC NP   /* dimension of covariance matrix          */
#define M 500   /* actual number of samples                */
#define N 4     /* actual number of unknowns                */

/* The unknowns we are going to solve for. */
#define C1 50.0
#define C2 -2.0
#define C3 -10.0
#define C4 25.0

{
    double a[MP][NP], w[NP], v[NP][NP], work[NP], b[MP], x[NP], cvm[NC][NC];
    double val, bf1, bf2, bf3, bf4, sd2, d, var;
    int i, j;

    /* Fill the design matrix. */
    for ( i = 0; i < M; i++ ) {

        /* Dummy independent variable. */
        val = ( (double) i ) / 10.0;

        /* Fill one row of the design matrix with the basis functions. */
```

¹⁴The internal functions `Gresid` and `Random` are useful in their own right. Functions like them are not included in SLALIB because they contain `static` variables and hence are not re-entrant.

```

    a[i][0] = bf1 = 1.0;
    a[i][1] = bf2 = val;
    a[i][2] = bf3 = sin ( val );
    a[i][3] = bf4 = cos ( val );

    /* The observed value, including noise. */
    b[i] = C1*bf1 + C2*bf2 + C3*bf3 + C4*bf4 + Gresid(2.0);
}

/* Factorize the design matrix, solve and generate covariance matrix. */
slaSvd ( M, N, MP, NP, (double*) a, w, (double*) v, work, &j );
slaSvdsol ( M, N, MP, NP, b, (double*) a, w, (double*) v, work, x );
slaSvdcov ( N, NP, NC, w, (double*) v, work, (double*) cvm );

/* Compute the variance. */
sd2 = 0.0;
for ( i = 0; i < M; i++ ) {
    val = ( (double) i ) / 10.0;
    bf1 = 1.0;
    bf2 = val;
    bf3 = sin ( val );
    bf4 = cos ( val );
    d = b[i] - ( x[0]*bf1 + x[1]*bf2 + x[2]*bf3 + x[3]*bf4 );
    sd2 += d*d;
}
var = sd2 / ( (double) M );

/* Report the RMS and the solution. */
printf ( "RMS = %5.2f\n\n", sqrt ( var ) );
for ( i = 0; i < N; i++ ) {
    printf ( "C%d = %+7.3f +/- %6.3f\n", i+1, x[i], sqrt(var*cvm[i][i]) );
}

return 0;
}

/* Pseudo-random normal deviate by Box-Muller method. */
double Gresid ( double s )
{
    static double seed = 1.0;
    static int jftf = 1;
    static double x, y;
    double w, rsq;

    if ( jftf ) {
        do {
            x = 2.0*Random(seed) - 1.0;
            y = 2.0*Random(seed) - 1.0;
            w = x*x + y*y;
            rsq = 1.0 - w;
            if ( rsq < 0.0 )
                continue;
            if ( rsq > 0.9 )
                break;
            seed = 1.0 - w;
        } while ( 1 );
        x = x*sqrt(rsq);
        y = y*sqrt(rsq);
    }
    return s*x;
}

```

```

        rsq = x*x + y*y;
    } while ( rsq >= 1.0 );
    w = ( rsq > 1e-30 ) ? sqrt(-2.0*log(rsq)/rsq) : 0.0;
    x *= w;
    y *= w;
    jftf = 0;
} else {
    x = y;
    jftf = 1;
}
return x * s;
}

/* Generate random number between 0.0 and 1.0. */
#include <stdlib.h>
#include <limits.h>
#include <slamac.h>
double Random ( double seed )
{
    static int jftf = 1;

    if ( jftf )
    {
        jftf = 0;
        srand ( gmin(UINT_MAX, (unsigned int) gmax(1.0,fabs(seed))) );
    }
    return (double) rand() / RAND_MAX;
}

```

The program produces output like (but not necessarily identical to, because of the platform-dependent vagaries of the random number generator) the following:

```

RMS = 2.01

C1 = +49.759 +/- 0.180
C2 = -1.992 +/- 0.006
C3 = -9.962 +/- 0.127
C4 = +25.186 +/- 0.127

```

In this above example, essentially identical results would be obtained if the more commonplace normal-equations method had been used, and the relatively large 1000×10 array would have been avoided. However, the SVD method comes into its own when the opportunity is taken to edit the W-matrix (the so-called “singular values”) in order to control possible ill-conditioning. The procedure involves replacing with zeroes any W-elements smaller than a nominated value, for example 0.001 times the largest W-element. Small W-elements indicate ill-conditioning, which in the case of the normal-equations method would produce spurious large coefficient values and

possible arithmetic overflows. Using SVD, the effect on the solution of setting suspiciously small W -elements to zero is to restrain the offending coefficients from moving very far. The fact that action was taken can be reported to show the program user that something is amiss. Furthermore, if element $W(j)$ was set to zero, the row numbers of the two biggest elements in the j th column of the V -matrix identify the pair of solution coefficients that are dependent.

A more detailed description of SVD and its use in least-squares problems would be out of place here, and the reader is urged to refer to the relevant sections of the book *Numerical Recipes* (Press *et al.*, Cambridge University Press, 1987).

The functions `slaCombn` and `slaPermut` are useful for problems which involve combinations (different subsets) and permutations (different orders). Both return the next in a sequence of results, cycling through all the possible results as the function is called repeatedly.

4 SUBPROGRAM SPECIFICATIONS

slaAddet	add E-terms of aberration	slaAddet
-----------------	---------------------------	-----------------

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

CALL : `slaAddet(rm, dm, eq, &rc, &dc);`

GIVEN :

<code>rm, dm</code>	<code>double</code>	[α, δ] without E-terms (radians)
<code>eq</code>	<code>double</code>	Besselian epoch of mean equator and equinox

RETURNED :

<code>rc, dc</code>	<code>double*</code>	[α, δ] with E-terms included (radians)
---------------------	----------------------	--

NOTE : Most star positions from pre-1984 optical catalogues (or obtained by astrometry with respect to such stars) have the E-terms built-in. 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 [α, δ] should be adjusted using this function.

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

slaAfin	sexagesimal character string to angle	slaAfin
----------------	---------------------------------------	----------------

ACTION : Decode a free-format sexagesimal string (degrees, arcminutes, arcseconds) into a single precision floating point number (radians).

CALL : `slaAfin(string, &nstrt, &reslt, &j);`

GIVEN :

<code>string</code>	<code>char*</code>	string containing deg, arcmin, arcsec fields
<code>nstrt</code>	<code>int*</code>	index to start of decode (beginning of string = 1)

RETURNED :

nstrt	int*	advanced past the decoded angle
reslt	float*	angle in radians
j	int*	status:
		0 = OK
		+1 = default, reslt unchanged (Note 2)
		-1 = bad degrees (Note 3)
		-2 = bad arcminutes (Note 3)
		-3 = bad arcseconds (Note 3)

EXAMPLE :

<i>argument</i>	<i>before</i>	<i>after</i>
string	'-57_17_44.806__12_34_56.7'	unchanged
nstrt	1	16 (<i>i.e.</i> pointing to 12...)
reslt	-	-1.00000
j	-	0

A further call to **slaAfin**, without adjustment of **nstrt**, will decode the second angle, 12° 34' 56".7.

NOTES :

1. 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 **slaDfltIn** function and is free-format.
2. 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 **reslt** is unchanged. In all other cases **reslt** is changed.
3. 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 **reslt** is computed and returned.
4. Further fields after the three expected ones are not treated as an error. The index **nstrt** is left in the correct state for further decoding with the present function or with **slaDfltIn** *etc.* See the example, above.

5. If **string** contains hours, minutes, seconds instead of degrees *etc.*, or if the required units are turns (or days) instead of radians, the result **reslt** should be multiplied as follows:

<i>for string</i>	<i>to obtain</i>	<i>multiply reslt by</i>
o / "	radians	1.0
o / "	turns	$1/2\pi = 0.1591549430918953358$
h m s	radians	15.0
h m s	days	$15/2\pi = 2.3873241463784300365$

slaAirmas

air mass

slaAirmas

ACTION : Air mass at given zenith distance.

CALL : `d = slaAirmas(zd);`

GIVEN :

zd	double	observed zenith distance (radians)
-----------	---------------	------------------------------------

RETURNED :

double	air mass (1 at zenith)
---------------	------------------------

NOTES :

1. The *observed* zenith distance referred to above means “as affected by refraction”.
2. The function 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.
3. The sign of the ZD is ignored.
4. At zenith distances greater than about $\zeta = 87^\circ$ the air mass is held constant to avoid arithmetic overflows.

REFERENCES :

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

slaAltaz velocities *etc.* for altazimuth mount **slaAltaz**

ACTION : Positions, velocities and accelerations for an altazimuth telescope mount that is tracking a star.

CALL : slaAltaz(ha, dec, phi,
 &az, &azd, &azdd, &el, &eld, &eldd, &pa, &pad, &padd);

GIVEN :

ha	double	hour angle
dec	double	declination
phi	double	observatory latitude

RETURNED :

az	double*	azimuth
azd	double*	azimuth velocity
azdd	double*	azimuth acceleration
el	double*	elevation
eld	double*	elevation velocity
eldd	double*	elevation acceleration
pa	double*	parallactic angle
pad	double*	parallactic angle velocity
padd	double*	parallactic angle acceleration

NOTES :

1. Natural units are used throughout. The arguments **ha**, **dec**, **phi**, **az**, **el** and **zd** are in radians. The velocities and accelerations assume constant declination and constant rate of change of hour angle (as for tracking a star); the units of **azd**, **eld** and **pad** are radians per radian of HA, while the units of **azdd**, **eldd** and **padd** are radians per radian of HA squared. To convert into practical degree- and second-based units:

angles	$\times 360/2\pi$	\rightarrow	degrees
velocities	$\times (2\pi/86400) \times (360/2\pi)$	\rightarrow	degree/sec
accelerations	$\times (2\pi/86400)^2 \times (360/2\pi)$	\rightarrow	degree/sec/sec

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

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

2. Azimuth is returned in the range $[0, 2\pi]$; north is zero, and east is $+\pi/2$. Elevation and parallactic angle are returned in the range $\pm\pi$. Position angle is positive for a star west of the meridian and is the angle NP–star–zenith.
3. The latitude is geodetic as opposed to geocentric. The hour angle and declination are topocentric. Refraction and deficiencies in the telescope mounting are ignored. The purpose of the function is to give the general form of the quantities. The details of a real telescope could profoundly change the results, especially close to the zenith.
4. No range checking of arguments is carried out.
5. In applications which involve many such calculations, rather than calling the present function 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.

slaAmp	apparent to mean	slaAmp
---------------	------------------	---------------

ACTION : Convert star $[\alpha, \delta]$ from geocentric apparent to mean place.

CALL : `slaAmp(ra, da, date, eq, &rm, &dm);`

GIVEN :

<code>ra, da</code>	<code>double</code>	apparent $[\alpha, \delta]$ (radians)
<code>date</code>	<code>double</code>	TDB for apparent place (JD–2400000.5)
<code>eq</code>	<code>double</code>	equinox: Julian epoch of mean place

RETURNED :

<code>rm, dm</code>	<code>double*</code>	mean $[\alpha, \delta]$ (radians)
---------------------	----------------------	-----------------------------------

NOTES :

1. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
2. Iterative techniques are used for the aberration and light deflection corrections so that the functions **slaAmp** (or **slaAmpqk**) and **slaMap** (or **slaMapqk**) are accurate inverses; even at the edge of the Sun's disc the discrepancy is only about 1 nanoarcsecond.

3. Where multiple apparent places are to be converted to mean places, for a fixed date and equinox, it is more efficient to use the `slaMappa` function to compute the required parameters once, followed by one call to `slaAmpqk` per star.
4. For `eq=2000.0`, the agreement with ICRS is sub-mas, limited by the precession-nutation model (IAU 1976 precession, Shirai & Fukushima 2001 forced nutation and precession corrections).
5. The accuracy is further limited by the function `slaEvp`, called by `slaMappa`, which computes the Earth position and velocity using the methods of Stumpff. The maximum error is about 0.3 milliarcsecond.

REFERENCES :

1. 1984 *Astronomical Almanac*, pp B39-B41.
2. Lederle & Schwan 1984, *Astr.Astrophys.* **134**, 1-6.
3. Shirai, T. & Fukushima, T. 2001, *Astron.J.*, **121**, 3270-3283.

slaAmpqk	quick apparent to mean	slaAmpqk
-----------------	------------------------	-----------------

ACTION : Convert star $[\alpha, \delta]$ from geocentric apparent to mean place. 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 `slaMappa` function.

CALL : `slaAmpqk(ra, da, amprms, &rm, &dm);`

GIVEN :

ra,da	double	apparent $[\alpha, \delta]$ (radians)
amprms	double[21]	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]	◦ (gravitational radius of Sun) $\times 2/(\text{Sun-Earth distance})$
	[8-10]	◦ v : barycentric Earth velocity in units of c
	[11]	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	[12-20]	◦ precession-nutation 3×3 matrix

RETURNED :

rm,dm	double*	mean $[\alpha, \delta]$ (radians)
-------	---------	-----------------------------------

NOTE : Iterative techniques are used for the aberration and light deflection corrections so that the functions `slaAmp` (or `slaAmpqk`) and `slaMap` (or `slaMapqk`) are accurate inverses; even at the edge of the Sun's disc the discrepancy is only about 1 nanoarcsecond.

REFERENCES :

1. 1984 *Astronomical Almanac*, pp B39-B41.
2. Lederle & Schwan, 1984. *Astr.Astrophys.* **134**, 1-6.

slaAop	apparent to observed	slaAop
---------------	----------------------	---------------

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

CALL : `slaAop(rap, dap, utc, dut, elongm, phim, hm, xp, yp,`
`tk, phpa, rh, wlfq, tlr, &aob, &zob, &hob, &dob, &rob);`

GIVEN :

<code>rap,dap</code>	<code>double</code>	geocentric apparent $[\alpha, \delta]$ (radians)
<code>utc</code>	<code>double</code>	UTC date/time (MJD, JD-2400000.5)
<code>dut</code>	<code>double</code>	Δ UT: UT1-UTC (UTC seconds)
<code>elongm</code>	<code>double</code>	observer's mean longitude (radians, east +ve)
<code>phim</code>	<code>double</code>	observer's mean geodetic latitude (radians)
<code>hm</code>	<code>double</code>	observer's height above sea level (metres)
<code>xp,yp</code>	<code>double</code>	polar motion $[x, y]$ coordinates (radians)
<code>tk</code>	<code>double</code>	local ambient temperature (K; std=273.15)
<code>phpa</code>	<code>double</code>	local atmospheric pressure (hPa; std=1013.25)
<code>rh</code>	<code>double</code>	local relative humidity (in the range 0.0-1.0)
<code>wlfq</code>	<code>double</code>	wavelength (μ m) or minus frequency (GHz)
<code>tlr</code>	<code>double</code>	tropospheric lapse rate (K per metre, <i>e.g.</i> 0.0065)

RETURNED :

<code>aob</code>	<code>double*</code>	observed azimuth (radians: N=0, E=90°)
<code>zob</code>	<code>double*</code>	observed zenith distance (radians)
<code>hob</code>	<code>double*</code>	observed hour angle (radians)
<code>dob</code>	<code>double*</code>	observed δ (radians)
<code>rob</code>	<code>double*</code>	observed α (radians)

NOTES :

1. This function returns zenith distance rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.
2. The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about $0''.1$ for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90° , the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the `slaRefr` function (*q.v.*) returns a fixed value of the refraction.
The complementary functions `slaAop` (or `slaAopqk`) and `slaOap` (or `slaOapqk`) are self-consistent to better than 1 microarcsecond all over the celestial sphere.
3. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
4. *Apparent* $[\alpha, \delta]$ means the geocentric apparent right ascension and declination, which is obtained from a catalogue mean place by allowing for space motion, parallax, the Sun's gravitational lens effect, annual aberration, and precession-nutation. For star positions in the FK5 system (*i.e.* J2000), these effects can be applied by means of the `slaMap` *etc.* functions. Starting from other mean place systems, additional transformations will be needed; for example, FK4 (*i.e.* B1950) mean places would first have to be converted to FK5, which can be done with the `slaFk425` *etc.* functions.
5. *Observed* $[Az, El]$ means the position that would be seen by a perfect theodolite located at the observer. This is obtained from the geocentric apparent $[\alpha, \delta]$ by allowing for Earth orientation and diurnal aberration, rotating from equator to horizon coordinates, and then adjusting for refraction. The $[h, \delta]$ is obtained by rotating back into equatorial coordinates, using the geodetic latitude corrected for polar motion, and is the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). Finally, the α is obtained by subtracting the h from the local apparent ST.
6. To predict the required setting of a real telescope, the observed place produced by this function would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. This function takes time to execute, due mainly to the rigorous integration used to evaluate the refraction. For processing multiple stars for one location and time, call `slaAoppa` once followed by one call per star to `slaAopqk`. Where a range of times within a limited period of a few hours is involved, and the highest precision is not required, call `slaAoppa` once, followed by a call to `slaAoppat` each time the time changes, followed by one call per star to `slaAopqk`.
8. The `utc` argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the ΔUT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap

ΔUT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the ΔUT changes by one second to its post-leap new value.

9. The ΔUT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep ΔUT within ± 0.9 .
10. **Important : take care with the longitude sign convention.** The longitude required by the present function is **east-positive**, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the `slaObs` function are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present function.
11. The polar motion coordinates x_p, y_p can be obtained from IERS bulletins. The maximum amplitude is about $0''.3$. If x_p, y_p values are unavailable, use $x_p = y_p = 0.0$. The x_p, y_p coordinates are reckoned as follows:
 - positive x_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 90° .
 - positive y_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 270° .

Internally, the polar motion is stored in a form rotated onto the local meridian.

12. The height above sea level of the observing station, `hm`, can be obtained from the *Astronomical Almanac* (Section J in the 1988 edition), or via the function `slaObs`. If `p`, the pressure in hPa (mb), is available, an adequate estimate of `hm` can be obtained from the following expression:

$$\text{hm} = -29.3 * \text{tsl} * \log(p/1013.25);$$

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

$$p = 1013.25 * \exp(-\text{hm}/(29.3 * \text{tsl}));$$

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

13. The azimuths *etc.* used by the present function are with respect to the celestial pole. The correction needed to refer instead to terrestrial north can be computed using `slaPolmo`.
14. The argument `wlfq` specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at $100 \mu\text{m}$ (about 3000 GHz).

slaAoppa	apparent-to-observed parameters	slaAoppa
-----------------	---------------------------------	-----------------

ACTION : Pre-compute the set of apparent to observed place parameters required by the “quick” functions `slaAopqk` and `sla0apqk`.

CALL : `slaAoppa(utc, dut, elongm, phim, hm, xp, yp,
tk, phpa, rh, wlfq, tlr, aoprms);`

GIVEN :

<code>utc</code>	double	UTC date/time (MJD, JD–2400000.5)
<code>dut</code>	double	Δ UT: UT1–UTC (UTC seconds)
<code>elongm</code>	double	observer’s mean longitude (radians, east +ve)
<code>phim</code>	double	observer’s mean geodetic latitude (radians)
<code>hm</code>	double	observer’s height above sea level (metres)
<code>xp,yp</code>	double	polar motion $[x, y]$ coordinates (radians)
<code>tk</code>	double	local ambient temperature (K; std=273.15)
<code>phpa</code>	double	local atmospheric pressure (hPa; std=1013.25)
<code>rh</code>	double	local relative humidity (in the range 0.0–1.0)
<code>wlfq</code>	double	wavelength (μ m) or minus frequency (GHz)
<code>tlr</code>	double	tropospheric lapse rate (K per metre, <i>e.g.</i> 0.0065)

RETURNED :

<code>aoprms</code>	double[14]	star-independent apparent-to-observed parameters:
	[0]	◦ geodetic latitude (radians)
	[1,2]	◦ sine and cosine of geodetic latitude
	[3]	◦ magnitude of diurnal aberration vector
	[4]	◦ height above sea level (metres)
	[5]	◦ temperature (K; std=273.15)
	[6]	◦ pressure (mb = hPa)
	[7]	◦ relative humidity (in the range 0.0–1.0)
	[8]	◦ wavelength (μ m) or minus frequency (GHz)
	[9]	◦ tropospheric lapse rate (K per metre)
	[10,11]	◦ refraction constants A and B (radians)
	[12]	◦ longitude + eqn of equinoxes + “sidereal Δ UT” (radians)
	[13]	◦ local apparent sidereal time (radians)

NOTES :

1. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.

2. The `utc` argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the ΔUT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap ΔUT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the ΔUT changes by one second to its post-leap new value.
3. The ΔUT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep ΔUT within $\pm 0^{\text{s}}.9$. The “sidereal ΔUT ” which forms part of `aoprms` [12] is the same quantity, but converted from solar to sidereal seconds and expressed in radians.
4. **Important : take care with the longitude sign convention.** The longitude required by the present function is **east-positive**, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the `slaObs` function are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present function.
5. The polar coordinates `xp,yp` can be obtained from IERS circulars and equivalent publications. The maximum amplitude is about $0^{\circ}.3$. If `xp,yp` values are unavailable, use `xp = yp = 0.0`. See page B60 of the 1988 *Astronomical Almanac* for a definition of the two angles.
6. The height above sea level of the observing station, `hm`, can be obtained from the *Astronomical Almanac* (Section J in the 1988 edition), or via the function `slaObs`. If `p`, the pressure in hPa (mb) (the same as hPa), is available, an adequate estimate of `hm` can be obtained from the following expression:

$$\text{hm} = -29.3 * \text{tsl} * \log(\text{p}/1013.25);$$

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

$$\text{p} = 1013.25 * \exp(-\text{hm}/(29.3 * \text{tsl}));$$

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

7. Repeated, computationally-expensive, calls to `slaAoppa` for times that are very close together can be avoided by calling `slaAoppa` just once and then using `slaAoppat` for the subsequent times. Fresh calls to `slaAoppa` will be needed only when changes in the equation of the equinoxes or the polar motion have grown to unacceptable levels or when anything affecting the refraction has changed.
8. The argument `wlfq` specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at $100\ \mu\text{m}$ (about 3000 GHz).

slaAoppat update apparent-to-observed parameters **slaAoppat**

ACTION : Recompute the sidereal time in the apparent to observed place star-independent parameter block.

CALL : `slaAoppat(utc, aoprms);`

GIVEN :

<code>utc</code>	<code>double</code>	UTC date/time (MJD, JD−2400000.5)
<code>aoprms</code>	<code>double[14]</code>	star-independent apparent-to-observed parameters:
	[0–11]	◦ not required
	[12]	◦ longitude + eqn of equinoxes + “sidereal Δ UT” (radians)
	[13]	◦ not required

RETURNED :

<code>aoprms</code>	<code>double[14]</code>	star-independent apparent-to-observed parameters:
	[0–12]	◦ not changed
	[13]	◦ local apparent sidereal time (radians)

NOTE : For more information, see `slaAoppa`.

slaAopqk quick apparent to observed **slaAopqk**

ACTION : Quick apparent to observed place (but see Note 8, below).

CALL : `slaAopqk(rap, dap, aoprms, &aob, &zob, &hob, &dob, &rob);`

GIVEN :

<code>rap,dap</code>	<code>double</code>	geocentric apparent $[\alpha, \delta]$ (radians)
<code>aoprms</code>	<code>double[14]</code>	star-independent apparent-to-observed parameters:
	[0]	◦ geodetic latitude (radians)
	[1,2]	◦ sine and cosine of geodetic latitude
	[3]	◦ magnitude of diurnal aberration vector
	[4]	◦ height above sea level (metres)
	[5]	◦ ambient temperature (K; std=273.15)
	[6]	◦ pressure (mb = hPa)
	[7]	◦ relative humidity (in the range 0.0–1.0)
	[8]	◦ wavelength (μm) or minus frequency (GHz)
	[9]	◦ tropospheric lapse rate (K per metre)
	[10,11]	◦ refraction constants A and B (radians)
	[12]	◦ longitude + eqn of equinoxes + “sidereal ΔUT ” (radians)
	[13]	◦ local apparent sidereal time (radians)

RETURNED :

<code>aob</code>	<code>double*</code>	observed azimuth (radians: N=0, E=90°)
<code>zob</code>	<code>double*</code>	observed zenith distance (radians)
<code>hob</code>	<code>double*</code>	observed hour angle (radians)
<code>dob</code>	<code>double*</code>	observed declination (radians)
<code>rob</code>	<code>double*</code>	observed right ascension (radians)

NOTES :

1. This function returns zenith distance rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.
2. The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about 0".1 for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90° , the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the `slaRefro` function (*q.v.*) returns a fixed value of the refraction.
The complementary functions `slaAop` (or `slaAopqk`) and `slaOap` (or `slaOapqk`) are self-consistent to better than 1 microarcsecond all over the celestial sphere.
3. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
4. *Apparent* $[\alpha, \delta]$ means the geocentric apparent right ascension and declination, which is obtained from a catalogue mean place by allowing for space motion, parallax, the Sun's gravitational lens effect, annual aberration and precession-nutation. For star positions in the FK5 system (*i.e.* J2000), these effects can be applied by means of the `slaMap` *etc.* functions. Starting from other mean place systems, additional transformations will be needed; for example, FK4 (*i.e.* B1950) mean places would first have to be converted to FK5, which can be done with the `slaFk425` *etc.* functions.

5. *Observed* $[Az, El]$ means the position that would be seen by a perfect theodolite located at the observer. This is obtained from the geocentric apparent $[\alpha, \delta]$ by allowing for Earth orientation and diurnal aberration, rotating from equator to horizon coordinates, and then adjusting for refraction. The $[h, \delta]$ is obtained by rotating back into equatorial coordinates, using the geodetic latitude corrected for polar motion, and is the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). Finally, the α is obtained by subtracting the h from the local apparent ST.
6. To predict the required setting of a real telescope, the observed place produced by this function would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. The star-independent apparent-to-observed-place parameters in `aoprms` may be computed by means of the `slaAoppa` function. If nothing has changed significantly except the time, the `slaAoppat` function may be used to perform the requisite partial re-computation of `aoprms`.
8. At zenith distances beyond about 76° , the need for special care with the corrections for refraction causes a marked increase in execution time. Moreover, the effect gets worse with increasing zenith distance. Adroit programming in the calling application may allow the problem to be reduced. Prepare an alternative `aoprms` array, computed for zero air-pressure; this will disable the refraction corrections and cause rapid execution. Using this `aoprms` array, a preliminary call to the present function will, depending on the application, produce a rough position which may be enough to establish whether the full, slow calculation (using the real `aoprms` array) is worthwhile. For example, there would be no need for the full calculation if the preliminary call had already established that the source was well below the elevation limits for a particular telescope.
9. The azimuths *etc.* used by the present function are with respect to the celestial pole. The correction needed to refer instead to terrestrial north can be computed using `slaPolmo`.

slaAtmdsp	atmospheric dispersion	slaAtmdsp
------------------	------------------------	------------------

ACTION : Apply atmospheric-dispersion adjustments to optical/IR refraction coefficients.

CALL : `slaAtmdsp(tk, phpa, rh, w11, a1, b1, w12, &a2, &b2);`

GIVEN :

tk	double	ambient temperature at the observer (K)
phpa	double	pressure at the observer (hPa = mb)
rh	double	relative humidity at the observer (range 0–1)
wl1	double	base wavelength (μm)
a1	double	refraction coefficient A for wavelength wl1 (radians)
b1	double	refraction coefficient B for wavelength wl1 (radians)
wl2	double	wavelength for which adjusted A,B required (μm)

RETURNED :

a2	double*	refraction coefficient A for wavelength wl2 (radians)
b2	double*	refraction coefficient B for wavelength wl2 (radians)

NOTES :

1. To use this function, first call **slaRefco** specifying **wl1** as the wavelength. This yields refraction coefficients **b1**, **b1**, correct for that wavelength. Subsequently, calls to **slaAtmdsp** specifying different wavelengths will produce new, slightly adjusted refraction coefficients **a2**, **b2**, which apply to the specified wavelength.
2. Most of the atmospheric dispersion happens between $0.7\mu\text{m}$ and the UV atmospheric cutoff, and the effect increases strongly towards the UV end. For this reason a blue reference wavelength is recommended, for example $0.4\mu\text{m}$.
3. The accuracy, for this set of conditions:

height above sea level	2000 m
latitude	29°
pressure	793 hPa
temperature	290°K
humidity	0.5 (50%)
lapse rate	$0.0065^\circ\text{m}^{-1}$
reference wavelength	$0.4\mu\text{m}$
star elevation	15°

is about 2.5 mas RMS between $0.3\mu\text{m}$ and $1.0\mu\text{m}$, and stays within 4 mas for the whole range longward of $0.3\mu\text{m}$ (compared with a total dispersion from 0.3 to $20\mu\text{m}$ of about $11''$). These errors are typical for ordinary conditions; in extreme conditions values a few times this size may occur.

4. If either wavelength exceeds $100\mu\text{m}$ (or is negative, used in in certain other functions to indicate frequency), the radio case is assumed and the returned refraction coefficients are the same as the given ones. Note that radio refraction coefficients cannot be turned into optical values using this function, nor vice versa.

5. The algorithm consists of calculation of the refractivity of the air at the observer for the two wavelengths, using the methods of the **slaRefro** function, and then scaling of the two refraction coefficients according to classical refraction theory. This amounts to scaling the A coefficient in proportion to $(\mu - 1)$ and the B coefficient almost in the same ratio (see R.M.Green, *Spherical Astronomy*, Cambridge University Press, 1985).

slaAv2m	rotation matrix from axial vector	slaAv2m
----------------	-----------------------------------	----------------

ACTION : Form the rotation matrix corresponding to a given axial vector (single precision).

CALL : `slaAv2m(axvec, rmat);`

GIVEN :

<code>axvec</code>	<code>float[3]</code>	axial vector (radians)
--------------------	-----------------------	------------------------

RETURNED :

<code>rmat</code>	<code>float[3][3]</code>	rotation matrix
-------------------	--------------------------	-----------------

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis, called the Euler axis. The *axial vector* supplied to this function has the same direction as the Euler axis, and its magnitude is the amount of rotation in radians.
2. If `axvec` is null, the unit matrix is returned.
3. The reference frame rotates clockwise as seen looking along the axial vector from the origin.

slaBear	direction between points on a sphere	slaBear
----------------	--------------------------------------	----------------

ACTION : Returns the bearing (position angle) of one point on a sphere seen from another (single precision).

CALL : `r = slaBear(a1, b1, a2, b2);`

GIVEN :

<code>a1,b1</code>	<code>float</code>	spherical coordinates of one point
<code>a2,b2</code>	<code>float</code>	spherical coordinates of the other point

RETURNED :

float	bearing from first point to second
-------	------------------------------------

NOTES :

1. The spherical coordinates are $[\alpha, \delta]$, $[\lambda, \beta]$ *etc.*, in radians.
2. The result is the bearing (position angle), in radians, of point $[a2, b2]$ as seen from point $[a1, b1]$. It is in the range $\pm\pi$. The sense is such that if $[a2, b2]$ is a small distance due east of $[a1, b1]$ the result is about $+\pi/2$. Zero is returned if the two points are coincident.
3. If either b-coordinate is outside the range $\pm\pi/2$, the result may correspond to “the long way round”.
4. The function `slaPav` performs an equivalent function except that the points are specified in the form of Cartesian unit vectors.

slaC2i	ICRS to CIRS	slaC2i
---------------	--------------	---------------

ACTION : Transform ICRS star data, epoch J2000, to $[\alpha, \delta]$ in the celestial intermediate reference system.

CALL : `slaC2i(rc, dc, pr, pd, px, rv, date, dx, dy, &ri, &di);`

GIVEN :

<code>rc,dc</code>	double	ICRS $[\alpha, \delta]$ at J2000 (radians)
<code>pr,pd</code>	double	proper motions: rate of change of $[\alpha, \delta]$ (radians per year)
<code>px</code>	double	parallax (arcsec)
<code>rv</code>	double	radial velocity (km s^{-1} , +ve if receding)
<code>date</code>	double	TDB for CIRS place (JD–2400000.5)
<code>dx,dy</code>	double	corrections to IAU CIP X, Y (radians)

RETURNED :

<code>ri,di</code>	double*	CIRS $[\alpha, \delta]$ (radians)
--------------------	---------	-----------------------------------

NOTES :

1. Star data for an epoch other than J2000 (for example from the Hipparcos catalogue) will require a preliminary call to `slaPm` before use.
2. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications, UTC is adequate.
3. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$.
4. This function may be wasteful for some applications because it recomputes the Earth position/velocity and the bias-precession-nutation matrix each time, and because it allows for parallax and proper motion. Where multiple transformations are to be carried out for one date, a faster method is to call the `slaC2ipa` function once and then either the `slaC2iqk` function (which includes parallax and proper motion) or `slaC2iqkz` (which assumes zero parallax and proper motion).
5. The accuracy is sub-mas, limited by the precession-nutation information (IAU 2006 model plus the CIP corrections, which are obtained from IERS bulletins).
6. The accuracy is in principle further limited by the function `slaEpv`, called by `slaC2ipa`, which computes the Earth position and velocity. The maximum error arising from this source is less than $5 \mu\text{s}$.

slaC2ipa	ICRS to CIRS parameters	slaC2ipa
-----------------	-------------------------	-----------------

ACTION : For a geocentric observer, compute star-independent parameters in preparation for conversions between ICRS star data CIRS $[\alpha, \delta]$.

The parameters produced by this function are required in the parallax, light deflection, aberration and bias-precession-nutation parts of the ICRS \Longleftrightarrow CIRS transformations.

CALL : `slaC2ipa(date, dx, dy, &ciprms, &eo);`

GIVEN :

<code>date</code>	<code>double</code>	TDB (JD–2400000.5)
<code>dx, dy</code>	<code>double</code>	corrections to IAU CIP X, Y (radians)

RETURNED :

<code>ciprms</code>	<code>CIpars*</code>	star-independent ICRS-to-CIRS parameters:
	<code>pmt</code>	◦ time interval for proper motion (Julian years)
	<code>eb[3]</code>	◦ barycentric position of the Earth (AU)
	<code>ehn[3]</code>	◦ heliocentric direction of the Earth (unit vector)
	<code>gr2e</code>	◦ (gravitational radius of Sun) $\times 2 / (\text{Sun-Earth distance})$
	<code>abv[3]</code>	◦ v : barycentric Earth velocity in units of c
	<code>ab1</code>	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	<code>bpn[3][3]</code>	◦ bias-precession-nutation 3×3 matrix
<code>eo</code>	<code>double*</code>	equation of the origins (ERA–GST)

NOTES :

1. For date, the distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
2. The vectors `ciprms->eb` and `ciprms->ehn` are in the BCRS.
3. The parameters `ciprms` produced by this function are used by `slaC2iqk`, `slaC2iqkz` and `slaI2cqk`. The bias-precession-nutation portion can be suppressed using `slaC2ipad`, allowing transformations to and from GCRS instead of CIRS.
4. The accuracy is sub-mas, limited by the precession-nutation information (IAU 2006 model plus the CIP corrections, which are obtained from IERS bulletins).
5. The accuracy is in principle further limited by the function `slaEpv`, which computes the Earth position and velocity. The maximum error arising from this source is less than $5\ \mu\text{s}$.

slaC2ipad make ICRS to CIRS parameters stop at GCRS **slaC2ipad**

ACTION : In the ICRS to CIRS star-independent parameters, disable bias-precession-nutation. Subsequent calls to `slaC2iqk` will consequently produce GCRS coordinates rather than CIRS.

CALL : `slaC2ipad(&ciprms);`

GIVEN :

<code>ciprms</code>	<code>CIpars*</code>	star-independent ICRS-to-CIRS parameters:
	<code>pmt</code>	◦ time interval for proper motion (Julian years)
	<code>eb[3]</code>	◦ barycentric position of the Earth (AU)
	<code>ehn[3]</code>	◦ heliocentric direction of the Earth (unit vector)
	<code>gr2e</code>	◦ (gravitational radius of Sun)×2/(Sun-Earth distance)
	<code>abv[3]</code>	◦ v : barycentric Earth velocity in units of <i>c</i>
	<code>ab1</code>	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	<code>bpn[3][3]</code>	◦ bias-precession-nutation 3×3 matrix

RETURNED :

<code>ciprms</code>	<code>CIpars*</code>	star-independent ICRS-to-GCRS parameters:
	<code>pmt</code>	◦ unchanged
	<code>eb[3]</code>	◦ unchanged
	<code>ehn[3]</code>	◦ unchanged
	<code>gr2e</code>	◦ unchanged
	<code>abv[3]</code>	◦ unchanged
	<code>ab1</code>	◦ unchanged
	<code>bpn[3][3]</code>	◦ 3×3 identity matrix

slaC2ipas ICRS to CIRS parameters, space observer **slaC2ipas**

ACTION : For an observer not necessarily coincident with the geocenter, compute star-independent parameters in preparation for transformations between ICRS star data and CIRS $[\alpha, \delta]$.

The parameters produced by this function are required in the parallax, light deflection, aberration and bias-precession-nutation parts of the ICRS \longleftrightarrow CIRS transformations.

CALL : `slaC2ipas(date, pv, dx, dy, &ciprms, &eo);`

GIVEN :

<code>date</code>	<code>double</code>	TDB (JD−2400000.5)
<code>pv</code>	<code>double[2][3]</code>	observer's geocentric position and velocity
<code>dx,dy</code>	<code>double</code>	corrections to IAU CIP X, Y (radians)

RETURNED :

<code>ciprms</code>	<code>CIpars*</code>	star-independent ICRS-to-CIRS parameters:
	<code>pmt</code>	◦ time interval for proper motion (Julian years)
	<code>eb[3]</code>	◦ barycentric position of the Earth (AU)
	<code>ehn[3]</code>	◦ heliocentric direction of the Earth (unit vector)
	<code>gr2e</code>	◦ (gravitational radius of Sun) $\times 2/(\text{Sun-Earth distance})$
	<code>abv[3]</code>	◦ \mathbf{v} : barycentric Earth velocity in units of c
	<code>ab1</code>	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	<code>bpn[3][3]</code>	◦ bias-precession-nutation 3×3 matrix
<code>eo</code>	<code>double*</code>	equation of the origins (ERA−GST)

NOTES :

1. For `date`, the distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
2. The vectors `ciprms->eb` and `ciprms->ehn` are in the BCRS.
3. The parameters `ciprms` produced by this function are used by `slaC2iqk`, `slaC2iqkz` and `slaI2cqk`. The bias-precession-nutation portion can be suppressed using `slaC2ipad`, allowing transformations to and from GCRS instead of CIRS.
4. The accuracy is sub-mas, limited by the precession-nutation information (IAU 2006 model plus the CIP corrections, which are obtained from IERS bulletins).
5. The accuracy is in principle further limited by the function `slaEpv`, called by `slaC2ipa`, which computes the Earth position and velocity. The maximum error arising from this source is less than $5 \mu\text{s}$.

6. The observer's position and velocity **pv** are in GCRS coordinates and in units of km and km/s.

slaC2iqk quick ICRS to CIRS **slaC2iqk**

ACTION : Quick ICRS place, epoch J2000, to geocentric CIRS, given the star-independent parameters.

CALL : `slaC2iqk(rc, dc, pr, pd, px, rv, &ciprms, &ri, &di);`

GIVEN :

<code>rc,dc</code>	double	ICRS $[\alpha, \delta]$ at J2000 (radians)
<code>pr,pd</code>	double	proper motions: rate of change of $[\alpha, \delta]$ (radians per year)
<code>px</code>	double	parallax (arcsec)
<code>rv</code>	double	radial velocity (km s^{-1} , +ve if receding)
<code>ciprms</code>	CIpars*	star-independent ICRS-to-CIRS parameters:
	<code>pmt</code>	◦ time interval for proper motion (Julian years)
	<code>eb[3]</code>	◦ barycentric position of the Earth (AU)
	<code>ehn[3]</code>	◦ heliocentric direction of the Earth (unit vector)
	<code>gr2e</code>	◦ $(\text{gravitational radius of Sun}) \times 2 / (\text{Sun-Earth distance})$
	<code>abv[3]</code>	◦ v : barycentric Earth velocity in units of <i>c</i>
	<code>ab1</code>	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	<code>bpn[3][3]</code>	◦ bias-precession-nutation 3×3 matrix

RETURNED :

<code>ri,di</code>	double*	CIRS $[\alpha, \delta]$ (radians)
--------------------	---------	-----------------------------------

NOTES :

1. Use of this function is appropriate when efficiency is important and where many star positions are to be transformed for one date. The star-independent parameters can be obtained by calling the `slaC2ipa` function.
2. If the parallax and proper motions are zero, the `slaC2iqkz` function can be used instead.
3. Star data for an epoch other than J2000 (for example from the Hipparcos catalogue) will require a preliminary call to `slaPm` before use.
4. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$.

5. Using `ciprms` parameters from `slaC2ipa`, the accuracy is sub-mas, limited by the precession-nutation information (IAU 2006 model plus the CIP corrections, which are obtained from IERS bulletins).
6. The accuracy is in principle further limited by the function `slaEpv`, which computes the Earth position and velocity. The maximum error arising from this source is less than $5\ \mu\text{s}$.
7. The vectors `ciprms->eb` and `ciprms->ehn` are in the BCRS.
8. Strictly speaking, the function is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about $920''$ of the centre of the Sun's disc. The term has a maximum value of about $1''.85$ at this radius, and decreases to zero as the centre of the disc is approached.

REFERENCE : Seidelmann, P.K. *et al.*, 1992, *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Chapter 3.

slaC2iqkz	quick ICRS to CIRS, no P.M. <i>etc.</i>	slaC2iqkz
------------------	---	------------------

ACTION : Quick ICRS $[\alpha, \delta]$, epoch J2000, to CIRS, given the star-independent parameters, and assuming zero parallax and proper motion.

CALL : `slaC2iqkz(rc, dc, &ciprms, &ri, &di);`

GIVEN :

<code>rc,dc</code>	<code>double</code>	ICRS $[\alpha, \delta]$ at J2000 (radians)
<code>ciprms</code>	<code>CIpars*</code>	star-independent ICRS-to-CIRS parameters:
	<code>pmt</code>	◦ time interval for proper motion (Julian years)
	<code>eb[3]</code>	◦ barycentric position of the Earth (AU)
	<code>ehn[3]</code>	◦ heliocentric direction of the Earth (unit vector)
	<code>gr2e</code>	◦ (gravitational radius of Sun) $\times 2 / (\text{Sun-Earth distance})$
	<code>abv[3]</code>	◦ v : barycentric Earth velocity in units of c
	<code>ab1</code>	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	<code>bpn[3][3]</code>	◦ bias-precession-nutation 3×3 matrix

RETURNED :

<code>ri,di</code>	<code>double*</code>	CIRS $[\alpha, \delta]$ (radians)
--------------------	----------------------	-----------------------------------

NOTES :

1. Use of this function is appropriate when efficiency is important and where many star positions are to be transformed for one date. The star-independent parameters can be obtained by calling the `slaC2ipa` function.
2. The corresponding function for the case of non-zero parallax and proper motion is `slaC2iqk`.
3. Using `ciprms` parameters from `slaC2ipa`, the accuracy is sub-mas, limited by the precession-nutation information (IAU 2006 model plus the CIP corrections, which are obtained from IERS bulletins).
4. The accuracy is in principle further limited by the function `slaEpv`, which computes the Earth position and velocity. The maximum error arising from this source is less than $5\ \mu\text{s}$.
5. The vectors `ciprms->eb` and `ciprms->ehn` are in the BCRS.
6. Strictly speaking, the function is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about $920''$ of the centre of the Sun's disc. The term has a maximum value of about $1''.85$ at this radius, and decreases to zero as the centre of the disc is approached.

REFERENCE : Seidelmann, P.K. *et al.*, 1992, *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Chapter 3.

slaCaf2r	deg, arcmin, arcsec to radians	slaCaf2r
-----------------	--------------------------------	-----------------

ACTION : Convert degrees, arcminutes, arcseconds to radians (single precision).

CALL : `slaCaf2r(ideg, iamin, asec, &rad, &j);`

GIVEN :

<code>ideg</code>	<code>int</code>	degrees
<code>iamin</code>	<code>int</code>	arcminutes
<code>asec</code>	<code>float</code>	arcseconds

RETURNED :

<code>rad</code>	<code>float*</code>	angle in radians
<code>j</code>	<code>int*</code>	status:
		1 = <code>ideg</code> outside range 0–359
		2 = <code>iamin</code> outside range 0–59
		3 = <code>asec</code> outside range 0–59.999...

NOTES :

1. The result is computed even if any of the range checks fail.
2. The sign must be dealt with outside this function.

slaCaldj	calendar date to MJD	slaCaldj
-----------------	----------------------	-----------------

ACTION : Gregorian Calendar to Modified Julian Date, with century default.

CALL : slaCaldj(iy, im, id, &djm, &j);

GIVEN :

iy, im, id	int	year, month, day in Gregorian calendar
------------	-----	--

RETURNED :

djm	double*	modified Julian Date (JD−2400000.5) for 0 ^h
j	int*	status:
		0 = OK
		1 = bad year (MJD not computed)
		2 = bad month (MJD not computed)
		3 = bad day (MJD computed)

NOTES :

1. This function supports the *century default* feature. Acceptable years are:
 - 00-49, interpreted as 2000–2049,
 - 50-99, interpreted as 1950–1999, and
 - 100 upwards, interpreted literally.
 For 1-100 AD use the function **slaCldj** instead.
2. For year n BC use $iy = -(n - 1)$.
3. When an invalid year or month is supplied (status $j = 1$ or 2) the MJD is **not** computed. When an invalid day is supplied (status $j = 3$) the MJD **is** computed.

slaCalyd	calendar date to year, day	slaCalyd
-----------------	----------------------------	-----------------

ACTION : Gregorian calendar date to year and day in year, in a Julian calendar aligned to the 20th/21st century Gregorian calendar, with century default.

CALL : slaCalyd(iy, im, id, &ny, &nd, &j);

GIVEN :

iy, im, id	int	year, month, day in Gregorian calendar: year may optionally omit the century
------------	-----	--

RETURNED :

ny	int*	year (re-aligned Julian calendar)
nd	int*	day in year (1 = January 1st)
j	int*	status:
		0 = OK
		1 = bad year (before -4711)
		2 = bad month
		3 = bad day

NOTES :

1. This function supports the *century default* feature. Acceptable years are:
 - 00-49, interpreted as 2000–2049,
 - 50-99, interpreted as 1950–1999, and
 - other years after -4712, interpreted literally.
 Use `slaClyd` for years before 100AD.
2. The purpose of `slaCaldj` is to support `slaEarth`, `slaMoon` and `slaEcor`.
3. Between 1900 March 1 and 2100 February 28 it returns answers which are consistent with the ordinary Gregorian calendar. Outside this range there will be a discrepancy which increases by one day for every non-leap century year.
4. When an invalid year or month is supplied (status `j = 1` or `j = 2`) the results are **not** computed. When a day is supplied which is outside the conventional range (status `j = 3`) the results **are** computed.

slaCc2s	Cartesian to spherical	slaCc2s
----------------	------------------------	----------------

ACTION : Cartesian coordinates to spherical coordinates (single precision).

CALL : `slaCc2s(v, &a, &b);`

GIVEN :

v	float[3]	$[x, y, z]$ vector
---	----------	--------------------

RETURNED :

a, b	float*	spherical coordinates in radians
------	--------	----------------------------------

NOTES :

1. The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.
2. If \mathbf{v} is null, zero \mathbf{a} and \mathbf{b} are returned.
3. At either pole, zero \mathbf{a} is returned.

slaCc62s	Cartesian 6-vector to spherical	slaCc62s
-----------------	---------------------------------	-----------------

ACTION : Transformation of position & velocity in Cartesian coordinates to spherical coordinates (single precision).

CALL : `slaCc62s(v, &a, &b, &r, &ad, &bd, &rd);`

GIVEN :

\mathbf{v}	float[6]	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$
--------------	----------	--

RETURNED :

a	float*	longitude (radians) – for example α
b	float*	latitude (radians) – for example δ
r	float*	radial coordinate
ad	float*	longitude derivative (radians per unit time)
bd	float*	latitude derivative (radians per unit time)
rd	float*	radial derivative

NOTE : The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.

slaCd2tf	days to hour,min,sec	slaCd2tf
-----------------	----------------------	-----------------

ACTION : Convert an interval in days to hours, minutes, seconds (single precision).

CALL : `slaCd2tf(ndp, days, &sign, ihmsf);`

GIVEN :

ndp	int	number of decimal places of seconds
days	float	interval in days

RETURNED :

<code>sign</code>	<code>char*</code>	<code>'+' or '-'</code>
<code>ihmsf</code>	<code>int[4]</code>	hours, minutes, seconds, fraction

NOTES :

1. `ndp` less than zero is interpreted as zero.
2. The largest useful value for `ndp` is determined by the size of `days`, the format of `float` numbers on the target machine, and the risk of overflowing `ihmsf[3]`. On some architectures, for `days` up to 1.0, the available floating-point precision corresponds roughly to `ndp` = 3. This is well below the ultimate limit of `ndp` = 9 set by the capacity of a typical 32-bit `ihmsf[3]`.
3. The absolute value of `days` may exceed 1.0. In cases where it does not, it is up to the caller to test for and handle the case where `days` is very nearly 1.0 and rounds up to 24 hours, by testing for `ihmsf[0] = 24` and setting `ihmsf[0-3]` to zero.

slaCldj	calendar date to MJD	slaCldj
----------------	----------------------	----------------

ACTION : Gregorian Calendar to Modified Julian Date.**CALL :** `slaCldj(iy, im, id, &djm, &j);`**GIVEN :**

<code>iy, im, id</code>	<code>int</code>	year, month, day in Gregorian calendar
-------------------------	------------------	--

RETURNED :

<code>djm</code>	<code>double*</code>	modified Julian Date (JD-2400000.5) for 0 ^h
<code>j</code>	<code>int*</code>	status:
		0 = OK
		1 = bad year
		2 = bad month
		3 = bad day

NOTES :

1. When an invalid year or month is supplied (status `j` = 1 or 2) the MJD is **not** computed. When an invalid day is supplied (status `j` = 3) the MJD **is** computed.
2. The year must be -4699 (*i.e.* 4700 BC) or later. For year n BC use `iy` = $-(n - 1)$.
3. An alternative to the present function is `slaCaldj`, which accepts a year with the century missing.

REFERENCE : The algorithm is adapted from Hatcher 1984, *Q. Jl. R. astr. Soc.* **25**, 53-55.

slaClyd	calendar date to year,day	slaClyd
----------------	---------------------------	----------------

ACTION : Gregorian calendar date to year and day in year, in a Julian calendar aligned to the 20th/21st century Gregorian calendar.

CALL : slaClyd(iy, im, id, &ny, &nd, &j);

GIVEN :

iy, im, id	int	year, month, day in Gregorian calendar
------------	-----	--

RETURNED :

ny	int*	year (re-aligned Julian calendar)
nd	int*	day in year (1 = January 1st)
j	int*	status:
		0 = OK
		1 = bad year (before -4711)
		2 = bad month
		3 = bad day

NOTES :

1. The purpose of **slaClyd** is to support **slaEarth**, **slaMoon** and **slaEcor**.
2. Between 1900 March 1 and 2100 February 28 it returns answers which are consistent with the ordinary Gregorian calendar. Outside this range there will be a discrepancy which increases by one day for every non-leap century year.
3. When an invalid year or month is supplied (status **j** = 1 or **j** = 2) the results are **not** computed. When a day is supplied which is outside the conventional range (status **j** = 3) the results **are** computed.

slaCombn	next combination	slaCombn
-----------------	------------------	-----------------

ACTION : Generate the next combination, a subset of a specified size chosen from a specified number of items.

CALL : slaCombn(nsel, ncand, list, &j);

GIVEN :

nset	int	number of items (subset size)
ncand	int	number of candidates (set size)

GIVEN and RETURNED :

list	I(NSEL)	latest combination, LIST(1)=0 to initialize
-------------	----------------	---

RETURNED :

j	int*	status:
		-1 = illegal nset or ncand
		0 = OK
		+1 = no more combinations available

NOTES :

1. **nset** and **ncand** must both be at least 1, and **nset** must be less than or equal to **ncand**.
2. This function returns, in the LIST array, a subset of **nset** integers chosen from the range 1 to **ncand** inclusive, in ascending order. Before calling the function for the first time, the caller must set the first element of the **list** array to zero (any value less than 1 will do) to cause initialization.
3. The first combination to be generated is:

$$\text{list}[0]=1, \text{list}[1]=2, \dots, \text{list}[\text{nset}-1]=\text{nset}$$
This is also the combination returned for the “finished” ($j=1$) case. The final permutation to be generated is:

$$\text{list}[0]=\text{ncand}, \text{list}[1]=\text{ncand}-1, \dots, \text{list}[\text{nset}-1]=\text{ncand}-\text{nset}+1$$
4. If the “finished” ($j=1$) status is ignored, the function continues to deliver combinations, the pattern repeating every $\text{ncand}!/(\text{nset}!(\text{ncand}-\text{nset})!)$ calls.
5. The algorithm is by R. F. Warren-Smith (private communication).

slaCr2af

radians to deg, arcmin, arcsec

slaCr2af**ACTION :** Convert an angle in radians to degrees, arcminutes, arcseconds (single precision).**CALL :** slaCr2af(ndp, angle, &sign, idmsf);**GIVEN :**

ndp	int	number of decimal places of arcseconds
angle	float	angle in radians

RETURNED :

sign	char*	'+' or '-'
idmsf	int[4]	degrees, arcminutes, arcseconds, fraction

NOTES :

1. `ndp` less than zero is interpreted as zero.
2. The largest useful value for `ndp` is determined by the size of `angle`, the format of `float` numbers on the target machine, and the risk of overflowing `idmsf[3]`. On some architectures, for `angle` up to 2π , the available floating-point precision corresponds roughly to `ndp` = 3. This is well below the ultimate limit of `ndp` = 9 set by the capacity of a typical 32-bit `idmsf[3]`.
3. The absolute value of `angle` may exceed 2π . In cases where it does not, it is up to the caller to test for and handle the case where `angle` is very nearly 2π and rounds up to 360° , by testing for `idmsf[0] = 360` and setting `idmsf[0-3]` to zero.

slaCr2tf	radians to hour,min,sec	slaCr2tf
-----------------	-------------------------	-----------------

ACTION : Convert an angle in radians to hours, minutes, seconds (single precision).

CALL : `slaCr2tf(ndp, angle, &sign, ihmsf);`

GIVEN :

<code>ndp</code>	int	number of decimal places of seconds
<code>angle</code>	float	angle in radians

RETURNED :

sign	char*	'+' or '-'
ihmsf	int[4]	hours, minutes, seconds, fraction

NOTES :

1. `ndp` less than zero is interpreted as zero.
2. The largest useful value for `ndp` is determined by the size of `angle`, the format of `float` numbers on the target machine, and the risk of overflowing `ihmsf[3]`. On some architectures, for `angle` up to 2π , the available floating-point precision corresponds roughly to `ndp` = 3. This is well below the ultimate limit of `ndp` = 9 set by the capacity of a typical 32-bit `ihmsf[3]`.
3. The absolute value of `angle` may exceed 2π . In cases where it does not, it is up to the caller to test for and handle the case where `angle` is very nearly 2π and rounds up to 24 hours, by testing for `ihmsf[0] = 24` and setting `ihmsf[0-3]` to zero.

slaCs2c	spherical to Cartesian	slaCs2c
----------------	------------------------	----------------

ACTION : Spherical coordinates to Cartesian coordinates (single precision).

CALL : `slaCs2c(a, b, v);`

GIVEN :

<code>a, b</code>	<code>float</code>	spherical coordinates in radians: $[\alpha, \delta]$ <i>etc.</i>
-------------------	--------------------	--

RETURNED :

<code>v</code>	<code>float[3]</code>	$[x, y, z]$ unit vector
----------------	-----------------------	-------------------------

NOTE : The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.

slaCs2c6	spherical position and velocity to Cartesian	slaCs2c6
-----------------	--	-----------------

ACTION : Transformation of position & velocity in spherical coordinates to Cartesian coordinates (single precision).

CALL : `slaCs2c6(a, b, r, ad, bd, rd, v);`

GIVEN :

<code>a</code>	<code>float</code>	longitude (radians) – for example α
<code>b</code>	<code>float</code>	latitude (radians) – for example δ
<code>float</code>	<code>float</code>	radial coordinate
<code>ad</code>	<code>float</code>	longitude derivative (radians per unit time)
<code>bd</code>	<code>float</code>	latitude derivative (radians per unit time)
<code>rd</code>	<code>float</code>	radial derivative

RETURNED :

<code>v</code>	<code>float[6]</code>	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$
----------------	-----------------------	--

NOTE : The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.

slaCtf2d	hour,min,sec to days	slaCtf2d
-----------------	----------------------	-----------------

ACTION : Convert hours, minutes, seconds to days (single precision).

CALL : `slaCtf2d(ihour, imin, sec, &days, &j);`

GIVEN :

<code>ihour</code>	<code>int</code>	hours
<code>imin</code>	<code>int</code>	minutes
<code>sec</code>	<code>float</code>	seconds

RETURNED :

<code>days</code>	<code>float*</code>	interval in days
<code>j</code>	<code>int</code>	status:
		0 = OK
		1 = <code>ihour</code> outside range 0-23
		2 = <code>imin</code> outside range 0-59
		3 = <code>sec</code> outside range 0-59.999...

NOTES :

1. The result is computed even if any of the range checks fail.
2. The sign must be dealt with outside this function.

slaCtf2r	hour,min,sec to radians	slaCtf2r
-----------------	-------------------------	-----------------

ACTION : Convert hours, minutes, seconds to radians (single precision).

CALL : `slaCtf2r(ihour, imin, sec, &rad, &j);`

GIVEN :

<code>ihour</code>	<code>int</code>	hours
<code>imin</code>	<code>int</code>	minutes
<code>sec</code>	<code>float</code>	seconds

RETURNED :

rad	float*	angle in radians
j	int*	status:
		0 = OK
		1 = <code>ihour</code> outside range 0-23
		2 = <code>imin</code> outside range 0-59
		3 = <code>sec</code> outside range 0-59.999...

NOTES :

1. The result is computed even if any of the range checks fail.
2. The sign must be dealt with outside this function.

slaDaf2r	deg,arcmin,arcsec to radians	slaDaf2r
-----------------	------------------------------	-----------------

ACTION : Convert degrees, arcminutes, arcseconds to radians (double precision).

CALL : `slaDaf2r(ideg, iamin, asec, &rad, &j);`

GIVEN :

ideg	int	degrees
iamin	int	arcminutes
asec	double	arcseconds

RETURNED :

rad	double*	angle in radians
j	int*	status:
		1 = <code>ideg</code> outside range 0–359
		2 = <code>iamin</code> outside range 0–59
		3 = <code>asec</code> outside range 0–59.999...

NOTES :

1. The result is computed even if any of the range checks fail.
2. The sign must be dealt with outside this function.

slaDafin	sexagesimal character string to angle	slaDafin
-----------------	---------------------------------------	-----------------

ACTION : Decode a free-format sexagesimal string (degrees, arcminutes, arcseconds) into a double precision floating point number (radians).

CALL : `slaDafin(string, &nstrt, &dreslt, &jf);`

GIVEN :

string	char*	string containing deg, arcmin, arcsec fields
nstrt	int	index to start of decode (beginning of string = 1)

RETURNED :

nstrt	int*	advanced past the decoded angle
dresult	double*	angle in radians
jf	int*	status:
		0 = OK
		+1 = default, dresult unchanged (Note 2)
		-1 = bad degrees (Note 3)
		-2 = bad arcminutes (Note 3)
		-3 = bad arcseconds (Note 3)

EXAMPLE :

<i>argument</i>	<i>before</i>	<i>after</i>
string	'-57_17_44.806__12_34_56.7'	unchanged
nstrt	1	16 (<i>i.e.</i> pointing to 12...)
result	-	-1.00000
jf	-	0

A further call to **slaDafin**, without adjustment of **nstrt**, will decode the second angle, 12° 34' 56".7.

NOTES :

1. 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 **slaDfltin** function and is free-format.
2. 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 **dresult** is unchanged. In all other cases **dresult** is changed.
3. 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 **dreslt** is computed and returned.
- 4. Further fields after the three expected ones are not treated as an error. The index **nstrt** is left in the correct state for further decoding with the present function or with **slaDfltin** *etc.* See the example, above.
- 5. If **string** contains hours, minutes, seconds instead of degrees *etc.*, or if the required units are turns (or days) instead of radians, the result **dreslt** should be multiplied as follows:

<i>for string</i>	<i>to obtain</i>	<i>multiply dreslt by</i>
o / "	radians	1.0
o / "	turns	$1/2\pi = 0.1591549430918953358$
h m s	radians	15.0
h m s	days	$15/2\pi = 2.3873241463784300365$

slaDat	TAI–UTC	slaDat
---------------	----------------	---------------

ACTION : Increment to be applied to Coordinated Universal Time UTC to give International Atomic Time TAI.

CALL : **d = slaDat(utc);**

GIVEN :

utc	double	UTC date/time (MJD, JD–2400000.5)
------------	---------------	-----------------------------------

RETURNED :

double*	TAI–UTC in seconds
----------------	--------------------

NOTES :

1. The UTC is specified to be a date rather than a time to indicate that care needs to be taken not to specify an instant which lies within a leap second. Although in most cases **utc** can include the fractional part, correct behaviour on the day of a leap second can be guaranteed only up to the end of the second 23^h 59^m 59^s.
2. For dates from 1961 January 1 onwards, the expressions from the file **ftp://maia.usno.navy.mil/ser7/tai-utc.dat** are used. A 5 ms time step at 1961 January 1 is taken from 2.58.1 (p87) of the 1992 Explanatory Supplement.
3. UTC began at 1960 January 1.0 (JD 2436934.5) and it is improper to call the function with an earlier date. However, if this is attempted, the TAI–UTC expression for the year 1960 is used.

4. This function has to be updated on each occasion that a leap second is announced, and programs using it relinked. Refer to the program source code for information on when the most recent leap second was added.

slaDav2m	rotation matrix from axial vector	slaDav2m
-----------------	-----------------------------------	-----------------

ACTION : Form the rotation matrix corresponding to a given axial vector (double precision).

CALL : `slaDav2m(axvec, rmat);`

GIVEN :

<code>axvec</code>	<code>double[3]</code>	axial vector (radians)
--------------------	------------------------	------------------------

RETURNED :

<code>rmat</code>	<code>double[3][3]</code>	rotation matrix
-------------------	---------------------------	-----------------

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis, called the Euler axis. The *axial vector* supplied to this function has the same direction as the Euler axis, and its magnitude is the amount of rotation in radians.
2. If `axvec` is null, the unit matrix is returned.
3. The reference frame rotates clockwise as seen looking along the axial vector from the origin.

slaDbear	direction between points on a sphere	slaDbear
-----------------	--------------------------------------	-----------------

ACTION : Returns the bearing (position angle) of one point on a sphere relative to another (double precision).

CALL : `d = slaDbear(a1, b1, a2, b2);`

GIVEN :

<code>a1,b1</code>	<code>double</code>	spherical coordinates of one point
<code>a2,b2</code>	<code>double</code>	spherical coordinates of the other point

RETURNED :

double* bearing from first point to second

NOTES :

1. The spherical coordinates are $[\alpha, \delta]$, $[\lambda, \beta]$ *etc.*, in radians.
2. The result is the bearing (position angle), in radians, of point $[a2, b2]$ as seen from point $[a1, b1]$. It is in the range $\pm\pi$. The sense is such that if $[a2, b2]$ is a small distance due east of $[a1, b1]$ the result is about $+\pi/2$. Zero is returned if the two points are coincident.
3. If either B-coordinate is outside the range $\pm\pi/2$, the result may correspond to “the long way round”.
4. The function `slaDpav` performs an equivalent function except that the points are specified in the form of Cartesian vectors.

slaDbjin	decode string to B/J epoch	slaDbjin
-----------------	----------------------------	-----------------

ACTION : Decode a character string into a double precision number, with special provision for Besselian and Julian epochs. The string syntax is as for `slaDfltln`, prefixed by an optional 'B' or 'J'.

CALL : `slaDbjin(string, &nstrt, &dreslt, &j1, &j2);`

GIVEN :

string	char*	string containing field to be decoded
nstrt	int	index to first character of field in string

RETURNED :

nstrt	int*	incremented past the decoded field
dreslt	double*	result
j1	int*	<code>slaDfltln</code> status: -1 = -OK 0 = +OK 1 = null field 2 = error
j2	int*	syntax flag: 0 = normal <code>slaDfltln</code> syntax 1 = 'B' or 'b' 2 = 'J' or 'j'

NOTES :

1. The purpose of the syntax extensions is to help cope with mixed FK4 and FK5 data, allowing fields such as 'B1950' or 'J2000' to be decoded.
2. In addition to the syntax accepted by `slaDfltIn`, the following two extensions are recognized by `slaDbjin`:
 - (a) A valid non-null field preceded by the character 'B' (or 'b') is accepted.
 - (b) A valid non-null field preceded by the character 'J' (or 'j') is accepted.
3. The calling program is told of the 'B' or 'J' through an supplementary status argument. The rest of the arguments are as for `slaDfltIn`.

slaDc62s	Cartesian 6-vector to spherical	slaDc62s
-----------------	---------------------------------	-----------------

ACTION : Transformation of position & velocity in Cartesian coordinates to spherical coordinates (double precision).

CALL : `slaDc62s(v, &a, &b, &r, &ad, &bd, &rd);`

GIVEN :

<code>v</code>	<code>double[6]</code>	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$
----------------	------------------------	--

RETURNED :

<code>a</code>	<code>double*</code>	longitude (radians)
<code>b</code>	<code>double*</code>	latitude (radians)
<code>r</code>	<code>double*</code>	radial coordinate
<code>ad</code>	<code>double*</code>	longitude derivative (radians per unit time)
<code>bd</code>	<code>double*</code>	latitude derivative (radians per unit time)
<code>rd</code>	<code>double*</code>	radial derivative

NOTE : The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.

slaDcc2s	Cartesian to spherical	slaDcc2s
-----------------	------------------------	-----------------

ACTION : Cartesian coordinates to spherical coordinates (double precision).

CALL : `slaDcc2s(v, &a, &b);`

GIVEN :

v	double[3]	$[x, y, z]$ vector
----------	------------------	--------------------

RETURNED :

a,b	double*	spherical coordinates in radians
------------	----------------	----------------------------------

NOTES :

1. The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.
2. If V is null, zero **a** and **b** are returned.
3. At either pole, zero **a** is returned.

slaDcmpf	interpret linear fit	slaDcmpf
-----------------	----------------------	-----------------

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

CALL : `slaDcmpf(coeffs, &xz, &yz, &xs, &ys, &perp, &orient);`

GIVEN :

coeffs	double[6]	transformation coefficients (see note)
---------------	------------------	--

RETURNED :

xz	double*	x zero point
yz	double*	y zero point
xs	double*	x scale
ys	double*	y scale
perp	double	nonperpendicularity (radians)
orient	double	orientation (radians)

NOTES :

1. The model relates two sets of $[x, y]$ coordinates as follows. Naming the six elements of `coeffs` a, b, c, d, e & f , the model transforms coordinates $[x_1, y_1]$ into coordinates $[x_2, y_2]$ as follows:

$$\begin{aligned}x_2 &= a + bx_1 + cy_1 \\ y_2 &= d + ex_1 + fy_1\end{aligned}$$

The `slaDcmpf` function decomposes this transformation into four steps:

- (a) Zero points:

$$\begin{aligned}x' &= x_1 + \mathbf{xz} \\ y' &= y_1 + \mathbf{yz}\end{aligned}$$

- (b) Scales:

$$\begin{aligned}x'' &= x' \times \mathbf{xs} \\ y'' &= y' \times \mathbf{ys}\end{aligned}$$

- (c) Nonperpendicularity:

$$\begin{aligned}x''' &= +x'' \cos(\mathbf{perp}/2) + y'' \sin(\mathbf{perp}/2) \\ y''' &= +x'' \sin(\mathbf{perp}/2) + y'' \cos(\mathbf{perp}/2)\end{aligned}$$

- (d) Orientation:

$$\begin{aligned}x_2 &= +x''' \cos(\mathbf{orient}) + y''' \sin(\mathbf{orient}) \\ y_2 &= -x''' \sin(\mathbf{orient}) + y''' \cos(\mathbf{orient})\end{aligned}$$

2. See also `slaFitxy`, `slaPxy`, `slaInvf`, `slaXy2xy`.

slaDcs2c	spherical to Cartesian	slaDcs2c
-----------------	------------------------	-----------------

ACTION : Spherical coordinates to Cartesian coordinates (double precision).

CALL : `slaDcs2c(a, b, v);`

GIVEN :

<code>a, b</code>	<code>double</code>	spherical coordinates in radians: $[\alpha, \delta]$ <i>etc.</i>
-------------------	---------------------	--

RETURNED :

<code>v</code>	<code>double[3]</code>	$[x, y, z]$ unit vector
----------------	------------------------	-------------------------

NOTE : The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.

slaDd2tf	days to hour,min,sec	slaDd2tf
-----------------	----------------------	-----------------

ACTION : Convert an interval in days into hours, minutes, seconds (double precision).

CALL : slaDd2tf(ndp, days, &sign, ihmsf);

GIVEN :

ndp	int	number of decimal places of seconds
days	double	interval in days

RETURNED :

sign	char*	'+' or '-'
ihmsf	int[4]	hours, minutes, seconds, fraction

NOTES :

1. ndp less than zero is interpreted as zero.
2. The largest useful value for ndp is determined by the size of days, the format of double numbers on the target machine, and the risk of overflowing ihmsf[3]. On some architectures, for days up to 1.0, the available floating-point precision corresponds roughly to ndp = 12. However, the practical limit is ndp = 9, set by the capacity of a typical 32-bit ihmsf[3].
3. The absolute value of days may exceed 1. In cases where it does not, it is up to the caller to test for and handle the case where days is very nearly 1.0 and rounds up to 24 hours, by testing for ihmsf[0] = 24 and setting ihmsf[0-3] to zero.

slaDe2h	$[h, \delta]$ to $[Az, El]$	slaDe2h
----------------	-----------------------------	----------------

ACTION : Equatorial to horizon coordinates (double precision).

CALL : slaDe2h(ha, dec, phi, &az, &el);

GIVEN :

ha	double	hour angle (radians)
dec	double	declination (radians)
phi	double	latitude (radians)

RETURNED :

az	double*	azimuth (radians)
el	double*	elevation (radians)

NOTES :

1. Azimuth is returned in the range $0-2\pi$; north is zero, and east is $+\pi/2$. Elevation is returned in the range $\pm\pi$.
2. The latitude must be geodetic. In critical applications, corrections for polar motion should be applied.
3. 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* $[h, \delta]$, and the elevation *in vacuo*, which would require the *topocentric* $[h, \delta]$. If the effects of diurnal aberration can be neglected, the *apparent* $[h, \delta]$ may be used instead of the topocentric $[h, \delta]$.
4. No range checking of arguments is carried out.
5. In applications which involve many such calculations, rather than calling the present function 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.

slaDeuler	Euler angles to rotation matrix	slaDeuler
------------------	---------------------------------	------------------

ACTION : Form a rotation matrix from the Euler angles – three successive rotations about specified Cartesian axes (double precision).

CALL : slaDeuler(order, phi, theta, psi, rmat);

GIVEN :

order	char*	specifies about which axes the rotations occur
phi	double	first rotation (radians)
theta	double	second rotation (radians)
psi	double	third rotation (radians)

RETURNED :

rmat	double[3][3]	rotation matrix
------	--------------	-----------------

NOTES :

1. 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.
2. The axis names can be any of the following, in any order or combination: 'X', 'Y', 'Z', uppercase or lowercase, '1', '2', '3'. Thus, the "zxz" example given above could be written "ZXZ" or "313" (or even "ZxZ" or "3xZ").
3. **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. Zero rotations produces the identity **rmat**.
4. Normal axis labelling/numbering conventions apply; the *xyz* (\equiv "123") triad is right-handed.
5. A rotation is positive when the reference frame rotates anticlockwise as seen looking towards the origin from the positive region of the specified axis.

slaDflt	decode a double precision number	slaDflt
----------------	----------------------------------	----------------

ACTION : Convert free-format input into double precision floating point.

CALL : `slaDflt(string, &nstrt, &dreslt, &jflag);`

GIVEN :

string	char*	string containing number to be decoded
nstrt	int*	index to where decoding is to commence
dreslt	double*	current value of result

RETURNED :

nstrt	int*	advanced to next number
dreslt	double*	result
jflag	int*	status: -1 = -OK, 0 = +OK, 1 = null result, 2 = error

NOTES :

1. The reason **slaDflt** has separate "OK" status values for '+' and '-' is to enable minus zero to be detected. This is of crucial importance when decoding mixed-radix numbers. For example, an angle expressed as degrees, arcminutes and arcseconds may have a leading minus sign but a zero degrees field.

2. A TAB is interpreted as a space, and lowercase characters are interpreted as upper-case.
3. The basic format is the sequence of fields $\pm n.nx \pm n$, where \pm is a sign character '+' or '-', n means a string of decimal digits, '.' is a decimal point, and x , which indicates an exponent, means 'D' or 'E'. Various combinations of these fields can be omitted, and embedded blanks are permissible in certain places.
4. Spaces:
 - Leading spaces are ignored.
 - Embedded spaces are allowed only after +, -, D or E, and after the decimal point if the first sequence of digits is absent.
 - Trailing spaces are ignored; the first signifies end of decoding and subsequent ones are skipped.
5. Delimiters:
 - Any character other than +, -, 0-9, period, D, E or space may be used to signal the end of the number and terminate decoding.
 - Comma is recognized by `slaDfltln` as a special case; it is skipped, leaving the index on the next character. See 13, below.
 - Decoding will in all cases terminate if end of string is reached.
6. Both signs are optional. The default is '+'.
7. The mantissa $n.n$ defaults to unity.
8. The exponent $x \pm n$ defaults to "D0".
9. The strings of decimal digits may be of any length.
10. The decimal point is optional for whole numbers.
11. A *null result* occurs when the string of characters being decoded does not begin with +, -, 0-9, period, D or E or consists entirely of spaces. When this condition is detected, `jflag` is set to 1 and `dresult` is left untouched.
12. `nstrt` = 1 for the first character in the string.
13. On return from `slaDfltln`, `nstrt` is set ready for the next decode – following trailing blanks and any comma. If a delimiter other than comma is being used, `nstrt` must be incremented before the next call to `slaDfltln`, otherwise all subsequent calls will return a null result.
14. Errors (`jflag` = 2) occur when:
 - a +, -, D or E is left unsatisfied; or
 - the decimal point is present without at least one decimal digit before or after it; or
 - an exponent more than 100 has been presented.
15. When an error has been detected, `nstrt` is left pointing to the character following the last one used before the error came to light. This may be after the point at which a more sophisticated program could have detected the error. For example, `slaDfltln` does not detect that "1e999" is unacceptable (on a platform where this is so) until the entire number has been decoded.
16. Certain highly unlikely combinations of mantissa and exponent can cause arithmetic faults during the decode, in some cases despite the fact that they together could be construed as a valid number.

17. Decoding is left to right, one pass.
18. See also `slaFlotin`, `slaInt2in` and `slaIntin`.

slaDh2e	$[Az, El]$ to $[h, \delta]$	slaDh2e
----------------	-----------------------------	----------------

ACTION : Horizon to equatorial coordinates (double precision).

CALL : `slaDh2e(az, el, phi, &ha, &dec);`

GIVEN :

<code>az</code>	<code>double</code>	azimuth (radians)
<code>el</code>	<code>double</code>	elevation (radians)
<code>phi</code>	<code>double</code>	latitude (radians)

RETURNED :

<code>ha</code>	<code>double*</code>	hour angle (radians)
<code>dec</code>	<code>double*</code>	declination (radians)

NOTES :

1. The sign convention for azimuth is north zero, east $+\pi/2$.
2. Hour angle is returned in the range $\pm\pi$. Declination is returned in the range $\pm\pi/2$.
3. The latitude is (in principle) geodetic. In critical applications, corrections for polar motion should be applied (see `slaPolmo`).
4. In some applications it will be important to specify the correct type of elevation in order to produce the required type of $[h, \delta]$. In particular, it may be important to distinguish between the elevation as affected by refraction, which will yield the *observed* $[h, \delta]$, and the elevation *in vacuo*, which will yield the *topocentric* $[h, \delta]$. If the effects of diurnal aberration can be neglected, the topocentric $[h, \delta]$ may be used as an approximation to the *apparent* $[h, \delta]$.
5. No range checking of arguments is carried out.
6. In applications which involve many such calculations, rather than calling the present function it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude.

slaDimxv	apply 3D reverse rotation	slaDimxv
-----------------	---------------------------	-----------------

ACTION : Multiply a 3-vector by the inverse of a rotation matrix (double precision).

CALL : `slaDimxv(dm, va, vb);`

GIVEN :

dm	double[3][3]	rotation matrix
va	double[3]	vector to be rotated

RETURNED :

vb	double[3]	result vector
-----------	------------------	---------------

NOTES :

1. This function performs the operation:

$$\mathbf{b} = \mathbf{M}^T \times \mathbf{a}$$

where **a** and **b** are the 3-vectors **va** and **vb** respectively, and **M** is the 3×3 matrix **dm**.

2. The purpose of this function is to apply an inverse rotation; under these circumstances, **M** must be *orthogonal*, so that its inverse is the same as its transpose.
3. The same array may be specified for both **va** and **vb**.

slaDjcal	MJD to Gregorian for output	slaDjcal
-----------------	-----------------------------	-----------------

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

CALL : `slaDjcal(ndp, djm, iymdf, &j);`

GIVEN :

ndp	int	number of decimal places of days in fraction
djm	double	modified Julian Date (JD−2400000.5)

RETURNED :

iymdf	int[4]	year, month, day, fraction in Gregorian calendar
j	int*	status: nonzero = out of range

NOTES :

1. Any date after 4701 BC March 1 is accepted.
2. Large **ndp** values risk internal overflows. It is typically safe to use up to **ndp** = 4.

REFERENCE : The algorithm is adapted from Hatcher 1984, *Q. Jl. R. astr. Soc.* **25**, 53-55.

slaDjcl	MJD to year,month,day,frac	slaDjcl
----------------	----------------------------	----------------

ACTION : Modified Julian Date to Gregorian year, month, day, and fraction of a day.

CALL : `slaDjcl(djm, &iy, &im, &id, &fd, &j);`

GIVEN :

<code>djm</code>	<code>double</code>	modified Julian Date (JD−2400000.5)
------------------	---------------------	-------------------------------------

RETURNED :

<code>iy</code>	<code>int*</code>	year
<code>im</code>	<code>int*</code>	month
<code>id</code>	<code>int*</code>	day
<code>fd</code>	<code>double*</code>	fraction of day
<code>j</code>	<code>int*</code>	status:
		0 = OK
		−1 = unacceptable date
		(before 4701 BC March 1)

REFERENCE : The algorithm is adapted from Hatcher 1984, *Q. Jl. R. astr. Soc.* **25**, 53-55.

slaDm2av	rotation matrix to axial vector	slaDm2av
-----------------	---------------------------------	-----------------

ACTION : From a rotation matrix, determine the corresponding axial vector (double precision).

CALL : `slaDm2av(rmat, axvec);`

GIVEN :

<code>rmat</code>	<code>double[3][3]</code>	rotation matrix
-------------------	---------------------------	-----------------

RETURNED :

<code>axvec</code>	<code>double[3]</code>	axial vector (radians)
--------------------	------------------------	------------------------

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis, called the Euler axis. The *axial vector* returned by this function has the same direction as the Euler axis, and its magnitude is the amount of rotation in radians.
2. The magnitude and direction of the axial vector can be separated by means of the function `slaDvn`.
3. The reference frame rotates clockwise as seen looking along the axial vector from the origin.
4. If `rmat` is null, so is the result.

slaDmat	solve simultaneous equations	slaDmat
----------------	------------------------------	----------------

ACTION : Matrix inversion and solution of simultaneous equations (double precision).

CALL : `slaDmat(n, a, y, d, &jf, iw);`

GIVEN :

<code>n</code>	<code>int</code>	number of unknowns
<code>a</code>	<code>double[n][n]</code>	matrix
<code>y</code>	<code>double[n]</code>	vector

RETURNED :

<code>a</code>	<code>double[n][n]</code>	matrix inverse
<code>y</code>	<code>double[n]</code>	solution
<code>d</code>	<code>double*</code>	determinant
<code>jf</code>	<code>int*</code>	singularity flag: 0 = OK
<code>iw</code>	<code>int[n]</code>	workspace

NOTES :

1. For the set of n simultaneous linear equations in n unknowns:

$$\mathbf{A} \times \mathbf{y} = \mathbf{x}$$

where:

- \mathbf{A} is a non-singular $n \times n$ matrix,
- \mathbf{y} is the vector of n unknowns, and
- \mathbf{x} is the known vector,

`slaDmat` computes:

- the inverse of matrix \mathbf{A} ,
- the determinant of matrix \mathbf{A} , and

- the vector of n unknowns \mathbf{y} .

Argument \mathbf{n} is the order n , \mathbf{a} (given) is the matrix \mathbf{A} , \mathbf{y} (given) is the vector \mathbf{x} and \mathbf{y} (returned) is the vector \mathbf{y} . The argument \mathbf{a} (returned) is the inverse matrix \mathbf{A}^{-1} , and \mathbf{d} is $\det \mathbf{A}$.

2. \mathbf{jf} is the singularity flag. If the matrix is non-singular, $\mathbf{jf} = 0$ is returned. If the matrix is singular, $\mathbf{jf} = -1$ and $\mathbf{d} = 0.0$ are returned. In the latter case, the contents of array \mathbf{a} on return are undefined.
3. The algorithm is Gaussian elimination with partial pivoting. This method is very fast; some much slower algorithms can give better accuracy, but only by a small factor.

slaDmoon approximate Moon position and velocity **slaDmoon**

ACTION : Approximate geocentric position and velocity of the Moon (double precision).

CALL : `slaDmoon(date, pv);`

GIVEN :

<code>date</code>	<code>double</code>	TDB as a Modified Julian Date (JD−2400000.5)
-------------------	---------------------	--

RETURNED :

<code>pv</code>	<code>double[6]</code>	Moon $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, mean equator and equinox of date (AU, AU s ^{−1})
-----------------	------------------------	--

NOTES :

1. This function is a full implementation of the algorithm published by Meeus (see reference).
2. Meeus quotes accuracies of 10" in longitude, 3" 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 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" and 0".50/hour in longitude, 11" and 0".24/hour in latitude, 40 km and 0.29 m/s in distance.
3. The original algorithm is expressed in terms of the obsolete time-scale *Ephemeris Time*. Either TDB or TT can be used, but not UT without incurring significant errors (30" at the present time) due to the Moon's 0".5/s movement.
4. 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.
5. Velocity is obtained by a complete analytical differentiation of the Meeus model.

REFERENCE : Meeus, *l'Astronomie*, June 1984, p348.

slaDmxm	product of two 3×3 matrices	slaDmxm
----------------	--------------------------------------	----------------

ACTION : Product of two 3×3 matrices (double precision).

CALL : `slaDmxm(a, b, c);`

GIVEN :

a	double[3][3]	matrix A
b	double[3][3]	matrix B

RETURNED :

c	double[3][3]	matrix result: A × B
---	--------------	------------------------------------

NOTE : The arguments do not have to be different arrays.

slaDmxv	apply 3D rotation	slaDmxv
----------------	-------------------	----------------

ACTION : Multiply a 3-vector by a rotation matrix (double precision).

CALL : `slaDmxv(dm, va, vb);`

GIVEN :

dm	double[3][3]	rotation matrix
va	double[3]	vector to be rotated

RETURNED :

vb	double[3]	result vector
----	-----------	---------------

NOTES :

1. This function performs the operation:

$$\mathbf{b} = \mathbf{M} \times \mathbf{a}$$

where \mathbf{a} and \mathbf{b} are the 3-vectors \mathbf{va} and \mathbf{vb} respectively, and \mathbf{M} is the 3×3 matrix \mathbf{dm} .

2. The main function of this function is apply a rotation; under these circumstances, \mathbf{M} is a *proper real orthogonal* matrix.
3. \mathbf{va} and \mathbf{vb} may be the same array.

slaDpav	position-angle between two directions	slaDpav
----------------	---------------------------------------	----------------

ACTION : Returns the bearing (position angle) of one celestial direction with respect to another (double precision).

CALL : `d = slaDpav(v1, v2);`

GIVEN :

v1	double[3]	vector to one point
v2	double[3]	vector to the other point

RETURNED :

double	position-angle of second point with respect to first
--------	--

NOTES :

1. The coordinate frames correspond to $[\alpha, \delta]$, $[\lambda, \beta]$ etc..
2. The result is the bearing (position angle), in radians, of point $\mathbf{v2}$ as seen from point $\mathbf{v1}$. It is in the range $\pm\pi$. The sense is such that if $\mathbf{v2}$ is a small distance due east of $\mathbf{v1}$ the result is about $+\pi/2$. Zero is returned if the two points are coincident.
3. There is no requirement for either vector to be of unit length.
4. The function **slaDbear** performs an equivalent function except that the points are specified in the form of spherical coordinates.

slaDr2af	radians to deg,min,sec,frac	slaDr2af
-----------------	-----------------------------	-----------------

ACTION : Convert an angle in radians to degrees, arcminutes, arcseconds, fraction (double precision).

CALL : `slaDr2af(ndp, angle, &sign, idmsf);`

GIVEN :

ndp	int	number of decimal places of arcseconds
angle	double	angle in radians

RETURNED :

sign	char*	'+' or '-'
idmsf	int[4]	degrees, arcminutes, arcseconds, fraction

NOTES :

1. **ndp** less than zero is interpreted as zero.
2. The largest useful value for **ndp** is determined by the size of **angle**, the format of **double** numbers on the target machine, and the risk of overflowing **idmsf[3]**. On some architectures, for **angle** up to 2π , the available floating-point precision corresponds roughly to **ndp** = 12. However, the practical limit is **ndp** = 9 set by the capacity of a typical 32-bit **idmsf[3]**.
3. The absolute value of **angle** may exceed 2π . In cases where it does not, it is up to the caller to test for and handle the case where **angle** is very nearly 2π and rounds up to 360° , by testing for **idmsf[0]** = 360 and setting **idmsf[0-3]** to zero.

slaDr2tf	radians to hour,min,sec,frac	slaDr2tf
-----------------	------------------------------	-----------------

ACTION : Convert an angle in radians to hours, minutes, seconds, fraction (double precision).**CALL :** `slaDr2tf(ndp, angle, &sign, ihmsf);`**GIVEN :**

ndp	int	number of decimal places of seconds
angle	double	angle in radians

RETURNED :

sign	char*	'+' or '-'
ihmsf	int[4]	hours, minutes, seconds, fraction

NOTES :

1. **ndp** less than zero is interpreted as zero.
2. The largest useful value for **ndp** is determined by the size of **angle**, the format of **double** numbers on the target machine, and the risk of overflowing **ihmsf**[3]. On some architectures, for **angle** up to 2π , the available floating-point precision corresponds roughly to **ndp**=12. However, the practical limit is **ndp**=9, set by the capacity of a typical 32-bit **ihmsf**[3].
3. The absolute value of **angle** may exceed 2π . In cases where it does not, it is up to the caller to test for and handle the case where **angle** is very nearly 2π and rounds up to 24 hours, by testing for **ihmsf**[0] = 24 and setting **ihmsf**[0-3] to zero.

slaDrange	put angle into range $\pm\pi$	slaDrange
------------------	-------------------------------	------------------

ACTION : Normalize an angle into the range $\pm\pi$ (double precision).

CALL : `d = slaDrange(angle);`

GIVEN :

angle	double	angle in radians
--------------	---------------	------------------

RETURNED :

double	angle expressed in the range $\pm\pi$.
---------------	---

slaDranrm	put angle into range $0-2\pi$	slaDranrm
------------------	-------------------------------	------------------

ACTION : Normalize an angle into the range $0-2\pi$ (double precision).

CALL : `d = slaDranrm(angle);`

GIVEN :

angle	double	angle in radians
--------------	---------------	------------------

RETURNED :

double angle expressed in the range $0-2\pi$

slaDs2c6 spherical position and velocity to Cartesian **slaDs2c6**

ACTION : Transformation of position & velocity in spherical coordinates to Cartesian coordinates (double precision).

CALL : slaDs2c6(a, b, r, ad, bd, rd, v);

GIVEN :

a	double	longitude (radians) – for example α
b	double	latitude (radians) – for example δ
float	double	radial coordinate
ad	double	longitude derivative (radians per unit time)
bd	double	latitude derivative (radians per unit time)
rd	double	radial derivative

RETURNED :

v double[6] $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$

NOTE : The spherical coordinates are longitude (positive anticlockwise looking from the positive latitude pole) and latitude. The Cartesian coordinates are right handed, with the x -axis at zero longitude and latitude, and the z -axis at the positive latitude pole.

slaDs2tp spherical to tangent plane **slaDs2tp**

ACTION : Projection of spherical coordinates onto the tangent plane (double precision).

CALL : slaDs2tp(ra, dec, raz, decz, &xi, &eta, &j);

GIVEN :

ra,dec	double	spherical coordinates of star (radians)
raz,decz	double	spherical coordinates of tangent point (radians)

RETURNED :

xi,eta	*double	tangent plane coordinates (radians)
j	int*	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

NOTES :

1. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
2. When working in $[x, y, z]$ rather than spherical coordinates, the equivalent Cartesian function `slaDv2tp` is available.

slaDsep	angle between two points on a sphere	slaDsep
----------------	--------------------------------------	----------------

ACTION : Angle between two points on a sphere (double precision).

CALL : `d = slaDsep(a1, b1, a2, b2);`

GIVEN :

a1,b1	double	spherical coordinates of one point (radians)
a2,b2	double	spherical coordinates of the other point (radians)

RETURNED :

double	angle between [a1,b1] and [a2,b2] in radians
--------	--

NOTES :

1. The spherical coordinates are right ascension and declination, longitude and latitude, *etc.*, in radians.
2. The result is always positive.

slaDsepv	angle between two vectors	slaDsepv
-----------------	---------------------------	-----------------

ACTION : Angle between two vectors (double precision).

CALL : `d = slaDsepv(v1, v2);`

GIVEN :

v1	double[3]	first vector
v2	double[3]	second vector

RETURNED :

double	angle between v1 and v2 in radians
---------------	--

NOTES :

1. There is no requirement for either vector to be of unit length.
2. If either vector is null, zero is returned.
3. The result is always positive.

slaDt	approximate ET minus UT	slaDt
--------------	-------------------------	--------------

ACTION : Estimate ΔT , the offset between dynamical time and Universal Time, for a given historical epoch.

CALL : `d = slaDt(epoch);`

GIVEN :

epoch	double	(Julian) epoch (<i>e.g.</i> 1850.0)
--------------	---------------	--------------------------------------

RETURNED :

double	approximate ET–UT (after 1984, TT–UT) in seconds
---------------	--

NOTES :

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

before AD 979	Stephenson & Morrison's 390 BC to AD 948 model
AD 979 to AD 1708	Stephenson & Morrison's AD 948 to AD 1600 model
after AD 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.
2. The accuracy is modest, with errors of up to 20^s during the interval since 1650, rising to perhaps 30^m by 1000 BC. Comparatively accurate values from AD 1600 are tabulated in the *Astronomical Almanac* (see section K8 of the 1995 edition).
3. The use of double precision for both argument and result is simply for compatibility with other SLALIB time functions.
4. The models used are based on a lunar tidal acceleration value of $-26''.00$ per century.

REFERENCE : Seidelmann, P.K. (ed), 1992, *Explanatory Supplement to the Astronomical Almanac*, ISBN 0-935702-68-7. This contains references to the papers by Stephenson & Morrison and by McCarthy & Babcock which describe the models used here.

slaDtf2d	hour,min,sec to days	slaDtf2d
-----------------	----------------------	-----------------

ACTION : Convert hours, minutes, seconds to days (double precision).

CALL : slaDtf2d(ihour, imin, sec, &days, &j);

GIVEN :

ihour	int	hours
imin	int	minutes
sec	double	seconds

RETURNED :

days	double*	interval in days
j	int*	status:
		0 = OK
		1 = ihour outside range 0-23
		2 = imin outside range 0-59
		3 = sec outside range 0-59.999...

NOTES :

1. The result is computed even if any of the range checks fail.
2. The sign must be dealt with outside this function.

slaDtf2r	hour,min,sec to radians	slaDtf2r
-----------------	-------------------------	-----------------

ACTION : Convert hours, minutes, seconds to radians (double precision).

CALL : slaDtf2r(ihour, imin, sec, &rad, &j);

GIVEN :

ihour	int	hours
imin	int	minutes
sec	double	seconds

RETURNED :

rad	double*	angle in radians
j*	int	status:
		0 = OK
		1 = <code>ihour</code> outside range 0-23
		2 = <code>imin</code> outside range 0-59
		3 = <code>sec</code> outside range 0-59.999...

NOTES :

1. The result is computed even if any of the range checks fail.
2. The sign must be dealt with outside this function.

slaDtp2s	tangent plane to spherical	slaDtp2s
-----------------	----------------------------	-----------------

ACTION : Transform tangent plane coordinates into spherical coordinates (double precision)**CALL :** `slaDtp2s(xi, eta, raz, decz, &ra, &dec);`**GIVEN :**

xi,eta	double	tangent plane rectangular coordinates (radians)
raz,decz	double	spherical coordinates of tangent point (radians)

RETURNED :

ra,dec	double*	spherical coordinates (radians)
--------	---------	---------------------------------

NOTES :

1. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
2. When working in $[x, y, z]$ rather than spherical coordinates, the equivalent Cartesian function `slaDtp2v` is available.

slaDtp2v	tangent plane to $[x, y, z]$	slaDtp2v
-----------------	------------------------------	-----------------

ACTION : Given the tangent-plane coordinates of a star and the direction cosines of the tangent point, determine the direction cosines of the star (double precision).**CALL :** `slaDtp2v(xi, eta, v0, v);`

GIVEN :

xi,eta	double	tangent plane coordinates of star (radians)
v0	double[3]	direction cosines of tangent point

RETURNED :

v	double[3]	direction cosines of star
---	-----------	---------------------------

NOTES :

1. If vector v0 is not of unit length, the returned vector v will be wrong.
2. If vector v0 points at a pole, the returned vector v will be based on the arbitrary assumption that $\alpha = 0$ at the tangent point.
3. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
4. This function is the Cartesian equivalent of the function `slaDtp2s`.

slaDtps2c	plate centre from $[\xi, \eta]$ and $[\alpha, \delta]$	slaDtps2c
------------------	--	------------------

ACTION : From the tangent plane coordinates of a star of known $[\alpha, \delta]$, determine the $[\alpha, \delta]$ of the tangent point (double precision)

CALL : `slaDtps2c(xi, eta, ra, dec, &raz1, &decz1, &raz2, &decz2, &n);`

GIVEN :

xi,eta	double	tangent plane rectangular coordinates (radians)
ra,dec	double	spherical coordinates (radians)

RETURNED :

raz1,decz1	double*	spherical coordinates of tangent point, solution 1
raz2,decz2	double*	spherical coordinates of tangent point, solution 2
n	int*	number of solutions:

0 = no solutions returned (Note 2)
 1 = only the first solution is useful (Note 3)
 2 = there are two useful solutions (Note 3)

NOTES :

1. The **raz1** and **raz2** values returned are in the range $0-2\pi$.
2. 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 ξ value, and hence it is meaningless to ask where the tangent point would have to be to bring about this combination of ξ and δ .
3. 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 “beyond the pole” case, and this is reflected in the values of **raz2** and **decz2** which are returned.
4. The **decz1** and **decz2** values returned are in the range $\pm\pi$, but in the ordinary, non-pole-crossing, case, the range is $\pm\pi/2$.
5. **ra**, **dec**, **raz1**, **decz1**, **raz2**, **decz2** are all in radians.
6. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
7. When working in $[x, y, z]$ rather than spherical coordinates, the equivalent Cartesian function **slaDtpv2c** is available.

slaDtpv2c	plate centre from $[\xi, \eta]$ and $[x, y, z]$	slaDtpv2c
------------------	---	------------------

ACTION : From the tangent plane coordinates of a star of known direction cosines, determine the direction cosines of the tangent point (double precision)

CALL : `slaDtpv2c(xi, eta, v, v01, v02, &n);`

GIVEN :

xi,eta	double	tangent plane coordinates of star (radians)
v	double[3]	direction cosines of star

RETURNED :

v01	double[3]	direction cosines of tangent point, solution 1
v02	double[3]	direction cosines of tangent point, solution 2
n	int*	number of solutions:
		0 = no solutions returned (Note 2)
		1 = only the first solution is useful (Note 3)
		2 = there are two useful solutions (Note 3)

NOTES :

1. The vector **v** must be of unit length or the result will be wrong.
2. 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.
3. 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 can be regarded as valid if the vector **v02** is interpreted as the “beyond the pole” case.
4. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
5. This function is the Cartesian equivalent of the function **slaDtps2c**.

slaDtt	TT minus UTC	slaDtt
---------------	--------------	---------------

ACTION : Compute ΔTT , the increment to be applied to Coordinated Universal Time UTC to give Terrestrial Time TT.

CALL : `d = slaDtt(utc);`

GIVEN :

utc	double	UTC date/time (MJD, JD–2400000.5)
------------	---------------	-----------------------------------

RETURNED :

double	TT–UTC in seconds
---------------	-------------------

NOTES :

1. The UTC is specified to be a date rather than a time to indicate that care needs to be taken not to specify an instant which lies within a leap second. Though in most cases **utc** can include the fractional part, correct behaviour on the day of a leap second can be guaranteed only up to the end of the second 23^h 59^m 59^s.
2. Pre 1972 January 1 a fixed value of 10 + ET–TAI is returned.
3. TT is one interpretation of the defunct time-scale *Ephemeris Time*, ET.
4. See also the function **slaDt**, which roughly estimates ET–UT for historical epochs.

slaDv2tp	$[x, y, z]$ to tangent plane	slaDv2tp
-----------------	------------------------------	-----------------

ACTION : Given the direction cosines of a star and of the tangent point, determine the star's tangent-plane coordinates (double precision).

CALL : `slaDv2tp(v, v0, &xi, &eta, &j);`

GIVEN :

<code>v</code>	<code>double[3]</code>	direction cosines of star
<code>v0</code>	<code>double[3]</code>	direction cosines of tangent point

RETURNED :

<code>xi,eta</code>	<code>double*</code>	tangent plane coordinates (radians)
<code>j</code>	<code>int*</code>	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

NOTES :

1. If vector `v0` is not of unit length, or if vector `v` is of zero length, the results will be wrong.
2. If `v0` points at a pole, the returned ξ, η will be based on the arbitrary assumption that $\alpha = 0$ at the tangent point.
3. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
4. This function is the Cartesian equivalent of the function `slaDs2tp`.

slaDvdv	scalar product	slaDvdv
----------------	----------------	----------------

ACTION : Scalar product of two 3-vectors (double precision).

CALL : `d = slaDvdv(va, vb);`

GIVEN :

<code>va</code>	<code>double[3]</code>	first vector
<code>vb</code>	<code>double[3]</code>	second vector

RETURNED :

double scalar product $\mathbf{va} \cdot \mathbf{vb}$

slaDvn normalize vector **slaDvn**

ACTION : Normalize a 3-vector, also giving the modulus (double precision).

CALL : slaDvn(v, uv, &vm);

GIVEN :

v double[3] vector

RETURNED :

uv double[3] unit vector $\hat{\mathbf{v}}$
vm double* modulus $|\mathbf{v}|$

NOTE : If the modulus of \mathbf{v} is zero, \mathbf{uv} is set to the null vector.

slaDvxv vector product **slaDvxv**

ACTION : Vector product of two 3-vectors (double precision).

CALL : slaDvxv(va, vb, vc);

GIVEN :

va double[3] first vector
vb double[3] second vector

RETURNED :

vc double[3] vector product $\mathbf{va} \times \mathbf{vb}$

slaE2h $[h, \delta]$ to $[Az, El]$ **slaE2h**

ACTION : Equatorial to horizon coordinates (single precision).

CALL : slaDe2h(ha, dec, phi, &az, &el);

GIVEN :

ha	float	hour angle (radians)
dec	float	declination (radians)
phi	float	latitude (radians)

RETURNED :

az	float*	azimuth (radians)
el	float*	elevation (radians)

NOTES :

1. Azimuth is returned in the range $0-2\pi$; north is zero, and east is $+\pi/2$. Elevation is returned in the range $\pm\pi$.
2. The latitude must be geodetic. In critical applications, corrections for polar motion should be applied.
3. 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* $[h, \delta]$, and the elevation *in vacuo*, which would require the *topocentric* $[h, \delta]$. If the effects of diurnal aberration can be neglected, the *apparent* $[h, \delta]$ may be used instead of the topocentric $[h, \delta]$.
4. No range checking of arguments is carried out.
5. In applications which involve many such calculations, rather than calling the present function 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.

slaEarth	approximate Earth position and velocity	slaEarth
-----------------	---	-----------------

ACTION : Approximate heliocentric position and velocity of the Earth (single precision).

CALL : slaEarth(iy, id, fd, pv);

GIVEN :

iy	int	year
id	int	day in year (1 = Jan 1st)
fd	float	fraction of day

RETURNED :

pv	float[6]	Earth $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ (AU, AU s ⁻¹)
----	----------	--

NOTES :

1. The date and time is TDB (TT, or even UTC, will do) in a Julian calendar which has been aligned to the ordinary Gregorian calendar for the interval 1900 March 1 to 2100 February 28. The year and day can be obtained by calling `slaCalyd` or `slaClyd`.
2. The Earth heliocentric 6-vector is referred to the FK4 mean equator and equinox of date.
3. Maximum/RMS errors 1950-2050:
 - $13/5 \times 10^{-5}$ AU = 19200/7600 km in position
 - $47/26 \times 10^{-10}$ AU s⁻¹ = 0.0070/0.0039 km s⁻¹ in speed
4. More accurate results are obtainable with the functions `slaEvp` and `slaEvpv`.

slaEclev	ecliptic to equatorial	slaEclev
-----------------	------------------------	-----------------

ACTION : Transformation from ecliptic longitude and latitude to J2000.0 $[\alpha, \delta]$.

CALL : `slaEclev(dl, db, date, &dr, &dd);`

GIVEN :

<code>dl,db</code>	<code>double</code>	ecliptic longitude and latitude (mean of date, IAU 1980 theory, radians)
<code>date</code>	<code>double</code>	TDB as Modified Julian Date (JD-2400000.5)

RETURNED :

<code>dr,dd</code>	<code>double*</code>	J2000.0 mean $[\alpha, \delta]$ (radians)
--------------------	----------------------	---

NOTE : TT, or even UTC, can be used instead of TDB.

slaEcmat	form $[\alpha, \delta] \rightarrow [\lambda, \beta]$ matrix	slaEcmat
-----------------	---	-----------------

ACTION : Form the equatorial to ecliptic rotation matrix (IAU 1980 theory).

CALL : `slaEcmat(date, rmat);`

GIVEN :

<code>date</code>	<code>double</code>	TDB as Modified Julian Date (JD-2400000.5)
-------------------	---------------------	--

RETURNED :

<code>rmat</code>	<code>double[3][3]</code>	rotation matrix
-------------------	---------------------------	-----------------

NOTES :

1. The distinction between the required TDB and TT (or even UTC) is always negligible.
2. `rmat` is matrix \mathbf{M} in the expression $\mathbf{v}_{ecl} = \mathbf{M} \cdot \mathbf{v}_{equ}$.
3. The equator, equinox and ecliptic are mean of date.

REFERENCE : Murray, C.A., *Vectorial Astrometry*, section 4.3.

slaEcor	radial velocity & light-time corrections to Sun	slaEcor
----------------	---	----------------

ACTION : Component of Earth orbit velocity and heliocentric light time in a given direction.

CALL : `slaEcor(rm, dm, iy, id, fd, &rv, &tl);`

GIVEN :

<code>rm, dm</code>	<code>float</code>	mean $[\alpha, \delta]$ of date (radians)
<code>iy</code>	<code>int</code>	year
<code>id</code>	<code>int</code>	day in year (1 = Jan 1st)
<code>fd</code>	<code>float</code>	fraction of day

RETURNED :

<code>rv</code>	<code>float*</code>	component of Earth orbital velocity (km s^{-1})
<code>tl</code>	<code>float*</code>	component of heliocentric light time (s)

NOTES :

1. The date and time is TDB (TT, or even UTC, will do) in a Julian calendar which has been aligned to the ordinary Gregorian calendar for the interval 1900 March 1 to 2100 February 28. The year and day can be obtained by calling `slaCalyd` or `slaClyd`.
2. Sign convention:
 - The velocity component is positive when the Earth is receding from the given point on the sky.
 - The light time component is positive when the Earth lies between the Sun and the given point on the sky.
3. Accuracy:

- The velocity component is usually within 0.004 km s^{-1} of the correct value and is never in error by more than 0.007 km s^{-1} .
- The error in light time correction is about $0^{\circ}03$ at worst, but is usually better than $0^{\circ}01$.

For applications requiring higher accuracy, see the `slaEvp` and `slaEpv` functions.

slaEg50	B1950 $[\alpha, \delta]$ to $[l^II, b^II]$	slaEg50
----------------	--	----------------

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

CALL : `slaEg50(dr, dd, &dl, &db);`

GIVEN :

<code>dr, dd</code>	<code>double</code>	B1950.0 $[\alpha, \delta]$ (radians)
---------------------	---------------------	--------------------------------------

RETURNED :

<code>dl, db</code>	<code>double*</code>	galactic longitude and latitude $[l^II, b^II]$ (radians)
---------------------	----------------------	--

NOTE : The equatorial coordinates are B1950.0 FK4. Use the function `slaEqgal` if transformation from J2000.0 FK5 coordinates is required.

REFERENCE : Blaauw *et al.*, 1960, *Mon.Not.R.astr.Soc.*, **121**, 123.

slaEl2ue	conventional to universal elements	slaEl2ue
-----------------	------------------------------------	-----------------

ACTION : Transform conventional osculating orbital elements into “universal” form.

CALL : `slaEl2ue(date, jform, epoch, orbinc, anode,
 perih, aorq, e, aorl, dm,
 u, &jstat);`

GIVEN :

<code>date</code>	<code>double</code>	epoch (TDB MJD) of osculation (Note 3)
<code>jform</code>	<code>int</code>	choice of element set (1-3; Note 6)
<code>epoch</code>	<code>double</code>	epoch of elements (t_0 or T , TDB MJD)
<code>orbinc</code>	<code>double</code>	inclination (i , radians)
<code>anode</code>	<code>double</code>	longitude of the ascending node (Ω , radians)
<code>perih</code>	<code>double</code>	longitude or argument of perihelion (ϖ or ω , radians)
<code>aorq</code>	<code>double</code>	mean distance or perihelion distance (a or q , AU)
<code>e</code>	<code>double</code>	eccentricity (e)
<code>aorl</code>	<code>double</code>	mean anomaly or longitude (M or L , radians, <code>jform</code> = 1,2 only)
<code>dm</code>	<code>double</code>	daily motion (n , radians, <code>jform</code> = 1 only)

RETURNED :

<code>u</code>	<code>double[13]</code>	universal orbital elements (Note 1)
	[0]	◦ combined mass ($M + m$)
	[1]	◦ total energy of the orbit (α)
	[2]	◦ reference (osculating) epoch (t_0)
	[3-5]	◦ position at reference epoch (\mathbf{r}_0)
	[6-8]	◦ velocity at reference epoch (\mathbf{v}_0)
	[9]	◦ heliocentric distance at reference epoch
	[10]	◦ $\mathbf{r}_0 \cdot \mathbf{v}_0$
	[11]	◦ date (t)
	[12]	◦ universal eccentric anomaly (ψ) of date, approx.

<code>jstat</code>	<code>int*</code>	status:
		0 = OK
		-1 = illegal <code>jform</code>
		-2 = illegal <code>e</code>
		-3 = illegal <code>aorq</code>
		-4 = illegal <code>dm</code>
		-5 = numerical error

NOTES :

1. 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) α , 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 ψ , the “universal eccentric anomaly” at a given date and (v) that date.
2. The companion function is `slaUe2pv`. This takes the set of numbers that the present function outputs and uses them to derive the object’s position and velocity. A single prediction requires one call to the present function followed by one call to `slaUe2pv`; for convenience, the two calls are packaged as the function `slaPlane1`. Multiple predictions may be made by again calling the present function once, but then calling `slaUe2pv` multiple times, which is faster than multiple calls to `slaPlane1`.
 3. `date` is the epoch of osculation. It is in the TDB time-scale (TT will do) and is a Modified Julian Date (JD–2400000.5).
 4. 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.
 5. The universal elements returned in the array `u` are in canonical units (solar masses, AU and canonical days).
 6. Three different element-format options are supported, as follows.

`jform = 1`, suitable for the major planets:

<code>epoch</code>	= epoch of elements t_0 (TDB MJD)
<code>orbinc</code>	= inclination i (radians)
<code>anode</code>	= longitude of the ascending node Ω (radians)
<code>perih</code>	= longitude of perihelion ϖ (radians)
<code>aorq</code>	= mean distance a (AU)
<code>e</code>	= eccentricity e ($0 \leq e < 1$)
<code>aorl</code>	= mean longitude L (radians)
<code>dm</code>	= daily motion n (radians)

`jform = 2`, suitable for minor planets:

<code>epoch</code>	= epoch of elements t_0 (TDB MJD)
<code>orbinc</code>	= inclination i (radians)
<code>anode</code>	= longitude of the ascending node Ω (radians)
<code>perih</code>	= argument of perihelion ω (radians)
<code>aorq</code>	= mean distance a (AU)
<code>e</code>	= eccentricity e ($0 \leq e < 1$)
<code>aorl</code>	= mean anomaly M (radians)

`jform = 3`, suitable for comets:

<code>epoch</code>	= epoch of perihelion T (TDB MJD)
<code>orbinc</code>	= inclination i (radians)
<code>anode</code>	= longitude of the ascending node Ω (radians)
<code>perih</code>	= argument of perihelion ω (radians)
<code>aorq</code>	= perihelion distance q (AU)
<code>e</code>	= eccentricity e ($0 \leq e \leq 10$)

7. Unused elements (`dm` for `jform = 2`, `aorl` and `dm` for `jform = 3`) are not accessed.

8. The algorithm was originally adapted from the **EPHSLA** Fortran program of D. H. P. Jones (private communication, 1996). The method is based on Stumpff's Universal Variables.

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

slaEo	equation of the origins.	slaEo
--------------	--------------------------	--------------

ACTION : Equation of the origins.

CALL : `d = slaEo(date);`

GIVEN :

date	double	TDB Modified Julian Date
------	--------	--------------------------

RETURNED :

double	the equation of the origins (radians)
--------	---------------------------------------

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). TT will do.
2. The equation of the origins is the distance between the true equinox and the celestial intermediate origin and, equivalently, the difference between Earth rotation angle and Greenwich apparent sidereal time (ERA–GST). It comprises the precession (since J2000.0) in right ascension plus the equation of the equinoxes (including the small correction terms).
3. To transform a geocentric apparent right ascension into the equivalent right ascension in the celestial intermediate reference system (CIRS), add the equation of the origins.

REFERENCE : Capitaine, N. & Wallace, P.T. 2006, *Astron.Astrophys.* 459, 981.

slaEors	EO from NPB matrix and <i>s</i>	slaEors
----------------	---------------------------------	----------------

ACTION : Equation of the origins, given the classical NPB matrix and the quantity *s*.

CALL : `d = slaEors(rnpb, s);`

GIVEN :

rnpb	double [3] [3]	classical NPB matrix
s	double	the quantity s (the CIO locator)

RETURNED :

double	the equation of the origins (radians)
--------	---------------------------------------

NOTES :

1. The equation of the origins is the distance between the true equinox and the celestial intermediate origin and, equivalently, the difference between Earth rotation angle and Greenwich apparent sidereal time (ERA–GST). It comprises the precession (since J2000.0) in right ascension plus the equation of the equinoxes (including the small correction terms).
2. To transform a geocentric apparent right ascension into the equivalent right ascension in the celestial intermediate reference system (CIRS), add the equation of the origins.
3. The algorithm is from Wallace & Capitaine (2006).

REFERENCE : Wallace, P.T. & Capitaine, N. 2006, *Astron.Astrophys.* 459, 981.

slaEpb	MJD to Besselian epoch	slaEpb
---------------	------------------------	---------------

ACTION : Transformation of Modified Julian Date to Besselian Epoch.

CALL : d = slaEpb(date);

GIVEN :

date	double	Modified Julian Date (JD–2400000.5)
------	--------	-------------------------------------

RETURNED :

double	Besselian Epoch
--------	-----------------

REFERENCE : Lieske, J.H. 1979, *Astr.Astrophys.* **73**, 282.

slaEpb2d	Besselian epoch to MJD	slaEpb2d
-----------------	------------------------	-----------------

ACTION : Transformation of Besselian Epoch to Modified Julian Date.

CALL : d = slaEpb2d(epb);

GIVEN :

epb	double	Besselian Epoch
-----	--------	-----------------

RETURNED :

double	Modified Julian Date (JD−2400000.5)
--------	-------------------------------------

REFERENCE : Lieske, J.H. 1979, *Astr.Astrophys.* **73**, 282.

slaEpco	convert epoch to B or J	slaEpco
----------------	-------------------------	----------------

ACTION : Convert an epoch to Besselian or Julian to match another one.

CALL : d = slaEpco(k0, k, e);

GIVEN :

k0	char*	form of result: 'B' = Besselian, 'J' = Julian
k	char*	form of given epoch: 'B' or 'J'
e	double	epoch

RETURNED :

double	the given epoch converted as necessary
--------	--

NOTES :

1. The result is always either equal to or very close to the given epoch **e**. The function is required only in applications where punctilious treatment of heterogeneous mixtures of star positions is necessary.
2. **k0** and **k** are not validated. They are interpreted as follows:
 - If **k0** and **k** are the same, the result is **e**.
 - If **k0** is 'B' and **k** isn't, the conversion is J to B.
 - In all other cases, the conversion is B to J.

slaEpj	MJD to Julian epoch	slaEpj
---------------	---------------------	---------------

ACTION : Convert Modified Julian Date to Julian Epoch.

CALL : d = slaEpj(date);

GIVEN :

date	double	Modified Julian Date (JD−2400000.5)
------	--------	-------------------------------------

RETURNED :

double	Julian Epoch
--------	--------------

REFERENCE : Lieske, J.H. 1979, *Astr.Astrophys.*, **73**, 282.

slaEpj2d	Julian epoch to MJD	slaEpj2d
-----------------	---------------------	-----------------

ACTION : Convert Julian Epoch to Modified Julian Date.

CALL : d = slaEpj2d(epj);

GIVEN :

epj	double	Julian Epoch
-----	--------	--------------

RETURNED :

double	Modified Julian Date (JD−2400000.5)
--------	-------------------------------------

REFERENCE : Lieske, J.H. 1979, *Astr.Astrophys.*, **73**, 282.

slaEpv	Earth position & velocity (high accuracy)	slaEpv
---------------	---	---------------

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

CALL : slaEpv(date, ph, vh, pb, vb);

GIVEN :

date	double	TDB Modified Julian Date (Note 1)
------	--------	-----------------------------------

RETURNED :

ph	double[3]	heliocentric $[x, y, z]$, AU
vh	double[3]	heliocentric $[\dot{x}, \dot{y}, \dot{z}]$, AU d ⁻¹
pb	double[3]	barycentric $[x, y, z]$, AU
vb	double[3]	barycentric $[\dot{x}, \dot{y}, \dot{z}]$, AU d ⁻¹

NOTES :

1. The date is TDB as MJD (= JD−2400000.5). TT can be used instead of TDB in most applications.
2. The vectors are with respect to the Barycentric Celestial Reference System (BCRS). Positions are in AU; velocities are in AU per TDB day.
3. The function is a *simplified solution* from the planetary theory VSOP2000 (X. Moisson, P. Bretagnon, 2001, Celes. Mechanics & Dyn. Astron., **80**, 3/4, 205-213) and is an adaptation of original Fortran code supplied by P. Bretagnon (private communication, 2000).
4. Comparisons over the time span 1900-2100 with this simplified solution and the JPL DE405 ephemeris give the following results:

	<i>RMS</i>	<i>max</i>	
<i>Heliocentric:</i>			
position error	3.7	11.2	km
velocity error	1.4	5.0	mm/s
<i>Barycentric:</i>			
position error	4.6	13.4	km
velocity error	1.4	4.9	mm/s

The results deteriorate outside this time span. Comparisons with DE406 show almost double the position errors by 1800 and 2200, a factor of ten by 1500 and 2500 and a factor of 60 by 1000 and 3000. The velocity accuracy falls off at about half that rate.

5. The function `slaEvp` is faster but less accurate. The present function targets the case where high accuracy is more important than CPU time, yet the extra complication of reading a pre-computed ephemeris is not justified.

slaEqecl	J2000 $[\alpha, \delta]$ to $[\lambda, \beta]$	slaEqecl
-----------------	--	-----------------

ACTION : Transformation from J2000.0 equatorial coordinates to ecliptic longitude and latitude.

CALL : `slaEqecl(dr, dd, date, &dl, &db);`

GIVEN :

dr,dd	double	J2000.0 mean $[\alpha, \delta]$ (radians)
date	double	TDB as Modified Julian Date (JD−2400000.5)

RETURNED :

dl,db	double*	ecliptic longitude and latitude (mean of date, IAU 1980 theory, radians)
-------	---------	---

NOTE : TT, or even UTC, can be used instead of TDB.

slaEgeqx	equation of the equinoxes	slaEgeqx
-----------------	---------------------------	-----------------

ACTION : Equation of the equinoxes (IAU 1994).

CALL : d = slaEgeqx(date);

GIVEN :

date	double	TDB as Modified Julian Date (JD−2400000.5)
------	--------	--

RETURNED :

double	The equation of the equinoxes (radians)
--------	---

NOTES :

1. The equation of the equinoxes is defined here as GAST−GMST: it is added to a *mean* sidereal time to give the *apparent* sidereal time.
2. The change from the classic “textbook” expression $\Delta\psi \cos \epsilon$ occurred with IAU Resolution C7, Recommendation 3 (1994). The new formulation takes into account cross-terms between the various precession and nutation quantities, amounting to about 3 milliarcsec. The transition from the old to the new model officially took place on 1997 February 27.
3. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.

REFERENCE : Capitaine, N. & Gontier, A.-M. 1993, *Astron.Astrophys.* **275**, 645-650.

slaEqgal	J2000 [α, δ] to [l^II, b^II]	slaEqgal
-----------------	--	-----------------

ACTION : Transformation from J2000.0 FK5 equatorial coordinates to IAU 1958 galactic coordinates.

CALL : slaEqgal(dr, dd, &dl, &db);

GIVEN :

<code>dr,dd</code>	<code>double</code>	J2000.0 $[\alpha, \delta]$ (radians)
--------------------	---------------------	--------------------------------------

RETURNED :

<code>dl,db</code>	<code>double*</code>	galactic longitude and latitude $[l^II, b^II]$ (radians)
--------------------	----------------------	--

NOTE : The equatorial coordinates are J2000.0 FK5. Use the function `slaEg50` if transformation from B1950.0 FK4 coordinates is required.

REFERENCE : Blaauw *et al.*, 1960, *Mon.Not.R.astr.Soc.*, **121**, 123.

slaEra	Earth rotation angle	slaEra
---------------	----------------------	---------------

ACTION : Earth rotation angle.

CALL : `d = slaEra(date, ut);`

GIVEN :

<code>date</code>	<code>double</code>	UT1 date (MJD: integer part of $JD - 2400000.5$)
<code>ut</code>	<code>double</code>	UT1 time (fraction of a day)

RETURNED :

<code>double</code>	Earth rotation angle (radians), in the range $0 - 2\pi$.
---------------------	---

NOTES :

1. There is, in fact, no restriction on how the UT is apportioned between the `date` and `ut` arguments. Either of the two arguments could, for example, be zero and the entire date+time supplied in the other. However, the function is designed to deliver maximum accuracy when the `date` argument is a whole number and the `ut` argument lies in the range 0 to 1, or *vice versa*.
2. The algorithm is adapted from Expression 22 of Capitaine *et al.* 2000. The time argument has been expressed in days directly, and, to retain precision, integer contributions have been eliminated at the earliest opportunity.

REFERENCE : Capitaine, N., Guinot, B. & McCarthy, D.D. 2000, *Astron.Astrophys.* **355**, 398-405

slaEtrms	E-terms of aberration	slaEtrms
-----------------	-----------------------	-----------------

ACTION : Compute the E-terms vector – the part of the annual aberration which arises from the eccentricity of the Earth’s orbit.

CALL : `slaEtrms(ep, ev);`

GIVEN :

ep	double	Besselian epoch
----	--------	-----------------

RETURNED :

ev	double[3]	E-terms as $[\Delta x, \Delta y, \Delta z]$
----	-----------	---

NOTE : Note the use of the J2000 aberration constant (20".49552). 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.

REFERENCES :

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

slaEuler	rotation matrix from Euler angles	slaEuler
-----------------	-----------------------------------	-----------------

ACTION : Form a rotation matrix from the Euler angles – three successive rotations about specified Cartesian axes (single precision).

CALL : `slaEuler(order, phi, theta, psi, rmat);`

GIVEN :

order	char*	specifies about which axes the rotations occur
phi	float	first rotation (radians)
theta	float	second rotation (radians)
psi	float	third rotation (radians)

RETURNED :

<code>rmat</code>	<code>float [3] [3]</code>	rotation matrix
-------------------	----------------------------	-----------------

NOTES :

1. The characters of **order** define which axes the three successive rotations are about. A typical value is "zxx", 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.
2. The axis names can be any of the following, in any order or combination: 'X', 'Y', 'Z', uppercase or lowercase, '1', '2', '3'. Thus, the "zxx" example given above could be written "ZXZ" or "313" (or even "ZxZ" or "3xZ").
3. **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. Zero rotations produces the identity **rmat**.
4. Normal axis labelling/numbering conventions apply; the *xyz* (\equiv "123") triad is right-handed.
5. A rotation is positive when the reference frame rotates anticlockwise as seen looking towards the origin from the positive region of the specified axis.

slaEvp	Earth position & velocity	slaEvp
---------------	---------------------------	---------------

ACTION : Barycentric and heliocentric velocity and position of the Earth.

CALL : `slaEvp(date, deqx, dvb, dpb, dvh, dph);`

GIVEN :

<code>date</code>	<code>double</code>	TDB as a Modified Julian Date (JD−2400000.5)
<code>deqx</code>	<code>double</code>	Julian Epoch (<i>e.g.</i> 2000.0) of mean equator and equinox of the vectors returned. If DEQX < 0, all vectors are referred to the mean equator and equinox (FK5) of date DATE.

RETURNED :

<code>dvb</code>	<code>double [3]</code>	barycentric $[\dot{x}, \dot{y}, \dot{z}]$, AU s ^{−1}
<code>dpb</code>	<code>double [3]</code>	barycentric $[x, y, z]$, AU
<code>dvh</code>	<code>double [3]</code>	heliocentric $[\dot{x}, \dot{y}, \dot{z}]$, AU s ^{−1}
<code>dph</code>	<code>double [3]</code>	heliocentric $[x, y, z]$, AU

NOTES :

1. This function is accurate enough for many purposes but faster and more compact than the `slaEpv` function. The maximum deviations from the JPL DE96 ephemeris are as follows:
 - velocity (barycentric or heliocentric): 420 mm s^{-1}
 - position (barycentric): 6900 km
 - position (heliocentric): 1600 km
2. The function is adapted from the `BARVEL` and `BARCOR` subroutines of Stumpff (1980). Most of the changes are merely cosmetic and do not affect the results at all. However, some adjustments have been made so as to give results that refer to the IAU 1976 ‘FK5’ equinox and precession, although the differences these changes make relative to the results from Stumpff’s original ‘FK4’ version are smaller than the inherent accuracy of the algorithm. One minor shortcoming in the original functions that has **not** been corrected is that slightly better numerical accuracy could be achieved if the various polynomial evaluations were to be so arranged that the smallest terms were computed first.
3. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.

REFERENCE : Stumpff, P., 1980., *Astron.Astrophys.Suppl.Ser.* **41**, 1-8.

slaFitxy	fit linear model to two $[x, y]$ sets	slaFitxy
-----------------	---------------------------------------	-----------------

ACTION : Fit a linear model to relate two sets of $[x, y]$ coordinates.

CALL : `slaFitxy(itype, np, xye, xym, coeffs, &j);`

GIVEN :

<code>itype</code>	<code>int</code>	type of model: 4 or 6 (Note 1)
<code>np</code>	<code>int</code>	number of samples (Note 2)
<code>xye</code>	<code>double[2] [np]</code>	expected $[x, y]$ for each sample
<code>xym</code>	<code>double[2] [np]</code>	measured $[x, y]$ for each sample

RETURNED :

<code>coeffs</code>	<code>double[6]</code>	coefficients of model (Note 3)
<code>j</code>	<code>int*</code>	status:
		0 = OK
		-1 = illegal <code>itype</code>
		-2 = insufficient data
		-3 = singular solution

NOTES :

1. `itype`, which must be either 4 or 6, selects the type of model fitted. Both allowed `itype` values produce a model `coeffs` which consists of six coefficients, namely the zero points and, for each of `xe` and `ye`, the coefficient of `xm` and `ym`. For `itype`=6, all six coefficients are independent, modelling squash and shear as well as origin, scale, and orientation. However, `itype`=4 selects the *solid body rotation* option; the model `coeffs` still consists of the same six coefficients, but now two of them are used twice (appropriately signed). Origin, scale and orientation are still modelled, but not squash or shear – the units of x and y have to be the same.
2. For `nc`=4, `np` must be at least 2. For `nc`=6, `np` must be at least 3.
3. The model is returned in the array `coeffs`. Writing x_m *etc.* for `xm` *etc.* and naming the six elements of `coeffs` a, b, c, d, e & f , the model transforms *measured* coordinates $[x_m, y_m]$ into *expected* coordinates $[x_e, y_e]$ as follows:

$$\begin{aligned}x_e &= a + bx_m + cy_m \\ y_e &= d + ex_m + fy_m\end{aligned}$$

For the *solid body rotation* option (`itype`=4), the magnitudes of b and f , and of c and e , are equal. The signs of these coefficients depend on whether there is a sign reversal between $[x_e, y_e]$ and $[x_m, y_m]$; fits are performed with and without a sign reversal and the best one chosen.

4. Error status values $j = -1$ and -2 leave `coeffs` unchanged; if $j = -3$ `coeffs` may have been changed.
5. See also `slaPxy`, `slaInvf`, `slaXy2xy`, `slaDcmpf`.

slaFk425

FK4 to FK5

slaFk425

ACTION : Convert B1950.0 FK4 star data to J2000.0 FK5. This function converts stars from the old, Bessel-Newcomb, FK4 system to the new, IAU 1976, FK5, Fricke system. The precepts of Smith *et al.* (see reference 1) are followed, using the implementation by Yallop *et al.* (reference 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.* (reference 3) are used canonically.

CALL : `slaFk425(r1950, d1950, dr1950, dd1950, p1950, v1950,`
`&r2000, &d2000, &dr2000, &dd2000, &p2000, &v2000);`

GIVEN :

r1950	double	B1950.0 α (radians)
d1950	double	B1950.0 δ (radians)
dr1950	double	B1950.0 proper motion in α (radians per tropical year)
dd1950	double	B1950.0 proper motion in δ (radians per tropical year)
p1950	double	B1950.0 parallax (arcsec)
v1950	double	B1950.0 radial velocity (km s^{-1} , +ve = moving away)

RETURNED :

r2000	double*	J2000.0 α (radians)
d2000	double*	J2000.0 δ (radians)
dr2000	double*	J2000.0 proper motion in α (radians per Julian year)
dd2000	double*	J2000.0 proper motion in δ (radians per Julian year)
p2000	double*	J2000.0 parallax (arcsec)
v2000	double*	J2000.0 radial velocity (km s^{-1} , +ve = moving away)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
2. Transformation from Besselian epoch 1950.0 to Julian epoch 2000.0 only is provided for. Transformations involving other epochs will require use of the appropriate precession, proper motion, and E-terms functions before and/or after `slaFk425` is called.
3. In the FK4 catalogue the proper motions of stars within 10° of the poles do not include the *differential E-terms* 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 function to include the effect of differential E-terms on the proper motions for all stars, whether polar or not. At epoch J2000, and measuring on the sky rather than in terms of $\Delta\alpha$, the errors resulting from this simplification are less than 1 milliarcsecond in position and 1 milliarcsecond per century in proper motion.
4. See also `slaFk45z`, `slaFk524`, `slaFk54z`.

REFERENCES :

1. Smith, C.A. *et al.*, 1989, *Astr.J.* **97**, 265.
2. Yallop, B.D. *et al.*, 1989, *Astr.J.* **97**, 274.
3. Seidelmann, P.K. (ed), 1992, *Explanatory Supplement to the Astronomical Almanac*, ISBN 0-935702-68-7.

slaFk45z FK4 to FK5, no proper motion or parallax **slaFk45z**

ACTION : Convert B1950.0 FK4 star data to J2000.0 FK5 assuming zero proper motion in the FK5 system. This function converts stars from the old, Bessel-Newcomb, FK4 system to the new, 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 function requires the epoch at which the position in the FK4 system was determined. The method is from appendix 2 of reference 1, but using the constants of reference 4.

CALL : `slaFk45z(r1950, d1950, bePOCH, &r2000, &d2000);`

GIVEN :

<code>r1950</code>	<code>double</code>	B1950.0 FK4 α at epoch BEPOCH (radians)
<code>d1950</code>	<code>double</code>	B1950.0 FK4 δ at epoch BEPOCH (radians)
<code>bePOCH</code>	<code>double</code>	Besselian epoch (<i>e.g.</i> 1979.3)

RETURNED :

<code>r2000</code>	<code>double*</code>	J2000.0 FK5 α (radians)
<code>d2000</code>	<code>double*</code>	J2000.0 FK5 δ (radians)

NOTES :

1. The epoch `bePOCH` is strictly speaking Besselian, but if a Julian epoch is supplied the result will be affected only to a negligible extent.
2. Transformation from Besselian epoch 1950.0 to Julian epoch 2000.0 only is provided for. Transformations involving other epochs will require use of the appropriate precession, proper motion, and E-terms functions before and/or after `slaFk45z` is called.
3. In the FK4 catalogue the proper motions of stars within 10° of the poles do not include the *differential E-terms* 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 function 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 $\Delta\alpha$, the errors resulting from this simplification are less than 1 milliarcsecond in position and 1 milliarcsecond per century in proper motion.

4. See also `slaFk425`, `slaFk524`, `slaFk54z`.

REFERENCES :

1. Aoki, S., *et al.*, 1983, *Astr.Astrophys.*, **128**, 263.
2. Smith, C.A. *et al.*, 1989, *Astr.J.* **97**, 265.
3. Yallop, B.D. *et al.*, 1989, *Astr.J.* **97**, 274.
4. Seidelmann, P.K. (ed), 1992, *Explanatory Supplement to the Astronomical Almanac*, ISBN 0-935702-68-7.

slaFk524	FK5 to FK4	slaFk524
-----------------	------------	-----------------

ACTION : Convert J2000.0 FK5 star data to B1950.0 FK4. This function converts stars from the new, IAU 1976, FK5, Fricke system, to the old, Bessel-Newcomb, FK4 system. The precepts of Smith *et al.* (reference 1) are followed, using the implementation by Yallop *et al.* (reference 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.* (reference 3) are used canonically.

CALL : `slaFk524(r2000, d2000, dr2000, dd2000, p2000, v2000,`
`&r1950, &d1950, &dr1950, &dd1950, &p1950, &v1950);`

GIVEN :

<code>r2000</code>	<code>double</code>	J2000.0 α (radians)
<code>d2000</code>	<code>double</code>	J2000.0 δ (radians)
<code>dr2000</code>	<code>double</code>	J2000.0 proper motion in α (radians per Julian year)
<code>dd2000</code>	<code>double</code>	J2000.0 proper motion in δ (radians per Julian year)
<code>p2000</code>	<code>double</code>	J2000.0 parallax (arcsec)
<code>v2000</code>	<code>double</code>	J2000 radial velocity (km s ⁻¹ , +ve = moving away)

RETURNED :

r1950	double*	B1950.0 α (radians)
d1950	double*	B1950.0 δ (radians)
dr1950	double*	B1950.0 proper motion in α (radians per tropical year)
dd1950	double*	B1950.0 proper motion in δ (radians per tropical year)
p1950	double*	B1950.0 parallax (arcsec)
v1950	double*	radial velocity (km s^{-1} , +ve = moving away)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
2. Note that transformation from Julian epoch 2000.0 to Besselian epoch 1950.0 only is provided for. Transformations involving other epochs will require use of the appropriate precession, proper motion, and E-terms functions before and/or after `slaFk524` is called.
3. In the FK4 catalogue the proper motions of stars within 10° of the poles do not include the *differential E-terms* 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 function 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 $\Delta\alpha$, the errors resulting from this simplification are less than 1 milliarcsecond in position and 1 milliarcsecond per century in proper motion.
4. See also `slaFk425`, `slaFk45z`, `slaFk54z`.

REFERENCES :

1. Smith, C.A. *et al.*, 1989, *Astr.J.* **97**, 265.
2. Yallop, B.D. *et al.*, 1989, *Astr.J.* **97**, 274.
3. Seidelmann, P.K. (ed), 1992, *Explanatory Supplement to the Astronomical Almanac*, ISBN 0-935702-68-7.

slaFk52h
FK5 to Hipparcos
slaFk52h

ACTION : Transform an FK5 (J2000) position and proper motion into the system of the Hipparcos catalogue.

CALL : `slaFk52h(r5, d5, dr5, dd5, &rh, &dh, &drh, &ddh);`

GIVEN :

r5	double	J2000.0 FK5 α (radians)
d5	double	J2000.0 FK5 δ (radians)
dr5	double	J2000.0 FK5 proper motion in α (radians per Julian year)
dd5	double	J2000.0 FK5 proper motion in δ (radians per Julian year)

RETURNED :

rh	double*	Hipparcos α (radians)
dh	double*	Hipparcos δ (radians)
drh	double*	Hipparcos proper motion in α (radians per Julian year)
ddh	double*	Hipparcos proper motion in δ (radians per Julian year)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	−19.9	−0.30
<i>y</i>	−9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians.

4. See also `slaFk5hz`, `slaH2fk5`, `slaHfk5z`.

REFERENCE : Feissel, M. & Mignard, F., 1998, *Astron.Astrophys.* **331**, L33-L36.

slaFk54z FK5 to FK4, no proper motion or parallax **slaFk54z**

ACTION : Convert a J2000.0 FK5 star position to B1950.0 FK4 assuming FK5 zero proper motion and parallax. This function converts star positions from the new, IAU 1976, FK5, Fricke system to the old, Bessel-Newcomb, FK4 system.

CALL : `slaFk54z(r2000, d2000, bepoch, &r1950, &d1950, &dr1950, &dd1950);`

GIVEN :

<code>r2000</code>	<code>double</code>	J2000.0 FK5 α (radians)
<code>d2000</code>	<code>double</code>	J2000.0 FK5 δ (radians)
<code>bepoch</code>	<code>double</code>	Besselian epoch (<i>e.g.</i> 1950.0)

RETURNED :

<code>r1950</code>	<code>double*</code>	B1950.0 FK4 α at epoch BEPOCH (radians)
<code>d1950</code>	<code>double*</code>	B1950.0 FK4 δ at epoch BEPOCH (radians)
<code>dr1950</code>	<code>double*</code>	B1950.0 FK4 proper motion in α (radians per tropical year)
<code>dd1950</code>	<code>double*</code>	B1950.0 FK4 proper motion in δ (radians per tropical year)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
2. Transformation from Julian epoch 2000.0 to Besselian epoch 1950.0 only is provided for. Transformations involving other epochs will require use of the appropriate precession functions before and after this function is called.
3. Unlike in the `slaFk524` function, the FK5 proper motions, the parallax and the radial velocity are presumed zero.
4. It was 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*.
5. The position returned by this function is in the B1950 reference system but at Besselian epoch `bepoch`. For comparison with catalogues the `bepoch` argument will frequently be 1950.0.
6. See also `slaFk425`, `slaFk45z`, `slaFk524`.

slaFk5hz FK5 to Hipparcos, no proper motion **slaFk5hz**

ACTION : Transform an FK5 (J2000) star position into the system of the Hipparcos catalogue, assuming zero Hipparcos proper motion.

CALL : slaFk5hz(r5, d5, epoch, &rh, &dh);

GIVEN :

r5	double	J2000.0 FK5 α (radians)
d5	double	J2000.0 FK5 δ (radians)
epoch	double	Julian epoch (TDB)

RETURNED :

rh	double*	Hipparcos α (radians)
dh	double*	Hipparcos δ (radians)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	−19.9	−0.30
<i>y</i>	−9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians.

4. TT, or even UTC, can be used instead of TDB.
5. See also slaFk52h, slaH2fk5, slaHfk5z.

REFERENCE : Feissel, M. & Mignard, F., 1998, *Astron.Astrophys.* **331**, L33-L36.

slaFlotin decode a single-precision real number **slaFlotin**

ACTION : Convert free-format input into single precision floating point.

CALL : slaFlotin(string, &nstrt, &reslt, &jflag);

GIVEN :

string	char*	string containing number to be decoded
nstrt	int*	index to where decoding is to commence
reslt	float*	current value of result

RETURNED :

nstrt	int*	advanced to next number
reslt	float*	result
jflag	int*	status: -1 = -OK, 0 = +OK, 1 = null result, 2 = error

NOTES :

1. The reason **slaFlotin** has separate “OK” status values for ‘+’ and ‘-’ is to enable minus zero to be detected. This is of crucial importance when decoding mixed-radix numbers. For example, an angle expressed as degrees, arcminutes and arcseconds may have a leading minus sign but a zero degrees field.
2. A TAB is interpreted as a space, and lowercase characters are interpreted as uppercase
3. The basic format is the sequence of fields $\pm n.nx \pm n$, where \pm is a sign character ‘+’ or ‘-’, n means a string of decimal digits, ‘.’ is a decimal point, and x , which indicates an exponent, means ‘D’ or ‘E’. Various combinations of these fields can be omitted, and embedded blanks are permissible in certain places.
4. Spaces:
 - Leading spaces are ignored.
 - Embedded spaces are allowed only after +, -, D or E, and after the decimal point if the first sequence of digits is absent.
 - Trailing spaces are ignored; the first signifies end of decoding and subsequent ones are skipped.
5. Delimiters:
 - Any character other than +, -, 0-9, period, D, E or space may be used to signal the end of the number and terminate decoding.

- Comma is recognized by `slaFlotin` as a special case; it is skipped, leaving the index on the next character. See 13, below.
 - Decoding will in all cases terminate if end of string is reached.
6. Both signs are optional. The default is '+ '.
 7. The mantissa $n.n$ defaults to unity.
 8. The exponent $x \pm n$ defaults to "E0".
 9. The strings of decimal digits may be of any length.
 10. The decimal point is optional for whole numbers.
 11. A *null result* occurs when the string of characters being decoded does not begin with +, -, 0-9, period, D or E or consists entirely of spaces. When this condition is detected, `jflag` is set to 1 and `reslt` is left untouched.
 12. `nstrt` = 1 for the first character in the string.
 13. On return from `slaFlotin`, `nstrt` is set ready for the next decode – following trailing blanks and any comma. If a delimiter other than comma is being used, `nstrt` must be incremented before the next call to `slaFlotin`, otherwise all subsequent calls will return a null result.
 14. Errors (`jflag` = 2) occur when:
 - a +, -, D or E is left unsatisfied; or
 - the decimal point is present without at least one decimal digit before or after it; or
 - an exponent more than 100 has been presented.
 15. When an error has been detected, `nstrt` is left pointing to the character following the last one used before the error came to light. This may be after the point at which a more sophisticated program could have detected the error. For example, `slaFlotin` does not detect that "1e999" is unacceptable (on a platform where this is so) until the entire number has been decoded.
 16. Certain highly unlikely combinations of mantissa and exponent can cause arithmetic faults during the decode, in some cases despite the fact that they together could be construed as a valid number.
 17. Decoding is left to right, one pass.
 18. See also `slaDfltln`, `slaInt2ln` and `slaIntln`.

slaFw2m	rotation matrix from F-W angles	slaFw2m
----------------	---------------------------------	----------------

ACTION : Form a rotation matrix from the Fukushima-Williams angles.

CALL : `slaFw2m(gamb, phib, psi, eps, r);`

GIVEN :

gamb	double*	F-W angle $\bar{\gamma}$ (radians)
phib	double*	F-W angle $\bar{\phi}$ (radians)
psi	double*	F-W angle ψ (radians)
eps	double*	F-W angle ϵ (radians)

RETURNED :

`r` `double[3][3]` rotation matrix

NOTES :

1. Naming the following points:

e = J2000 ecliptic pole,
 p = GCRS pole,
 E = ecliptic pole of date, and
 P = CIP,

the four Fukushima-Williams angles are as follows:

`gamb` = $\bar{\gamma}$ = epE
`phib` = $\bar{\phi}$ = pE
`psi` = ψ = pEP
`eps` = ϵ = EP

2. The matrix representing the combined effects of frame bias **B**, precession **P** and nutation **N** is:

$$\mathbf{N} \cdot \mathbf{P} \cdot \mathbf{B} = R_1(-\epsilon) \cdot R_3(-\psi) \cdot R_1(\bar{\phi}) \cdot R_3(\bar{\gamma})$$

3. Three different matrices can be constructed, depending on the supplied angles:

- To obtain the $\mathbf{N} \cdot \mathbf{P} \cdot \mathbf{B}$ matrix, generate the four precession angles, generate the nutation components and add them to the $\bar{\psi}$ and ϵ_A angles, and call the present function.
- To obtain the $\mathbf{P} \cdot \mathbf{B}$ matrix, generate the four precession angles and call the present function.
- To obtain the frame bias matrix, **B**, generate the four precession angles for date J2000.0 and call the present function.

The **N** and **P** matrices can if necessary be obtained by combining these three appropriately.

REFERENCE : Hilton, J. *et al.* 2006, *Celest. Mech. Dyn. Astron.* 94, 351)

slaFw2xy CIP X, Y given F-W NPB angles. **slaFw2xy**

ACTION : CIP X, Y given Fukushima-Williams bias-precession-nutation angles.

CALL : `slaFw2xy(gamb, phib, psi, eps, &x, &y);`

GIVEN :

gamb	double*	F-W angle $\bar{\gamma}$ (radians)
phib	double*	F-W angle $\bar{\phi}$ (radians)
psi	double*	F-W angle ψ (radians)
eps	double*	F-W angle ϵ (radians)

RETURNED :

x,y	double*	CIP X, Y in “radians”
------------	----------------	-------------------------

NOTES :

1. Naming the following points:

e = J2000 ecliptic pole,
 p = GCRS pole,
 E = ecliptic pole of date, and
 P = CIP,

the four Fukushima-Williams angles are as follows:

$\mathbf{gamb} = \bar{\gamma} = epE$
 $\mathbf{phib} = \bar{\phi} = pE$
 $\mathbf{psi} = \psi = pEP$
 $\mathbf{eps} = \epsilon = EP$

2. The matrix representing the combined effects of frame bias \mathbf{B} , precession \mathbf{P} and nutation \mathbf{N} is:

$$\mathbf{N} \cdot \mathbf{P} \cdot \mathbf{B} = R_1(-\epsilon) \cdot R_3(-\psi) \cdot R_1(\bar{\phi}) \cdot R_3(\bar{\gamma})$$

X and Y are elements (3,1) and (3,2) of the matrix $\mathbf{N} \cdot \mathbf{P} \cdot \mathbf{B}$.

REFERENCE : Hilton, J. *et al.* 2006, *Celest. Mech. Dyn. Astron.* 94, 351)

slaG2ixys	GCRS to CIRS matrix given X, Y, s	slaG2ixys
------------------	-------------------------------------	------------------

ACTION : Form the GCRS to CIRS matrix given the CIP X, Y and the CIO locator s .

CALL : slaG2ixys($x, y, s, \text{rg2i}$);

GIVEN :

x,y	double	celestial intermediate pole (Note 1)
s	double	the CIO locator s (Note 2)

NOTES :

1. All arguments are in radians.
2. The equatorial coordinates are J2000.0 FK5. Use the function `slaGe50` if transformation to B1950.0 FK4 coordinates is required.

slaGalsup	galactic to supergalactic	slaGalsup
------------------	---------------------------	------------------

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

CALL : `slaGalsup(dl, db, &dsl, &dsb);`

GIVEN :

<code>dl,db</code>	double	galactic longitude and latitude [l^II, b^II] (radians)
--------------------	--------	--

RETURNED :

<code>dsl,dsb</code>	double*	supergalactic longitude and latitude (radians)
----------------------	---------	--

REFERENCES :

1. de Vaucouleurs, de Vaucouleurs, & Corwin, 1976, *Second Reference Catalogue of Bright Galaxies*, U.Texas, p8.
2. 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 $l^II = 137.37^\circ$.)

slaGe50	galactic to B1950 [α, δ]	slaGe50
----------------	--	----------------

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

CALL : `slaGe50(dl, db, &dr, &dd);`

GIVEN :

<code>dl,db</code>	double	galactic longitude and latitude [l^II, b^II]
--------------------	--------	--

RETURNED :

<code>dr,dd</code>	<code>double*</code>	B1950.0 $[\alpha, \delta]$
--------------------	----------------------	----------------------------

NOTES :

1. All arguments are in radians.
2. The equatorial coordinates are B1950.0 FK4. Use the function `slaGaleq` if transformation to J2000.0 FK5 coordinates is required.

REFERENCE : Blaauw *et al.*, 1960, *Mon.Not.R.astr.Soc.*, **121**, 123.

slaGeoc	geodetic to geocentric	slaGeoc
----------------	------------------------	----------------

ACTION : Convert geodetic position to geocentric.

CALL : `slaGeoc(p, h, &r, &z);`

GIVEN :

<code>p</code>	<code>double</code>	latitude (geodetic, radians)
<code>h</code>	<code>double</code>	height above reference spheroid (geodetic, metres)

RETURNED :

<code>r</code>	<code>double*</code>	distance from Earth axis (AU)
<code>z</code>	<code>double*</code>	distance from plane of Earth equator (AU)

NOTES :

1. Geocentric latitude can be obtained by evaluating `atan2(z,r)`.
2. IAU 1976 constants are used.

REFERENCE : Green, R.M. 1985, *Spherical Astronomy*, Cambridge U.P., p98.

slaGmst	UT to GMST	slaGmst
----------------	------------	----------------

ACTION : Transformation from universal time UT1 to Greenwich mean sidereal time.

CALL : `d = slaGmst(ut1);`

GIVEN :

<code>ut1</code>	<code>double</code>	universal time (strictly UT1) expressed as modified Julian Date (JD−2400000.5)
------------------	---------------------	--

RETURNED :

<code>double</code>	Greenwich mean sidereal time (radians)
---------------------	--

NOTES :

1. The IAU 1982 expression (see page S15 of the 1984 *Astronomical Almanac*) is used, but rearranged to reduce rounding errors. This expression is always described as giving the GMST at 0^hUT; in fact, it gives the difference between the GMST and the UT, which happens to equal the GMST (modulo 24 hours) at 0^hUT each day. In `slaGmst`, the entire UT is used directly as the argument for the canonical formula, and the fractional part of the UT is added separately; note that the factor 1.0027379... does not appear.
2. See also the function `slaGmsta`, which delivers better numerical precision by accepting the UT date and time as separate arguments.

<code>slaGmsta</code>	UT to GMST (extra precision)	<code>slaGmsta</code>
------------------------------	------------------------------	------------------------------

ACTION : Transformation from universal time UT1 to Greenwich mean sidereal time, with rounding errors minimized.

CALL : `d = slaGmsta(date, ut1);`

GIVEN :

<code>date</code>	<code>double</code>	UT1 date as Modified Julian Date (integer part of JD−2400000.5)
<code>ut1</code>	<code>double</code>	UT1 time (fraction of a day)

RETURNED :

<code>double</code>	Greenwich mean sidereal time (radians)
---------------------	--

NOTES :

1. The algorithm is derived from the IAU 1982 expression (see page S15 of the 1984 Astronomical Almanac).
2. There is no restriction on how the UT is apportioned between the **date** and **ut1** arguments. Either of the two arguments could, for example, be zero and the entire date + time supplied in the other. However, the function is designed to deliver maximum accuracy when the **date** argument is a whole number and the **ut1** argument lies in the range $[0, 1]$, or *vice versa*.
3. See also the function **slaGmst**, which accepts the UT1 as a single argument. Compared with **slaGmst**, the extra numerical precision delivered by the present function is unlikely to be important in an absolute sense, but may be useful when critically comparing algorithms and in applications where two sidereal times close together are differenced.

slaGst	GST (from ERA and EO)	slaGst
---------------	-----------------------	---------------

ACTION : Greenwich (apparent) sidereal time, from ERA and EO.

CALL : `d = slaGst(date, utdate, ut);`

GIVEN :

date	double	TDB Modified Julian Date
utdate	double	UT1 date (MJD: integer part of $JD - 2400000.5$)
ut	double	UT1 time (fraction of a day)

RETURNED :

double	Greenwich apparent sidereal time (radians, $0 - 2\pi$)
--------	---

NOTES :

1. The date is TDB as an MJD ($= JD - 2400000.5$). TT will do.
2. This Greenwich sidereal time is consistent with the latest and most accurate of SLALIB's precession-nutation models. It does not use the conventional GMST formula and equation of the equinoxes, but instead computes GST directly from the Earth rotation angle and the equation of the origins (the distance between the true equinox of date and the celestial intermediate origin).
3. There is no restriction on how the UT is apportioned between the **utdate** and **ut** arguments. Either of the two arguments could, for example, be zero and the entire date+time supplied in the other. However, the function is designed to deliver maximum accuracy when the **utdate** argument is a whole number and **ut** lies in the range $0 - 1$ (or *vice versa*).

slaH2e	$[Az, El]$ to $[h, \delta]$	slaH2e
---------------	-----------------------------	---------------

ACTION : Horizon to equatorial coordinates (single precision).

CALL : slaH2e(az, el, phi, &ha, &dec);

GIVEN :

az	float	azimuth (radians)
el	float	elevation (radians)
phi	float	latitude (radians)

RETURNED :

ha	float*	hour angle (radians)
dec	float*	declination (radians)

NOTES :

1. The sign convention for azimuth is north zero, east $+\pi/2$.
2. Hour angle is returned in the range $\pm\pi$. Declination is returned in the range $\pm\pi$.
3. The latitude is (in principle) geodetic. In critical applications, corrections for polar motion should be applied (see `slaPolmo`).
4. In some applications it will be important to specify the correct type of elevation in order to produce the required type of $[h, \delta]$. In particular, it may be important to distinguish between the elevation as affected by refraction, which will yield the *observed* $[h, \delta]$, and the elevation *in vacuo*, which will yield the *topocentric* $[h, \delta]$. If the effects of diurnal aberration can be neglected, the topocentric $[h, \delta]$ may be used as an approximation to the *apparent* $[h, \delta]$.
5. No range checking of arguments is carried out.
6. In applications which involve many such calculations, rather than calling the present function it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude.

slaH2fk5	Hipparcos to FK5	slaH2fk5
-----------------	------------------	-----------------

ACTION : Transform a Hipparcos star position and proper motion into the FK5 (J2000) system.

CALL : slaH2fk5(rh, dh, drh, ddh, &r5, &d5, &dr5, &dd5);

GIVEN :

rh	double	Hipparcos α (radians)
dh	double	Hipparcos δ (radians)
drh	double	Hipparcos proper motion in α (radians per Julian year)
ddh	double	Hipparcos proper motion in δ (radians per Julian year)

RETURNED :

r5	double*	J2000.0 FK5 α (radians)
d5	double*	J2000.0 FK5 δ (radians)
dr5	double*	J2000.0 FK5 proper motion in α (radians per Julian year)
dd5	double*	FK5 J2000.0 proper motion in δ (radians per Julian year)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	−19.9	−0.30
<i>y</i>	−9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians.

4. See also `slaFk52h`, `slaFk5hz`, `slaHfk5z`.

REFERENCE : Feissel, M. & Mignard, F., 1998, *Astron.Astrophys.* **331**, L33-L36.

slaHfk5z Hipparcos to FK5, no proper motion **slaHfk5z**

ACTION : Transform a Hipparcos star position into the FK5 (J2000) system assuming zero Hipparcos proper motion.

CALL : `slaHfk5z(rh, dh, epoch, &r5, &d5, &dr5, &dd5);`

GIVEN :

rh	double	Hipparcos α (radians)
dh	double	Hipparcos δ (radians)
epoch	double	Julian epoch (TDB)

RETURNED :

r5	double*	J2000.0 FK5 α (radians)
d5	double*	J2000.0 FK5 δ (radians)
dr5	double*	J2000.0 FK5 proper motion in α (radians per Julian year)
dd5	double*	FK5 J2000.0 proper motion in δ (radians per Julian year)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	−19.9	−0.30
<i>y</i>	−9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians.

4. It was the intention that Hipparcos 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 FK5, and this function returns those *fictitious proper motions*.
5. The position returned by this function is in the FK5 J2000 reference system but at Julian epoch **epoch**.
6. TT, or even UTC, can be used instead of TDB.
7. See also **slaFk52h**, **slaFk5hz**, **slaH2fk5**.

REFERENCE : Feissel, M. & Mignard, F., 1998, *Astron.Astrophys.* **331**, L33-L36.

slaI2c	CIRS to ICRS	slaI2c
---------------	--------------	---------------

ACTION : Transform star $[\alpha, \delta]$ from CIRS to ICRS astrometric..

CALL : `slaI2c(ri, di, date, dx, dy, &rc, &dc);`

GIVEN :

<code>ri,di</code>	<code>double</code>	CIRS $[\alpha, \delta]$ (radians)
<code>date</code>	<code>double</code>	TDB (JD-2400000.5)
<code>dx,dy</code>	<code>double</code>	corrections to IAU CIP X, Y (radians)

RETURNED :

<code>rc,dc</code>	<code>double*</code>	ICRS astrometric $[\alpha, \delta]$ (radians)
--------------------	----------------------	---

NOTES :

1. For date, the distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
2. Iterative techniques are used for the aberration and light deflection corrections so that the functions `slaI2c` (or `slaI2cqk`) and `slaC2i` (or `slaC2iqk`) are accurate inverses; even at the edge of the Sun's disc the discrepancy is only about 1 nanoarcsecond.
3. The accuracy is sub-mas, limited by the precession-nutation information (IAU 2006 model plus the CIP corrections, which are obtained from IERS bulletins).
4. The accuracy is in principle further limited by the function `slaEpv`, which computes the Earth position and velocity. The maximum error arising from this source is less than $5 \mu\text{s}$.

REFERENCE : Seidelmann, P.K. *et al.*, 1992, *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Chapter 3.

slaI2cqk	quick CIRS to ICRS	slaI2cqk
-----------------	--------------------	-----------------

ACTION : Quick CIRS $[\alpha, \delta]$ to ICRS astrometric place, given the star-independent parameters.

CALL : `slaI2cqk(ri, di, &ciprms, &rc, &dc);`

GIVEN :

ri,di	double	CIRS $[\alpha, \delta]$ (radians)
ciprms	CIpars*	star-independent ICRS-to-CIRS parameters:
	pmt	◦ time interval for proper motion (Julian years)
	eb[3]	◦ barycentric position of the Earth (AU)
	ehn[3]	◦ heliocentric direction of the Earth (unit vector)
	gr2e	◦ (gravitational radius of Sun) $\times 2$ /(Sun-Earth distance)
	abv[3]	◦ v : barycentric Earth velocity in units of c
	ab1	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	bpn[3][3]	◦ bias-precession-nutation 3×3 matrix

RETURNED :

rc,dc	double*	ICRS astrometric $[\alpha, \delta]$ (radians)
--------------	----------------	---

NOTES :

1. Use of this function is appropriate when efficiency is important and where many star positions are all to be transformed for one date. The star-independent parameters can be obtained by calling the **slaC2ipa** function.
2. Iterative techniques are used for the aberration and light deflection corrections so that the functions **slaI2c** (or **slaI2cqk**) and **slaC2i** (or **slaC2iqk**) are accurate inverses; even at the edge of the Sun's disc the discrepancy is only about 1 nanoarcsecond.
3. Using **ciprms** parameters from **slaC2ipa**, the accuracy is sub-mas, limited by the precession-nutation information (IAU 2006 model plus the CIP corrections, which are obtained from IERS bulletins).
4. The accuracy is in principle further limited by the function **slaEpv**, which computes the Earth position and velocity. The maximum error arising from this source is less than $5 \mu\text{s}$.

REFERENCE : Seidelmann, P.K. *et al.*, 1992, *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Chapter 3.

slaI2o	CIRS to observed place	slaI2o
---------------	------------------------	---------------

ACTION : CIRS $[\alpha, \delta]$ to observed place at a ground based site, for sources distant from the solar system.

CALL : **slaI2o**(**ri**, **di**, **utc**, **dut**, **elongm**, **phim**, **hm**, **xp**, **yp**,
tk, **phpa**, **rh**, **wlfq**, **tlr**, **&aob**, **&zob**, **&hob**, **&dob**, **&rob**);

GIVEN :

<code>ri,di</code>	<code>double</code>	CIRS $[\alpha, \delta]$ (radians)
<code>utc</code>	<code>double</code>	UTC date/time (MJD, JD–2400000.5)
<code>dut</code>	<code>double</code>	Δ UT: UT1–UTC (UTC seconds)
<code>elongm</code>	<code>double</code>	observer’s mean longitude (radians, east +ve)
<code>phim</code>	<code>double</code>	observer’s mean geodetic latitude (radians)
<code>hm</code>	<code>double</code>	observer’s height above sea level (metres)
<code>xp,yp</code>	<code>double</code>	polar motion $[x, y]$ coordinates (radians)
<code>tk</code>	<code>double</code>	local ambient temperature (K; std=273.15)
<code>phpa</code>	<code>double</code>	local atmospheric pressure (hPa; std=1013.25)
<code>rh</code>	<code>double</code>	local relative humidity (in the range 0.0–1.0)
<code>wlfq</code>	<code>double</code>	wavelength (μ m) or minus frequency (GHz)
<code>tlr</code>	<code>double</code>	tropospheric lapse rate (K per metre, <i>e.g.</i> 0.0065)

RETURNED :

<code>aob</code>	<code>double*</code>	observed azimuth (radians: N=0, E=90°)
<code>zob</code>	<code>double*</code>	observed zenith distance (radians)
<code>hob</code>	<code>double*</code>	observed hour angle (radians)
<code>dob</code>	<code>double*</code>	observed δ (radians)
<code>rob</code>	<code>double*</code>	observed α (radians)

NOTES :

1. This function returns zenith distance rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.
2. The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about 0".1 for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90° , the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the `slaRefro` function (*q.v.*) returns a fixed value of the refraction.

The complementary functions `slaI2o` (or `slaI2oqk`) and `sla02i` (or `sla02iqk`) are self-consistent to better than 1 microarcsecond all over the celestial sphere.

3. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
4. The CIRS $[\alpha, \delta]$ is obtained from a catalogue mean place by allowing for space motion, parallax, the Sun’s gravitational lens effect, annual aberration, and precession-nutation. For star positions in the ICRS (the successor to FK5 J2000), these effects can be applied by means of the `slaC2i` *etc.* functions. Starting from the formerly used “mean place”, additional transformations will be needed first.

5. *Observed* $[Az, El]$ means the position that would be seen by a perfect theodolite located at the observer. This is obtained from the CIRS $[\alpha, \delta]$ by allowing for Earth orientation and diurnal aberration, rotating from equator to horizon coordinates, and then adjusting for refraction. The $[h, \delta]$ is obtained by rotating back into equatorial coordinates, using the geodetic latitude corrected for polar motion, and is the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). Finally, the α is obtained by subtracting the h from the local ERA.
6. To predict the required setting of a real telescope, the observed place produced by this function would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. This function takes time to execute, due mainly to the rigorous integration used to evaluate the refraction. For processing multiple stars for one location and time, call `slaI2opa` once followed by one call per star to `slaI2oqk`. Where a range of times within a limited period of a few hours is involved, and the highest precision is not required, call `slaI2opa` once, followed by a call to `slaI2opat` each time the time changes, followed by one call per star to `slaI2oqk`.
8. The `utc` argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the ΔUT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap ΔUT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the ΔUT changes by one second to its post-leap new value.
9. The ΔUT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep ΔUT within $\pm 0^s.9$.
10. **Important : take care with the longitude sign convention.** The longitude required by the present function is **east-positive**, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the `slaObs` function are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present function.
11. The polar motion coordinates x_p, y_p can be obtained from IERS bulletins. The maximum amplitude is about $0''.3$. If x_p, y_p values are unavailable, use $xp = yp = 0.0$. The x_p, y_p coordinates are reckoned as follows:
 - positive x_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 90° .
 - positive y_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 270° .

Internally, the polar motion is stored in a form rotated onto the local meridian.

12. The height above sea level of the observing station, **hm**, can be obtained from the *Astronomical Almanac* (Section J in the 1988 edition), or via the function **slaObs**. If **p**, the pressure in hPa (mb), is available, an adequate estimate of **hm** can be obtained from the following expression:

$$\text{hm} = -29.3 * \text{tsl} * \log(\text{p}/1013.25);$$

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

$$\text{p} = 1013.25 * \exp(-\text{hm}/(29.3 * \text{tsl}));$$

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

13. The argument **wlfq** specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at 100 μm (about 3000 GHz).

slaI2opa	CIRS-to-observed parameters	slaI2opa
-----------------	-----------------------------	-----------------

ACTION : Pre-compute the set of CIRS to observed place parameters required by the “quick” functions **slaI2oqk** and **sla02iqk**.

CALL : **slaI2opa**(**utc**, **dut**, **elongm**, **phim**, **hm**, **xp**, **yp**,
tk, **phpa**, **rh**, **wlfq**, **tlr**, **&ioprms**);

GIVEN :

utc	double	UTC date/time (MJD, JD–2400000.5)
dut	double	ΔUT : UT1–UTC (UTC seconds)
elongm	double	observer’s mean longitude (radians, east +ve)
phim	double	observer’s mean geodetic latitude (radians)
hm	double	observer’s height above sea level (metres)
xp,yp	double	polar motion [<i>x</i> , <i>y</i>] coordinates (radians)
tk	double	local ambient temperature (K; std=273.15)
phpa	double	local atmospheric pressure (hPa; std=1013.25)
rh	double	local relative humidity (in the range 0.0–1.0)
wlfq	double	wavelength (μm) or minus frequency (GHz)
tlr	double	tropospheric lapse rate (K per metre, <i>e.g.</i> 0.0065)

RETURNED :

ioprms	IOpars*	star-independent CIRS-to-observed parameters:
	along	◦ longitude + $s' + \text{dERA}(\Delta\text{UT})$ (radians)
	phi	◦ geodetic latitude (radians)
	hm	◦ height above sea level (metres)
	xpl	◦ polar motion x_p wrt local meridian (radians)
	ypl	◦ polar motion y_p wrt local meridian (radians)
	sphi	◦ sine of geodetic latitude
	cphi	◦ cosine of geodetic latitude
	diurab	◦ magnitude of diurnal aberration vector
	p	◦ pressure (hPa = mb)
	tk	◦ ambient temperature (K)
	rh	◦ relative humidity (0–1)
	tlr	◦ tropospheric lapse rate (K per metre)
	wl	◦ wavelength (μm) or minus frequency (GHz)
	refa	◦ refraction constant A (radians)
	refb	◦ refraction constant B (radians)
	eral	◦ “local” Earth rotation angle (radians)

NOTES :

1. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
2. The `utc` argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the ΔUT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap ΔUT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the ΔUT changes by one second to its post-leap new value.
3. The ΔUT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep ΔUT within ± 0.9 .
4. **Important : take care with the longitude sign convention.** The longitude required by the present function is **east-positive**, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the `slaObs` function are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present function.
5. The polar motion coordinates x_p, y_p can be obtained from IERS bulletins. The maximum amplitude is about $0''.3$. If x_p, y_p values are unavailable, use $\mathbf{xp} = \mathbf{yp} = 0.0$. The x_p, y_p coordinates are reckoned as follows:
 - positive x_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 90° .
 - positive y_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 270° .

Internally, the polar motion is stored in a form rotated onto the local meridian.

6. The height above sea level of the observing station, `hm`, can be obtained from the *Astronomical Almanac* (Section J in the 1988 edition), or via the function `slaObs`. If `p`, the pressure in hPa (mb) (the same as hPa), is available, an adequate estimate of `hm` can be obtained from the following expression:

$$\text{hm} = -29.3 * \text{tsl} * \log(p/1013.25);$$

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

$$p = 1013.25 * \exp(-\text{hm}/(29.3 * \text{tsl}));$$

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

7. The “adjusted longitude” `ioprms->along` is the observer’s longitude plus the tiny quantity s' plus the small amount of ERA that corresponds to the current UT1–UTC.
8. The “local Earth rotation angle” `ioprms->eral` is ERA plus the adjusted longitude.
9. Repeated, computationally-expensive, calls to `slaI2opa` for times that are very close together can be avoided by calling `slaI2opa` just once and then using `slaI2opat` for the subsequent times. Fresh calls to `slaI2opa` will be needed only when changes in the polar motion have grown to unacceptable levels or when anything affecting the refraction has changed.
10. The argument `wlfq` specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at $100\ \mu\text{m}$ (about 3000 GHz).

slaI2opad in CIRS-to-observed, disable diurnal aberration **slaI2opad**

ACTION : In the CIRS to observed place star-independent parameter structure, disable the diurnal aberration.

CALL : `slaI2opad(&ioprms);`

GIVEN :

ioprms	IOpars*	star-independent CIRS-to-observed parameters:
	along	◦ longitude + s' + dERA(Δ UT) (radians)
	phi	◦ geodetic latitude (radians)
	hm	◦ height above sea level (metres)
	xpl	◦ polar motion x_p wrt local meridian (radians)
	ypl	◦ polar motion y_p wrt local meridian (radians)
	sphi	◦ sine of geodetic latitude
	cphi	◦ cosine of geodetic latitude
	diurab	◦ magnitude of diurnal aberration vector
	p	◦ pressure (hPa = mb)
	tk	◦ ambient temperature (K)
	rh	◦ relative humidity (0–1)
	tlr	◦ tropospheric lapse rate (K per metre)
	wl	◦ wavelength (μ m) or minus frequency (GHz)
	refa	◦ refraction constant A (radians)
	refb	◦ refraction constant B (radians)
	eral	◦ “local” Earth rotation angle (radians)

RETURNED :

ioprms	IOpars*	star-independent CIRS-to-observed parameters:
	along	◦ unchanged
	phi	◦ unchanged
	hm	◦ unchanged
	xpl	◦ unchanged
	ypl	◦ unchanged
	sphi	◦ unchanged
	cphi	◦ unchanged
	diurab	◦ set to zero
	p	◦ unchanged
	tk	◦ unchanged
	rh	◦ unchanged
	tlr	◦ unchanged
	wl	◦ unchanged
	refa	◦ unchanged
	refb	◦ unchanged
	eral	◦ unchanged

NOTES :

1. This capability is provided to bypass the geocentric stage in the transformation from ICRS to CIRS coordinates; the site velocity and position can be specified at that point. One motivation for doing so would be to include (stellar) geocentric parallax, which is neglected in the `slaI2o` functions.
2. To re-instate the correction for diurnal aberration, use `slaI2opa`.
3. For more information, see `slaI2opa`.

slaI2opat

update CIRS-to-observed parameters

slaI2opat

ACTION : Recompute the Earth rotation angle in the CIRS to observed place star-independent parameter structure.

CALL : `slaI2opat(utc, &ioprms);`

GIVEN :

<code>utc</code>	<code>double</code>	UTC date/time (MJD, $JD - 2400000.5$)
<code>ioprms</code>	<code>I0pars*</code>	star-independent CIRS-to-observed parameters:
	<code>along</code>	◦ longitude + $s' + \text{dERA}(\Delta\text{UT})$ (radians)
	<code>phi</code>	◦ geodetic latitude (radians)
	<code>hm</code>	◦ height above sea level (metres)
	<code>xpl</code>	◦ polar motion x_p wrt local meridian (radians)
	<code>ypl</code>	◦ polar motion y_p wrt local meridian (radians)
	<code>sphi</code>	◦ sine of geodetic latitude
	<code>cphi</code>	◦ cosine of geodetic latitude
	<code>diurab</code>	◦ magnitude of diurnal aberration vector
	<code>p</code>	◦ pressure (hPa = mb)
	<code>tk</code>	◦ ambient temperature (K)
	<code>rh</code>	◦ relative humidity (0–1)
	<code>tlr</code>	◦ tropospheric lapse rate (K per metre)
	<code>wl</code>	◦ wavelength (μm) or minus frequency (GHz)
	<code>refa</code>	◦ refraction constant A (radians)
	<code>refb</code>	◦ refraction constant B (radians)
	<code>eral</code>	◦ “local” Earth rotation angle (radians)

RETURNED :

<code>ioprms</code>	<code>I0pars*</code>	star-independent CIRS-to-observed parameters:
	<code>along</code>	◦ unchanged
	<code>phi</code>	◦ unchanged
	<code>hm</code>	◦ unchanged
	<code>xpl</code>	◦ unchanged
	<code>ypl</code>	◦ unchanged
	<code>sphi</code>	◦ unchanged
	<code>cphi</code>	◦ unchanged
	<code>diurab</code>	◦ unchanged
	<code>p</code>	◦ unchanged
	<code>tk</code>	◦ unchanged
	<code>rh</code>	◦ unchanged
	<code>tlr</code>	◦ unchanged
	<code>wl</code>	◦ unchanged
	<code>refa</code>	◦ unchanged
	<code>refb</code>	◦ unchanged
	<code>eral</code>	◦ updated

NOTES :

1. The `utc` argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the ΔUT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap ΔUT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the ΔUT changes by one second to its post-leap new value.
2. The ΔUT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep ΔUT within ± 0.9 .
3. The “adjusted longitude” `ioprms->along` is the observer’s longitude plus the tiny quantity s' plus the small amount of ERA that corresponds to the current UT1–UTC.
4. The “local Earth rotation angle” `ioprms->eral` is ERA plus the adjusted longitude.

slaI2oqk	quick CIRS to observed place	slaI2oqk
-----------------	------------------------------	-----------------

ACTION : Quick CIRS to observed place (but see Note 8, below, for remarks about speed).

CALL : `slaI2oqk(ri, di, &ioprms, &aob, &zob, &hob, &dob, &rob);`

GIVEN :

<code>ri,di</code>	<code>double</code>	CIRS $[\alpha, \delta]$ (radians)
<code>ioprms</code>	<code>I0pars*</code>	star-independent CIRS-to-observed parameters:
	<code>along</code>	◦ longitude + s' + dERA(ΔUT) (radians)
	<code>phi</code>	◦ geodetic latitude (radians)
	<code>hm</code>	◦ height above sea level (metres)
	<code>xpl</code>	◦ polar motion x_p wrt local meridian (radians)
	<code>ypl</code>	◦ polar motion y_p wrt local meridian (radians)
	<code>sphi</code>	◦ sine of geodetic latitude
	<code>cphi</code>	◦ cosine of geodetic latitude
	<code>diurab</code>	◦ magnitude of diurnal aberration vector
	<code>p</code>	◦ pressure (hPa = mb)
	<code>tk</code>	◦ ambient temperature (K)
	<code>rh</code>	◦ relative humidity (0–1)
	<code>tlr</code>	◦ tropospheric lapse rate (K per metre)
	<code>wl</code>	◦ wavelength (μm) or minus frequency (GHz)
	<code>refa</code>	◦ refraction constant A (radians)
	<code>refb</code>	◦ refraction constant B (radians)
	<code>eral</code>	◦ “local” Earth rotation angle (radians)

RETURNED :

aob	double*	observed azimuth (radians: N=0, E=90°)
zob	double*	observed zenith distance (radians)
hob	double*	observed hour angle (radians)
dob	double*	observed δ (radians)
rob	double*	observed α (radians)

NOTES :

1. This function returns zenith distance rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.
2. The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about 0".1 for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90° , the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the **slaRefro** function (*q.v.*) returns a fixed value of the refraction.

The complementary functions **slaI2o** (or **slaI2oqk**) and **sla02i** (or **sla02iqk**) are self-consistent to better than 1 microarcsecond all over the celestial sphere.

3. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
4. The CIRS $[\alpha, \delta]$ is obtained from a catalogue mean place by allowing for space motion, parallax, the Sun's gravitational lens effect, annual aberration, and precession-nutation. For star positions in the ICRS (the successor to FK5 J2000), these effects can be applied by means of the **slaC2i** *etc.* functions. Starting from the formerly used "mean place", additional transformations will be needed first.
5. *Observed* $[Az, El]$ means the position that would be seen by a perfect theodolite located at the observer. This is obtained from the CIRS $[\alpha, \delta]$ by allowing for Earth orientation and diurnal aberration, rotating from equator to horizon coordinates, and then adjusting for refraction. The $[h, \delta]$ is obtained by rotating back into equatorial coordinates, using the geodetic latitude corrected for polar motion, and is the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). Finally, the α is obtained by subtracting the h from the local ERA.
6. To predict the required setting of a real telescope, the observed place produced by this function would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. The star-independent CIRS-to-observed-place parameters in **ioprms** may be computed by means of the **slaI2opa** function. If nothing has changed significantly except the time, the **slaI2opat** function may be used to perform the requisite partial recomputation of **ioprms**.

8. At zenith distances beyond about 76° , the need for special care with the corrections for refraction causes a marked increase in execution time. Moreover, the effect gets worse with increasing zenith distance. Adroit programming in the calling application may allow the problem to be reduced. Prepare an alternative `ioprms` structure, computed for zero air-pressure; this will disable the refraction corrections and cause rapid execution. Using this `ioprms` structure, a preliminary call to the present function will, depending on the application, produce a rough position which may be enough to establish whether the full, slow calculation (using the real `ioprms` structure) is worthwhile. For example, there would be no need for the full calculation if the preliminary call had already established that the source was well below the elevation limits for a particular telescope.

slaImxv	apply 3D reverse rotation	slaImxv
----------------	---------------------------	----------------

ACTION : Multiply a 3-vector by the inverse of a rotation matrix (single precision).

CALL : `slaImxv(rm, va, vb);`

GIVEN :

<code>rm</code>	<code>float[3][3]</code>	rotation matrix
<code>va</code>	<code>float[3]</code>	vector to be rotated

RETURNED :

<code>vb</code>	<code>float[3]</code>	result vector
-----------------	-----------------------	---------------

NOTES :

1. This function performs the operation:

$$\mathbf{b} = \mathbf{M}^T \times \mathbf{a}$$
 where \mathbf{a} and \mathbf{b} are the 3-vectors `va` and `vb` respectively, and \mathbf{M} is the 3×3 matrix `rm`.
2. The main function of this function is apply a rotation; under these circumstances, \mathbf{M} is a *proper real orthogonal* matrix.
3. `va` and `vb` may be the same array.

slaInt2in	decode an integer number	slaInt2in
------------------	--------------------------	------------------

ACTION : Convert free-format input into an integer.

CALL : `slaInt2in(string, &nstrt, &iresult, &jflag);`

GIVEN :

string	char*	string containing number to be decoded
nstrt	int*	index to where decoding is to commence
ireslt	int*	current value of result

RETURNED :

nstrt	int*	advanced to next number
ireslt	int*	result
jflag	int*	status: -1 = -OK, 0 = +OK, 1 = null result, 2 = error

NOTES :

1. The reason **slaInt2in** has separate “OK” status values for ‘+’ and ‘-’ is to enable minus zero to be detected. This is of crucial importance when decoding mixed-radix numbers. For example, an angle expressed as degrees, arcminutes and arcseconds may have a leading minus sign but a zero degrees field.
2. A TAB is interpreted as a space.
3. The basic format is the sequence of fields $\pm n$, where \pm is a sign character ‘+’ or ‘-’, and n means a string of decimal digits.
4. Spaces:
 - Leading spaces are ignored.
 - Spaces between the sign and the number are allowed.
 - Trailing spaces are ignored; the first signifies end of decoding and subsequent ones are skipped.
5. Delimiters:
 - Any character other than +, -, 0-9 or space may be used to signal the end of the number and terminate decoding.
 - Comma is recognized by **slaInt2in** as a special case; it is skipped, leaving the index on the next character. See 9, below.
 - Decoding will in all cases terminate if end of string is reached.
6. The sign is optional. The default is +.
7. A *null result* occurs when the string of characters being decoded does not begin with +, - or 0-9, or consists entirely of spaces. When this condition is detected, **jflag** is set to 1 and **ireslt** is left untouched.
8. **nstrt** = 1 for the first character in the string.
9. On return from **slaInt2in**, **nstrt** is set ready for the next decode – following trailing blanks and any comma. If a delimiter other than comma is being used, **nstrt** must be incremented before the next call to **slaInt2in**, otherwise all subsequent calls will return a null result.
10. Errors (**jflag** = 2) occur when:

- there is a '+' or '-' but no number; or
 - the number is greater than $2^{31} - 1$.
11. When an error has been detected, `nstrt` is left pointing to the character following the last one used before the error came to light.
 12. See also `slaIntin`, `slaFlotin` and `slaDfltin`.

slaIntin	decode a long integer number	slaIntin
-----------------	------------------------------	-----------------

ACTION : Convert free-format input into an integer.

CALL : `slaIntin(string, &nstrt, &lresult, &jflag);`

GIVEN :

<code>string</code>	<code>char*</code>	string containing number to be decoded
<code>nstrt</code>	<code>int*</code>	index to where decoding is to commence
<code>lresult</code>	<code>long*</code>	current value of result

RETURNED :

<code>nstrt</code>	<code>int*</code>	advanced to next number
<code>lresult</code>	<code>long*</code>	result
<code>jflag</code>	<code>int*</code>	status: -1 = -OK, 0 = +OK, 1 = null result, 2 = error

NOTES :

1. The reason `slaIntin` has separate “OK” status values for ‘+’ and ‘-’ is to enable minus zero to be detected. This is of crucial importance when decoding mixed-radix numbers. For example, an angle expressed as degrees, arcminutes and arcseconds may have a leading minus sign but a zero degrees field.
2. A TAB is interpreted as a space.
3. The basic format is the sequence of fields $\pm n$, where \pm is a sign character '+' or '-', and n means a string of decimal digits.
4. Spaces:
 - Leading spaces are ignored.
 - Spaces between the sign and the number are allowed.
 - Trailing spaces are ignored; the first signifies end of decoding and subsequent ones are skipped.
5. Delimiters:
 - Any character other than +, -, 0-9 or space may be used to signal the end of the number and terminate decoding.

- Comma is recognized by `slaIntin` as a special case; it is skipped, leaving the index on the next character. See 9, below.
 - Decoding will in all cases terminate if end of string is reached.
6. The sign is optional. The default is '+'.
 7. A *null result* occurs when the string of characters being decoded does not begin with +, - or 0-9, or consists entirely of spaces. When this condition is detected, `jflag` is set to 1 and `lresult` is left untouched.
 8. `nstrt` = 1 for the first character in the string.
 9. On return from `slaIntin`, `nstrt` is set ready for the next decode – following trailing blanks and any comma. If a delimiter other than comma is being used, `nstrt` must be incremented before the next call to `slaIntin`, otherwise all subsequent calls will return a null result.
 10. Errors (`jflag` = 2) occur when:
 - there is a '+' or '-' but no number; or
 - the number is greater than $2^{31} - 1$.
 11. When an error has been detected, `nstrt` is left pointing to the character following the last one used before the error came to light.
 12. See also `slaInt2in`, `slaFlotin` and `slaDfltin`.

slaInvf	invert linear model	slaInvf
----------------	---------------------	----------------

ACTION : Invert a linear model of the type produced by the `slaFitxy` function.

CALL : `slaInvf(fwds, bkws, &j);`

GIVEN :

<code>fwds</code>	<code>double[6]</code>	model coefficients
-------------------	------------------------	--------------------

RETURNED :

<code>bkws</code>	<code>double[6]</code>	inverse model
<code>j</code>	<code>int*</code>	status: 0 = OK, -1 = no inverse

NOTES :

1. The models relate two sets of $[x, y]$ coordinates as follows. Naming the six elements of `fwds` a, b, c, d, e & f , where two sets of coordinates $[x_1, y_1]$ and $[x_2, y_2]$ are related thus:

$$\begin{aligned} x_2 &= a + bx_1 + cy_1 \\ y_2 &= d + ex_1 + fy_1 \end{aligned}$$

The present function generates a new set of coefficients p, q, r, s, t & u (the array `bkwds`) such that:

$$\begin{aligned}x_1 &= p + qx_2 + ry_2 \\ y_1 &= s + tx_2 + uy_2\end{aligned}$$

2. Two successive calls to this function will deliver a set of coefficients equal to the starting values.
3. `fwds` and `bkwds` can be the same array.
4. See also `slaFitxy`, `slaPxy`, `slaXy2xy`, `slaDcmpf`.

slaKbj	select epoch prefix	slaKbj
---------------	---------------------	---------------

ACTION : Select epoch prefix 'B' or 'J'.

CALL : `slaKbj(jb, e, &k, &j);`

GIVEN :

<code>jb</code>	<code>int</code>	sladbjin prefix status: 0 = none, 1 = 'B', 2 = 'J'
<code>e</code>	<code>double</code>	epoch – Besselian or Julian

RETURNED :

<code>k</code>	<code>char*</code>	'B' or 'J'
<code>j</code>	<code>int*</code>	status: 0 = OK

NOTE : The function is mainly intended for use in conjunction with the `sladbjin` function.

If the value of `jb` indicates that an explicit 'B' or 'J' prefix was detected by `sladbjin`, a 'B' or 'J' is returned to match. If `jb` indicates that no explicit 'B' or 'J' was supplied, the choice is made on the basis of the epoch itself; 'B' is assumed for `e` < 1984, otherwise 'J'.

slaM2av	rotation matrix to axial vector	slaM2av
----------------	---------------------------------	----------------

ACTION : From a rotation matrix, determine the corresponding axial vector (single precision).

CALL : `slaM2av(rmat, axvec);`

GIVEN :

<code>rmat</code>	<code>float [3] [3]</code>	rotation matrix
-------------------	----------------------------	-----------------

RETURNED :

<code>axvec</code>	<code>float[3]</code>	axial vector (radians)
--------------------	-----------------------	------------------------

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis, called the Euler axis. The *axial vector* returned by this function has the same direction as the Euler axis, and its magnitude is the amount of rotation in radians.
2. The magnitude and direction of the axial vector can be separated by means of the function `slaVn`.
3. The reference frame rotates clockwise as seen looking along the axial vector from the origin.
4. If `rmat` is null, so is the result.

slaMap	mean to apparent	slaMap
---------------	------------------	---------------

ACTION : Transform star $[\alpha, \delta]$ from mean place to geocentric apparent. The reference systems and time-scales used are post IAU 1976.

CALL : `slaMap(rm, dm, pr, pd, px, rv, eq, date, &ra, &da);`

GIVEN :

<code>rm, dm</code>	<code>double</code>	mean $[\alpha, \delta]$ (radians)
<code>pr, pd</code>	<code>double</code>	proper motions: $[\alpha, \delta]$ changes per Julian year
<code>px</code>	<code>double</code>	parallax (arcsec)
<code>rv</code>	<code>double</code>	radial velocity (km s^{-1} , +ve if receding)
<code>eq</code>	<code>double</code>	epoch and equinox of star data (Julian)
<code>date</code>	<code>double</code>	TDB for apparent place ($\text{JD}-2400000.5$)

RETURNED :

<code>ra, da</code>	<code>double*</code>	apparent $[\alpha, \delta]$ (radians)
---------------------	----------------------	---------------------------------------

NOTES :

1. `eq` is the Julian epoch specifying both the reference system and the epoch of the position – usually 2000. For positions where the epoch and equinox are different, use the function `slaPm` to apply proper motion corrections before using this function.
2. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.

3. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are per year rather than per century.
4. This function may be wasteful for some applications because it recomputes the Earth position/velocity and the precession-nutation matrix each time, and because it allows for parallax and proper motion. Where multiple transformations are to be carried out for one epoch, a faster method is to call the `slaMappa` function once and then either the `slaMapqk` function (which includes parallax and proper motion) or `slaMapqkz` (which assumes zero parallax and FK5 proper motion).
5. The accuracy, starting from ICRS star data, is limited to about 1 mas by the precession-nutation model used, SF2001 (IAU 1976 precession, Shirai & Fukushima 2001 forced nutation and precession corrections). A different precession-nutation model can be introduced by using `slaMappa` and `slaMapqk` (see the previous note) and replacing the precession-nutation matrix into the parameter array directly.
6. The accuracy is further limited by the function `slaEvp`, called by `slaMappa`, which computes the Earth position and velocity using the methods of Stumpff. The maximum error is about 0.3 milliarcsecond.

REFERENCES :

1. 1984 *Astronomical Almanac*, pp B39-B41.
2. Lederle & Schwan 1984, *Astr.Astrophys.* **134**, 1-6.
3. Shirai, T. & Fukushima, T. 2001, *Astron.J.*, **121**, 3270-3283.

slaMappa	mean to apparent parameters	slaMappa
-----------------	-----------------------------	-----------------

ACTION : 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, light deflection, aberration, and precession-nutation parts of the mean/apparent transformations. The reference systems and time-scales used are post IAU 1976.

CALL : `slaMappa(eq, date, amprms);`

GIVEN :

eq	double	epoch of mean equinox to be used (Julian)
date	double	TDB (JD-2400000.5)

RETURNED :

amprms	double [21]	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]	◦ (gravitational radius of Sun)×2/(Sun-Earth distance)
	[8-10]	◦ v : barycentric Earth velocity in units of <i>c</i>
	[11]	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	[12-20]	◦ precession-nutation 3 × 3 matrix

NOTES :

1. For **date**, the distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
2. The vectors **amprms**[1-3] and **amprms**[4-6] are (in essence) referred to the mean equinox and equator of epoch **eq**. For **eq** = 2000.0, they are referred to the ICRS.
3. The parameters produced by this function are used by **slaMapqk**, **slaMapqkz** and **slaAmpqk**.
4. The accuracy, starting from ICRS star data, is limited to about 1 mas by the precession-nutation model used, SF2001 (IAU 1976 precession, Shirai & Fukushima 2001 forced nutation and precession corrections). A different precession-nutation model can be introduced by first calling the present function and then replacing the precession-nutation matrix in **amprms**[12-20] directly.
5. A further limit to the accuracy of functions using the parameter array **amprms** is imposed by the function **slaEvp**, used here to compute the Earth position and velocity by the methods of Stumpff. The maximum error in the resulting aberration corrections is about 0.3 milliarcsecond.

REFERENCES :

1. 1984 *Astronomical Almanac*, pp B39-B41.
2. Lederle & Schwan 1984, *Astr.Astrophys.* **134**, 1-6.
3. Shirai, T. & Fukushima, T. 2001, *Astron.J.*, **121**, 3270-3283.

slaMapqk	quick mean to apparent	slaMapqk
-----------------	------------------------	-----------------

ACTION : Quick mean to apparent place: transform a star $[\alpha, \delta]$ from mean place to geocentric apparent place, given the star-independent parameters. The reference systems and time-scales used are post IAU 1976.

CALL : **slaMapqk**(*rm*, *dm*, *pr*, *pd*, *px*, *rv*, *amprms*, *&ra*, *&da*);

GIVEN :

<code>rm, dm</code>	<code>double</code>	mean $[\alpha, \delta]$ (radians)
<code>pr, pd</code>	<code>double</code>	proper motions: $[\alpha, \delta]$ changes per Julian year
<code>px</code>	<code>double</code>	parallax (arcsec)
<code>rv</code>	<code>double</code>	radial velocity (km s^{-1} , +ve if receding)
<code>amprms</code>	<code>double[21]</code>	star-independent mean-to-apparent parameters:
	[1]	◦ 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]	◦ (gravitational radius of Sun) $\times 2 / (\text{Sun-Earth distance})$
	[8-10]	◦ \mathbf{v} : barycentric Earth velocity in units of c
	[11]	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	[12-20]	◦ precession-nutation 3×3 matrix

RETURNED :

<code>ra, da</code>	<code>double*</code>	apparent $[\alpha, \delta]$ (radians)
---------------------	----------------------	---------------------------------------

NOTES :

1. Use of this function 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 `slaMappa` function.
2. If the parallax and proper motions are zero the `slaMapqkz` function can be used instead.
3. The vectors `amprms[1-3]` and `amprms[4-6]` are (in essence) referred to the mean equinox and equator of epoch `eq`. For `eq = 2000.0`, they are referred to the ICRS.
4. Strictly speaking, the function is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about $920''$ of the centre of the Sun's disc. The term has a maximum value of about $1''.85$ at this radius, and decreases to zero as the centre of the disc is approached.

REFERENCES :

1. 1984 *Astronomical Almanac*, pp B39-B41.
2. Lederle & Schwan 1984, *Astr. Astrophys.* **134**, 1-6.

slaMapqkz	quick mean to apparent, no P.M. <i>etc.</i>	slaMapqkz
------------------	---	------------------

ACTION : Quick mean to apparent place: transform a star $[\alpha, \delta]$ from mean place to geocentric apparent place, given the star-independent parameters, and assuming zero parallax and FK5 proper motion. The reference systems and time-scales used are post IAU 1976.

CALL : `slaMapqkz(rm, dm, amprms, &ra, &da);`

GIVEN :

<code>rm, dm</code>	<code>double</code>	mean $[\alpha, \delta]$ (radians)
<code>amprms</code>	<code>double[21]</code>	star-independent mean-to-apparent parameters:
	<code>[0]</code>	◦ time interval for proper motion (Julian years)
	<code>[1-3]</code>	◦ barycentric position of the Earth (AU)
	<code>[4-6]</code>	◦ heliocentric direction of the Earth (unit vector)
	<code>[7]</code>	◦ (gravitational radius of Sun) $\times 2$ /(Sun-Earth distance)
	<code>[8-10]</code>	◦ \mathbf{v} : barycentric Earth velocity in units of c
	<code>[11]</code>	◦ $(1 - \mathbf{v} ^2)^{1/2}$
	<code>[12-20]</code>	◦ precession-nutation 3×3 matrix

RETURNED :

<code>ra, da</code>	<code>double*</code>	apparent $[\alpha, \delta]$ (radians)
---------------------	----------------------	---------------------------------------

NOTES :

1. 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 **slaMappa** function.
2. The corresponding function for the case of non-zero parallax and ICRS proper motion is **slaMapqk**.
3. The vectors `amprms[1-3]` and `amprms[4-6]` are (in essence) referred to the mean equinox and equator of epoch `eq`. For `eq = 2000.0`, they are referred to the ICRS.
4. Strictly speaking, the function is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about $920''$ of the centre of the Sun's disc. The term has a maximum value of about $1''.85$ at this radius, and decreases to zero as the centre of the disc is approached.

REFERENCES :

1. 1984 *Astronomical Almanac*, pp B39-B41.
2. Lederle & Schwan 1984, *Astr.Astrophys.* **134**, 1-6.

slaMoon	approximate Moon position and velocity	slaMoon
----------------	--	----------------

ACTION : Approximate geocentric position and velocity of the Moon (single precision).

CALL : slaMoon(iy, id, fd, pv);

GIVEN :

iy	int	year
id	int	day in year (1 = Jan 1st)
fd	R	fraction of day

RETURNED :

pv	float[6]	Moon $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, mean equator and equinox of date (AU, AU s ⁻¹)
----	----------	--

NOTES :

1. The date and time is TDB (TT will do, but not UTC) in a Julian calendar which has been aligned to the ordinary Gregorian calendar for the interval 1900 March 1 to 2100 February 28. The year and day can be obtained by calling **slaCalyd** or **slaClyd**.
2. The position is accurate to better than 0.5 arcminute in direction and 1000 km in distance. The velocity is accurate to better than 0".5 per hour in direction and 4 metres per second in distance. (RMS figures with respect to JPL DE200 for the interval 1960-2025 are 14" and 0".2 per hour in longitude, 9" and 0".2 per hour in latitude, 350 km and 2 metres per second in distance.) Note that the distance accuracy is comparatively poor because this function is principally intended for computing topocentric direction.
3. This function is only a partial implementation of the original Meeus algorithm (reference below), which offers 4 times the accuracy in direction and 20 times the accuracy in distance when fully implemented (as it is in **slaDmoon**).

REFERENCE : Meeus, *l'Astronomie*, June 1984, p348.

slaMxm	product of two 3×3 matrices	slaMxm
---------------	--------------------------------------	---------------

ACTION : Product of two 3×3 matrices (single precision).

CALL : slaMxm(a, b, c);

GIVEN :

a	float [3] [3]	matrix A
b	float [3] [3]	matrix B

RETURNED :

c	float [3] [3]	matrix result: A × B
----------	---------------	------------------------------------

NOTE : The arguments do not have to be different arrays.

slaMxv	apply 3D rotation	slaMxv
---------------	-------------------	---------------

ACTION : Multiply a 3-vector by a rotation matrix (single precision).

CALL : slaMxv(rm, va, vb);

GIVEN :

rm	float [3] [3]	rotation matrix
va	float [3]	vector to be rotated

RETURNED :

vb	float [3]	result vector
-----------	-----------	---------------

NOTES :

1. This function performs the operation:

$$\mathbf{b} = \mathbf{M} \times \mathbf{a}$$
where **a** and **b** are the 3-vectors **va** and **vb** respectively, and **M** is the 3×3 matrix **rm**.
2. The main function of this function is apply a rotation; under these circumstances, **M** is a *proper real orthogonal* matrix.
3. **va** and **vb** do not have to be different arrays.

slaNu00a	nututation, IAU 2000A	slaNu00a
-----------------	-----------------------	-----------------

ACTION : Nutation, IAU 2000A model (MHB_2000 luni-solar and planetary nutation nutation with free core nutation omitted).

CALL : slaNu00a(date, &dpsi, &deps);

GIVEN :

date double TDB Modified Julian Date (Note 1)

RETURNED :

dpsi,deps double* nutation, luni-solar + planetary (Note 2)

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). TT will do.
2. The nutation components in longitude and obliquity are with respect to the equinox and ecliptic of date. The obliquity at J2000 is assumed to be the Lieske *et al.* (1977) value of $84381''.448$.
3. Both the luni-solar and planetary nutations are included. The latter are due to direct planetary nutations and the perturbations of the lunar and terrestrial orbits.
4. The function computes the MHB_2000 nutation series with the associated corrections for planetary nutations. It is an implementation of the nutation part of the IAU 2000A precession-nutation model, formally adopted by the IAU General Assembly in 2000, namely MHB2000 (Mathews *et al.* 2002), but with the free core nutation (FCN – see Note 5) omitted.
5. The full MHB_2000 model also contains contributions to the nutations in longitude and obliquity due to the free-excitation of the free-core-nutation during the period 1979-2000. These FCN terms, which are time-dependent and unpredictable, are **not** included in the present function and, if required, must be independently computed. With the FCN corrections included, the present function delivers a pole which is at current epochs accurate to a few hundred microarcseconds. Neglecting FCN introduces further errors of about that size.
6. The present function provides classical nutation. The MHB_2000 algorithm, from which it is adapted, deals also with (i) the offsets between the ICRS and mean poles and (ii) the adjustments in longitude and obliquity due to the changed precession rates with respect to IAU 1976 precession.

REFERENCES :

1. Chapront, J., Chapront-Touze, M. & Francou, G. 2002, *Astron.Astrophys.* **387**, 700
2. Lieske, J.H., Lederle, T., Fricke, W. & Morando, B. 1977, *Astron.Astrophys.* **58**, 1-16.
3. Mathews, P.M., Herring, T.A., Buffet, B.A. 2002, *J.Geophys.Res.* **107**, B4. The MHB_2000 code itself was obtained on 9th September 2002 from <ftp://maia.usno.navy.mil/conv2000/chapter5/IAU2000A>.
4. Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M., Francou, G., Laskar, J. 1994, *Astron.Astrophys.* **282**, 663-683

5. Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999, *Astron.Astrophys.Supp.Ser.* **135**, 111

slaNu	nutaton components, IAU 2006	slaNu
--------------	------------------------------	--------------

ACTION : Nutation, IAU 2006 model: IAU 2000A nutation with corrections applied to match the IAU 2006 precession.

CALL : `slaNu(date, &dpsi, &deps);`

GIVEN :

date	double	TDB Modified Julian Date
------	--------	--------------------------

RETURNED :

dpsi,deps	double*	nutation, luni-solar + planetary
-----------	---------	----------------------------------

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). TT will do.
2. The nutation components in longitude and obliquity are with respect to the mean equinox and ecliptic of date, IAU 2006 precession model (Hilton *et al.* 2006, Capitaine *et al.* 2003).
3. The function first computes the IAU 2000A nutation, then applies corrections for (i) the consequences of the change in obliquity from the IAU 1980 ecliptic to the IAU 2006 ecliptic and (ii) the secular variation in the Earth's dynamical flattening, not taken into account in the IAU 2000A model.
4. The present function provides classical nutation, supplementing the IAU 2000 frame bias and IAU 2006 precession. It delivers a pole which is at current epochs accurate to a few tens of microarcseconds, apart from the omitted free core nutation.

REFERENCES :

1. Capitaine, N., Wallace, P.T. & Chapront, J., 2003, *Astron.Astrophys.* **432**, 355-357
2. Hilton, J. *et al.* 2006, *Celest. Mech. Dyn. Astron.* 94, 351)

slaNut	nutation matrix	slaNut
---------------	-----------------	---------------

ACTION : Form the matrix of nutation (SF2001 theory) for a given date.

CALL : `slaNut(date, rmatn);`

GIVEN :

date double TDB as Modified Julian Date (JD−2400000.5)

RETURNED :

rmatn double[3][3] nutation matrix

NOTES :

1. The matrix is in the sense:

$$\mathbf{v}_{true} = \mathbf{M} \times \mathbf{v}_{mean}$$

where \mathbf{v}_{true} is the star vector relative to the true equator and equinox of date, \mathbf{M} is the 3×3 matrix **rmatn** and \mathbf{v}_{mean} is the star vector relative to the mean equator and equinox of date.

2. The matrix represents forced nutation (but not free core nutation) plus corrections to the IAU 1976 precession model.
3. Earth attitude predictions made by combining the present nutation matrix with IAU 1976 precession are accurate to 1 mas (with respect to the ICRS) for a few decades around 2000.
4. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.

REFERENCES :

1. Kaplan, G.H., 1981, *USNO circular No. 163*, pA3-6.
2. Shirai, T. & Fukushima, T. 2001, *Astron.J.*, **121**, 3270-3283.

slaNutc	nutation components	slaNutc
----------------	---------------------	----------------

ACTION : Nutation (SF2001 theory): longitude & obliquity components, and mean obliquity.

CALL : slaNutc (date, &dpsi, &deps, &eps0);

GIVEN :

date double TDB as Modified Julian Date (JD−2400000.5)

RETURNED :

dpsi,deps	double*	nutation in longitude and obliquity (radians)
eps0	double*	mean obliquity (radians)

NOTES :

1. The function predicts forced nutation (but not free core nutation) plus corrections to the IAU 1976 precession model.
2. Earth attitude predictions made by combining the present nutation model with IAU 1976 precession are accurate to 1 mas (with respect to the ICRS) for a few decades around 2000.
3. The `slaNutc80` function is the equivalent of the present function but using the IAU 1980 nutation theory. The older theory is less accurate, leading to errors as large as 350 mas over the interval 1900-2100, mainly because of the error in the IAU 1976 precession.
4. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.

REFERENCES :

1. Shirai, T. & Fukushima, T. 2001, *Astron.J.*, **121**, 3270-3283.
2. Fukushima, T. 1991, *Astron.Astrophys.* **244**, L11.
3. Simon, J. L., Bretagnon, P., Chapront, J., Chapront-Touze, M., Francou, G. & Laskar, J. 1994, *Astron.Astrophys.* **282**, 663.

slaNutc80	nutation components, IAU 1980	slaNutc80
------------------	-------------------------------	------------------

ACTION : Nutation (IAU 1980 theory): longitude & obliquity components, and mean obliquity.

CALL : `slaNutc80 (date, &dpsi, &deps, &eps0);`

GIVEN :

<code>date</code>	<code>double</code>	TDB as Modified Julian Date (JD−2400000.5)
-------------------	---------------------	--

RETURNED :

<code>dpsi,deps</code>	<code>double*</code>	nutation in longitude and obliquity (radians)
<code>eps0</code>	<code>double*</code>	mean obliquity (radians)

NOTES :

1. The IAU 1980 theory used in the present function has errors as large as 350 mas over the interval 1900-2100, mainly because of the error in the IAU 1976 precession. For more accurate results, either the corrections published in IERS *Bulletin B* must be applied, or the `slaNutc` function can be used. The latter is based upon the more recent SF2001 nutation theory and is of better than 1 mas accuracy.
2. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.

REFERENCES :

1. Final report of the IAU Working Group on Nutation, chairman P.K.Seidelmann, 1980.
2. Kaplan, G.H. 1981, *USNO circular no. 163*, pA3-6.

slaO2i	observed place to CIRS	slaO2i
---------------	------------------------	---------------

ACTION : Observed place to CIRS: observed place at a ground based site to CIRS $[\alpha, \delta]$ for sources distant from the solar system.

CALL : sla02i(type, ob1, ob2, utc, dut, elongm, phim,
 hm, xp, yp, tk, phpa, rh, wlfq, tlr, &ri, &di);

GIVEN :

type	char*	type of coordinates – 'R', 'H' or 'A' (see below)
ob1	double	observed Az, HA or RA (radians; Az is N=0, E=90°)
ob2	double	observed zenith distance or δ (radians)
utc	double	UTC date/time (MJD, JD–2400000.5)
dut	double	Δ UT: UT1–UTC (UTC seconds)
elongm	double	observer's mean longitude (radians, east +ve)
phim	double	observer's mean geodetic latitude (radians)
hm	double	observer's height above sea level (metres)
xp,yp	double	polar motion $[x, y]$ coordinates (radians)
tk	double	local ambient temperature (K; std=273.15)
phpa	double	local atmospheric pressure (hPa; std=1013.25)
rh	double	local relative humidity (in the range 0.0–1.0)
wlfq	double	wavelength (μ m) or minus frequency (GHz)
tlr	double	tropospheric lapse rate (K per metre, e.g. 0.0065)

RETURNED :

ri,di	double	CIRS $[\alpha, \delta]$ (radians)
-------	--------	-----------------------------------

NOTES :

1. Only the first character of the **type** argument is significant. 'R' or 'r' indicates that obs1 and obs2 are the observed right ascension and declination; 'H' or 'h' indicates that they are hour angle (west positive) and declination; anything else ('A' or 'a' is recommended) indicates that obs1 and obs2 are azimuth (north zero, east 90°) and zenith distance. (Zenith distance is used rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.)

2. The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about $0''.1$ for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90° , the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the `slaRefro` function (*q.v.*) returns a fixed value of the refraction.

The complementary functions `slaI2o` (or `slaI2oqk`) and `sla02i` (or `sla02iqk`) are self-consistent to better than 1 microarcsecond all over the celestial sphere.

3. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
4. *Observed* [*Az*, *El*] means the position that would be seen by a perfect theodolite located at the observer. This is related to the observed [*h*, δ] via the standard rotation, using the geodetic latitude (corrected for polar motion), while the observed HA and RA are related simply through the local ERA. *Observed* [α , δ] or [*h*, δ] thus means the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). By removing from the observed place the effects of atmospheric refraction and diurnal aberration, the CIRS [α , δ] is obtained.
5. Frequently, ICRS [α , δ] rather than CIRS [α , δ] will be required, in which case further transformations will be necessary. The `slaI2c` *etc.* functions will convert the CIRS [α , δ] produced by the present function into an ICRS astrometric place, by allowing for the Sun's gravitational lens effect, annual aberration, nutation and precession.
6. To convert to CIRS [α , δ] the coordinates read from a real telescope, corrections would have to be applied for encoder zero points, gear and encoder errors, tube flexure, the position of the rotator axis and the pointing axis relative to it, non-perpendicularity between the mounting axes, and finally for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures). Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. This function takes time to execute, due mainly to the rigorous integration used to evaluate the refraction. For processing multiple stars for one location and time, call `slaI2opa` once followed by one call per star to `sla02iqk`. Where a range of times within a limited period of a few hours is involved, and the highest precision is not required, call `slaI2opa` once, followed by a call to `slaI2opat` each time the time changes, followed by one call per star to `sla02iqk`.
8. The `utc` argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the ΔUT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap ΔUT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the ΔUT changes by one second to its post-leap new value.
9. The ΔUT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep ΔUT within $\pm 0''.9$.

10. **Important : take care with the longitude sign convention.** The longitude required by the present function is **east-positive**, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the `slaObs` function are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present function.
11. The polar motion coordinates x_p, y_p can be obtained from IERS bulletins. The maximum amplitude is about $0''.3$. If x_p, y_p values are unavailable, use $x_p = y_p = 0.0$. The x_p, y_p coordinates are reckoned as follows:
- positive x_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 90° .
 - positive y_p means that the point on the geoid directly beneath the CIP is in the hemisphere centred on longitude 270° .

Internally, the polar motion is stored in a form rotated onto the local meridian.

12. The height above sea level of the observing station, `hm`, can be obtained from the *Astronomical Almanac* (Section J in the 1988 edition), or via the function `slaObs`. If `p`, the pressure in hPa (mb) (the same as hPa), is available, an adequate estimate of `hm` can be obtained from the following expression:

$$hm = -29.3 * tsl * \log(p/1013.25)$$

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

$$p = 1013.25 * \exp(-hm/(29.3 * tsl))$$

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

13. The argument `wlfq` specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at $100 \mu m$ (about 3000 GHz).

slaO2iqk	quick observed place to CIRS	slaO2iqk
-----------------	------------------------------	-----------------

ACTION : Quick observed place to CIRS: observed place at a ground based site to CIRS $[\alpha, \delta]$ for sources distant from the solar system, given the star-independent CIRS-to-observed parameters.

CALL : `slaO2iqk(type, ob1, ob2, &ioprms, &ri, &di);`

GIVEN :

type	char*	type of coordinates – ‘R’, ‘H’ or ‘A’ (see below)
ob1	double	observed Az, HA or RA (radians; Az is N=0, E=90°)
ob2	double	observed zenith distance or δ (radians)
ioprms	IOpars*	star-independent CIRS-to-observed parameters:
	along	◦ longitude + s' + dERA(Δ UT) (radians)
	phi	◦ geodetic latitude (radians)
	hm	◦ height above sea level (metres)
	xpl	◦ polar motion x_p wrt local meridian (radians)
	ypl	◦ polar motion y_p wrt local meridian (radians)
	sphi	◦ sine of geodetic latitude
	cphi	◦ cosine of geodetic latitude
	diurab	◦ magnitude of diurnal aberration vector
	p	◦ pressure (hPa = mb)
	tk	◦ ambient temperature (K)
	rh	◦ relative humidity (0–1)
	tlr	◦ tropospheric lapse rate (K per metre)
	wl	◦ wavelength (μ m) or minus frequency (GHz)
	refa	◦ refraction constant A (radians)
	refb	◦ refraction constant B (radians)
	eral	◦ “local” Earth rotation angle (radians)

RETURNED :

ri,di	double	CIRS [α, δ] (radians)
--------------	---------------	-------------------------------------

NOTES :

1. Only the first character of the **type** argument is significant. ‘R’ or ‘r’ indicates that **obs1** and **obs2** are the observed right ascension and declination; ‘H’ or ‘h’ indicates that they are hour angle (west positive) and declination; anything else (‘A’ or ‘a’ is recommended) indicates that **obs1** and **obs2** are azimuth (north zero, east 90°) and zenith distance. (Zenith distance is used rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.)
2. The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about 0".1 for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90°, the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the **slaRefro** function (*q.v.*) returns a fixed value of the refraction.

The complementary functions **slaI2o** (or **slaI2oqk**) and **sla02i** (or **sla02iqk**) are self-consistent to better than 1 microarcsecond all over the celestial sphere.

3. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.

4. *Observed* $[Az, El]$ means the position that would be seen by a perfect theodolite located at the observer. This is related to the observed $[h, \delta]$ via the standard rotation, using the geodetic latitude (corrected for polar motion), while the observed HA and RA are related simply through the local ERA. *Observed* $[\alpha, \delta]$ or $[h, \delta]$ thus means the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). By removing from the observed place the effects of atmospheric refraction and diurnal aberration, the CIRS $[\alpha, \delta]$ is obtained.
5. Frequently, ICRS $[\alpha, \delta]$ rather than CIRS $[\alpha, \delta]$ will be required, in which case further transformations will be necessary. The `slaI2c` *etc.* functions will convert the CIRS $[\alpha, \delta]$ produced by the present function into an ICRS astrometric place, by allowing for the Sun's gravitational lens effect, annual aberration, nutation and precession.
6. To convert to CIRS $[\alpha, \delta]$ the coordinates read from a real telescope, corrections would have to be applied for encoder zero points, gear and encoder errors, tube flexure, the position of the rotator axis and the pointing axis relative to it, non-perpendicularity between the mounting axes, and finally for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures). Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. The star-independent CIRS-to-observed-place parameters in `ioprms` may be computed by means of the `slaI2opa` function. If nothing has changed significantly except the time, the `slaI2opat` function may be used to perform the requisite partial recomputation of `ioprms`.

slaOap	observed to apparent	slaOap
---------------	----------------------	---------------

ACTION : Observed to apparent place.

CALL : `slaOap(type, ob1, ob2, utc, dut, elongm, phim,
 hm, xp, yp, tk, phpa, rh, wlfq, tlr, &rap, &dap);`

GIVEN :

type	char*	type of coordinates – ‘R’, ‘H’ or ‘A’ (see below)
ob1	double	observed Az, HA or RA (radians; Az is N=0, E=90°)
ob2	double	observed zenith distance or δ (radians)
utc	double	UTC date/time (MJD, JD–2400000.5)
dut	double	Δ UT: UT1–UTC (UTC seconds)
elongm	double	observer’s mean longitude (radians, east +ve)
phim	double	observer’s mean geodetic latitude (radians)
hm	double	observer’s height above sea level (metres)
xp,yp	double	polar motion $[x, y]$ coordinates (radians)
tk	double	local ambient temperature (K; std=273.15)
phpa	double	local atmospheric pressure (hPa; std=1013.25)
rh	double	local relative humidity (in the range 0.0–1.0)
wlfq	double	wavelength (μ m) or minus frequency (GHz)
tlr	double	tropospheric lapse rate (K per metre, <i>e.g.</i> 0.0065)

RETURNED :

rap,dap	double*	geocentric apparent $[\alpha, \delta]$
----------------	----------------	--

NOTES :

- Only the first character of the **type** argument is significant. ‘R’ or ‘r’ indicates that **obs1** and **obs2** are the observed right ascension and declination; ‘H’ or ‘h’ indicates that they are hour angle (west positive) and declination; anything else (‘A’ or ‘a’ is recommended) indicates that **obs1** and **obs2** are azimuth (north zero, east 90°) and zenith distance. (Zenith distance is used rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.)
- The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about 0".1 for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90°, the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the **slaRefro** function (*q.v.*) returns a fixed value of the refraction.
The complementary functions **slaAop** (or **slaAopqk**) and **slaOap** (or **slaOapqk**) are self-consistent to better than 1 microarcsecond all over the celestial sphere.
- It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
- Observed* $[Az, El]$ means the position that would be seen by a perfect theodolite located at the observer. This is related to the observed $[h, \delta]$ via the standard rotation, using the geodetic latitude (corrected for polar motion), while the observed HA and RA are related simply through the local apparent ST. *Observed* $[\alpha, \delta]$ or $[h, \delta]$ thus means the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth’s axis of rotation (*n.b.* not to the refracted pole). By removing from the observed place the effects of atmospheric refraction and diurnal aberration, the geocentric apparent $[\alpha, \delta]$ is obtained.

5. Frequently, *mean* rather than *apparent* $[\alpha, \delta]$ will be required, in which case further transformations will be necessary. The `slaAmp` etc. functions will convert the apparent $[\alpha, \delta]$ produced by the present function into an FK5 J2000 mean place, by allowing for the Sun's gravitational lens effect, annual aberration, nutation and precession. Should FK4 B1950 coordinates be required, the functions `slaFk524` etc. will also have to be applied.
6. To convert to apparent $[\alpha, \delta]$ the coordinates read from a real telescope, corrections would have to be applied for encoder zero points, gear and encoder errors, tube flexure, the position of the rotator axis and the pointing axis relative to it, non-perpendicularity between the mounting axes, and finally for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures). Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. This function takes time to execute, due mainly to the rigorous integration used to evaluate the refraction. For processing multiple stars for one location and time, call `slaAoppa` once followed by one call per star to `sla0apqk`. Where a range of times within a limited period of a few hours is involved, and the highest precision is not required, call `slaAoppa` once, followed by a call to `slaAoppat` each time the time changes, followed by one call per star to `sla0apqk`.
8. The `utc` argument is UTC expressed as an MJD. This is, strictly speaking, wrong, because of leap seconds. However, as long as the ΔUT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap ΔUT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the ΔUT changes by one second to its post-leap new value.
9. The ΔUT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep ΔUT within ± 0.9 .
10. **Important : take care with the longitude sign convention.** The longitude required by the present function is **east-positive**, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the `sla0bs` function are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present function.
11. The polar coordinates `xp,yp` can be obtained from IERS circulars and equivalent publications. The maximum amplitude is about $0''.3$. If `xp,yp` values are unavailable, use `xp=yp=0.0`. See page B60 of the 1988 *Astronomical Almanac* for a definition of the two angles.
12. The height above sea level of the observing station, `hm`, can be obtained from the *Astronomical Almanac* (Section J in the 1988 edition), or via the function `sla0bs`. If `p`, the pressure in hPa (mb) (the same as hPa), is available, an adequate estimate of `hm` can be obtained from the following expression:

$$hm = -29.3 * ts1 * \log(p/1013.25)$$

where `ts1` is the approximate sea-level air temperature in K (see *Astrophysical Quantities*, C.W.Allen, 3rd edition, §52). Similarly, if the pressure `p` is not known, it can be estimated from the height of the observing station, `hm` as follows:

$$p = 1013.25 \cdot \exp(-hm / (29.3 \cdot ts1))$$

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

13. The azimuths *etc.* used by the present function are with respect to the celestial pole. The correction needed to refer instead to terrestrial north can be computed using `slaPolmo`.
14. The argument `wlfq` specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at $100 \mu\text{m}$ (about 3000 GHz). $100 \mu\text{m}$

slaOapqk	quick observed to apparent	slaOapqk
-----------------	----------------------------	-----------------

ACTION : Quick observed to apparent place.

CALL : `slaOapqk(type, ob1, ob2, aoprms, &rap, &dap);`

GIVEN :

type	char*	type of coordinates – ‘R’, ‘H’ or ‘A’ (see below)
ob1	double	observed Az, HA or RA (radians; Az is N=0, E=90°)
ob2	double	observed zenith distance or δ (radians)
aoprms	double[14]	star-independent apparent-to-observed parameters:
	[0]	◦ geodetic latitude (radians)
	[1,2]	◦ sine and cosine of geodetic latitude
	[3]	◦ magnitude of diurnal aberration vector
	[4]	◦ height above sea level (metres)
	[5]	◦ temperature (K; std=273.15)
	[6]	◦ pressure (hPa = mb)
	[7]	◦ relative humidity (in the range 0.0–1.0)
	[8]	◦ wavelength (μm) or minus frequency (GHz)
	[9]	◦ tropospheric lapse rate (K per metre)
	[10,11]	◦ refraction constants A and B (radians)
	[12]	◦ longitude + eqn of equinoxes + “sidereal ΔUT ” (radians)
	[13]	◦ local apparent sidereal time (radians)

RETURNED :

<code>rap,dap</code>	double*	geocentric apparent $[\alpha, \delta]$
----------------------	---------	--

NOTES :

1. Only the first character of the `type` argument is significant. 'R' or 'r' indicates that `obs1` and `obs2` are the observed right ascension and declination; 'H' or 'h' indicates that they are hour angle (west positive) and declination; anything else ('A' or 'a' is recommended) indicates that `obs1` and `obs2` are azimuth (north zero, east 90°) and zenith distance. (Zenith distance is used rather than elevation in order to reflect the fact that no allowance is made for depression of the horizon.)
2. The accuracy of the result is limited by the corrections for refraction. Providing the meteorological parameters are known accurately and there are no gross local effects, the predicted azimuth and elevation should be within about 0".1 for $\zeta < 70^\circ$. Even at a topocentric zenith distance of 90°, the accuracy in elevation should normally be better than 1 arcminute; useful results are available a little lower still, beyond which the `slaRefro` function (*q.v.*) returns a fixed value of the refraction.
The complementary functions `slaAop` (or `slaAopqk`) and `slaOap` (or `slaOapqk`) are self-consistent to better than 1 microarcsecond all over the celestial sphere.
3. It is advisable to take great care with units, as even unlikely values of the input parameters are accepted and processed in accordance with the models used.
4. *Observed* [*Az*, *El*] means the position that would be seen by a perfect theodolite located at the observer. This is related to the observed [*h*, δ] via the standard rotation, using the geodetic latitude (corrected for polar motion), while the observed HA and RA are related simply through the local apparent ST. *Observed* [α , δ] or [*h*, δ] thus means the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). By removing from the observed place the effects of atmospheric refraction and diurnal aberration, the geocentric apparent [α , δ] is obtained.
5. Frequently, *mean* rather than *apparent* [α , δ] will be required, in which case further transformations will be necessary. The `slaAmp` *etc.* functions will convert the apparent [α , δ] produced by the present function into an FK5 J2000 mean place, by allowing for the Sun's gravitational lens effect, annual aberration, nutation and precession. Should FK4 B1950 coordinates be required, the functions `slaFk524` *etc.* will also have to be applied.
6. To convert to apparent [α , δ] the coordinates read from a real telescope, corrections would have to be applied for encoder zero points, gear and encoder errors, tube flexure, the position of the rotator axis and the pointing axis relative to it, non-perpendicularity between the mounting axes, and finally for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures). Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. The star-independent apparent-to-observed-place parameters in `aoprms` may be computed by means of the `slaAoppa` function. If nothing has changed significantly except the time, the `slaAoppat` function may be used to perform the requisite partial re-computation of `aoprms`.
8. The azimuths *etc.* used by the present function are with respect to the celestial pole. The correction needed to refer instead to terrestrial north can be computed using `slaPolmo`.

slaObs	observatory parameters	slaObs
---------------	------------------------	---------------

ACTION : Look up an entry in a standard list of groundbased observing stations parameters.

CALL : `slaObs(n, c, name, &w, &p, &h);`

GIVEN :

n	int	number specifying observing station
----------	------------	-------------------------------------

GIVEN or RETURNED :

c	char*	identifier specifying observing station
----------	--------------	---

RETURNED :

name	char*	name of specified observing station
w	double*	longitude (radians, west +ve)
p	double*	geodetic latitude (radians, north +ve)
h	double*	height above sea level (metres)

NOTES :

1. Station identifiers **c** may be up to 10 characters long, and station names **name** may be up to 40 characters long.
2. **c** and **n** are *alternative* ways of specifying the observing station. The **c** option, which is the most generally useful, may be selected by specifying an **n** value of zero or less. If **n** is 1 or more, the parameters of the **n**th station in the currently supported list are interrogated, and the station identifier **c** is returned as well as **name**, **w**, **p** and **h**.
3. If the station parameters are not available, either because the station identifier **c** is not recognized, or because an **n** value greater than the number of stations supported is given, a name of "?" is returned and **w**, **p** and **h** are left in their current states.
4. Programs can obtain a list of all currently supported stations by calling the function repeatedly, with **n** = 1,2,3... When **name** = "?" is seen, the list of stations has been exhausted. The stations at the time of writing are listed below.
5. Station numbers, identifiers, names and other details are subject to change and should not be hardwired into application programs.
6. All station identifiers **c** are uppercase only; lower case characters must be converted to uppercase by the calling program. The station names returned may contain both upper- and lowercase. All characters 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.

7. **Important : take care with the longitude sign convention.** The longitude returned by `slaObs` is **west-positive**, following the pre-1984 *Astronomical Almanac*. However, this sign convention is left-handed and is the opposite of the one now used; elsewhere in SLALIB the preferable east-positive convention is used. In particular, note that for use in `slaAop`, `slaAoppa` and `slaOap` the sign of the longitude must be reversed.
8. Feedback is invited, to inform the author of possible improvements and extensions. For example:
- typographical corrections
 - more accurate parameters
 - better station identifiers or names
 - additional stations

Stations supported by `slaObs` at the time of writing:

<i>ID</i>	<i>name</i>
AAT	Anglo-Australian 3.9m Telescope
ANU2.3	Siding Spring 2.3m
AP03.5	Apache Point 3.5m
ARECIBO	Arecibo 1000 foot
ATCA	Australia Telescope Compact Array
BLOEMF	Bloemfontein 1.52m
BOSQALEGRE	Bosque Alegre 1.54m
CAMB1MILE	Cambridge 1 mile
CAMB5KM	Cambridge 5 km
CATALINA61	Catalina 61 inch
CFHT	Canada-France-Hawaii 3.6m Telescope
CSO	Caltech Sub-mm Observatory, Mauna Kea
DAO72	DAO Victoria BC 1.85m
DUNLAP74	David Dunlap 74 inch
DUPONT	Du Pont 2.5m Telescope, Las Campanas
EFFELSBERG	Effelsberg 100m
ESO3.6	ESO 3.6m
ESONTT	ESO 3.5m NTT
ESOSCHM	ESO 1m Schmidt, La Silla
FCRAO	Five College Radio Astronomy Obs
FLAGSTF61	USNO 61 inch astrograph, Flagstaff
GBVA140	Greenbank 140 foot
GBVA300	Greenbank 300 foot
GEMININ	Gemini North 8m
GEMINIS	Gemini South 8m
HARVARD	Harvard College Observatory 1.55m
HET	Hobby-Eberly Telescope
HPROV1.52	Haute Provence 1.52m
HPROV1.93	Haute Provence 1.93m
IRTF	NASA IR Telescope Facility, Mauna Kea
JCMT	JCMT 15m

JODRELL1	Jodrell Bank 250 foot
KECK1	Keck 10m Telescope 1
KECK2	Keck 10m Telescope 2
KISO	Kiso 1.05m Schmidt, Japan
KOSMA3M	Cologne Submillimeter Observatory 3m
KOTTAMIA	Kottamia 74 inch
KPNO158	Kitt Peak 158 inch
KPNO36FT	Kitt Peak 36 foot
KPNO84	Kitt Peak 84 inch
KPNO90	Kitt Peak 90 inch
LICK120	Lick 120 inch
LOWELL72	Perkins 72 inch, Lowell
LP01	Jacobus Kapteyn 1m Telescope
LP02.5	Isaac Newton 2.5m Telescope
LP04.2	William Herschel 4.2m Telescope
MAGELLAN1	Magellan 1, 6.5m, Las Campanas
MAGELLAN2	Magellan 2, 6.5m, Las Campanas
MAUNAK88	Mauna Kea 88 inch
MCDONLD2.1	McDonald 2.1m
MCDONLD2.7	McDonald 2.7m
MMT	MMT, Mt Hopkins
MOPRA	ATNF Mopra Observatory
MTEKAR	Mt Ekar 1.82m
MTHOP1.5	Mt Hopkins 1.5m
MTLEMMON60	Mt Lemmon 60 inch
NOBEYAMA	Nobeyama 45m
OKAYAMA	Okayama 1.88m
PALOMAR200	Palomar 200 inch
PALOMAR48	Palomar 48-inch Schmidt
PALOMAR60	Palomar 60 inch
PARKES	Parkes 64m
QUEBEC1.6	Quebec 1.6m
SAA074	Sutherland 74 inch
SANPM83	San Pedro Martir 83 inch
ST.ANDREWS	St Andrews University Observatory
STEWART90	Steward 90 inch
STROMLO74	Mount Stromlo 74 inch
SUBARU	Subaru 8m
SUGARGROVE	Sugar Grove 150 foot
TAUTNBG	Tautenburg 2m
TAUTSCHM	Tautenberg 1.34m Schmidt
TIDBINBLA	Tidbinbilla 64m
TOLOLO1.5M	Cerro Tololo 1.5m
TOLOLO4M	Cerro Tololo 4m
UKIRT	UK Infra Red Telescope
UKST	UK 1.2m Schmidt, Siding Spring
USSR6	USSR 6m
USSR600	USSR 600 foot

VLA	Very Large Array
VLT1	ESO VLT 8m, UT1
VLT2	ESO VLT 8m, UT2
VLT3	ESO VLT 8m, UT3
VLT4	ESO VLT 8m, UT4

slaPa	$[h, \delta]$ to parallactic angle	slaPa
--------------	------------------------------------	--------------

ACTION : Hour angle and declination to parallactic angle (double precision).

CALL : `d = slaPa(ha, dec, phi);`

GIVEN :

ha	double	hour angle in radians (geocentric apparent)
dec	double	declination in radians (geocentric apparent)
phi	double	latitude in radians (geodetic)

RETURNED :

double	parallactic angle (radians, in the range $\pm\pi$)
--------	---

NOTES :

1. 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 $[h, \delta]$ and to consider separately the effects of atmospheric refraction and telescope mount errors.
2. At the pole a zero result is returned.
3. There is some debate about whether “parallactic angle” is the correct term to use here. What is certain, however, is that the meaning is conventionally as presented in Note 1.

slaPav	position-angle between two directions	slaPav
---------------	---------------------------------------	---------------

ACTION : Returns the bearing (position angle) of one celestial direction with respect to another (single precision).

CALL : `r = slaPav(v1, v2);`

GIVEN :

v1	float[3]	vector to one point
v2	float[3]	vector to the other point

RETURNED :

float	position-angle of second point with respect to first
--------------	--

NOTES :

1. The coordinate frames correspond to $[\alpha, \delta]$, $[\lambda, \beta]$ *etc.*
2. The result is the bearing (position angle), in radians, of point **v2** as seen from point **v1**. It is in the range $\pm\pi$. The sense is such that if **v2** is a small distance due east of **v1** the result is about $+\pi/2$. Zero is returned if the two points are coincident.
3. There is no requirement for either vector to be of unit length.
4. The function **slaBear** performs an equivalent function except that the points are specified in the form of spherical coordinates.

slaPcd	apply radial distortion	slaPcd
---------------	-------------------------	---------------

ACTION : Apply pincushion/barrel distortion to a tangent-plane $[x, y]$.

CALL : `slaPcd(disco, &x, &y);`

GIVEN :

disco	double	pincushion/barrel distortion coefficient
x,y	double*	tangent-plane $[x, y]$

RETURNED :

x,y	double*	distorted $[x, y]$
------------	----------------	--------------------

NOTES :

1. The distortion is of the form $\rho = r(1 + cr^2)$, where r is the radial distance from the tangent point, c is the **disco** argument, and ρ is the radial distance in the presence of the distortion.
2. For *pincushion* distortion, c is positive; for *barrel* distortion, c is negative.
3. For x, y in units of one projection radius (in the case of a photographic plate, the focal length), the following **disco** values apply:

Geometry	disco
astrograph	0.0
Schmidt	-0.3333
AAT PF doublet	+147.069
AAT PF triplet	+178.585
AAT f/8	+21.20
JKT f/8	+14.6

4. There is a companion function, **slaUnpcd**, which performs the inverse operation.

slaPda2h	hour angle for a given azimuth	slaPda2h
-----------------	--------------------------------	-----------------

ACTION : Hour angle corresponding to a given azimuth (double precision).

CALL : `slaPda2h(p, d, a, &h1, &j1, &h2, &j2);`

GIVEN :

p	double	latitude
d	double	declination
a	double	azimuth

RETURNED :

h1	double*	hour angle: first solution if any
j1	int*	flag: 0 = solution 1 is valid
h2	double*	hour angle: second solution if any
j2	int*	flag: 0 = solution 2 is valid

slaPdq2h	hour angle for a given parallactic angle	slaPdq2h
-----------------	--	-----------------

ACTION : Hour angle corresponding to a given parallactic angle (double precision).

CALL : `slaPdq2h(p, d, q, &h1, &j1, &h2, &j2);`

GIVEN :

p	double	latitude
d	double	declination
q	double	azimuth

RETURNED :

h1	double*	hour angle: first solution if any
j1	int*	flag: 0 = solution 1 is valid
h2	double*	hour angle: second solution if any
j2	int*	flag: 0 = solution 2 is valid

NOTE : The *parallactic angle* at a point in the sky (as conventionally defined – there is some dispute about whether the term is correctly used for this purpose) is the position angle of the vertical, *i.e.* the angle between the directions to the pole and zenith.

slaPermut	next permutation	slaPermut
------------------	------------------	------------------

ACTION : Generate the next permutation of a specified number of items.

CALL : slaPermut(n, istate, iorder, &j);

GIVEN :

n	int	number of items: there will be n! permutations
istate	int[n]	state, istate[0] = -1 to initialize

RETURNED :

istate	int[n]	state, updated ready for next time
iorder	int[n]	next permutation of numbers 1,2,...,n
j	int*	status:
		-1 = illegal n (zero or less is illegal)
		0 = OK
		+1 = no more permutations available

NOTES :

1. This function returns, in the **iorder** array, the integers 1 to **n** inclusive, in an order that depends on the current contents of the **istate** array. Before calling the function for the first time, the caller must set the first element of the **istate** array to -1 (any negative number will do) to cause the **istate** array to be fully initialized.
2. The first permutation to be generated is:

$$\mathbf{iorder}[0] = \mathbf{n}, \mathbf{iorder}[1] = \mathbf{n} - 1, \dots, \mathbf{iorder}[\mathbf{n} - 1] = 1$$
This is also the permutation returned for the “finished” ($j = 1$) case. The final permutation to be generated is:

$$\mathbf{iorder}[0] = 1, \mathbf{iorder}[1] = 2, \dots, \mathbf{iorder}[\mathbf{n} - 1] = \mathbf{n}$$
3. If the “finished” ($j = 1$) status is ignored, the function continues to deliver permutations, the pattern repeating every $\mathbf{n}!$ calls.

slaPertel	perturbed orbital elements	slaPertel
------------------	----------------------------	------------------

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

CALL : `slaPertel(jform, date0, date1,
 epoch0, orbi0, anode0, perih0, aorq0, e0, am0,
 &epoch1, &orbi1, &anode1, &perih1, &aorq1, &e1, &am1,
 &jstat);`

GIVEN (format and dates) :

jform	int	choice of element set (2 or 3; Note 1)
date0	double	date of osculation (TDB MJD) for the given elements
date1	double	date of osculation (TDB MJD) for the updated elements

GIVEN (the unperturbed elements) :

epoch0	double	epoch of the given element set (t_0 or T , TDB MJD;
		Note 2)
orbi0	double	inclination (i , radians)
anode0	double	longitude of the ascending node (Ω , radians)
perih0	double	argument of perihelion (ω , radians)
aorq0	double	mean distance or perihelion distance (a or q , AU)
e0	double	eccentricity (e)
am0	double	mean anomaly (M , radians, jform = 2 only)

RETURNED (the updated elements) :

epoch1	double	epoch of the updated element set (t_0 or T , TDB MJD; Note 2)
orbi1	double	inclination (i , radians)
anode1	double	longitude of the ascending node (Ω , radians)
perih1	double	argument of perihelion (ω , radians)
aorq1	double	mean distance or perihelion distance (a or q , AU)
e1	double	eccentricity (e)
am1	double	mean anomaly (M , radians, jform = 2 only)

RETURNED (status flag) :

jstat	int*	status:
		+102 = warning, distant epoch
		+101 = warning, large timespan (> 100 years)
		+1 to +10 = coincident with major planet (Note 6)
		0 = OK
		-1 = illegal jform
		-2 = illegal e0
		-3 = illegal aorq0
		-4 = internal error
		-5 = numerical error

NOTES :

1. Two different element-format options are supported, as follows.

jform = 2, suitable for minor planets:

epoch	=	epoch of elements t_0 (TDB MJD)
orbinc	=	inclination i (radians)
anode	=	longitude of the ascending node Ω (radians)
perih	=	argument of perihelion ω (radians)
aorq	=	mean distance a (AU)
e	=	eccentricity e ($0 \leq e < 1$)
aorl	=	mean anomaly M (radians)

jform = 3, suitable for comets:

epoch	=	epoch of perihelion T (TDB MJD)
orbinc	=	inclination i (radians)
anode	=	longitude of the ascending node Ω (radians)
perih	=	argument of perihelion ω (radians)
aorq	=	perihelion distance q (AU)
e	=	eccentricity e ($0 \leq e \leq 10$)

2. **date0**, **date1**, **epoch0** and **epoch1** are all instants of time in the TDB time-scale (TT will do), 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.
3. The elements are with respect to the J2000 ecliptic and mean equinox.
4. Unused elements (**am0** and **am1** for **jform** = 3) are not accessed.
5. See the **slaPertue** function for details of the algorithm used.
6. This function 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 function, sensible results will, in fact, be produced. This happens because the **slaPertue** function 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.

REFERENCE : Sterne, Theodore E., *An Introduction to Celestial Mechanics*, Interscience Publishers, 1960. Section 6.7, p199.

slaPertue	perturbed universal elements	slaPertue
------------------	------------------------------	------------------

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

CALL : **slaPertue**(**date**, **u**, **&jstat**);

GIVEN :

date	double	final epoch (TDB MJD) for the updated elements
-------------	---------------	--

GIVEN and RETURNED :

u	double[13]	universal elements (updated in place)
	[0]	◦ combined mass ($M + m$)
	[1]	◦ total energy of the orbit (α)
	[2]	◦ reference (osculating) epoch (t_0)
	[3–5]	◦ position at reference epoch (\mathbf{r}_0)
	[6–8]	◦ velocity at reference epoch (\mathbf{v}_0)
	[9]	◦ heliocentric distance at reference epoch
	[10]	◦ $\mathbf{r}_0 \cdot \mathbf{v}_0$
	[11]	◦ date (t)
	[12]	◦ universal eccentric anomaly (ψ) of date, approx.

RETURNED :

jstat	int*	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 :

1. 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) α , 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 ψ , the “universal eccentric anomaly” at a given date and (v) that date.
2. The universal elements are with respect to the J2000 equator and equinox.
3. The epochs **date**, **u[2]** and **u[11]** are all TDBs (TT will do) as Modified Julian Dates (JD−2400000.5).
4. 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 & 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 function 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 & 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 are obtained from the function `slaPlanet`. Two levels of interpolation are used, to enhance speed without significantly degrading accuracy. At a low frequency, the function `slaPlanet` 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 (internal) parameter `RNE`, which incurs a time penalty but improves accuracy for near-Earth objects.

5. This function 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 function 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 function 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.

REFERENCES :

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

slaPfw	IAU 2006 precession angles	slaPfw
--------	----------------------------	--------

ACTION : Precession angles, IAU 2006 (Fukushima-Williams 4-angle formulation).

CALL : slaPfw(date, &gamb, &phib, &psib, &epsa);

GIVEN :

date	double	TDB Modified Julian Date
------	--------	--------------------------

RETURNED :

gamb	double*	F-W angle $\bar{\gamma}$ (radians)
phib	double*	F-W angle $\bar{\phi}$ (radians)
psib	double*	F-W angle $\bar{\psi}$ (radians)
epsa	double*	F-W angle ϵ_A (radians)

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). TT will do.
2. Naming the following points:
 e = J2000 ecliptic pole,
 p = GCRS pole,
 E = mean ecliptic pole of date, and
 P = mean pole of date,

the four Fukushima-Williams angles are as follows:

$$\begin{aligned} \text{gamb} &= \bar{\gamma} = epE \\ \text{phib} &= \bar{\phi} = pE \\ \text{psib} &= \bar{\psi} = pEP \\ \text{epsa} &= \epsilon_A = EP \end{aligned}$$

3. The matrix representing the combined effects of frame bias **B** and precession **P** is:

$$\mathbf{P} \cdot \mathbf{B} = R_1(-\epsilon_A) \cdot R_3(-\bar{\psi}) \cdot R_1(\bar{\phi}) \cdot R_3(\bar{\gamma})$$

4. The matrix representing the combined effects of frame bias **B**, precession **P** and nutation **N** is simply:

$$\mathbf{N} \cdot \mathbf{P} \cdot \mathbf{B} = R_1(-\epsilon) \cdot R_3(-\psi) \cdot R_1(\bar{\phi}) \cdot R_3(\bar{\gamma})$$

where $\psi = \bar{\psi} + \Delta\psi$ and $\epsilon = \bar{\epsilon} + \Delta\epsilon$, and $\Delta\psi$ and $\Delta\epsilon$ are the nutation components with respect to the ecliptic of date.

REFERENCE : Hilton, J. *et al.* 2006, *Celest. Mech. Dyn. Astron.* 94, 351)

slaPlanel	planet position from elements	slaPlanel
------------------	-------------------------------	------------------

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

CALL : slaPlanel(date, jform, epoch, orbinc, anode, perih,
aorq, e, aorl, dm, pv, &jstat);

GIVEN :

date	double	TDB MJD of observation (JD−2400000.5, Note 1)
jform	int	choice of element set (1-3, Note 3)
epoch	double	epoch of elements (t_0 or T , TDB MJD, Note 4)
orbinc	double	inclination (i , radians)
anode	double	longitude of the ascending node (Ω , radians)
perih	double	longitude or argument of perihelion (ϖ or ω , radians)
aorq	double	mean distance or perihelion distance (a or q , AU)
e	double	eccentricity (e)
aorl	double	mean anomaly or longitude (M or L , radians, jform = 1, 2 only)
dm	double	daily motion (n , radians, jform = 1 only)

RETURNED :

pv	double[6]	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s)
jstat	int*	status: 0 = OK −1 = illegal jform −2 = illegal e −3 = illegal aorq −4 = illegal dm −5 = numerical error

NOTES :

1. DATE is the instant for which the prediction is required. It is in the TDB time-scale (TT will do) and is a Modified Julian Date (JD−2400000.5).
2. The elements are with respect to the J2000 ecliptic and equinox.

3. A choice of three different element-format options is available, as follows.

`jform = 1`, suitable for the major planets:

```
epoch  = epoch of elements  $t_0$  (TDB MJD)
orbinc = inclination  $i$  (radians)
anode  = longitude of the ascending node  $\Omega$  (radians)
perih  = longitude of perihelion  $\varpi$  (radians)
aorq   = mean distance  $a$  (AU)
e      = eccentricity  $e$ 
aorl   = mean longitude  $L$  (radians)
dm     = daily motion  $n$  (radians)
```

`jform = 2`, suitable for minor planets:

```
epoch  = epoch of elements  $t_0$  (TDB MJD)
orbinc = inclination  $i$  (radians)
anode  = longitude of the ascending node  $\Omega$  (radians)
perih  = argument of perihelion  $\omega$  (radians)
aorq   = mean distance  $a$  (AU)
e      = eccentricity  $e$ 
aorl   = mean anomaly  $M$  (radians)
```

`jform = 3`, suitable for comets:

```
epoch  = epoch of perihelion  $T$  (TDB MJD)
orbinc = inclination  $i$  (radians)
anode  = longitude of the ascending node  $\Omega$  (radians)
perih  = argument of perihelion  $\omega$  (radians)
aorq   = perihelion distance  $q$  (AU)
e      = eccentricity  $e$ 
```

Unused elements (`dm` for `jform = 2`, `aorl` and `dm` for `jform = 3`) are not accessed.

4. 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 perturbations, 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 function:

- 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 **slaPertel** may be used to update the element-set (and its associated osculating epoch) by applying planetary perturbations.
5. The reference frame for the result is equatorial and is with respect to the mean equinox and ecliptic of epoch J2000.
 6. The algorithm was originally adapted from the **EPHSLA** Fortran program of D. H. P. Jones (private communication, 1996). The method is based on Stumpff's Universal Variables.

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

slaPlanet	planetary ephemerides, approximate	slaPlanet
------------------	------------------------------------	------------------

ACTION : Approximate heliocentric position and velocity of a planet.

CALL : `slaPlanet(date, np, pv, &jstat);`

GIVEN :

date	double	Modified Julian Date (JD−2400000.5)
np	int	body:
		1 = Mercury
		2 = Venus
		3 = Earth-Moon Barycentre
		4 = Mars
		5 = Jupiter
		6 = Saturn
		7 = Uranus
		8 = Neptune
		9 = Pluto

RETURNED :

pv	double[6]	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s)
jstat	int*	status: +1 = warning: date outside of range 0 = OK -1 = illegal np (outside 1-9) -2 = solution didn't converge

NOTES :

1. The epoch, **date**, is in the TDB time scale and is in the form of a Modified Julian Date (JD-2400000.5). TT will do, but not UTC.
2. The reference frame is equatorial and is with respect to the mean equinox and ecliptic of epoch J2000.
3. If a planet number, **np**, outside the range 1-9 is supplied, an error status is returned (**jstat** = -1) and the **pv** vector is set to zeroes.
4. The algorithm for obtaining the mean elements of the planets (Mercury to Neptune) is due to J.L. Simon, P. Bretagnon, J. Chapront, M. Chapront-Touze, G. Francou and J. Laskar (Bureau des Longitudes, Paris, France). The (completely different) algorithm for calculating the ecliptic coordinates of the dwarf planet Pluto is by Meeus.
5. Comparisons of the present function with the JPL DE200 ephemeris give the following RMS errors over the interval 1960-2025:

	<i>position (km)</i>	<i>speed (metre/sec)</i>
Mercury	334	0.437
Venus	1060	0.855
EMB	2010	0.815
Mars	7690	1.98
Jupiter	71700	7.70
Saturn	199000	19.4
Uranus	564000	16.4
Neptune	158000	14.4
Pluto	36400	0.137

From comparisons with DE102, Simon *et al.* quote the following longitude accuracies over the interval 1800-2200:

Mercury	4"
Venus	5"
EMB	6"
Mars	17"
Jupiter	71"
Saturn	81"
Uranus	86"
Neptune	11"

In the case of Pluto, Meeus quotes an accuracy of 0".6 in longitude and 0".2 in latitude for the period 1885-2099.

For all except Pluto, over the period 1000-3000, the accuracy is better than 1.5 times that over 1800-2200. Outside the interval 1000-3000 the accuracy declines. For Pluto the accuracy declines rapidly outside the period 1885-2099. Outside these ranges (1885-2099 for Pluto, 1000-3000 for the rest) a "date out of range" warning status (`jstat = +1`) is returned.

6. The algorithms for (i) Mercury through Neptune and (ii) Pluto are completely independent. In the Mercury through Neptune case, the present SLALIB implementation differs from the original Simon *et al.* Fortran code in the following respects:
 - The date is supplied as a Modified Julian Date rather a Julian Date ($\text{MJD} = \text{JD} - 2400000.5$).
 - The result is returned only in equatorial Cartesian form; the ecliptic longitude, latitude and radius vector are not returned.
 - The velocity is in AU per second, not AU per day.
 - Different error/warning status values are used.
 - Kepler's Equation is not solved inline.
 - Polynomials in T are nested to minimize rounding errors.
 - Explicit double-precision constants are used to avoid mixed-mode expressions.
 - There are other, cosmetic, changes to comply with Starlink/SLALIB style guidelines.

None of the above changes affects the result significantly.

7. For `np = 3` the result is for the Earth-Moon Barycentre. To obtain the heliocentric position and velocity of the Earth, either use the SLALIB function `slaEvp` (or `slaEpv`) or call `slaDmoon` and subtract 0.012150581 times the geocentric Moon vector from the EMB vector produced by the present function. (The Moon vector should be pre-processed to J2000 first, but this can be omitted for modern epochs without introducing significant inaccuracy.)

REFERENCES :

1. Simon *et al.* 1994, *Astron.Astrophys.* **282**, 663.
2. Meeus, J. 1991. *Astronomical Algorithms*, Willmann-Bell.

slaPlante	$[\alpha, \delta]$ of planet from elements	slaPlante
------------------	--	------------------

ACTION : Topocentric apparent $[\alpha, \delta]$ of a Solar-System object whose heliocentric orbital elements are known.

CALL : `slaPlante(date, elong, phi, jform, epoch, orbinc, anode, perih, aorq, e, aorl, dm, &ra, &dec, &r, &jstat);`

GIVEN :

date	double	TDB MJD of observation (JD–2400000.5, Notes 1,5)
elong,phi	double	observer's longitude (east +ve) and latitude (radians, Note 2)
jform	int	choice of element set (1-3, Notes 3-6)
epoch	double	epoch of elements (t_0 or T , TDB MJD, Note 5)
orbinc	double	inclination (i , radians)
anode	double	longitude of the ascending node (Ω , radians)
perih	double	longitude or argument of perihelion (ϖ or ω , radians)
aorq	double	mean distance or perihelion distance (a or q , AU)
e	double	eccentricity (e)
aorl	double	mean anomaly or longitude (M or L , radians, jform = 1,2 only)
dm	double	daily motion (n , radians, jform = 1 only)

RETURNED :

ra,dec	double*	topocentric apparent $[\alpha, \delta]$ (radians)
r	double*	distance from observer (AU)
jstat	int*	status: 0 = OK –1 = illegal jform –2 = illegal e –3 = illegal aorq –4 = illegal dm –5 = numerical error

NOTES :

1. **date** is the instant for which the prediction is required. It is in the TDB time-scale (TT will do) and is a Modified Julian Date (JD–2400000.5).
2. 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 the functions **slaEvp** (or **slaEpv**) and **slaPlanel**.
3. The elements are with respect to the J2000 ecliptic and equinox.
4. A choice of three different element-format options is available, as follows.

jform = 1, suitable for the major planets:

epoch	=	epoch of elements t_0 (TDB MJD)
orbinc	=	inclination i (radians)
anode	=	longitude of the ascending node Ω (radians)

perih = longitude of perihelion ϖ (radians)
aorq = mean distance a (AU)
e = eccentricity e
aorl = mean longitude L (radians)
dm = daily motion n (radians)

jform = 2, suitable for minor planets:

epoch = epoch of elements t_0 (TDB MJD)
orbinc = inclination i (radians)
anode = longitude of the ascending node Ω (radians)
perih = argument of perihelion ω (radians)
aorq = mean distance a (AU)
e = eccentricity e
aorl = mean anomaly M (radians)

jform = 3, suitable for comets:

epoch = epoch of perihelion T (TDB MJD)
orbinc = inclination i (radians)
anode = longitude of the ascending node Ω (radians)
perih = argument of perihelion ω (radians)
aorq = perihelion distance q (AU)
e = eccentricity e

Unused elements (**dm** for **jform** = 2, **aorl** and **dm** for **jform** = 3) are not accessed.

5. 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 perturbations, 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 function:

- 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 **slaPertel** may be used to update the element-set (and its associated osculating epoch) by applying planetary perturbations.
6. 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. For further details, see Section 3.20.

slaPlantu	$[\alpha, \delta]$ from universal elements	slaPlantu
------------------	--	------------------

ACTION : Topocentric apparent $[\alpha, \delta]$ of a Solar-System object whose heliocentric universal orbital elements are known.

CALL : `slaPlantu(date, elong, phi, u, &ra, &dec, &r, &jstat);`

GIVEN :

<code>date</code>	double	TDB MJD of observation (JD−2400000.5)
<code>elong, phi</code>	double	observer's longitude (east +ve) and latitude (radians)

GIVEN and RETURNED :

<code>u</code>	double[13]	universal orbital elements
	[0]	◦ combined mass ($M + m$)
	[1]	◦ total energy of the orbit (α)
	[2]	◦ reference (osculating) epoch (t_0)
	[3–5]	◦ position at reference epoch (\mathbf{r}_0)
	[6–8]	◦ velocity at reference epoch (\mathbf{v}_0)
	[9]	◦ heliocentric distance at reference epoch
	[10]	◦ $\mathbf{r}_0 \cdot \mathbf{v}_0$
	[11]	◦ date (t)
	[12]	◦ universal eccentric anomaly (ψ) of date, approx.

RETURNED :

<code>ra, dec</code>	double*	topocentric apparent $[\alpha, \delta]$ (radians)
<code>r</code>	double*	distance from observer (AU)
<code>jstat</code>	int*	status:
		0 = OK
		−1 = radius vector zero
		−2 = failed to converge

NOTES :

1. **date** is the instant for which the prediction is required. It is in the TDB time-scale (TT will do) and is a Modified Julian Date (JD−2400000.5).
2. 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 the functions **slaEvp** (or **slaEpv**) and **slaUe2pv**.
3. 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) α , 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 ψ , the “universal eccentric anomaly” at a given date and (v) that date.
4. The universal elements are with respect to the J2000 ecliptic and equinox.

REFERENCES :

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

slaPm	proper motion	slaPm
--------------	---------------	--------------

ACTION : Apply corrections for proper motion to a star $[\alpha, \delta]$.

CALL : `slaPm(r0, d0, pr, pd, px, rv, ep0, ep1, &r1, &d1);`

GIVEN :

r0,d0	double	$[\alpha, \delta]$ at epoch ep0 (radians)
pr,pd	double	proper motions: rate of change of $[\alpha, \delta]$ (radians per year)
px	double	parallax (arcsec)
rv	double	radial velocity (km s^{-1} , +ve if receding)
ep0	double	start epoch in years (<i>e.g.</i> Julian epoch)
ep1	double	end epoch in years (same system as ep0)

RETURNED :

r1,d1 **double*** $[\alpha, \delta]$ at epoch **ep1** (radians)

NOTES :

1. The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha} \cos \delta$, and are in the same coordinate system as **r0,d0**.
2. If the available proper motions are pre-FK5 they will be per tropical year rather than per Julian year, and so the epochs must both be Besselian rather than Julian. In such cases, a scaling factor of 365.2422/365.25 should be applied to the radial velocity before use also.

REFERENCES :

1. 1984 *Astronomical Almanac*, pp B39-B41.
2. Lederle & Schwan 1984, *Astr. Astrophys.*, **134**, 1-6.

slaPncio CIO based NPB matrix **slaPncio**

ACTION : Form the CIO based precession-nutation matrix.

CALL : `slaPncio(date, rnpb);`

GIVEN :

date **double** TDB Modified Julian Date

RETURNED :

rnpb **double[3][3]** NPB matrix, CIO based

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). TT will do.
2. The matrix is in the sense:

$$\mathbf{v}_{CIRS} = \mathbf{M} \times \mathbf{v}_{GCRS}$$

where \mathbf{v}_{CIRS} is the star vector relative to the CIP equator and CIO of date, \mathbf{M} is the 3×3 matrix **rnpb** and \mathbf{v}_{GCRS} is the star vector in the Geocentric Celestial Reference System.

slaPneqx	equinox based NPB matrix	slaPneqx
-----------------	--------------------------	-----------------

ACTION : Form the classical precession-nutation matrix, with respect to the GCRS.

CALL : slaPneqx(date, rnpb);

GIVEN :

date	double	TDB Modified Julian Date
------	--------	--------------------------

RETURNED :

rnpb	double[3][3]	NPB matrix, equinox based
------	--------------	---------------------------

NOTES :

1. The date is TDB as an MJD (= JD−2400000.5). TT will do.
2. The matrix is in the sense:

$$\mathbf{v}_{true} = \mathbf{M} \times \mathbf{v}_{GCRS}$$

where \mathbf{v}_{true} is the star vector relative to the true equator and equinox of date, \mathbf{M} is the 3×3 matrix **rnpb** and \mathbf{v}_{GCRS} is the star vector in the Geocentric Celestial Reference System.

slaPolmo	polar motion	slaPolmo
-----------------	--------------	-----------------

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

CALL : slaPolmo(elongm, phim, xp, yp, &elong, &phi, &daz);

GIVEN :

elongm	double	mean longitude of the site (radians, east +ve)
phim	double	mean geodetic latitude of the site (radians)
xp	double	polar motion x -coordinate (radians)
yp	double	polar motion y -coordinate (radians)

RETURNED :

elong	double*	true longitude of the site (radians, east +ve)
phi	double*	true geodetic latitude of the site (radians)
daz	double*	azimuth correction (terrestrial−celestial, radians)

NOTES :

1. Polar motion can be regarded as those components of the changing orientation of the Earth's axis that are around 1 day in period, compared with the much slower precession-nutation effects. The simplest way of dealing with polar motion is to apply a full 3D coordinate rotation, transforming between the International Celestial Reference System (ITRS) and the Terrestrial Intermediate Reference System. However, it has long been a tradition instead to regard polar motion as something that leads to small variations in the latitude and longitude of the observing site, and the SLALIB functions `slaAop` *etc.* work in this way. However, the traditional treatment omits one component of the transformation, namely the changing azimuth of the observing site reckoned with respect to the Celestial Intermediate Pole (CIP). The present function addresses that missing element.
2. "Mean" longitude and latitude are the (quasi-fixed) values for the site's location with respect to the ITRS; the latitude is geodetic. **Important : take care with the longitude sign convention.** The longitudes used by the present function are east-positive, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the `slaObs` function are west-positive, following astronomical usage, and must be reversed in sign before use in the present function.
3. `xp` and `yp` are the (changing) coordinates of the CIP with respect to the ITRS pole. `xp` is positive along the meridian at longitude 0° , and `yp` is positive along the meridian at longitude 270° (*i.e.* 90° west). Values for `xp,yp` can be obtained from IERS circulars and equivalent publications; the maximum amplitude observed so far is about $0''.3$.
4. "True" longitude and latitude are the (changing) values for the site's location with respect to the CIP and the ITRS meridian which corresponds to the Greenwich apparent sidereal time (for right ascensions referred to the equinox; Earth Rotation Angle for right ascensions referred to the Celestial Intermediate Origin). The true longitude and latitude link the terrestrial coordinates with the standard celestial models (for precession, nutation, sidereal time *etc.*).
5. The azimuths produced by `slaAop` and `slaAopqk` are with respect to due north as defined by the CIP, and can therefore be called "celestial azimuths". However, a telescope fixed to the Earth measures azimuth essentially with respect to due north as defined by the ITRS Pole, and can therefore be called "terrestrial azimuth". Uncorrected, this would manifest itself as a changing "azimuth zero-point error". The value `daz` is the correction to be added to a celestial azimuth to produce a terrestrial azimuth.
6. The present function is rigorous. For most practical purposes, the following simplified formulae provide an adequate approximation:

```

elong = elongm+xp*cos(elongm)-yp*sin(elongm);
phi   = phim+(xp*sin(elongm)+yp*cos(elongm))*tan(phim);
daz   = -SQRT(xp*xp+yp*yp)*cos(elongm-atan2(xp,yp))/cos(phim);

```

An alternative formulation for `daz` is:

```

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

```

7. The modern approach is to treat polar motion as a full 3-D rotation from the Terrestrial Intermediate Reference System into the International Terrestrial Reference System. The requisite matrix can be computed using the function `slaPomom`.

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

slaPomom	polar motion matrix	slaPomom
-----------------	---------------------	-----------------

ACTION : Form the matrix of polar motion for a given date.

CALL : `slaPomom(date, xp, yp, rpom);`

GIVEN :

<code>date</code>	<code>double</code>	TDB Modified Julian Date (Note 1)
<code>xp,yp</code>	<code>double</code>	coordinates of the pole (radians, Note 2)

RETURNED :

<code>rpom</code>	<code>double[3][3]</code>	polar-motion matrix (Note 4)
-------------------	---------------------------	------------------------------

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). The result is extremely insensitive to the date, so TT or even UTC can be used.
2. `xp` and `yp` are the “coordinates of the pole”, in radians, which position the Celestial Intermediate Pole in the International Terrestrial Reference System (see IERS Conventions 2003). In a geocentric right-handed triad u, v, w , where the w -axis points at the north geographic pole, the v -axis points towards the origin of longitudes and the u -axis completes the system, $\text{xp} = +u$ and $\text{yp} = -v$.
3. The other angle needed is the quantity s' , which positions the terrestrial intermediate origin on the equator. A standard model is used.
4. The matrix operates in the sense:

$$\mathbf{v}_{ITRS} = \mathbf{M} \times \mathbf{v}_{TIRS}$$

where \mathbf{v}_{ITRS} is the terrestrial vector at the sub-stellar point, \mathbf{M} is the 3×3 matrix `rpom` and \mathbf{v}_{TIRS} is the sub-stellar point in the Intermediate Terrestrial Reference System, meaning that it is the final rotation when computing the pointing direction to a celestial source.

REFERENCE : McCarthy, D.D., Petit, G. (eds.) 2004, *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG

slaPrebn	precession matrix (FK4)	slaPrebn
-----------------	-------------------------	-----------------

ACTION : Generate the matrix of precession between two epochs, using the old, pre IAU 1976, Bessel-Newcomb model, in Andoyer's formulation.

CALL : slaPrebn(bep0, bep1, rmatp);

GIVEN :

bep0	double	beginning Besselian epoch
bep1	double	ending Besselian epoch

RETURNED :

rmatp	double[3][3]	precession matrix
-------	--------------	-------------------

NOTE : The matrix is in the sense:

$$\mathbf{v}_1 = \mathbf{M} \times \mathbf{v}_0$$

where \mathbf{v}_1 is the star vector relative to the mean equator and equinox of epoch bep1, \mathbf{M} is the 3×3 matrix rmatp and \mathbf{v}_0 is the star vector relative to the mean equator and equinox of epoch bep0.

REFERENCE : Smith *et al.*, 1989, *Astr.J.* **97**, 269.

slaPrec	precession matrix (IAU 1976)	slaPrec
----------------	------------------------------	----------------

ACTION : Form the matrix of precession between two epochs (IAU 1976, FK5).

CALL : slaPrec(ep0, ep1, rmatp);

GIVEN :

ep0	double	beginning epoch
ep1	double	ending epoch

RETURNED :

`rmatp` `double[3][3]` precession matrix

NOTES :

1. The epochs are TDB Julian epochs. TT (or even UTC) will do.
2. The matrix is in the sense:

$$\mathbf{v}_1 = \mathbf{M} \times \mathbf{v}_0$$

where \mathbf{v}_1 is the star vector relative to the mean equator and equinox of epoch `ep1`, \mathbf{M} is the 3×3 matrix `rmatp` and \mathbf{v}_0 is the star vector relative to the mean equator and equinox of epoch `ep0`.

3. Though the matrix method itself is rigorous, the precession angles are expressed through canonical polynomials which are valid only for a limited time span. There are also errors in the IAU 1976 precession rate at the level of 0".3 per century. The absolute accuracy of the present formulation is better than 0".1 from 1960 AD to 2040 AD, better than 1" from 1640 AD to 2360 AD, and remains below 3" for the whole of the period 500 BC to 3000 AD. The errors exceed 10" outside the range 1200 BC to 3900 AD, exceed 100" outside 4200 BC to 5600 AD and exceed 1000" outside 6800 BC to 8200 AD. The SLALIB function `slaPrecl` implements a more elaborate model which is suitable for problems spanning several thousand years.

REFERENCES :

1. Lieske, J.H. 1979, *Astr.Astrophys.*, **73**, 282; equations 6 & 7, p283.
2. Kaplan, G.H. 1981, *USNO circular no. 163*, pA2.

slaPreces	precession	slaPreces
------------------	------------	------------------

ACTION : Precession – either the old “FK4” (Bessel-Newcomb, pre IAU 1976) or newer “FK5” (Fricke, post IAU 1976) as required.

CALL : `slaPreces(system, ep0, ep1, &ra, &dc);`

GIVEN :

<code>system</code>	<code>char*</code>	precession to be applied: "FK4" or "FK5"
<code>ep0,ep1</code>	<code>double</code>	starting and ending epoch
<code>ra,dc</code>	<code>double</code>	$[\alpha, \delta]$, mean equator & equinox of epoch tt <code>ep0</code>

RETURNED :

<code>ra,dc</code>	<code>double*</code>	$[\alpha, \delta]$, mean equator & equinox of epoch tt <code>ep1</code>
--------------------	----------------------	--

NOTES :

1. Lowercase characters in **system** are acceptable.
2. 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:

```
slaPreces( "FK4", 1900.0, 1950.0, &ra, &dc )
```
3. This function will **NOT** correctly convert between the old and the new systems – for example transformation from B1950 to J2000. For these purposes see **slaFk425**, **slaFk524**, **slaFk45z** and **slaFk54z**.
4. If an invalid **system** is supplied, values of $-99.0, -99.0$ are returned for both **ra** and **dc**.

slaPrecl	precession matrix (more accurate)	slaPrecl
-----------------	-----------------------------------	-----------------

ACTION : Form the matrix of precession between two epochs, using the model of Simon *et al.* (1994), which is suitable for long periods of time.

CALL : `slaPrecl(ep0, ep1, rmatp);`

GIVEN :

ep0	double	beginning epoch
ep1	double	ending epoch

RETURNED :

rmatp	double[3][3]	precession matrix
--------------	--------------	-------------------

NOTES :

1. The epochs are TDB Julian epochs. TT (or even UTC) will do.
2. The matrix is in the sense:

$$\mathbf{v}_1 = \mathbf{M} \times \mathbf{v}_0$$

where \mathbf{v}_1 is the star vector relative to the mean equator and equinox of epoch **ep1**, \mathbf{M} is the 3×3 matrix **rmatp** and \mathbf{v}_0 is the star vector relative to the mean equator and equinox of epoch **ep0**.

3. The absolute accuracy of the model is limited by the uncertainty in the general precession, about $0''.3$ per 1000 years. The remainder of the formulation provides a precision of 1 milliarcsecond over the interval from 1000 AD to 3000 AD, $0''.1$ from 1000 BC to 5000 AD and $1''$ from 4000 BC to 8000 AD.

REFERENCE : Simon, J.L. *et al.* 1994, *Astr.Astrophys.* **282**, 663.

slaPrenut precession-nutation matrix **slaPrenut**

ACTION : Form the matrix of precession and nutation (SF2001).

CALL : slaPrenut(epoch, date, rmatpn);

GIVEN :

epoch	double	Julian Epoch for mean coordinates
date	double	Modified Julian Date (JD−2400000.5) for true co-ordinates

RETURNED :

rmatpn	double[3][3]	combined precession-nutation matrix
--------	--------------	-------------------------------------

NOTES :

1. The epoch and date are TDB. TT (or even UTC) will do.
2. The matrix is in the sense:

$$\mathbf{v}_{true} = \mathbf{M} \times \mathbf{v}_{mean}$$

where \mathbf{v}_{true} is the star vector relative to the true equator and equinox of **date**, \mathbf{M} is the 3×3 matrix **rmatpn** and \mathbf{v}_{mean} is the star vector relative to the mean equator and equinox of **epoch**.

slaPv2el orbital elements from position & velocity **slaPv2el**

ACTION : Heliocentric osculating elements obtained from instantaneous position and velocity.

CALL : slaPv2el(pv, date, pmass, jformr, &jform, &epoch, &orbinc, &anode, &perih, &aorq, &e, &aorl, &dm, &jstat);

GIVEN :

pv	double[6]	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s; Note 1)
date	double	date (TT Modified Julian Date = JD−2400000.5)
pmass	double	mass of the planet (Sun = 1; Note 2)
jformr	int	requested element set (1-3; Note 3)

RETURNED :

jform	int*	element set actually returned (1-3; Note 4)
epoch	double*	epoch of elements (t_0 or T , TDB MJD)
orbinc	double*	inclination (i , radians)
anode	double*	longitude of the ascending node (Ω , radians)
perih	double*	longitude or argument of perihelion (ϖ or ω , radians)
aorq	double	mean distance or perihelion distance (a or q , AU)
e	double	eccentricity (e)
aorl	double	mean anomaly or longitude (M or L , radians, jform = 1, 2 only)
dm	double	daily motion (n , radians, jform = 1 only)
jstat	int*	status: 0 = OK -1 = illegal pmass -2 = illegal jformr -3 = position/velocity out of allowed range

NOTES :

1. 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.
2. The mass, **pmass**, is important only for the larger planets. For most purposes (*e.g.* asteroids) use 0.0. Values less than zero are illegal.
3. Three different element-format options are supported, as follows.

jform = 1, suitable for the major planets:

epoch	=	epoch of elements t_0 (TDB MJD)
orbinc	=	inclination i (radians)
anode	=	longitude of the ascending node Ω (radians)
perih	=	longitude of perihelion ϖ (radians)
aorq	=	mean distance a (AU)
e	=	eccentricity e ($0 \leq e < 1$)
aorl	=	mean longitude L (radians)
dm	=	daily motion n (radians)

jform = 2, suitable for minor planets:

epoch	=	epoch of elements t_0 (TDB MJD)
orbinc	=	inclination i (radians)
anode	=	longitude of the ascending node Ω (radians)
perih	=	argument of perihelion ω (radians)
aorq	=	mean distance a (AU)
e	=	eccentricity e ($0 \leq e < 1$)
aorl	=	mean anomaly M (radians)

jform = 3, suitable for comets:

<code>epoch</code>	= epoch of perihelion T (TDB MJD)
<code>orbinc</code>	= inclination i (radians)
<code>anode</code>	= longitude of the ascending node Ω (radians)
<code>perih</code>	= argument of perihelion ω (radians)
<code>aorq</code>	= perihelion distance q (AU)
<code>e</code>	= eccentricity e ($0 \leq e \leq 10$)

4. 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:

<code>jformr</code>	<code>jform</code>	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

5. The arguments returned for each value of `jform` (*cf.* Note 5: `jform` might not be the same as `jformr`) are as follows:

<code>jform</code>	1	2	3
<code>epoch</code>	t_0	t_0	T
<code>orbinc</code>	i	i	i
<code>anode</code>	Ω	Ω	Ω
<code>perih</code>	ϖ	ω	ω
<code>aorq</code>	a	a	q
<code>e</code>	e	e	e
<code>aorl</code>	L	M	-
<code>dm</code>	n	-	-

where:

t_0	is the epoch of the elements (MJD, TT)
T	is the epoch of perihelion (MJD, TT)
i	is the inclination (radians)
Ω	is the longitude of the ascending node (radians)
ϖ	is the longitude of perihelion (radians)
ω	is the argument of perihelion (radians)
a	is the mean distance (AU)
q	is the perihelion distance (AU)
e	is the eccentricity
L	is the longitude (radians, $0 - 2\pi$)
M	is the mean anomaly (radians, $0 - 2\pi$)

n is the daily motion (radians)
 - means no value is set

6. 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.
7. The osculating epoch for the returned elements is the argument **date**.

REFERENCE : Sterne, Theodore E. 1960, *An Introduction to Celestial Mechanics*, Interscience Publishers.

slaPv2ue position and velocity to universal elements **slaPv2ue**

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

CALL : slaPv2ue(pv, date, pmass, u, &jstat);

GIVEN :

pv	double[6]	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s; Note 1)
date	double	date (TT Modified Julian Date = JD-2400000.5)
pmass	double	mass of the planet (Sun = 1; Note 2)

RETURNED :

u	double[13]	universal orbital elements (Note 3)
	[0]	◦ combined mass ($M + m$)
	[1]	◦ total energy of the orbit (α)
	[2]	◦ reference (osculating) epoch (t_0)
	[3-5]	◦ position at reference epoch (\mathbf{r}_0)
	[6-8]	◦ velocity at reference epoch (\mathbf{v}_0)
	[9]	◦ heliocentric distance at reference epoch
	[10]	◦ $\mathbf{r}_0 \cdot \mathbf{v}_0$
	[11]	◦ date (t)
	[12]	◦ universal eccentric anomaly (ψ) of date, approx.
jstat	int*	status:
		0 = OK
		-1 = illegal pmass
		-2 = too close to Sun
		-3 = too slow

NOTES :

1. 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.
2. The mass, **pmass**, is important only for the larger planets. For most purposes (*e.g.* asteroids) use 0.0. Values less than zero are illegal.
3. 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) α , 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 ψ , the “universal eccentric anomaly” at a given date and (v) that date.

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

slaPvobs	observatory position & velocity	slaPvobs
-----------------	---------------------------------	-----------------

ACTION : Position and velocity of an observing station.

CALL : slaPvobs(p, h, stl, pv);

GIVEN :

p	double	latitude (geodetic, radians)
h	double	height above reference spheroid (geodetic, metres)
 stl	 double	 local apparent sidereal time (radians)

RETURNED :

pv	double[6]	[$x, y, z, \dot{x}, \dot{y}, \dot{z}$] (AU, AU s ⁻¹ , true equator and equinox of date)
-----------	------------------	--

NOTE : IAU 1976 constants are used.

slaPxy	apply linear model	slaPxy
---------------	--------------------	---------------

ACTION : Given arrays of *expected* and *measured* $[x, y]$ coordinates, and a linear model relating them (as produced by `slaFitxy`), compute the array of *predicted* coordinates and the RMS residuals.

CALL : `slaPxy(np, xye, xym, coeffs, xyp, &xrms, &y rms, &rrms);`

GIVEN :

<code>np</code>	<code>int</code>	number of samples
<code>xye</code>	<code>double[2][np]</code>	expected $[x, y]$ for each sample
<code>xym</code>	<code>double[2][np]</code>	measured $[x, y]$ for each sample
<code>coeffs</code>	<code>double[6]</code>	coefficients of model (see below)

RETURNED :

<code>xyp</code>	<code>double[2][np]</code>	predicted $[x, y]$ for each sample
<code>xrms</code>	<code>double*</code>	RMS in x
<code>y rms</code>	<code>double*</code>	RMS in y
<code>rrms</code>	<code>double*</code>	total RMS (RSS of <code>xrms</code> and <code>y rms</code>)

NOTES :

1. The model is supplied in the array `coeffs`. Naming the six elements of `coeffs` a, b, c, d, e & f , the model transforms *measured* coordinates $[x_m, y_m]$ into *predicted* coordinates $[x_p, y_p]$ as follows:

$$x_p = a + bx_m + cy_m$$

$$y_p = d + ex_m + fy_m$$
2. The residuals are $(x_p - x_e)$ and $(y_p - y_e)$.
3. If `np` is less than or equal to zero, no coordinates are transformed, and the RMS residuals are all zero.
4. See also `slaFitxy`, `slaInvf`, `slaXy2xy`, `slaDcmpf`

slaRange	put angle into range $\pm\pi$	slaRange
-----------------	-------------------------------	-----------------

ACTION : Normalize an angle into the range $\pm\pi$ (single precision).

CALL : `r = slaRange(angle);`

GIVEN :

angle	float	angle in radians
-------	-------	------------------

RETURNED :

float	angle expressed in the range $\pm\pi$.
-------	---

slaRanorm	put angle into range $0-2\pi$	slaRanorm
------------------	-------------------------------	------------------

ACTION : Normalize an angle into the range $0-2\pi$ (single precision).

CALL : `r = slaRanorm(angle);`

GIVEN :

angle	float	angle in radians
-------	-------	------------------

RETURNED :

float	angle expressed in the range $0-2\pi$
-------	---------------------------------------

slaRcc	barycentric coordinate time	slaRcc
---------------	-----------------------------	---------------

CALL : `d = slaRcc(tdb, ut1, wlfq, u, v);`

ACTION : The relativistic clock correction: the difference between *proper time* at a point on the Earth and *coordinate time* in the solar system barycentric space-time frame of reference. The proper time is Terrestrial Time, TT; the coordinate time is an implementation of Barycentric Dynamical Time, TDB.

GIVEN :

tdb	double	TDB (MJD: JD-2400000.5)
ut1	double	universal time (fraction of one day)
wl	double	clock longitude (radians west)
u	double	clock distance from Earth spin axis (km)
v	double	clock distance north of Earth equatorial plane (km)

RETURNED :

double TDB–TT (s)

NOTES :

1. TDB is coordinate time in the solar system barycentre frame of reference, in units chosen to eliminate the scale difference with respect to terrestrial time. TT is the proper time for clocks at mean sea level on the Earth.
2. The number returned by `slaRcc` is dominated by an annual sinusoidal term of amplitude approximately 1.66 ms. Its second harmonic is at $14\ \mu\text{s}$ amplitude, slightly below the largest planetary term which is of $21\ \mu\text{s}$ amplitude. Diurnal terms start at $2\ \mu\text{s}$. The variation arises from the transverse Doppler effect and the gravitational red-shift as the observer varies in speed and moves through different gravitational potentials.
3. The argument `tdb` is, strictly, the barycentric coordinate time; however, the terrestrial time (TT) can in practice be used without significant loss of accuracy.
4. The geocentric model is that of Fairhead & Bretagnon (1990), in its full form. It was supplied by Fairhead (private communication) as a Fortran subroutine. A number of coding changes were made to this subroutine in order match the calling sequence of previous versions of the present function, to comply with Starlink programming standards and to avoid compilation problems on certain machines. The numerical results are essentially unaffected by the changes.
5. The topocentric model is from Moyer (1981) and Murray (1983). It is an approximation to the expression

$$\frac{\mathbf{v}_e \cdot (\mathbf{x} - \mathbf{x}_e)}{c^2}$$

where \mathbf{v}_e is the barycentric velocity of the Earth, \mathbf{x} and \mathbf{x}_e are the barycentric positions of the observer and the Earth respectively, and c is the speed of light. It can be disabled, if necessary, by setting the arguments `u` and `v` to zero.

6. During the interval 1950-2050, the absolute accuracy is better than ± 3 nanoseconds relative to direct numerical integrations using the JPL DE200/LE200 solar system ephemeris.
7. The IAU 1976 definition of TDB was that it must differ from TT only by periodic terms. Though practical, this is an imprecise definition that ignores the existence of very long-period and secular effects in the dynamics of the solar system. As a consequence, different implementations of TDB will, in general, differ in zero-point and will drift linearly relative to one other. In 1991 the IAU introduced new time-scales designed to overcome these objections: geocentric coordinate time, TCG, and barycentric coordinate time, TCB. In principle, therefore, TDB is obsolete. However, `slaRcc` can be used to implement the periodic part of TCB–TCG.
8. TDB is for all practical purposes the same as the T_{eph} time-scale used with JPL solar-system ephemerides.

REFERENCES :

1. Fairhead, L., & Bretagnon, P. 1990, *Astron.Astrophys.* **229**, 240-247.
2. Moyer, T.D. 1981, *Cel.Mech.*, **23**, 33.
3. Murray, C.A. 1983, *Vectorial Astrometry*, Adam Hilger.
4. Seidelmann, P.K. *et al.*, 1992, *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Chapter 2.
5. Simon, J.L., Bretagnon, P., Chapront, J., Chapront-Touze, M., Francou, G. & Laskar, J. 1994, *Astron.Astrophys.* **282**, 663-683.

slaRdplan	apparent $[\alpha, \delta]$ of planet	slaRdplan
------------------	---------------------------------------	------------------

ACTION : Approximate topocentric apparent $[\alpha, \delta]$ and angular size of a planet (also Sun, Moon and Pluto).

CALL : slaRdplan(date, np, elong, phi, &ra, &dec, &diam);

GIVEN :

date	double	MJD of observation (JD−2400000.5)
np	int	body:
		1 = Mercury
		2 = Venus
		3 = Moon
		4 = Mars
		5 = Jupiter
		6 = Saturn
		7 = Uranus
		8 = Neptune
		9 = Pluto
		else = Sun
elong, phi	double	observer's longitude (east +ve) and latitude (radians)

RETURNED :

ra, dec	double*	topocentric apparent $[\alpha, \delta]$ (radians)
diam	double*	angular diameter (equatorial, radians)

NOTES :

1. The date is in a dynamical time-scale (TDB) and is in the form of a Modified Julian Date (JD−2400000.5). For all practical purposes, TT can be used instead of TDB, and for many applications UTC will do (except for the Moon).

2. 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 function its effect on planetary positions is small (negligible for the outer planets). Geocentric positions can be generated by appropriate use of the functions `slaDmoon` and `slaPlanet`.
3. The direction accuracy (arcsec, 1000-3000 AD) is of order:

Sun	5
Mercury	2
Venus	10
Moon	30
Mars	50
Jupiter	90
Saturn	90
Uranus	90
Neptune	10
Pluto	1 (1885-2099 AD only)

The angular diameter accuracy is about 0.4% for the Moon, and 0.01% or better for the other bodies. For more information on accuracy, refer to the functions `slaPlanet` and `slaDmoon`, which the present function uses.

slaRefco	refraction constants	slaRefco
-----------------	----------------------	-----------------

ACTION : Determine the constants A and B in the atmospheric refraction model $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$, where ζ is the *observed* zenith distance (*i.e.* affected by refraction) and $\Delta\zeta$ is what to add to ζ to give the *topocentric* (*i.e. in vacuo*) zenith distance.

CALL : `slaRefco(hm, tk, phpa, rh, wlfq, phi, tlr, eps, &refa, &refb);`

GIVEN :

<code>hm</code>	<code>double</code>	height of the observer above sea level (metre)
<code>tk</code>	<code>double</code>	ambient temperature at the observer K)
<code>phpa</code>	<code>double</code>	pressure at the observer (hPa)
<code>rh</code>	<code>double</code>	relative humidity at the observer (range 0–1)
<code>wlfq</code>	<code>double</code>	wavelength (μm) or minus frequency (GHz)
<code>phi</code>	<code>double</code>	latitude of the observer (radian, astronomical)
<code>tlr</code>	<code>double</code>	tropospheric lapse rate (K per metre)
<code>eps</code>	<code>double</code>	precision required to terminate iteration (radian)

RETURNED :

<code>refa</code>	<code>double*</code>	A : the $\tan \zeta$ coefficient (radians)
<code>refb</code>	<code>double*</code>	B : the $\tan^3 \zeta$ coefficient (radians)

NOTES :

1. Suggested values for the **tlr** and **eps** arguments are 0.0065 and 10^{-8} respectively. The signs of both are immaterial.
2. The argument **wlfq** specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at $100\ \mu\text{m}$ (about 3000 GHz).
3. The radio refraction is chosen by specifying $wl > 100\ \mu\text{m}$.
4. The function is a slower but more accurate alternative to the **slaRefcoq** function. The constants it produces give perfect agreement with **slaRefro** at zenith distances $\tan^{-1} 1$ (45°) and $\tan^{-1} 4$ ($\sim 76^\circ$). At other zenith distances, the model achieves: $0''.5$ accuracy for $\zeta < 80^\circ$, $0''.01$ accuracy for $\zeta < 60^\circ$, and $0''.001$ accuracy for $\zeta < 45^\circ$.
5. The sign convention for B is that it is a negative number.

slaRefcoq	refraction constants (fast)	slaRefcoq
------------------	-----------------------------	------------------

ACTION : Determine the constants A and B in the atmospheric refraction model $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$, where ζ is the *observed* zenith distance (*i.e.* affected by refraction) and $\Delta\zeta$ is what to add to ζ to give the *topocentric* (*i.e. in vacuo*) zenith distance. (This is a fast alternative to the **slaRefco** function – see notes.)

CALL : `slaRefcoq(tk, phpa, rh, wl, &refa, &refb);`

GIVEN :

tk	double	ambient temperature at the observer (K)
phpa	double	pressure at the observer (hPa)
rh	double	relative humidity at the observer (range 0–1)
wl	double	wavelength (μm)

RETURNED :

refa	double*	A: the $\tan \zeta$ coefficient (radians)
refb	double*	B: the $\tan^3 \zeta$ coefficient (radians)

NOTES :

1. The model is an approximation, for moderate zenith distances, to the predictions of the **slaRefro** function. The approximation is maintained across a range of conditions, and applies to both optical/IR and radio.

2. The algorithm is a fast alternative to the `slaRefco` function. The latter calls the `slaRefro` function itself: this involves integrations through a model atmosphere, and is costly in processor time. However, the model which is produced is precisely correct for two zenith distances (45° and $\sim 76^\circ$) and at other zenith distances is limited in accuracy only by the $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ formulation itself. The present function is not as accurate, though it satisfies most practical requirements.
3. The model omits the effects of (i) height above sea level (apart from the reduced pressure itself), (ii) latitude (*i.e.* the flattening of the Earth), (iii) variations in tropospheric lapse rate and (iv) dispersive effects in the radio.
4. The model has been tested using the following range of conditions:
 - lapse rates 0.0055, 0.0065, 0.0075 K per metre
 - latitudes 0° , 25° , 50° , 75°
 - heights 0, 2500, 5000 metres above sea level
 - pressures, mean for height -10% to $+5\%$ in steps of 5%
 - temperatures -10° to $+20^\circ$ with respect to 280K at sea level
 - relative humidity 0, 0.5, 1
 - wavelength 0.4, 0.6, ... $2\mu\text{m}$, + radio
 - zenith distances 15° , 45° , 75°

For the above conditions, the comparison with `slaRefro` was as follows:

	<i>worst</i>	<i>RMS</i>
optical/IR	62	8
radio	319	49
	mas	mas

For this particular set of conditions:

- lapse rate 6.5 K km^{-1}
- latitude 50°
- sea level
- pressure 1005 mb
- temperature 7°C
- humidity 80%
- wavelength 5740 \AA

the results were as follows:

ζ	<code>slaRefro</code>	<code>slaRefcoq</code>	Saastamoinen
10	10.27	10.27	10.27
20	21.19	21.20	21.19
30	33.61	33.61	33.60
40	48.82	48.83	48.81
45	58.16	58.18	58.16
50	69.28	69.30	69.27
55	82.97	82.99	82.95
60	100.51	100.54	100.50
65	124.23	124.26	124.20
70	158.63	158.68	158.61
72	177.32	177.37	177.31
74	200.35	200.38	200.32
76	229.45	229.43	229.42
78	267.44	267.29	267.41
80	319.13	318.55	319.10
deg	arcsec	arcsec	arcsec

The values for Saastamoinen's formula (which includes terms up to \tan^5) are taken from Hohenkerk & Sinclair (1985).

The results from the much slower but more accurate `slaRefco` function have not been included in the tabulation as they are identical to those in the `slaRefro` column to the 0".01 resolution used.

5. A `w1` value in the range 0-100 selects the optical/IR case and is wavelength in μm . Any value outside this range selects the radio case. Radio dispersion effects, important above 100 GHz, are neglected.
6. Outlandish input parameters are silently limited to mathematically safe values. Zero pressure is permissible, and causes zeroes to be returned.
7. The algorithm draws on several sources, as follows:
 - The formula for the saturation vapour pressure of water as a function of temperature and temperature is taken from Equations (A4.5-A4.7) of Gill (1982).
 - The formula for the water vapour pressure, given the saturation pressure and the relative humidity is from Crane (1976), Equation (2.5.5).
 - The refractivity of air is a function of temperature, total pressure, water-vapour pressure and, in the case of optical/IR, wavelength. The formulae for the two cases are developed from Hohenkerk & Sinclair (1985) and Rueger (2002).
 - The formula for β ($= H_0/r_0$) is an adaption of Equation (9) from Stone (1996). The adaptations, arrived at empirically, consist of (i) a small adjustment to the coefficient and (ii) a humidity term for the radio case only.
 - The formulae for the refraction constants as a function of $n - 1$ and β are from Green (1987), Equation (4.31).
8. The sign convention for B is that it is a negative number.

REFERENCES :

1. Crane, R.K. 1976, "Refraction Effects in the Neutral Atmosphere", *Methods of Experimental Physics: Astrophysics 12B*, Meeks, M.L. (ed), Academic Press.
2. Gill, Adrian E. 1982, *Atmosphere-Ocean Dynamics*, Academic Press.
3. Green, R.M. 1987, *Spherical Astronomy*, Cambridge University Press.
4. Hohenkerk, C.Y., & Sinclair, A.T. 1985, *NAO Technical Note* No. 63, Royal Greenwich Observatory.
5. Rueger, J.M. 2002, *Refractive Index Formulae for Electronic Distance Measurement with Radio and Millimetre Waves*, in Unisurv Report S-68, School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia.
6. Stone, Ronald C. 1996, P.A.S.P. **108** 1051-1058.

slaRefro	refraction	slaRefro
-----------------	------------	-----------------

ACTION : Atmospheric refraction, for radio or optical/IR wavelengths.

CALL : `slaRefro(zobs, hm, tk, phpa, rh, wlfq, phi, tlr, eps, &ref);`

GIVEN :

<code>zobs</code>	<code>double</code>	observed zenith distance of the source (radians)
<code>hm</code>	<code>double</code>	height of the observer above sea level (metre)
<code>tk</code>	<code>double</code>	ambient temperature at the observer (K)
<code>phpa</code>	<code>double</code>	pressure at the observer (hPa)
<code>rh</code>	<code>double</code>	relative humidity at the observer (range 0–1)
<code>wlfq</code>	<code>double</code>	wavelength (μm) or minus frequency (GHz)
<code>phi</code>	<code>double</code>	latitude of the observer (radian, astronomical)
<code>tlr</code>	<code>double</code>	tropospheric lapse rate (K per metre)
<code>eps</code>	<code>double</code>	precision required to terminate iteration (radian)

RETURNED :

<code>ref</code>	<code>double*</code>	refraction: <i>in vacuo</i> ZD minus observed ZD (radians)
------------------	----------------------	--

NOTES :

1. A suggested value for the `tlr` argument is 0.0065 (sign immaterial). The refraction is significantly affected by `tlr`, and if studies of the local atmosphere have been carried out a better value may be available.
2. A suggested value for the `eps` argument is 10^{-8} . The result is usually at least two orders of magnitude more computationally precise than the supplied `eps` value, and the choice of `eps` value has a large effect on computational speed, particularly for large zenith distances.

3. The function computes the refraction for zenith distances up to and a little beyond 90° (see Note 6) using the Biot-Auer-Standish method (Young 2004) in the form given by Hohenkerk & Sinclair (1985): see Section 3.281 of the *Explanatory Supplement to the Astronomical Almanac* (Seidelmann 1992).
4. The code is based on the **AREF** optical/IR refraction subroutine (HMNAO, September 1984, RGO: Hohenkerk 1985), with extensions to support the radio case. The following functional changes to the original HMNAO optical/IR refraction code have been made:
 - The angle arguments have been changed to radians, any value of **zobs** is allowed (see Note 6, below) and other argument values have been limited to safe values.
 - Gas constants from Murray (1983) have been used.
 - A better model for $P_s(T, P)$ has been adopted, taken from Buck (1981).
 - The formula for the water vapour pressure, given the saturation pressure and the relative humidity, is from Crane (1976), Equation (2.5.5).
 - The IAG (1999) optical refractivity for dry air is used.
 - Provision for radio wavelengths has been added using expressions originally devised by A.T.Sinclair, RGO (private communication 1989). The low frequency refractivity model currently used is from Rueger (2002). Corrections for O_2 and H_2O features, important in the mm/submm region, are applied using the MPM93 model of Liebe et al. (1993) as set out in Fortran code **N93AIR** by Hufford & Liebe. Ionospheric effects are neglected, compromising accuracy below about 30 MHz.
5. The argument **wlfq** specifies whether the refraction corrections are for the optical or radio, and the wavelength/frequency. Positive values specify wavelength in microns and are the usual way of selecting the optical case. Negative values specify frequency in GHz and are the usual way of selecting the radio case. The transition from optical to radio is assumed to occur at $100\ \mu\text{m}$ (about 3000 GHz).
6. Before use, the value of **zobs** is expressed in the range $\pm\pi$. If this ranged **zobs** is negative, the result **ref** is computed from its absolute value before being made negative to match. In addition, if it has an absolute value greater than 93° (optical/IR) or 90.5° (radio/mm/submm), the refraction for that angle is returned, appropriately signed. This precaution is necessary because the integration fails when the ray curvature approaches that of the Earth's surface.
7. As in the original Hohenkerk & Sinclair algorithm (see also Sinclair 1982), fixed values of the water vapour polytrope exponent, the height of the tropopause, and the height at which refraction is negligible are used.
8. The optical refraction has been compared with the Pulkovo tables (Abalakin 1985). For the example provided in the Preface (p9), **slaRefro** predicts refraction of $331''.40$, compared with the Pulkovo figure of $331''.38$. For the main tables the agreement over most of the range is within 0.03%. For the extreme zenith distances 88° , 89° and 90° , the Pulkovo predictions are $1064''.54$, $1409''.42$ and $1977''.97$, whereas **slaRefro** returns values that are $0''.07$, $0''.49$ and $3''.37$ lower, respectively.

The radio refraction has been tested against work done by Iain Coulson, JACH (1995) for the James Clerk Maxwell Telescope, Mauna Kea. For typical conditions, agreement at the $0''.1$ level is achieved for moderate zenith distances, worsening to perhaps $0''.5$ - $1''.0$ at $\zeta = 80^\circ$. At hot and humid sea-level sites the agreement would not be as good.

9. It should be noted that the relative humidity **rh** is formally defined in terms of “mixing ratio” rather than pressures or densities as is often stated. It is the mass of water per unit mass of dry air divided by that for saturated air at the same temperature and pressure (see Gill 1982). The familiar $\nu = p_w/p_s$ or $\nu = \rho_w/\rho_s$ expressions can differ from the formal definition by several percent, significant in the radio case.
10. The algorithm is designed for observers in the troposphere. The supplied temperature, pressure and lapse rate are assumed to be for a point in the troposphere and are used to define a model atmosphere with the tropopause at 11 km altitude and a constant temperature above that. However, in practice, the refraction values returned for stratospheric observers, at altitudes up to 25km, are quite usable.

REFERENCES :

1. Abalakin, V.K. (ed) 1985, *Refraction Tables of Pulkovo Observatory*, 5th edition, Central Astronomical Observatory, Academy of Sciences of the USSR: Nauka Publishing House, Leningrad Section, Leningrad, USSR.
2. Buck, A. 1981, *New Equations for Computing Vapor Pressure and Enhancement Factor*, J.Appl.Met. 20, 1527-1532.
3. Coulsen, I. 1995, private communication.
4. Crane, R.K. 1976, “Refraction Effects in the Neutral Atmosphere”, *Methods of Experimental Physics: Astrophysics 12B*, Meeks, M.L. (ed), Academic Press.
5. Gill, A.E. 1982, *Atmosphere-Ocean Dynamics*, Academic Press.
6. Hohenkerk, C.Y. 1985, private communication.
7. Hohenkerk, C.Y., & Sinclair, A.T. 1985, *The computation of angular atmospheric refraction at large zenith angles*, NAO Technical Note No. 63, Royal Greenwich Observatory.
8. International Association of Geodesy, XXIIInd General Assembly, Birmingham, UK, 1999, Resolution 3.
9. Murray, C.A. 1983, *Vectorial Astrometry*, Adam Hilger, Bristol.
10. Rueger, J.M. 2002, *Refractive Index Formulae for Electronic Distance Measurement with Radio and Millimetre Waves*, in Unisurv Report S-68, School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia.
11. Seidelmann, P.K. *et al.*, 1992, *Explanatory Supplement to the Astronomical Almanac*, Chapter 3, University Science Books.
12. Sinclair, A.T. 1982, *The effect of atmospheric refraction on laser ranging data*, NAO Technical Note No. 59, Royal Greenwich Observatory.
13. Sinclair, A.T. 1989, private communication.
14. Young, A.T. 2004, *Sunset science. IV. Low-altitude refraction*, Astron.J., 127, 3622-3637.

slaRefv	apply refraction to vector	slaRefv
----------------	----------------------------	----------------

ACTION : Adjust an unrefracted Cartesian vector to include the effect of atmospheric refraction, using the simple $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ model.

CALL : `slaRefv(vu, refa, refb, vr);`

GIVEN :

<code>vu</code>	<code>double[3]</code>	unrefracted position of the source ($[Az, El]$ 3-vector)
<code>refa</code>	<code>double</code>	$\tan \zeta$ coefficient (radians)
<code>refb</code>	<code>double</code>	$\tan^3 \zeta$ coefficient (radians)

RETURNED :

<code>vr</code>	<code>double[3]</code>	refracted position of the source ($[Az, El]$ 3-vector)
-----------------	------------------------	---

NOTES :

1. This function applies the adjustment for refraction in the opposite sense to the usual one – it takes an unrefracted (*in vacuo*) position and produces an observed (refracted) position, whereas the $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ model strictly applies to the case where an observed position is to have the refraction removed. The unrefracted to refracted case is harder, and requires an inverted form of the text-book refraction models; the algorithm used here is equivalent to one iteration of the Newton-Raphson method applied to the above formula.
2. Though optimized for speed rather than precision, the present function achieves consistency with the refracted-to-unrefracted $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ model at better than 1 microarcsecond within 30° of the zenith and remains within 1 milliarcsecond to $\zeta = 70^\circ$. The inherent accuracy of the model is, of course, far worse than this – see the documentation for `slaRefco` for more information.
3. At low elevations (below about 3°) the refraction correction is held back to prevent arithmetic problems and wildly wrong results. For optical/IR wavelengths, over a wide range of observer heights and corresponding temperatures and pressures, the following levels of accuracy (worst case) are achieved, relative to numerical integration through a model atmosphere:

ζ_{obs}	<i>error</i>	
80°	0".7	
81°	1".3	
82°	2".5	
83°	5"	
84°	10"	
85°	20"	
86°	55"	
87°	160"	
88°	360"	
89°	640"	
90°	1100"	
91°	1700"	< high-altitude
92°	2600"	< sites only

The results for radio are slightly worse over most of the range, becoming significantly worse below $\zeta = 88^\circ$ and unusable beyond $\zeta = 90^\circ$.

4. See also the function `slaRefz`, which performs the adjustment to the zenith distance rather than in $[x, y, z]$. The present function is faster than `slaRefz` and, except very low down, is equally accurate for all practical purposes. However, beyond about $\zeta = 84^\circ$ `slaRefz` should be used, and for the utmost accuracy iterative use of `slaRefro` should be considered.

slaRefz	apply refraction to ZD	slaRefz
----------------	------------------------	----------------

ACTION : Adjust an unrefracted zenith distance to include the effect of atmospheric refraction, using the simple $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ model.

CALL : `slaRefz(zu, refa, refb, &zr);`

GIVEN :

<code>zu</code>	<code>double</code>	unrefracted zenith distance of the source (radians)
<code>refa</code>	<code>double</code>	$\tan \zeta$ coefficient (radians)
<code>refb</code>	<code>double</code>	$\tan^3 \zeta$ coefficient (radians)

RETURNED :

<code>zr</code>	<code>double*</code>	refracted zenith distance (radians)
-----------------	----------------------	-------------------------------------

NOTES :

1. This function applies the adjustment for refraction in the opposite sense to the usual one – it takes an unrefracted (*in vacuo*) position and produces an observed (refracted) position, whereas the $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ model strictly applies to the case where an observed position is to have the refraction removed. The unrefracted to refracted case is harder, and requires an inverted form of the text-book refraction models; the formula used here is based on the Newton-Raphson method. For the utmost numerical consistency with the refracted to unrefracted model, two iterations are carried out, achieving agreement at the 10^{-11} arcsecond level for $\zeta = 80^\circ$. The inherent accuracy of the model is, of course, far worse than this – see the documentation for `slaRefco` for more information.
2. At $\zeta = 83^\circ$, the rapidly-worsening $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ model is abandoned and an empirical formula takes over:

$$\Delta\zeta = F \left(\frac{0.55445 - 0.01133E + 0.00202E^2}{1 + 0.28385E + 0.02390E^2} \right)$$

where $E = 90^\circ - \zeta_{true}$ and F is a factor chosen to meet the $\Delta\zeta = A \tan \zeta + B \tan^3 \zeta$ formula at $\zeta = 83^\circ$.

For optical/IR wavelengths, over a wide range of observer heights and corresponding temperatures and pressures, the following levels of accuracy (worst case) are achieved, relative to numerical integration through a model atmosphere:

ζ_{obs}	error
80°	0".7
81°	1".3
82°	2".4
83°	4".7
84°	6".2
85°	6".4
86°	8"
87°	10"
88°	15"
89°	30"
90°	60"
91°	150" < high-altitude
92°	400" < sites only

For radio wavelengths the errors are typically 50% larger than the optical figures and by $\zeta = 85^\circ$ are twice as bad, worsening rapidly below that. To maintain 1" accuracy down to $\zeta = 85^\circ$ at the Green Bank site, Condon (2004) has suggested amplifying the amount of refraction predicted by `slaRefz` below 10.8 elevation by the factor $(1 + 0.00195 * (10.8 - E_{topo}))$, where E_{topo} is the unrefracted elevation in degrees.

The high-ZD model is scaled to match the normal model at the transition point; there is no glitch.

3. See also the function `slaRefv`, which performs the adjustment in $[x, y, z]$, and with the emphasis on speed rather than numerical accuracy.

REFERENCE : Condon, J.J. 2004, *Refraction Corrections for the GBT*, PTCS/PN/35.2, NRAO Green Bank.

slaRverot radial velocity correction to Earth centre **slaRverot**

ACTION : Velocity component in a given direction due to Earth rotation.

CALL : `r = slaRverot(phi, ra, da, st);`

GIVEN :

<code>phi</code>	<code>float</code>	geodetic latitude of observing station (radians)
<code>ra,da</code>	<code>float</code>	apparent $[\alpha, \delta]$ (radians)
<code>st</code>	<code>float</code>	local apparent sidereal time (radians)

RETURNED :

<code>float</code>	Component of Earth rotation in direction $[\text{ra}, \text{da}]$ (km s^{-1})
--------------------	--

NOTES :

1. Sign convention: the result is positive when the observatory is receding from the given point on the sky.
2. Accuracy: the simple algorithm used assumes a spherical Earth and an observing station at sea level; for actual observing sites, the error is unlikely to be greater than 0.0005 km s^{-1} . For applications requiring greater accuracy, use the function `slaPvobs`.

slaRvgalc radial velocity correction to galactic centre **slaRvgalc**

ACTION : Velocity component in a given direction due to the rotation of the Galaxy.

CALL : `r = slaRvgalc(r2000, d2000);`

GIVEN :

<code>r2000,d2000</code>	<code>float</code>	J2000.0 mean $[\alpha, \delta]$ (radians)
--------------------------	--------------------	---

RETURNED :

float	Component of dynamical LSR motion in direction [r2000,d2000] (km s ⁻¹)
--------------	---

NOTES :

1. Sign convention: the result is positive when the LSR is receding from the given point on the sky.
2. 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.
3. The dynamical LSR velocity due to galactic rotation is assumed to be 220 km s⁻¹ towards $l^{\text{II}} = 90^\circ$, $b^{\text{II}} = 0$.

REFERENCE : Kerr F.J., & Lynden-Bell, D. 1986, *Mon.Not.R.astr.Soc.*, **221**, 1023.

slaRvlg	radial velocity correction to local group	slaRvlg
----------------	---	----------------

ACTION : 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.

CALL : r = slaRvlg(r2000, d2000);

GIVEN :

r2000,d2000	float	J2000.0 mean [α, δ] (radians)
-------------	--------------	---

RETURNED :

float	Component of solar (<i>n.b.</i>) motion in direction [r2000,d2000] (km s ⁻¹)
--------------	--

NOTE : Sign convention: the result is positive when the Sun is receding from the given point on the sky.

REFERENCE : *IAU Trans.* 1976, **16B**, p201.

slaRvlsrd radial velocity correction to dynamical LSR **slaRvlsrd**

ACTION : Velocity component in a given direction due to the Sun's motion with respect to the "dynamical" Local Standard of Rest.

CALL : `r = slaRvlsrd(r2000, d2000);`

GIVEN :

`r2000,d2000` `float` J2000.0 mean $[\alpha, \delta]$ (radians)

RETURNED :

`float` Component of *peculiar* solar motion in direction
[`r2000,d2000`] (km s^{-1})

NOTES :

1. Sign convention: the result is positive when the Sun is receding from the given point on the sky.
2. 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.
3. 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. The dynamical LSR is seldom used by observational astronomers, who conventionally use a kinematical LSR such as the one implemented in the function `slaRvlsrk`.
4. The peculiar solar motion is from Delhaye (1965): in galactic Cartesian coordinates $(+9, +12, +7) \text{ km s}^{-1}$. This corresponds to about 16.6 km s^{-1} towards galactic coordinates $l^{\text{II}} = 53^\circ$, $b^{\text{II}} = +25^\circ$.

REFERENCE : Delhaye, J., 1965, in *Stars and Stellar Systems*, vol 5, *Galactic Structure*, ed. Blaauw & Schmidt, Univ. of Chicago Press, pp73-4.

slaRvlsrk radial velocity correction to kinematical LSR **slaRvlsrk**

ACTION : Velocity component in a given direction due to the Sun's motion with respect to a kinematical Local Standard of Rest.

CALL : `r = slaRvlsrk(r2000, d2000);`

GIVEN :

r2000,d2000	float	J2000.0 mean $[\alpha, \delta]$ (radians)
-------------	-------	---

RETURNED :

float	Component of <i>standard</i> solar motion in direction [r2000,d2000] (km s^{-1})
-------	--

NOTES :

1. Sign convention: the result is positive when the Sun is receding from the given point on the sky.
2. 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.
3. There is another sort of LSR, seldom used by observational astronomers, called the *dynamical* LSR. This 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 function `slaRvlsrd`.
4. The adopted standard solar motion is 20 km s^{-1} towards $\alpha = 18^{\text{h}}$, $\delta = +30^{\circ}$ (1900).

REFERENCES :

1. Delhaye, J., 1965, in *Stars and Stellar Systems*, vol 5, *Galactic Structure*, ed. Blaauw & Schmidt, Univ. of Chicago Press, pp73-4.
2. *Methods of Experimental Physics* (ed Meeks), vol 12, part C, sec 6.1.5.2, p281.

slaS	the CIO locator <i>s</i>	slaS
-------------	--------------------------	-------------

ACTION : For a given date, and given the *X, Y* coordinates of the celestial intermediate pole, predict the quantity *s*, positioning the celestial intermediate origin on the CIP equator.

CALL : `d = slaS(date, x, y);`

GIVEN :

date	double	TDB Modified Julian Date (Note 1)
x,y	double	CIP coordinates (Note 3)

RETURNED :

double the quantity s , in radians (Note 2)

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). TT (or even UTC) will do.
2. The CIO locator s is the difference between the right ascensions of the same point in two frames: the two systems are the ICRS and the CIP,CEO, and the point is the ascending node of the respective equators. The quantity s remains below $0''.1$ throughout 1900-2100.
3. The series used to compute s is in fact for $s + XY/2$, where X and Y are the x and y components of the CIP unit vector; this series is more compact than a direct series for s would be. This function requires X, Y to be supplied by the caller, who is responsible for providing values that are consistent with the supplied date.
4. The model is consistent with the “P03” precession (Capitaine *et al.* 2003).

REFERENCES :

1. Capitaine, N., Wallace, P.T. & Chapront, J., 2003, *Astron.Astrophys.*, **432**, 355-357
2. McCarthy, D.D., Petit, G. (eds.) 2004, *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG

slaS2tp	spherical to tangent plane	slaS2tp
----------------	----------------------------	----------------

ACTION : Projection of spherical coordinates onto the tangent plane (single precision).

CALL : slaS2tp(ra, dec, raz, decz, &xi, &eta, &j);

GIVEN :

ra,dec	float	spherical coordinates of star (radians)
raz,decz	float	spherical coordinates of tangent point (radians)

RETURNED :

xi,eta	float*	tangent plane coordinates (radians)
j	int*	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

NOTES :

1. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
2. When working in $[x, y, z]$ rather than spherical coordinates, the equivalent Cartesian function `slaV2tp` is available.

slaSep	angle between two points on sphere	slaSep
---------------	------------------------------------	---------------

ACTION : Angle between two points on a sphere (single precision).

CALL : `r = slaSep(a1, b1, a2, b2);`

GIVEN :

<code>a1,b1</code>	<code>float</code>	spherical coordinates of one point (radians)
<code>a2,b2</code>	<code>float</code>	spherical coordinates of the other point (radians)

RETURNED :

<code>float</code>	angle between <code>[a1,b1]</code> and <code>[a2,b2]</code> in radians
--------------------	--

NOTES :

1. The spherical coordinates are right ascension and declination, longitude and latitude, *etc.* in radians.
2. The result is always positive.

slaSepv	angle between two vectors	slaSepv
----------------	---------------------------	----------------

ACTION : Angle between two vectors (single precision).

CALL : `r = slaSepv(v1, v2);`

GIVEN :

<code>v1</code>	<code>float[3]</code>	first vector
<code>v2</code>	<code>float[3]</code>	second vector

RETURNED :

float	angle between v1 and v2 in radians
-------	------------------------------------

NOTES :

1. There is no requirement for either vector to be of unit length.
2. If either vector is null, zero is returned.
3. The result is always positive.

slaSmat	solve simultaneous equations	slaSmat
----------------	------------------------------	----------------

ACTION : Matrix inversion and solution of simultaneous equations (single precision).**CALL :** slaSmat(n, a, y, &d, &jf, iw);**GIVEN :**

n	int	number of unknowns
a	R(N,N)	matrix
y	R(N)	vector

RETURNED :

a	float[n][n]	matrix inverse
y	float[n]	solution
d	float*	determinant
jf	int*	singularity flag: 0 = OK
iw	int[n]	workspace

NOTES :

1. For the set of n simultaneous linear equations in n unknowns:

$$\mathbf{A} \times \mathbf{y} = \mathbf{x}$$

where:

- \mathbf{A} is a non-singular $n \times n$ matrix,
- \mathbf{y} is the vector of n unknowns, and
- \mathbf{x} is the known vector,

slaSmat computes:

- the inverse of matrix \mathbf{A} ,
- the determinant of matrix \mathbf{A} , and

- the vector of n unknowns \mathbf{y} .

Argument \mathbf{n} is the order n , \mathbf{a} (given) is the matrix \mathbf{A} , \mathbf{y} (given) is the vector \mathbf{x} and \mathbf{y} (returned) is the vector \mathbf{y} . The argument \mathbf{a} (returned) is the inverse matrix \mathbf{A}^{-1} , and \mathbf{d} is $\det \mathbf{A}$.

2. \mathbf{jf} is the singularity flag. If the matrix is non-singular, $\mathbf{jf} = 0$ is returned. If the matrix is singular, $\mathbf{jf} = -1$ and $\mathbf{d} = 0.0$ are returned. In the latter case, the contents of array \mathbf{a} on return are undefined.
3. The algorithm is Gaussian elimination with partial pivoting. This method is very fast; some much slower algorithms can give better accuracy, but only by a small factor.

slaSp	the CIO locator s'	slaSp
--------------	----------------------	--------------

ACTION : For a given date, predict the quantity s' , positioning the terrestrial intermediate origin on the CIP equator.

CALL : `d = slaSp(date);`

GIVEN :

date	double	TDB Modified Julian Date (Note 1)
------	--------	-----------------------------------

RETURNED :

double	the quantity s' , in radians (Note 2)
--------	---

NOTES :

1. The date is TDB as an MJD ($= \text{JD} - 2400000.5$). TT (or even UTC) will do.
2. The quantity s' is obtained from polar motion observations by numerical integration, and so is in essence unpredictable. However, it is dominated by a secular drift of about $47 \mu\text{s}$ per century, which is the approximation evaluated by the present function.

REFERENCE : McCarthy, D.D., Petit, G. (eds.) 2004, *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG

slaSubet	remove E-terms	slaSubet
-----------------	----------------	-----------------

ACTION : Remove the E-terms (elliptic component of annual aberration) from a pre IAU 1976 catalogue $[\alpha, \delta]$ to give a mean place.

CALL : `slaSubet(rc, dc, eq, &rm, &dm);`

GIVEN :

rc,dc	double	$[\alpha, \delta]$ with E-terms included (radians)
eq	double	Besselian epoch of mean equator and equinox

RETURNED :

rm,dm	double*	$[\alpha, \delta]$ without E-terms (radians)
-------	---------	--

NOTE : Most star positions from pre-1984 optical catalogues (or obtained by astrometry with respect to such stars) have the E-terms built-in. This function converts such a position to a formal mean place (allowing, for example, comparison with a pulsar timing position).

REFERENCE : Seidelmann, P.K. *et al.*, 1992, *Explanatory Supplement to the Astronomical Almanac*, University Science Books, section 2D, p48.

slaSupgal	supergalactic to galactic	slaSupgal
------------------	---------------------------	------------------

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

CALL : slaSupgal(dsl, dsb, &dl, &db);

GIVEN :

dsl,dsb	double	supergalactic longitude and latitude (radians)
---------	--------	--

RETURNED :

dl,db	double*	galactic longitude and latitude $[l^II, b^II]$ (radians)
-------	---------	--

REFERENCES :

1. de Vaucouleurs, de Vaucouleurs, & Corwin, 1976, *Second Reference Catalogue of Bright Galaxies*, U.Texas, p8.
2. 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 $l^II = 137.37^\circ$.)

slaSvd singular value decomposition **slaSvd**

ACTION : Singular value decomposition. This function expresses a given matrix **A** as the product of three matrices **U**, **W**, **V^T**:

$$\mathbf{A} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T$$

where:

A is any m (rows) $\times n$ (columns) matrix, where $m \geq n$
U is an $m \times n$ column-orthogonal matrix
W is an $n \times n$ diagonal matrix with $w_{ii} \geq 0$
V^T is the transpose of an $n \times n$ orthogonal matrix

CALL : slaSvd(m, n, mp, np, a, w, v, work, &jstat);

GIVEN :

m,n	int	m, n , the numbers of rows and columns in matrix A
mp,np	int	physical dimensions of array containing matrix A
a	double[mp*np]	array containing $m \times n$ matrix A

RETURNED :

a	double[mp*np]	array containing $m \times n$ column-orthogonal matrix U
w	double[n]	$n \times n$ diagonal matrix W (diagonal elements only)
v	double[np*np]	array containing $n \times n$ orthogonal matrix V (<i>n.b.</i> not V^T)
work	double[n]	workspace
jstat	int*	0 = OK, -1 = array a wrong shape, >0 = index of w for which convergence failed (see note 3, below)

NOTES :

1. **m** and **n** are the *logical* dimensions of the matrices and vectors concerned, which can be located in arrays of larger *physical* dimensions, given by **mp** and **np**.
2. **v** contains matrix **V**, not the transpose of matrix **V**.
3. If the status **jstat** is greater than zero, this need not necessarily be treated as a failure. It means that, due to chance properties of the matrix **A**, the QR transformation phase of the function did not fully converge in a predefined number of iterations,

something that very seldom occurs. When this condition does arise, it is possible that the elements of the diagonal matrix \mathbf{W} have not been correctly found. However, in practice the results are likely to be trustworthy. Applications should report the condition as a warning, but then proceed normally.

4. The algorithm is an adaptation of the SVD subroutine in the *EISPACK* library (Garbow *et al.* 1977), which is a FORTRAN 66 implementation of the Algol procedure SVD of Wilkinson & Reinsch (1971). These references give full details of the algorithm used here. A good account of the use of SVD in least squares problems is given in Press *et al.* (1987), which includes another variant of the EISPACK code.
5. The lack of an “adjustable dimension” feature in C means that appropriate casts are needed in the application code. Here is an example:

```

:
double a[np][np], w[np], v[np][np], work[np];
int m, n, j;
:
slaSvd ( m, n, mp, np, (double*) a, w, (double*) v, work, &j );
:

```

REFERENCES :

1. Garbow *et al.* 1977, *EISPACK Guide Extension*, Springer Verlag.
2. Wilkinson & Reinsch 1971, *Handbook for Automatic Computation*, vol 2, ed, Bauer *et al.*, Springer Verlag).
3. Press *et al.* 1987, *Numerical Recipes*, Cambridge University Press.

slaSvdcov	covariance matrix from SVD	slaSvdcov
------------------	----------------------------	------------------

ACTION : From the \mathbf{W} and \mathbf{V} matrices from the SVD factorization of a matrix (as obtained from the `slaSvd` function), obtain the covariance matrix.

CALL : `slaSvdcov(n, np, nc, w, v, work, cvm);`

GIVEN :

n	int	n , the number of rows and columns in matrices \mathbf{W} and \mathbf{V}
np	int	first dimension of array containing $n \times n$ matrix \mathbf{V}
nc	int	first dimension of covariance matrix
w	double[n]	$n \times n$ diagonal matrix \mathbf{W} (diagonal elements only)
v	double[np*np]	array containing $n \times n$ orthogonal matrix \mathbf{V}

RETURNED :

work	double[n]	workspace
cvm	double[nc*nc]	array to receive covariance matrix

NOTE : The lack of an “adjustable dimension” feature in C means that appropriate casts are needed in the application code. Here is an example:

```

:
double w[np], v[np][np], work[np], c[nc][nc];
int n;
:
slaSvdcov ( n, np, nc, w, (double *) v, work, (double *) c );
:

```

REFERENCE : Press *et al.* 1987, *Numerical Recipes*, Cambridge University Press, §14.3.

slaSvdsol	solution vector from SVD	slaSvdsol
------------------	--------------------------	------------------

ACTION : From a given vector and the SVD of a matrix (as obtained from the slaSvd function), obtain the solution vector. This function solves the equation:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

where:

A	is a given m (rows) $\times n$ (columns) matrix, where $m \geq n$
x	is the n -vector we wish to find, and
b	is a given m -vector

by means of the *Singular Value Decomposition* method (SVD).

CALL : slaSvdsol(m, n, mp, np, b, u, w, v, work, x);

GIVEN :

m,n	int	m, n , the numbers of rows and columns in matrix A
mp,np	int	physical dimensions of array containing matrix A
b	D(M)	known vector b
u	D(MP,NP)	array containing $m \times n$ matrix U
w	D(N)	$n \times n$ diagonal matrix W (diagonal elements only)
v	D(NP,NP)	array containing $n \times n$ orthogonal matrix V

RETURNED :

work	double[n]	workspace
x	double[n]	unknown vector x

NOTES :

1. In the Singular Value Decomposition method (SVD), the matrix **A** is first factorized (for example by the function `slaSvd`) into the following components:

$$\mathbf{A} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T$$

where:

A	is any m (rows) $\times n$ (columns) matrix, where $m > n$
U	is an $m \times n$ column-orthogonal matrix
W	is an $n \times n$ diagonal matrix with $w_{ii} \geq 0$
V^T	is the transpose of an $n \times n$ orthogonal matrix

Note that m and n are the *logical* dimensions of the matrices and vectors concerned, which can be located in arrays of larger *physical* dimensions MP and NP. The solution is then found from the expression:

$$\mathbf{x} = \mathbf{V} \cdot [\text{diag}(1/\mathbf{W}_j)] \cdot (\mathbf{U}^T \cdot \mathbf{b})$$

2. If matrix **A** is square, and if the diagonal matrix **W** is not altered, the method is equivalent to conventional solution of simultaneous equations.
3. If $m > n$, the result is a least-squares fit.
4. If the solution is poorly determined, this shows up in the SVD factorization as very small or zero \mathbf{W}_j values. Where a \mathbf{W}_j value is small but non-zero it can be set to zero to avoid ill effects. The present function detects such zero \mathbf{W}_j values and produces a sensible solution, with highly correlated terms kept under control rather than being allowed to elope to infinity, and with meaningful values for the other terms.
5. The lack of an “adjustable dimension” feature in C means that appropriate casts are needed in the application code. Here is an example:

```
double a[mp][np], w[np], v[np][np], work[np], b[mp], x[np];
int m, n;
:
slaSvdsol ( m, n, mp, np, b, (double *) a, w, (double *) v, work, x );
:
```

REFERENCE : Press *et al.* 1987, *Numerical Recipes*, Cambridge University Press, §2.9.

slaTp2s

tangent plane to spherical

slaTp2s

ACTION : Transform tangent plane coordinates into spherical coordinates (single precision)

CALL : `slaTp2s(xi, eta, raz, decz, &ra, &dec);`

GIVEN :

xi,eta	float	tangent plane rectangular coordinates (radians)
raz,decz	float	spherical coordinates of tangent point (radians)

RETURNED :

ra,dec	float*	spherical coordinates (radians)
---------------	---------------	---------------------------------

NOTES :

1. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
2. When working in $[x, y, z]$ rather than spherical coordinates, the equivalent Cartesian function **slaTp2v** is available.

slaTp2v	tangent plane to $[x, y, z]$	slaTp2v
----------------	------------------------------	----------------

ACTION : Given the tangent-plane coordinates of a star and the direction cosines of the tangent point, determine the direction cosines of the star (single precision).

CALL : **slaTp2v**(**xi**, **eta**, **v0**, **v**);

GIVEN :

xi,eta	float	tangent plane coordinates of star (radians)
v0	float[3]	direction cosines of tangent point

RETURNED :

v	float[3]	direction cosines of star
----------	-----------------	---------------------------

NOTES :

1. If vector **v0** is not of unit length, the returned vector **v** will be wrong.
2. If vector **v0** points at a pole, the returned vector **v** will be based on the arbitrary assumption that $\alpha = 0$ at the tangent point.
3. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.

4. This function is the Cartesian equivalent of the function `slaTp2s`.

slaTps2c	plate centre from $[\xi, \eta]$ and $[\alpha, \delta]$	slaTps2c
-----------------	--	-----------------

ACTION : From the tangent plane coordinates of a star of known $[\alpha, \delta]$, determine the $[\alpha, \delta]$ of the tangent point (single precision)

CALL : `slaTps2c(xi, eta, ra, dec, &raz1, &decz1, &raz2, &decz2, &n);`

GIVEN :

<code>xi,eta</code>	<code>float</code>	tangent plane rectangular coordinates (radians)
<code>ra,dec</code>	<code>float</code>	spherical coordinates (radians)

RETURNED :

<code>raz1,decz1</code>	<code>float*</code>	spherical coordinates of tangent point, solution 1
<code>raz2,decz2</code>	<code>float*</code>	spherical coordinates of tangent point, solution 2
<code>n</code>	<code>int*</code>	number of solutions:
		0 = no solutions returned (Note 2)
		1 = only the first solution is useful (Note 3)
		2 = there are two useful solutions (Note 3)

NOTES :

1. The `raz1` and `raz2` values returned are in the range $0-2\pi$.
2. 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 ξ value, and hence it is meaningless to ask where the tangent point would have to be to bring about this combination of ξ and δ .
3. 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 “beyond the pole” case, and this is reflected in the values of `raz2` and `decz2` which are returned.
4. The `decz1` and `decz2` values returned are in the range $\pm\pi$, but in the ordinary, non-pole-crossing, case, the range is $\pm\pi/2$.
5. `ra, dec, raz1, decz1, raz2, decz2` are all in radians.
6. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
7. When working in $[x, y, z]$ rather than spherical coordinates, the equivalent Cartesian function `slaTpv2c` is available.

slaTpv2c	plate centre from $[\xi, \eta]$ and $[x, y, z]$	slaTpv2c
-----------------	---	-----------------

ACTION : From the tangent plane coordinates of a star of known direction cosines, determine the direction cosines of the tangent point (single precision)

CALL : slaTpv2c(xi, eta, v, v01, v02, &n);

GIVEN :

xi,eta	float	tangent plane coordinates of star (radians)
v	float[3]	direction cosines of star

RETURNED :

v01	float[3]	direction cosines of tangent point, solution 1
v01	float[3]	direction cosines of tangent point, solution 2
n	int*	number of solutions:
		0 = no solutions returned (Note 2)
		1 = only the first solution is useful (Note 3)
		2 = there are two useful solutions (Note 3)

NOTES :

1. The vector **v** must be of unit length or the result will be wrong.
2. 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 ξ value.
3. 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 can be regarded as valid if the vector **v02** is interpreted as the “beyond the pole” case.
4. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
5. This function is the Cartesian equivalent of the function **slaTps2c**.

slaUe2el	universal to conventional elements	slaUe2el
-----------------	------------------------------------	-----------------

ACTION : Transform universal elements into conventional heliocentric osculating elements.

CALL : slaUe2el(u, jformr,
 &jform, &epoch, &orbinc, &anode, &perih,
 &aorq, &e, &aorl, &dm, &jstat);

GIVEN :

u	double[13]	universal orbital elements (updated; Note 1)
	[0]	◦ combined mass ($M + m$)
	[1]	◦ total energy of the orbit (α)
	[2]	◦ reference (osculating) epoch (t_0)
	[3–5]	◦ position at reference epoch (\mathbf{r}_0)
	[6–8]	◦ velocity at reference epoch (\mathbf{v}_0)
	[9]	◦ heliocentric distance at reference epoch
	[10]	◦ $\mathbf{r}_0 \cdot \mathbf{v}_0$
	[11]	◦ date (t)
	[12]	◦ universal eccentric anomaly (ψ) of date, approx.
jformr	int	requested element set (1-3; Note 3)

RETURNED :

jform	int*	element set actually returned (1-3; Note 4)
epoch	double*	epoch of elements (t_0 or T , TDB MJD)
orbinc	double*	inclination (i , radians)
anode	double*	longitude of the ascending node (Ω , radians)
perih	double*	longitude or argument of perihelion (ϖ or ω , radians)
aorq	double*	mean distance or perihelion distance (a or q , AU)
e	double*	eccentricity (e)
aorl	double*	mean anomaly or longitude (M or L , radians, jform = 1, 2 only)
dm	double*	daily motion (n , radians, jform = 1 only)
jstat	int*	status:
		0 = OK
		-1 = illegal pmass
		-2 = illegal jformr
		-3 = position/velocity out of allowed range

NOTES :

1. 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) α , 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 ψ , the “universal eccentric anomaly” at a given date and (v) that date.

2. 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.
3. Three different element-format options are supported, as follows.

`jform` = 1, suitable for the major planets:

<code>epoch</code>	= epoch of elements t_0 (TDB MJD)
<code>orbinc</code>	= inclination i (radians)
<code>anode</code>	= longitude of the ascending node Ω (radians)
<code>perih</code>	= longitude of perihelion ϖ (radians)
<code>aorq</code>	= mean distance a (AU)
<code>e</code>	= eccentricity e ($0 \leq e < 1$)
<code>aorl</code>	= mean longitude L (radians)
<code>dm</code>	= daily motion n (radians)

`jform` = 2, suitable for minor planets:

<code>epoch</code>	= epoch of elements t_0 (TDB MJD)
<code>orbinc</code>	= inclination i (radians)
<code>anode</code>	= longitude of the ascending node Ω (radians)
<code>perih</code>	= argument of perihelion ω (radians)
<code>aorq</code>	= mean distance a (AU)
<code>e</code>	= eccentricity e ($0 \leq e < 1$)
<code>aorl</code>	= mean anomaly M (radians)

`jform` = 3, suitable for comets:

<code>epoch</code>	= epoch of perihelion T (TDB MJD)
<code>orbinc</code>	= inclination i (radians)
<code>anode</code>	= longitude of the ascending node Ω (radians)
<code>perih</code>	= argument of perihelion ω (radians)
<code>aorq</code>	= perihelion distance q (AU)
<code>e</code>	= eccentricity e ($0 \leq e \leq 10$)

4. 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:

<code>jformr</code>	<code>jform</code>	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

5. The arguments returned for each value of **jform** (*cf.* Note 5: **jform** might not be the same as **jformr**) are as follows:

jform	1	2	3
epoch	t_0	t_0	T
orbinc	i	i	i
anode	Ω	Ω	Ω
perih	ϖ	ω	ω
aorq	a	a	q
e	e	e	e
aorl	L	M	-
dm	n	-	-

where:

t_0	is the epoch of the elements (MJD, TT)
T	is the epoch of perihelion (MJD, TT)
i	is the inclination (radians)
Ω	is the longitude of the ascending node (radians)
ϖ	is the longitude of perihelion (radians)
ω	is the argument of perihelion (radians)
a	is the mean distance (AU)
q	is the perihelion distance (AU)
e	is the eccentricity
L	is the longitude (radians, $0 - 2\pi$)
M	is the mean anomaly (radians, $0 - 2\pi$)
n	is the daily motion (radians)
-	means no value is set

6. 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.
7. The various epochs are in the TDB time-scale (TT will do) and are Modified Julian Dates (JD-2400000.5).

REFERENCES :

1. Sterne, Theodore E. 1960 *An Introduction to Celestial Mechanics*, Interscience Publishers, Section 6.7, p199.

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

slaUe2pv position & velocity from universal elements **slaUe2pv**

ACTION : Heliocentric position and velocity of a planet, asteroid or comet, starting from orbital elements in the “universal variables” form.

CALL : slaUe2pv(date, u, pv, &jstat);

GIVEN :

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

GIVEN and RETURNED :

u	double[13]	universal orbital elements (updated; Note 1)
	[0]	◦ combined mass ($M + m$)
	[1]	◦ total energy of the orbit (α)
	[2]	◦ reference (osculating) epoch (t_0)
	[3–5]	◦ position at reference epoch (\mathbf{r}_0)
	[6–8]	◦ velocity at reference epoch (\mathbf{v}_0)
	[9]	◦ heliocentric distance at reference epoch
	[10]	◦ $\mathbf{r}_0 \cdot \mathbf{v}_0$
	[11]	◦ date (t)
	[12]	◦ universal eccentric anomaly (ψ) of date, approx.

RETURNED :

pv	double[6]	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s; Note 1)
jstat	int*	status:
		0 = OK
		–1 = radius vector zero
		–2 = failed to converge

NOTES :

1. 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) α , 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 ψ , the “universal eccentric anomaly” at a given date and (v) that date.

2. The companion function is `slaE12ue`. This takes the conventional orbital elements and transforms them into the set of numbers needed by the present function. A single prediction requires one call to `slaE12ue` followed by one call to the present function; for convenience, the two calls are packaged as the function `slaPlanel`. Multiple predictions may be made by again calling `slaE12ue` once, but then calling the present function multiple times, which is faster than multiple calls to `slaPlanel`.

It is not obligatory to use `slaE12ue` to obtain the parameters. However, it should be noted that because `slaE12ue` performs its own validation, no checks on the contents of the array `U` are made by the present function.

3. `date` is the instant for which the prediction is required. It is in the TDB time-scale (TT will do) and is a Modified Julian Date (JD−2400000.5).
4. The universal elements supplied in the array `u` are in canonical units (solar masses, AU and canonical days). The position and velocity are not sensitive to the choice of reference frame. The `slaE12ue` function in fact produces coordinates with respect to the J2000 equator and equinox.
5. The algorithm was originally adapted from the EPHSLA Fortran program of D. H. P. Jones (private communication, 1996). The method is based on Stumpff’s Universal Variables.

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

slaUnpcd	remove radial distortion	slaUnpcd
-----------------	--------------------------	-----------------

ACTION : Remove pincushion/barrel distortion from a distorted $[x, y]$ to give tangent-plane $[x, y]$.

CALL : `slaUnpcd(disco, &x, &y);`

GIVEN :

<code>disco</code>	double	pincushion/barrel distortion coefficient
<code>x,y</code>	double*	distorted $[x, y]$

RETURNED :

<code>x,y</code>	double*	tangent-plane $[x, y]$
------------------	---------	------------------------

NOTES :

1. The distortion is of the form $\rho = r(1 + cr^2)$, where r is the radial distance from the tangent point, c is the `disco` argument, and ρ is the radial distance in the presence of the distortion.
2. For *pincushion* distortion, c is positive; for *barrel* distortion, c is negative.
3. For x, y in units of one projection radius (in the case of a photographic plate, the focal length), the following `disco` values apply:

<i>geometry</i>	disco
astrograph	0.0
Schmidt	-0.3333
AAT PF doublet	+147.069
AAT PF triplet	+178.585
AAT f/8	+21.20
JKT f/8	+14.6

4. The present function is a rigorous inverse of the companion function `slaPcd`. The expression for ρ in Note 1 is rewritten in the form $x^3 + ax + b = 0$ and solved by standard techniques.
5. Cases where the cubic has multiple real roots can sometimes occur, corresponding to extreme instances of barrel distortion where up to three different undistorted $[x, y]$ s all produce the same distorted $[x, y]$. However, only one solution is returned, the one that produces the smallest change in $[x, y]$.

slaV2tp	$[x, y, z]$ to tangent plane	slaV2tp
----------------	------------------------------	----------------

ACTION : Given the direction cosines of a star and of the tangent point, determine the star's tangent-plane coordinates (single precision).

CALL : `slaV2tp(v, v0, &xi, &eta, &j);`

GIVEN :

<code>v</code>	<code>float[3]</code>	direction cosines of star
<code>v0</code>	<code>float[3]</code>	direction cosines of tangent point

RETURNED :

<code>xi,eta</code>	<code>float*</code>	tangent plane coordinates (radians)
<code>j</code>	<code>int*</code>	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

NOTES :

1. If vector **v0** is not of unit length, or if vector **v** is of zero length, the results will be wrong.
2. If **v0** points at a pole, the returned ξ, η will be based on the arbitrary assumption that $\alpha = 0$ at the tangent point.
3. The projection is called the *gnomonic* projection; the Cartesian coordinates $[\xi, \eta]$ are called *standard coordinates*. The latter are in units of the distance from the tangent plane to the projection point, *i.e.* radians near the origin.
4. This function is the Cartesian equivalent of the function **slaS2tp**.

slaVdv	scalar product	slaVdv
---------------	----------------	---------------

ACTION : Scalar product of two 3-vectors (single precision).

CALL : **r** = **slaVdv**(**va**, **vb**);

GIVEN :

va	float[3]	first vector
vb	float[3]	second vector

RETURNED :

float*	scalar product va.vb
--------	-----------------------------

slaVn	normalize vector	slaVn
--------------	------------------	--------------

ACTION : Normalize a 3-vector, also giving the modulus (single precision).

CALL : **slaVn**(**v**, **uv**, **&vm**);

GIVEN :

v	float[3]	vector
----------	----------	--------

RETURNED :

uv	float[3]	unit vector \hat{v}
vm	float*	modulus $ v $

NOTE : If the modulus of v is zero, uv is set to the null vector.

slaVxv	vector product	slaVxv
---------------	----------------	---------------

ACTION : Vector product of two 3-vectors (single precision).

CALL : `slaVxv(va, vb, vc);`

GIVEN :

<code>va</code>	<code>float[3]</code>	first vector
<code>vb</code>	<code>float[3]</code>	second vector

RETURNED :

<code>vc</code>	<code>float[3]</code>	vector product $va \times vb$
-----------------	-----------------------	-------------------------------

slaXy2xy	apply linear model to an $[x, y]$	slaXy2xy
-----------------	-----------------------------------	-----------------

ACTION : Transform one $[x, y]$ into another using a linear model of the type produced by the `slaFitxy` function.

CALL : `slaXy2xy(x1, y1, coeffs, &x2, &y2);`

GIVEN :

<code>x1,y1</code>	<code>double</code>	[x, y] before transformation
<code>coeffs</code>	<code>double[6]</code>	transformation coefficients (see note 1)

RETURNED :

<code>x2,y2</code>	<code>double*</code>	[x, y] after transformation
--------------------	----------------------	---------------------------------

NOTES :

1. The model relates two sets of $[x, y]$ coordinates as follows. Naming the six elements of `coeffs` a, b, c, d, e & f , the present function performs the transformation:

$$x_2 = a + bx_1 + cy_1$$

$$y_2 = d + ex_1 + fy_1$$

2. See also `slaFitxy`, `slaPxy`, `slaInvf`, `slaDcmpf`.

slaZd	$[h, \delta]$ to zenith distance	slaZd
--------------	----------------------------------	--------------

ACTION : Hour angle and declination to zenith distance.

CALL : `d = slaZd(ha, dec, phi);`

GIVEN :

<code>ha</code>	<code>double</code>	hour angle in radians
<code>dec</code>	<code>double</code>	declination in radians
<code>phi</code>	<code>double</code>	latitude in radians

RETURNED :

<code>double</code>	zenith distance (radians, $0-\pi$)
---------------------	-------------------------------------

NOTES :

1. The latitude must be geodetic. In critical applications, corrections for polar motion should be applied (see `slaPolmo`).
2. 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 zenith distance. In particular, it may be important to distinguish between the zenith distance as affected by refraction, which would require the *observed* $[h, \delta]$, and the zenith distance *in vacuo*, which would require the *topocentric* $[h, \delta]$. If the effects of diurnal aberration can be neglected, the *apparent* $[h, \delta]$ may be used instead of the *topocentric* $[h, \delta]$.
3. No range checking of arguments is done.
4. In applications which involve many zenith distance calculations, rather than calling the present function it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude, and perhaps sine and cosine of declination.

5 SUMMARY OF CALLS

The basic trigonometrical and numerical facilities are supplied in both single precision (`float`) and double precision (`double`) versions. Most of the more esoteric position and time functions use double precision arguments only, even in cases where single precision would normally be adequate in practice. Certain functions with modest accuracy objectives are supplied in single precision versions only. In the calling sequences which follow, no attempt has been made to distinguish between single and double precision argument names, and frequently the same name is used on different occasions to mean different things. However, none of the functions uses a mixture of single and double precision arguments; each function is either wholly single precision or wholly double precision.

In the classified list, below, functions with calls beginning '`r=`' return a `float` whereas those beginning '`d=`' return a `double`; the rest are `void`.

The list is, of course, merely for quick reference; inexperienced users **must** refer to the detailed specifications given later. In particular, **don't guess** whether arguments are single or double precision; the result could be a program that happens to work on one platform but not on another.

String Decoding

```
slaInt2in( string, &nstrt, &ireslt, &j );
slaIntin( string, &nstrt, &lreslt, &j );
    Convert free-format string into integer

slaFlotin( string, &nstrt, &reslt, &j );
slaDfltin( string, &nstrt, &dreslt, &j );
    Convert free-format string into floating-point number

slaAfin( string, &nstrt, &reslt, &j );
slaDafin( string, &nstrt, &dreslt, &j );
    Convert free-format string from deg,arcmin,arcsec to radians

slaDbjin( string, &nstrt, &dreslt, &j1, &j2 );
    Like slaDfltin but with extensions to accept leading 'B' and 'J'
```

Sexagesimal Conversions

```
slaCtf2d( ihour, imin, sec, &days, &j );
slaDtf2d( ihour, imin, sec, &days, &j );
    Hours, minutes, seconds to days

slaCd2tf( ndp, days, &sign, ihmsf );
slaDd2tf( ndp, days, &sign, ihmsf );
    Days to hours, minutes, seconds
```

```

slaCtf2r( ihour, imin, sec, &rad, &j );
slaDtf2r( ihour, imin, sec, &rad, &j );
    Hours, minutes, seconds to radians

slaCr2tf( ndp, angle, &sign, ihmsf );
slaDr2tf( ndp, angle, &sign, ihmsf );
    Radians to hours, minutes, seconds

slaCaf2r( ideg, iamin, asec, &rad, &j );
slaDaf2r( ideg, iamin, asec, &rad, &j );
    Degrees, arcminutes, arcseconds to radians

slaCr2af( ndp, angle, &sign, idmsf );
slaDr2af( ndp, angle, &sign, idmsf );
    Radians to degrees, arcminutes, arcseconds

```

Angles, Vectors and Rotation Matrices

```

r = slaRange( angle );
d = slaDrange( angle );
    Normalize angle into range  $\pm\pi$ 

r = slaRanorm( angle );
d = slaDranrm( angle );
    Normalize angle into range  $0-2\pi$ 

slaCs2c( a, b, v );
slaDcs2c( a, b, v );
    Spherical coordinates to  $[x, y, z]$ 

slaCc2s( v, &a, &b );
slaDcc2s( v, &a, &b );
     $[x, y, z]$  to spherical coordinates

r = slaVdv( va, vb );
d = slaDvdv( va, vb );
    Scalar product of two 3-vectors

slaVxv( va, vb, vc );
slaDvxv( va, vb, vc );
    Vector product of two 3-vectors

slaVn( v, uv, &vm );
slaDvn( v, uv, &vm );
    Normalize a 3-vector also giving the modulus

r = slaSep( a1, b1, a2, b2 );
d = slaDsep( a1, b1, a2, b2 );
    Angle between two points on a sphere

```

```

r = slaSepv( v1, v2 );
d = slaDsepv( v1, v2 );
    Angle between two  $[x, y, z]$  vectors

r = slaBear( a1, b1, a2, b2 );
d = slaDbear( a1, b1, a2, b2 );
    Direction of one point on a sphere seen from another

r = slaPav( v1, v2 );
d = slaDpav( v1, v2 );
    Position-angle of one  $[x, y, z]$  with respect to another

slaEuler( order, phi, theta, psi, rmat );
slaDeuler( order, phi, theta, psi, rmat );
    Form rotation matrix from three Euler angles

slaAv2m( axvec, rmat );
slaDav2m( axvec, rmat );
    Form rotation matrix from axial vector

slaM2av( rmat, axvec );
slaDm2av( rmat, axvec );
    Determine axial vector from rotation matrix

slaMxv( rm, va, vb );
slaDmxv( dm, va, vb );
    Rotate vector forwards

slaImxv( rm, va, vb );
slaDimxv( dm, va, vb );
    Rotate vector backwards

slaMxm( a, b, c );
slaDmxm( a, b, c );
    Product of two  $3 \times 3$  matrices

slaCs2c6( a, b, r, ad, bd, rd, v );
slaDs2c6( a, b, r, ad, bd, rd, v );
    Transformation of position and velocity in spherical coordinates to Cartesian coordinates

slaCc62s( v, &a, &b, &r, &ad, &bd, &rd );
slaDc62s( v, &a, &b, &r, &ad, &bd, &rd );
    Transformation of position and velocity in Cartesian coordinates to spherical coordinates

```

Calendars

```

slaCldj( iy, im, id, &djm, &j );
    Gregorian Calendar to Modified Julian Date

```



```

slaCaldj( iy, im, id, &djm, &j );
    Gregorian Calendar to Modified Julian Date, permitting century default

slaDjcal( ndp, djm, iymdf, &j );
    Modified Julian Date to Gregorian Calendar, in a form convenient for formatted
    output

slaDjcl( djm, &iy, &im, &id, &fd, &j );
    Modified Julian Date to Gregorian Year, Month, Day, Fraction

slaCalyd( iy, im, id, &ny, &nd, &j );
    Calendar to year and day in year, permitting century default

slaClyd( iy, im, id, &ny, &nd, &j );
    Calendar to year and day in year

d = slaEpb( date );
    Modified Julian Date to Besselian Epoch

d = slaEpb2d( epb );
    Besselian Epoch to Modified Julian Date

d = slaEpj( date );
    Modified Julian Date to Julian Epoch

d = slaEpj2d( epj );
    Julian Epoch to Modified Julian Date

```

Time-Scales

```

d = slaDat( utc );
    Offset of Atomic Time from Coordinated Universal Time: TAI–UTC

d = slaDt( epoch );
    Approximate offset between dynamical time and universal time

d = slaDtt( utc );
    Offset of Terrestrial Time from Coordinated Universal Time: TT–UTC

d = slaRcc( tdb, ut1, wlfq, u, v );
    Relativistic clock correction: TDB–TT

```

Earth Rotation and Sidereal Time

```

d = slaEqeqx( date );
    Equation of the equinoxes, SF2001

d = slaEra( date, ut1 );
    Earth rotation angle, IAU 2000

```

```
d = slaGmst( ut1 );
    Transformation from Universal Time to sidereal time, IAU 1982

d = slaGmsta( date, ut1 );
    Transformation from Universal Time to sidereal time, IAU 1982, rounding errors
    minimized

d = slaGst( date, utdate, ut1 );
    Greenwich apparent sidereal time, IAU 2006
```

Precession and Nutation

```
d = slaEors( date );
    Equation of the origins, IAU 2006

slaFw2m( gamb, phib, psi, eps, r );
    Rotation matrix, given the Fukushima-Williams angles

slaFw2xy( gamb, phib, psi, eps, &x, &y );
    CIP  $X, Y$ , given Fukushima-Williams bias-precession-nutation angles

slaG2ixys( x, y, s, rg2i );
    Form the GCRS to CIRS matrix, given  $X, Y$  and  $s$ 

slaNu( date, &dpsi, &deps );
    Components of nutation, IAU 2006

slaNu00a( date, &dpsi, &deps );
    Components of nutation, IAU 2000A

slaNut( date, rmatn );
    Nutation matrix, SF2001

slaNutc( date, &dpsi, &deps, &eps0 );
    Components of nutation, and mean obliquity, SF2001

slaNutc80( date, &dpsi, &deps, &eps0 );
    Components of nutation, and mean obliquity, IAU 1980

slaPfw( date, &gamb, &phib, &psib, &epsa );
    Precession angles, Fukushima-Williams, IAU 2006

slaPncio( date, rnpb );
    CIO based precession-nutation matrix, IAU 2006

slaPneqx( date, rnpb );
    Equinox based precession-nutation matrix, IAU 2006

slaPrec( ep0, ep1, rmatp );
    Precession matrix, IAU 1976

slaPrebn( bep0, bep1, rmatp );
    Precession matrix, FK4 system
```

```
slaPreces( system, ep0, ep1, &ra, &dc );
    Precession, in either the FK4 or FK5 system

slaPrecl( ep0, ep1, rmatp );
    Precession matrix, Simon et al.

slaPrenut( epoch, date, rmatpn );
    Combined precession-nutation matrix, SF2001

d = slaS( date, x, y );
    The CIO locator s
```

Proper Motion

```
slaPm( r0, d0, pr, pd, px, rv, ep0, ep1, &r1, &d1 );
    Adjust for proper motion
```

FK4/FK5/Hipparcos Transformations

```
slaFk425( r1950, d1950, dr1950, dd1950, p1950, v1950,
           &r2000, &d2000, &dr2000, &dd2000, &p2000, &v2000 );
    Convert B1950.0 FK4 star data to J2000.0 FK5

slaFk45z( r1950, d1950, epoch, &r2000, &d2000 );
    Convert B1950.0 FK4 position to J2000.0 FK5 assuming zero FK5 proper motion
    and no parallax

slaFk524( r2000, d2000, dr2000, dd2000, p2000, v2000,
           &r1950, &d1950, &dr1950, &dd1950, &p1950, &v1950 );
    Convert J2000.0 FK5 star data to B1950.0 FK4

slaFk54z( r2000, d2000, bepoch, &r1950, &d1950, &dr1950, &dd1950 );
    Convert J2000.0 FK5 position to B1950.0 FK4 assuming zero FK5 proper motion
    and no parallax

slaFk52h( r5, d5, dr5, dd5, &rh, &dh, &drh, &ddh );
    Convert J2000.0 FK5 star data to Hipparcos

slaFk5hz( r5, d5, epoch, &rh, &dh );
    Convert J2000.0 FK5 position to Hipparcos assuming zero Hipparcos proper motion

slaH2fk5( rh, dh, drh, ddh, &r5, &d5, &dr5, &dd5 );
    Convert Hipparcos star data to J2000.0 FK5

slaHfk5z( rh, dh, epoch, &r5, &d5, &dr5, &dd5 );
    Convert Hipparcos position to J2000.0 FK5 assuming zero Hipparcos proper motion

slaKbj( jb, e, &k, &j );
    Select epoch prefix 'B' or 'J'

d = slaEpco ( k0, k, e );
    Convert an epoch into the appropriate form – 'B' or 'J'
```

Elliptic Aberration

```
slaEtrms( ep, ev );
    E-terms

slaSubet( rc, dc, eq, &rm, &dm );
    Remove the E-terms

slaAddet( rm, dm, eq, &rc, &dc );
    Add the E-terms
```

Geographical and Geocentric Coordinates

```
slaObs( number, id, name, &wlong, &phi, &height );
    Interrogate list of observatory parameters

slaGeoc( p, h, &r, &z );
    Convert geodetic position to geocentric

slaPolmo( elongm, phim, xp, yp, &elong, &phi, &daz );
    Effect of polar motion on site coordinates and azimuth

slaPomom( date, xp, yp, rpom );
    Polar motion rotation matrix

slaPvobs( p, h, stl, pv );
    Position and velocity of observatory

d = slaSp ( date );
    TIO locator  $s'$ 
```

ICRS-CIRS-Observed Transformations

```
slaC2i( rc, dc, pr, pd, px, rv, date, dx, dy, &ri, &di );
    ICRS to CIRS

slaC2ipa( date, dx, dy, &ciprms, &eo );
    Precompute ICRS to CIRS parameters, terrestrial observer

slaC2ipad( &ciprms );
    In ICRS to CIRS parameters, disable bias-precession-nutation

slaC2ipas( date, pv, dx, dy, &ciprms, &eo );
    Precompute ICRS to CIRS parameters given observer's position and velocity

slaC2iqk( rc, dc, pr, pd, px, rv, &ciprms, &ri, &di );
    ICRS to CIRS using precomputed parameters

slaC2iqkz( rc, dc, &ciprms, &ri, &di );
    ICRS to CIRS using precomputed parameters, for zero proper motion and parallax
```

```

slaI2c( ri, di, date, dx, dy, &rc, &dc );
    CIRS to ICRS astrometric

slaI2cqk( ri, di, &ciprms, &rc, &dc );
    CIRS to ICRS astrometric using precomputed parameters

slaI2o( ri, di, utc, dut, elongm, phim, hm, xp, yp,
        tk, phpa, rh, wlfq, tlr, &aob, &zob, &hob, &dob, &rob );
    CIRS to observed place

slaI2opa( utc, dut, elongm, phim, hm, xp, yp,
        tk, phpa, rh, wlfq, tlr, &ioprms );
    Precompute CIRS to observed parameters

slaI2opad( &ioprms );
    In the CIRS to observed place parameters, disable diurnal aberration

slaI2opat( utc, &ioprms );
    Update Earth rotation angle in CIRS to observed parameters

slaI2oqk( ri, di, &ioprms, &aob, &zob, &hob, &dob, &rob );
    CIRS to observed using precomputed parameters

slaO2i( type, ob1, ob2, utc, dut, elongm, phim, hm, xp, yp,
        tk, phpa, rh, wlfq, tlr, &ri, &di );
    Observed to CIRS

slaO2iqk( type, ob1, ob2, &ioprms, &ri, &di );
    Observed to CIRS using precomputed parameters

```

Mean-Apparent-Observed Transformations

```

slaMap( rm, dm, pr, pd, px, rv, eq, date, &ra, &da );
    Mean place to geocentric apparent place

slaMappa( eq, date, amprms );
    Precompute mean to apparent parameters

slaMapqk( rm, dm, pr, pd, px, rv, amprms, &ra, &da );
    Mean to apparent using precomputed parameters

slaMapqkz( rm, dm, amprms, &ra, &da );
    Mean to apparent using precomputed parameters, for zero proper motion and par-
    allax

slaAmp( ra, da, date, eq, &rm, &dm );
    Geocentric apparent place to mean place

slaAmpqk( ra, da, amprms, &rm, &dm );
    Apparent to mean using precomputed parameters

slaAop( rap, dap, utc, dut, elongm, phim, hm, xp, yp,
        tk, phpa, rh, wlfq, tlr, &aob, &zob, &hob, &dob, &rob );
    Apparent place to observed place

```

```

slaAoppa( utc, dut, elongm, phim, hm, xp, yp,
          tk, phpa, rh, wlfq, tlr, aoprms );
    Precompute apparent to observed parameters

slaAoppat( utc, aoprms );
    Update sidereal time in apparent to observed parameters

slaAopqk( rap, dap, aoprms, &aob, &zob, &hob, &dob, &rob );
    Apparent to observed using precomputed parameters

slaOap( type, ob1, ob2, utc, dut, elongm, phim, hm, xp, yp,
        tk, phpa, rh, wlfq, tlr, &rap, &dap );
    Observed to apparent

slaOapqk( type, ob1, ob2, aoprms, &ra, &da );
    Observed to apparent using precomputed parameters

```

Azimuth and Elevation

```

slaAltaz( ha, dec, phi,
          &az, &azd, &azdd, &el, &eld, &eldd, &pa, &pad, &padd );
    Positions, velocities etc. for an altazimuth mount

slaE2h( ha, dec, phi, &az, &el );
slaDe2h( ha, dec, phi, &az, &el );
     $[h, \delta]$  to  $[Az, El]$ 

slaH2e( az, el, phi, &ha, &dec );
slaDh2e( az, el, phi, &ha, &dec );
     $[Az, El]$  to  $[h, \delta]$ 

slaPda2h( p, d, a, &h1, &j1, &h2, &j2 );
    Hour angle corresponding to a given azimuth

slaPdq2h( p, d, q, &h1, &j1, &h2, &j2 );
    Hour angle corresponding to a given parallactic angle

d = slaPa( ha, dec, phi );
     $[h, \delta]$  to parallactic angle

d = slaZd( ha, dec, phi );
     $[h, \delta]$  to zenith distance

```

Refraction and Air Mass

```

slaRefro( zobs, hm, tk, phpa, rh, wlfq, phi, tlr, eps, &ref );
    Change in zenith distance due to refraction

slaRefco( hm, tk, phpa, rh, wlfq, phi, tlr, eps, &refa, &refb );
    Constants for simple refraction model ( accurate )

```

```

slaRefcoq( tk, phpa, rh, wl, &refa, &refb );
    Constants for simple refraction model ( fast )

slaAtmdsp( tk, phpa, rh, wl1, refa1, refb1, wl2, &refa2, &refb2 );
    Adjust refraction constants for colour

slaRefz( zu, refa, refb, &zr );
    Unrefracted to refracted ZD, simple model

slaRefv( vu, refa, refb, vr );
    Unrefracted to refracted [ Az, El ] vector, simple model

d = slaAirmas( zd );
    Air mass

```

Ecliptic Coordinates

```

slaEcmat( date, rmat );
    Equatorial to ecliptic rotation matrix

slaEqec1( dr, dd, date, &dl, &db );
    J2000.0 ‘FK5’ to ecliptic coordinates

slaEc1eq( dl, db, date, &dr, &dd );
    Ecliptic coordinates to J2000.0 ‘FK5’

```

Galactic Coordinates

```

slaEg50( dr, dd, &dl, &db );
    B1950.0 ‘FK4’ to galactic

slaGe50( dl, db, &dr, &dd );
    Galactic to B1950.0 ‘FK4’

slaEqgal( dr, dd, &dl, &db );
    J2000.0 ‘FK5’ to galactic

slaGaleq( dl, db, &dr, &dd );
    Galactic to J2000.0 ‘FK5’

```

Supergalactic Coordinates

```

slaGalsup( dl, db, &dsl, &dsb );
    Galactic to supergalactic

slaSupgal( dsl, dsb, &dl, &db );
    Supergalactic to galactic

```

Ephemerides

```

slaDmoon( date, pv );
    Approximate geocentric position and velocity of the Moon

slaEarth( iy, id, fd, pv );
    Approximate heliocentric position and velocity of the Earth

slaEpv( date, dph, dvh, dpb, dvb );
    Heliocentric and barycentric position and velocity of the Earth (high-accuracy)

slaEvp( date, deqx, dvb, dpb, dvh, dph );
    Barycentric and heliocentric velocity and position of the Earth

slaMoon( iy, id, fd, pv );
    Approximate geocentric position and velocity of the Moon

slaPlanet( date, np, pv, &jstat );
    Approximate heliocentric position and velocity of a planet

slaRdplan( date, np, elong, phi, &ra, &dec, &diam );
    Approximate topocentric apparent place of a planet

slaPlanel( date, jform, epoch, orbinc, anode, perih,
            aorq, e, aorl, dm, pv, &jstat );
    Heliocentric position and velocity of a planet, asteroid or comet, starting from orbital
    elements

slaPlante( date, elong, phi, jform, epoch, orbinc, anode,
            perih, aorq, e, aorl, dm, &ra, &dec, &r, &jstat );
    Topocentric apparent place of a Solar-System object whose heliocentric orbital ele-
    ments are known

slaPlantu( date, elong, phi, u, &ra, &dec, &r, &jstat );
    Topocentric apparent place of a Solar-System object whose heliocentric universal
    orbital elements are known

slaPv2el( pv, date, pmass, jformr, &jform, &epoch, &orbinc,
            &anode, &perih, &aorq, &e, &aorl, &dm, &jstat );
    Orbital elements of a planet from instantaneous position and velocity

slaPertel( jform, date0, date1,
            epoch0, orbi0, anode0, perih0, aorq0, e0, am0,
            &epoch1, &orbi1, &anode1, &perih1, &aorq1, &e1, &am1,
            &jstat );
    Update elements by applying perturbations

slaEl2ue( date, jform, epoch, orbinc, anode,
            perih, aorq, e, aorl, dm,
            u, &jstat );
    Transform conventional elements to universal elements

```



```

slaUe2el( u, jformr,
          &jform, &epoch, &orbinc, &anode, &perih,
          &aorq, &e, &aorl, &dm, &jstat );
    Transform universal elements to conventional elements

slaPv2ue( pv, date, pmass, u, &jstat );
    Package a position and velocity for use as universal elements

slaUe2pv( date, u, pv, &jstat );
    Extract the position and velocity from universal elements

slaPertue( date, u, &jstat );
    Update universal elements by applying perturbations

r = slaRverot( phi, ra, da, st );
    Velocity component due to rotation of the Earth

slaEcor( rm, dm, iy, id, fd, &rv, &tl );
    Components of velocity and light time due to Earth orbital motion

r = slaRvlsrd( r2000, d2000 );
    Velocity component due to solar motion wrt dynamical LSR

r = slaRvlsrk( r2000, d2000 );
    Velocity component due to solar motion wrt kinematical LSR

r = slaRvgalc( r2000, d2000 );
    Velocity component due to rotation of the Galaxy

r = slaRvlg( r2000, d2000 );
    Velocity component due to rotation and translation of the Galaxy, relative to the
    mean motion of the local group

```

Astrometry

```

slaS2tp( ra, dec, raz, decz, &xi, &eta, &j );
slaDs2tp( ra, dec, raz, decz, &xi, &eta, &j );
    Transform spherical coordinates into tangent plane

slaV2tp( v, v0, &xi, &eta, &j );
slaDv2tp( v, v0, &xi, &eta, &j );
    Transform  $[x, y, z]$  into tangent plane coordinates

slaDtp2s( xi, eta, raz, decz, &ra, &dec );
slaTp2s( xi, eta, raz, decz, &ra, &dec );
    Transform tangent plane coordinates into spherical coordinates

slaDtp2v( xi, eta, v0, v );
slaTp2v( xi, eta, v0, v );
    Transform tangent plane coordinates into  $[x, y, z]$ 

```

```

slaDtps2c( xi, eta, ra, dec, &raz1, &decz1, &raz2, &decz2, &n );
slaTps2c( xi, eta, ra, dec, &raz1, &decz1, &raz2, &decz2, &n );
    Get plate centre from star  $[\alpha, \delta]$  and tangent plane coordinates

slaDtpv2c( xi, eta, v, v01, v02, &n );
slaTpv2c( xi, eta, v, v01, v02, &n );
    Get plate centre from star  $[x, y, z]$  and tangent plane coordinates

slaPcd( disco, &x, &y );
    Apply pincushion/barrel distortion

slaUnpcd( disco, &x, &y );
    Remove pincushion/barrel distortion

slaFitxy( itype, np, xye, xym, coeffs, &j );
    Fit a linear model to relate two sets of  $[x, y]$  coordinates

slaPxy( np, xye, xym, coeffs, xyp, &xrms, &yrrms, &rrms );
    Compute predicted coordinates and residuals

slaInvf( fwds, bkws, &j );
    Invert a linear model

slaXy2xy( x1, y1, coeffs, &x2, &y2 );
    Transform one  $[x, y]$ 

slaDcmpf( coeffs, &xz, &yz, &xs, &ys, &perp, &orient );
    Decompose a linear fit into scales etc.

```

Numerical Methods

```

slaCombn( nsel, ncand, list, &j );
    Next combination ( subset from a specified number of items )

slaPermut( n, istate, iorder, &j );
    Next permutation of a specified number of items

slaSmat( n, a, y, &d, &jf, iw );
slaDmat( n, a, y, &d, &jf, iw );
    Matrix inversion and solution of simultaneous equations

slaSvd( m, n, mp, np, a, w, v, work, &jstat );
    Singular value decomposition of a matrix

slaSvdsol( m, n, mp, np, b, u, w, v, work, x );
    Solution from given vector plus SVD

slaSvdcov( n, np, nc, w, v, work, cvm );
    Covariance matrix from SVD

```