CCLRC / Rutherford Appleton Laboratory
Particle Physics & Astronomy Research Council
Starlink Project
Starlink User Note 67.45

P. T. Wallace

12 October 1999

SLALIB — Positional Astronomy Library

2.4-0

Programmer's Manual

Abstract

SLALIB is a library used by writers of positional-astronomy applications. Most of the 183 routines are concerned with astronomical position and time, but a number have wider trigonometrical, numerical or general applications.

Contents

1 INTRODUCTION

1.1 Purpose

SLALIB[1] is a library of routines intended to make accurate and reliable positional-astronomy applications easier to write. Most SLALIB routines are concerned with astronomical position and time, but a number have wider trigonometrical, numerical or general applications. The applications ASTROM, COCO, RV and TPOINT all make extensive use of the SLALIB routines, as do a number of telescope control systems around the world. The SLALIB versions currently in service are written in Fortran 77 and run on VAX/VMS, several Unix platforms and PC. A generic ANSI C version is also available from the author; it is functionally similar to the Fortran version upon which the present document concentrates.

1.2 Example Application

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

```
      PROGRAM FK4FK5
*
*  Read a B1950 position from I/O unit 5 and reply on I/O unit 6
*  with the J2000 equivalent.  Enter a period to quit.
*
      IMPLICIT NONE
      CHARACTER C*80,S
      INTEGER I,J,IHMSF(4),IDMSF(4)
      DOUBLE PRECISION R4,D4,R5,D5
      LOGICAL BAD

*   Loop until a period is entered
      C = ' '
      DO WHILE (C(:1).NE.'.')

*     Read h m s d ' "
         READ (5,'(A)') C
         IF (C(:1).NE.'.') THEN
            BAD = .TRUE.

*        Decode the RA
            I = 1
            CALL sla_DAFIN(C,I,R4,J)
            IF (J.EQ.0) THEN
               R4 = 15D0*R4

*           Decode the Dec
               CALL sla_DAFIN(C,I,D4,J)
               IF (J.EQ.0) THEN
```

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

```
*              FK4 to FK5
                  CALL sla_FK45Z(R4,D4,1950D0,R5,D5)

*              Format and output the result
                  CALL sla_DR2TF(2,R5,S,IHMSF)
                  CALL sla_DR2AF(1,D5,S,IDMSF)
                  WRITE (6,
     :       '(1X,I2.2,2I3.2,''.'',I2.2,2X,A,I2.2,2I3.2,''.'',I1)')
     :                                                IHMSF,S,IDMSF
                  BAD = .FALSE.
               END IF
            END IF
            IF (BAD) WRITE (6,'(1X,''?'')')
         END IF
      END DO

      END
```

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. See the program COCO (SUN/56) for further ideas.

1.3 Scope

SLALIB contains 183 routines covering the following topics:

- String Decoding, Sexagesimal Conversions
- Angles, Vectors & Rotation Matrices
- Calendars, Timescales
- Precession & Nutation
- Proper Motion
- FK4/FK5/Hipparcos, Elliptic Aberration
- Geocentric Coordinates
- Apparent & Observed Place
- Azimuth & Elevation
- 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 were accurate and easy to use. To this end, the library is:

- Readily available, including source code and documentation.
- Supported and maintained.
- Portable – coded in standard languages and available for multiple computers and operating systems.
- Thoroughly commented, both for maintainability and to assist those wishing to cannibalize the code.
- Stable.
- Trustworthy – some care has gone into testing SLALIB, both by comparison with published data and by checks for internal consistency.
- 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 routines 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.
- The functions aren't orthogonal – there are several cases of different routines 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 omissions – for example there are no routines for calculating physical ephemerides of Solar-System bodies.

- SLALIB is not homogeneous, though important subsets (for example the FK4/FK5 routines) 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 Fortran Version

The Fortran versions of SLALIB use ANSI Fortran 77 with a few commonplace extensions. Just three out of the 183 routines require platform-specific techniques and accordingly are supplied in different forms. SLALIB has been implemented on the following platforms: VAX/VMS, PC (Microsoft Fortran, Linux), DECstation (Ultrix), DEC Alpha (DEC Unix), Sun (SunOS, Solaris), Hewlett Packard (HP-UX), CONVEX, Perkin-Elmer and Fujitsu.

1.6 C Version

An ANSI C version of SLALIB is available from the author but is not part of the Starlink release. The functionality of this (proprietary) C version closely matches that of the Starlink Fortran SLALIB, partly for the convenience of existing users of the Fortran version, some of whom have in the past implemented C "wrappers". The function names cannot be the same as the Fortran versions because of potential linking problems when both forms of the library are present; the C routine which is the equivalent of (for example) `SLA_REFRO` is `slaRefro`. The types of arguments follow the Fortran version, except that integers are `int` rather than `long`. Argument passing is 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.

The Fortran routines `sla_GRESID`, `sla_RANDOM` and `sla_WAIT` have no C counterparts.

Further details of the C version of SLALIB are available from the author. The definitive guide to the calling sequences is the file `slalib.h`.

1.7 Future Versions

The homogeneity and ease of use of SLALIB could perhaps be improved in the future by turning to C++ and object-oriented techniques. For example "celestial position" could be a class and many of the transformations could happen automatically. This requires further study and would almost certainly result in a complete redesign. Similarly, the impact of Fortran 90 has yet to be assessed. Once compilers become widely available, some internal recoding may be worthwhile in order to simplify parts of the code. However, as with C++, a redesign of the application interfaces will be needed if the capabilities of the new language are to be exploited to the full.

1.8 New Functions

In a package like SLALIB it is difficult to know how far to go. Is it enough 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 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 routines 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. Suggestions for additional SLALIB routines should be addressed to the author.

1.9 Acknowledgements

SLALIB is descended from a package of routines written for the AAO 16-bit minicomputers in the mid-1970s. The coming of the VAX allowed a much more comprehensive and thorough package to be designed for Starlink, especially important at a time when the adoption of the IAU 1976 resolutions meant that astronomers would have to cope with a mixture of reference frames, timescales and nomenclature.

Much of the preparatory work on SLALIB was done by Althea Wilkinson of Manchester University. During its development, Andrew Murray, Catherine Hohenkerk, Andrew Sinclair, Bernard Yallop and Brian Emerson of Her Majesty's Nautical Almanac Office were consulted on many occasions; their advice was indispensable. I am especially grateful to Catherine Hohenkerk for supplying preprints of papers, and test data. 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 Maddalena, NRAO. Mark Calabretta, CSIRO Radiophysics, Sydney supplied changes to suit Convex. I am indebted to Derek Jones (RGO) for introducing me to the "universal variables" method of calculating orbits.

The first C version of SLALIB was a hand-coded transcription of the Starlink Fortran version carried out by Steve Eaton (University of Leeds) in the course of MSc work. This was later enhanced by John Straede (AAO) and Martin Shepherd (Caltech). The current C SLALIB is a complete rewrite by the present author and includes a comprehensive validation suite. Additional comments on the C version came from Bob Payne (NRAO) and Jeremy Bailey (AAO).

2 LINKING

On Unix systems (Sun, DEC Alpha *etc.*):

```
% f77 progname.o -L/star/lib `sla_link` -o progname
```

(The above assumes that all Starlink directories have been added to the `LD_LIBRARY_PATH` and `PATH` environment variables as described in SUN/202.)

On VAX/VMS:

```
$ LINK progname,SLALIB_DIR:SLALIB/LIB
```

3 SUBPROGRAM SPECIFICATIONS

SLA_ADDET — Add E-terms of Aberration — SLA_ADDET

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 : `CALL sla_ADDET (RM, DM, EQ, RC, DC)`

GIVEN :

RM,DM	**D**	$[\alpha, \delta]$ without E-terms (radians)
EQ	**D**	Besselian epoch of mean equator and equinox

RETURNED :

RC,DC	**D**	$[\alpha, \delta]$ 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 $[\alpha, \delta]$ should be adjusted using this routine.

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

SLA_AFIN — Sexagesimal character string to angle — SLA_AFIN

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

CALL : `CALL sla_AFIN (STRING, NSTRT, RESLT, JF)`

GIVEN :

STRING	**C*(*)**	string containing deg, arcmin, arcsec fields
NSTRT	**I**	pointer to start of decode (beginning of STRING = 1)

RETURNED :

NSTRT	**I**	advanced past the decoded angle
RESLT	**R**	angle in radians
JF	**I**	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 :

argument	*before*	*after*
STRING	`'-57␣17␣44.806␣␣12␣34␣56.7'`	unchanged
NSTRT	1	16 (*i.e.* pointing to 12...)
RESLT	-	-1.00000
JF	-	0

A further call to sla_AFIN, without adjustment of NSTRT, will decode the second angle, $12^\circ\ 34'\ 56''\!.7$.

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 sla_DFLTIN routine 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 pointer NSTRT is left in the correct state for further decoding with the present routine or with sla_DFLTIN *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:

for STRING	*to obtain*	*multiply RESLT by*
$\circ$ $\prime$ $\prime\prime$	radians	1.0
$\circ$ $\prime$ $\prime\prime$	turns	$1/2\pi = 0.1591549430918953358$
h m s	radians	15.0
h m s	days	$15/2\pi = 2.3873241463784300365$

SLA_AIRMAS — Air Mass — SLA_AIRMAS

ACTION : Air mass at given zenith distance (double precision).

CALL : `D = sla_AIRMAS (ZD)`

GIVEN :

ZD	**D**	observed zenith distance (radians)

RETURNED :

sla_AIRMAS	**D**	air mass (1 at zenith)

NOTES :

1. The *observed* zenith distance referred to above means "as affected by refraction".
2. The routine 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.

SLA_ALTAZ Velocities *etc.* for Altazimuth Mount SLA_ALTAZ

ACTION : Positions, velocities and accelerations for an altazimuth telescope mount tracking a star (double precision).

CALL :
```
CALL sla_ALTAZ (HA, DEC, PHI,
                AZ, AZD, AZDD, EL, ELD, ELDD, PA, PAD, PADD)
```

GIVEN :

HA	**D**	hour angle
DEC	**D**	declination
PHI	**D**	observatory latitude

RETURNED :

AZ	**D**	azimuth
AZD	**D**	azimuth velocity
AZDD	**D**	azimuth acceleration
EL	**D**	elevation
ELD	**D**	elevation velocity
ELDD	**D**	elevation acceleration
PA	**D**	parallactic angle
PAD	**D**	parallactic angle velocity
PADD	**D**	parallactic angle acceleration

NOTES :

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

$$\begin{array}{rlcl} \text{angles} & \times 360/2\pi & \rightarrow & \text{degrees} \\ \text{velocities} & \times (2\pi/86400) \times (360/2\pi) & \rightarrow & \text{degree/sec} \\ \text{accelerations} & \times (2\pi/86400)^2 \times (360/2\pi) & \rightarrow & \text{degree/sec/sec} \end{array}$$

   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 $\pm\pi/2$. Position angle is +ve 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 routine is to give the general form of the quantities. The details of a real telescope could profoundly change the results, especially close to the zenith.

4. No range checking of arguments is carried out.

5. In applications which involve many such calculations, rather than calling the present routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude, and (for tracking a star) sine and cosine of declination.

SLA_AMP — Apparent to Mean — SLA_AMP

ACTION : Convert star $[\alpha, \delta]$ from geocentric apparent to mean place (post IAU 1976).

CALL : `CALL sla_AMP (RA, DA, DATE, EQ, RM, DM)`

GIVEN :

RA,DA	**D**	apparent $[\alpha, \delta]$ (radians)
DATE	**D**	TDB for apparent place (JD−2400000.5)
EQ	**D**	equinox: Julian epoch of mean place

RETURNED :

RM,DM	**D**	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. The accuracy is limited by the routine sla_EVP, called by sla_MAPPA, which computes the Earth position and velocity using the methods of Stumpff. The maximum error is about 0.3 milliarcsecond.

3. Iterative techniques are used for the aberration and light deflection corrections so that the routines sla_AMP (or sla_AMPQK) and sla_MAP (or sla_MAPQK) are accurate inverses; even at the edge of the Sun's disc the discrepancy is only about 1 nanoarcsecond.

4. Where multiple apparent places are to be converted to mean places, for a fixed date and equinox, it is more efficient to use the sla_MAPPA routine to compute the required parameters once, followed by one call to sla_AMPQK per star.

REFERENCES :

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

SLA_AMPQK — Quick Apparent to Mean — SLA_AMPQK

ACTION : Convert star $[\alpha, \delta]$ from geocentric apparent to mean place (post IAU 1976). Use of this routine 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 sla_MAPPA routine.

CALL : `CALL sla_AMPQK (RA, DA, AMPRMS, RM, DM)`

GIVEN :

RA,DA	**D**	apparent $[\alpha, \delta]$ (radians)
AMPRMS	**D(21)**	star-independent mean-to-apparent parameters:
(1)		time interval for proper motion (Julian years)
(2-4)		barycentric position of the Earth (AU)
(5-7)		heliocentric direction of the Earth (unit vector)
(8)		(gravitational radius of Sun)$\times$2/(Sun-Earth distance)
(9-11)		$\mathbf{v}$: barycentric Earth velocity in units of c
(12)		$\sqrt{1-\lvert\mathbf{v}\rvert^2}$
(13-21)		precession/nutation 3×3 matrix

RETURNED :

RM,DM	**D**	mean $[\alpha, \delta]$ (radians)

NOTES :

1. The accuracy is limited by the routine sla_EVP, called by sla_MAPPA, which computes the Earth position and velocity using the methods of Stumpff. The maximum error is about 0.3 milliarcsecond.
2. Iterative techniques are used for the aberration and light deflection corrections so that the routines sla_AMP (or sla_AMPQK) and sla_MAP (or sla_MAPQK) 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.

SLA_AOP — Apparent to Observed — SLA_AOP

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

CALL :
```
CALL sla_AOP (RAP, DAP, DATE, DUT, ELONGM, PHIM, HM, XP, YP,
              TDK, PMB, RH, WL, TLR, AOB, ZOB, HOB, DOB, ROB)
```

GIVEN :

RAP,DAP	**D**	geocentric apparent $[\,\alpha,\delta\,]$ (radians)
DATE	**D**	UTC date/time (Modified Julian Date, JD−2400000.5)
DUT	**D**	ΔUT: UT1−UTC (UTC seconds)
ELONGM	**D**	observer's mean longitude (radians, east +ve)
PHIM	**D**	observer's mean geodetic latitude (radians)
HM	**D**	observer's height above sea level (metres)
XP,YP	**D**	polar motion $[\,x,y\,]$ coordinates (radians)
TDK	**D**	local ambient temperature (degrees K; std=273.155D0)
PMB	**D**	local atmospheric pressure (mB; std=1013.25D0)
RH	**D**	local relative humidity (in the range 0D0 – 1D0)
WL	**D**	effective wavelength (μm, *e.g.* 0.55D0)
TLR	**D**	tropospheric lapse rate (degrees K per metre, *e.g.* 0.0065D0)

RETURNED :

AOB	**D**	observed azimuth (radians: N=0, E=90°)
ZOB	**D**	observed zenith distance (radians)
HOB	**D**	observed Hour Angle (radians)
DOB	**D**	observed δ (radians)
ROB	**D**	observed α (radians)

NOTES :

1. This routine 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 be better than

1 arcminute; useful results are available for a further 3°, beyond which the sla_REFRO routine returns a fixed value of the refraction. The complementary routines sla_AOP (or sla_AOPQK) and sla_OAP (or sla_OAPQK) 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, precession, nutation, annual aberration, and the Sun's gravitational lens effect. For star positions in the FK5 system (*i.e.* J2000), these effects can be applied by means of the sla_MAP *etc.* routines. Starting from other mean place systems, additional transformations will be needed; for example, FK4 (*i.e.* B1950) mean places would first have to be converted to FK5, which can be done with the sla_FK425 *etc.* routines.
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 routine would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.
7. This routine takes time to execute, due mainly to the rigorous integration used to evaluate the refraction. For processing multiple stars for one location and time, call sla_AOPPA once followed by one call per star to sla_AOPQK. Where a range of times within a limited period of a few hours is involved, and the highest precision is not required, call sla_AOPPA once, followed by a call to sla_AOPPAT each time the time changes, followed by one call per star to sla_AOPQK.
8. The DATE 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 $\pm 0\overset{s}{.}9$.
10. IMPORTANT – TAKE CARE WITH THE LONGITUDE SIGN CONVENTION. The longitude required by the present routine is **east-positive**, in accordance with

geographical convention (and right-handed). In particular, note that the longitudes returned by the sla_OBS routine are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present routine.

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=0D0. 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 routine sla_OBS. If P, the pressure in mB, is available, an adequate estimate of HM can be obtained from the following expression:

    ```
    HM=-29.3D0*TSL*LOG(P/1013.25D0)
    ```

    where TSL is the approximate sea-level air temperature in degrees 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=1013.25D0*EXP(-HM/(29.3D0*TSL))
    ```

    Note, however, that the refraction is proportional to the pressure and that an accurate P value is important for precise work.
13. The azimuths *etc.* used by the present routine are with respect to the celestial pole. Corrections to the terrestrial pole can be computed using sla_POLMO.

SLA_AOPPA — Appt-to-Obs Parameters — SLA_AOPPA

ACTION : Pre-compute the set of apparent to observed place parameters required by the "quick" routines sla_AOPQK and sla_OAPQK.

CALL :
```
CALL sla_AOPPA (DATE, DUT, ELONGM, PHIM, HM, XP, YP,
                TDK, PMB, RH, WL, TLR, AOPRMS)
```

GIVEN :

DATE	**D**	UTC date/time (Modified Julian Date, JD−2400000.5)
DUT	**D**	ΔUT: UT1−UTC (UTC seconds)
ELONGM	**D**	observer's mean longitude (radians, east +ve)
PHIM	**D**	observer's mean geodetic latitude (radians)
HM	**D**	observer's height above sea level (metres)
XP,YP	**D**	polar motion $[x, y]$ coordinates (radians)
TDK	**D**	local ambient temperature (degrees K; std=273.155D0)
PMB	**D**	local atmospheric pressure (mB; std=1013.25D0)
RH	**D**	local relative humidity (in the range 0D0 – 1D0)
WL	**D**	effective wavelength (μm, *e.g.* 0.55D0)
TLR	**D**	tropospheric lapse rate (degrees K per metre, *e.g.* 0.0065D0)

RETURNED :

AOPRMS	**D(14)**	star-independent apparent-to-observed parameters:
(1)		geodetic latitude (radians)
(2,3)		sine and cosine of geodetic latitude
(4)		magnitude of diurnal aberration vector
(5)		height (HM)
(6)		ambient temperature (TDK)
(7)		pressure (PMB)
(8)		relative humidity (RH)
(9)		wavelength (WL)
(10)		lapse rate (TLR)
(11,12)		refraction constants A and B (radians)
(13)		longitude + eqn of equinoxes + "sidereal ΔUT" (radians)
(14)		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 DATE 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 $\pm 0^{\mathrm{s}}\!.9$. The "sidereal ΔUT" which forms part of AOPRMS(13) 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 routine is **east-positive**, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the sla_OBS routine are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present routine.
5. 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=0D0. 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 routine sla_OBS. If P, the pressure in mB, is available, an adequate estimate of HM can be obtained from the following expression:

```
HM=-29.3D0*TSL*LOG(P/1013.25D0)
```

where TSL is the approximate sea-level air temperature in degrees 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=1013.25D0*EXP(-HM/(29.3D0*TSL))
```

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

SLA_AOPPAT Update Appt-to-Obs Parameters SLA_AOPPAT

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

CALL : `CALL sla_AOPPAT (DATE, AOPRMS)`

GIVEN :

DATE	**D**	UTC date/time (Modified Julian Date, JD−2400000.5)
AOPRMS	**D(14)**	star-independent apparent-to-observed parameters:
(1-12)		not required
(13)		longitude + eqn of equinoxes + "sidereal ΔUT" (radians)
(14)		not required

RETURNED :

AOPRMS	**D(14)**	star-independent apparent-to-observed parameters:
(1-13)		not changed
(14)		local apparent sidereal time (radians)

NOTE : For more information, see sla_AOPPA.

SLA_AOPQK Quick Appt-to-Observed SLA_AOPQK

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

CALL : `CALL sla_AOPQK (RAP, DAP, AOPRMS, AOB, ZOB, HOB, DOB, ROB)`

GIVEN :

RAP,DAP	**D**	geocentric apparent [α, δ] (radians)
AOPRMS	**D(14)**	star-independent apparent-to-observed parameters:
(1)		geodetic latitude (radians)
(2,3)		sine and cosine of geodetic latitude
(4)		magnitude of diurnal aberration vector
(5)		height (metres)
(6)		ambient temperature (degrees K)
(7)		pressure (mB)
(8)		relative humidity (0 – 1)
(9)		wavelength (μm)
(10)		lapse rate (degrees K per metre)
(11,12)		refraction constants A and B (radians)
(13)		longitude + eqn of equinoxes + "sidereal ΔUT" (radians)
(14)		local apparent sidereal time (radians)

RETURNED :

AOB	**D**	observed azimuth (radians: N=0, E=90°)
ZOB	**D**	observed zenith distance (radians)
HOB	**D**	observed Hour Angle (radians)
DOB	**D**	observed Declination (radians)
ROB	**D**	observed Right Ascension (radians)

NOTES :

1. This routine 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 be better than 1 arcminute; useful results are available for a further 3°, beyond which the sla_REFRO routine returns a fixed value of the refraction. The complementary routines sla_AOP (or sla_AOPQK) and sla_OAP (or sla_OAPQK) 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* [α, δ] means the geocentric apparent right ascension and declination, which is obtained from a catalogue mean place by allowing for space motion, parallax, precession, nutation, annual aberration, and the Sun's gravitational lens effect. For star positions in the FK5 system (*i.e.* J2000), these effects can be applied by means of the sla_MAP *etc.* routines. Starting from other mean place systems, additional transformations will be needed; for example, FK4 (*i.e.* B1950) mean places would first have to be converted to FK5, which can be done with the sla_FK425 *etc.* routines.

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 [α, δ] by allowing for Earth orientation and diurnal aberration, rotating from equator to horizon coordinates, and then adjusting for refraction. The [h, δ] 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 routine would have to be adjusted for the tilt of the azimuth or polar axis of the mounting (with appropriate corrections for mount flexures), for non-perpendicularity between the mounting axes, for the position of the rotator axis and the pointing axis relative to it, for tube flexure, for gear and encoder errors, and finally for encoder zero points. Some telescopes would, of course, exhibit other properties which would need to be accounted for at the appropriate point in the sequence.

7. The star-independent apparent-to-observed-place parameters in AOPRMS may be computed by means of the sla_AOPPA routine. If nothing has changed significantly except the time, the sla_AOPPAT routine may be used to perform the requisite partial recomputation of AOPRMS.

8. The "sidereal ΔUT" which forms part of AOPRMS(13) is UT1−UTC converted from solar to sidereal seconds and expressed in radians.

9. 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 routine will, depending on the application, produce a rough position which may be enough to establish whether the full, slow calculation (using the real AOPRMS array) is worthwhile. For example, there would be no need for the full calculation if the preliminary call had already established that the source was well below the elevation limits for a particular telescope.

10. The azimuths *etc.* used by the present routine are with respect to the celestial pole. Corrections to the terrestrial pole can be computed using sla_POLMO.

SLA_ATMDSP — Atmospheric Dispersion — SLA_ATMDSP

ACTION : Apply atmospheric-dispersion adjustments to refraction coefficients.

CALL : `CALL sla_ATMDSP (TDK, PMB, RH, WL1, A1, B1, WL2, A2, B2)`

GIVEN :

TDK	**D**	ambient temperature at the observer (degrees K)
PMB	**D**	pressure at the observer (mB)
RH	**D**	relative humidity at the observer (range 0 – 1)
WL1	**D**	base wavelength (μm)
A1	**D**	refraction coefficient A for wavelength WL1 (radians)
B1	**D**	refraction coefficient B for wavelength WL1 (radians)
WL2	**D**	wavelength for which adjusted A,B required (μm)

RETURNED :

A2	**D**	refraction coefficient A for wavelength WL2 (radians)
B2	**D**	refraction coefficient B for wavelength WL2 (radians)

NOTES :

1. To use this routine, first call sla_REFCO specifying WL1 as the wavelength. This yields refraction coefficients A1, B1, correct for that wavelength. Subsequently, calls to sla_ATMDSP 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 μm 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 μm.
3. The accuracy, for this set of conditions:

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

   is about 2.5 mas RMS between 0.3 and 1.0 μm, and stays within 4 mas for the whole range longward of 0.3 μm (compared with a total dispersion from 0.3 to 20 μm 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 μm, the radio case is assumed and the returned refraction coefficients are the same as the given ones.
5. The algorithm consists of calculation of the refractivity of the air at the observer for the two wavelengths, using the methods of the sla_REFRO routine, 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).

SLA_AV2M — Rotation Matrix from Axial Vector — SLA_AV2M

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

CALL : `CALL sla_AV2M (AXVEC, RMAT)`

GIVEN :

AXVEC	**R(3)**	axial vector (radians)

RETURNED :

RMAT	**R(3,3)**	rotation matrix

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis. The axis is called the *Euler axis*, and the angle through which the reference frame rotates is called the Euler angle. The axial vector supplied to this routine has the same direction as the Euler axis, and its magnitude is the Euler angle 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.

SLA_BEAR — Direction Between Points on a Sphere — SLA_BEAR

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

CALL : `R = sla_BEAR (A1, B1, A2, B2)`

GIVEN :

A1,B1	**R**	spherical coordinates of one point
A2,B2	**R**	spherical coordinates of the other point

RETURNED :

sla_BEAR	**R**	bearing from first point to second

NOTES :

1. The spherical coordinates are $[\alpha, \delta]$, $[\lambda, \phi]$ *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 routine sla_PAV performs an equivalent function except that the points are specified in the form of Cartesian unit vectors.

SLA_CAF2R Deg,Arcmin,Arcsec to Radians SLA_CAF2R

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

CALL : `CALL sla_CAF2R (IDEG, IAMIN, ASEC, RAD, J)`

GIVEN :

IDEG	**I**	degrees
IAMIN	**I**	arcminutes
ASEC	**R**	arcseconds

RETURNED :

RAD	**R**	angle in radians
J	**I**	status:
		1 = IDEG outside range 0−359
		2 = IAMIN outside range 0−59
		3 = ASEC 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 routine.

SLA_CALDJ Calendar Date to MJD SLA_CALDJ

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

CALL : `CALL sla_CALDJ (IY, IM, ID, DJM, J)`

GIVEN :

IY,IM,ID	**I**	year, month, day in Gregorian calendar

RETURNED :

DJM	**D**	modified Julian Date (JD−2400000.5) for $0^{\rm h}$
J	**I**	status: 0 = OK 1 = bad year (MJD not computed) 2 = bad month (MJD not computed) 3 = bad day (MJD computed)

NOTES :

1. This routine 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 1-100AD use the routine sla_CLDJ instead.
2. For year nBC 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.

SLA_CALYD — Calendar to Year, Day — SLA_CALYD

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 : `CALL sla_CALYD (IY, IM, ID, NY, ND, J)`

GIVEN :

IY,IM,ID	**I**	year, month, day in Gregorian calendar: year may optionally omit the century

RETURNED :

NY	**I**	year (re-aligned Julian calendar)
ND	**I**	day in year (1 = January 1st)
J	**I**	status: 0 = OK 1 = bad year (before −4711) 2 = bad month 3 = bad day

NOTES :

1. This routine 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 sla_CLYD for years before 100AD.
2. The purpose of sla_CALDJ is to support sla_EARTH, sla_MOON and sla_ECOR.
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.

SLA_CC2S Cartesian to Spherical SLA_CC2S

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

CALL : `CALL sla_CC2S (V, A, B)`

GIVEN :

V	**R(3)**	$[x, y, z]$ vector

RETURNED :

A,B	**R**	spherical coordinates in radians

NOTES :

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

SLA_CC62S Cartesian 6-Vector to Spherical SLA_CC62S

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

CALL : `CALL sla_CC62S (V, A, B, R, AD, BD, RD)`

GIVEN :

V	**R(6)**	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$

RETURNED :

A	**R**	longitude (radians) – for example α
B	**R**	latitude (radians) – for example δ
R	**R**	radial coordinate
AD	**R**	longitude derivative (radians per unit time)
BD	**R**	latitude derivative (radians per unit time)
RD	**R**	radial derivative

SLA_CD2TF — Days to Hour,Min,Sec — SLA_CD2TF

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

CALL : `CALL sla_CD2TF (NDP, DAYS, SIGN, IHMSF)`

GIVEN :

NDP	**I**	number of decimal places of seconds
DAYS	**R**	interval in days

RETURNED :

SIGN	**C**	'+' or '−'
IHMSF	**I(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 REAL floating-point numbers on the target machine, and the risk of overflowing IHMSF(4). For example, on a VAX computer, 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 the 32-bit integer IHMSF(4).
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(1)=24 and setting IHMSF(1-4) to zero.

SLA_CLDJ — Calendar to MJD — SLA_CLDJ

ACTION : Gregorian Calendar to Modified Julian Date.

CALL : `CALL sla_CLDJ (IY, IM, ID, DJM, J)`

GIVEN :

IY,IM,ID	**I**	year, month, day in Gregorian calendar

RETURNED :

DJM	**D**	modified Julian Date (JD−2400000.5) for $0^{\rm h}$
J	**I**	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.* 4700BC) or later. For year nBC use IY = $-(n-1)$.
3. An alternative to the present routine is sla_CALDJ, which accepts a year with the century missing.

REFERENCE : The algorithm is derived from that of Hatcher, *Q. Jl. R. astr. Soc.* (1984) **25**, 53-55.

SLA_CLYD — Calendar to Year, Day — SLA_CLYD

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

CALL : `CALL sla_CLYD (IY, IM, ID, NY, ND, J)`

GIVEN :

IY,IM,ID	**I**	year, month, day in Gregorian calendar

RETURNED :

NY	**I**	year (re-aligned Julian calendar)
ND	**I**	day in year (1 = January 1st)
J	**I**	status: 0 = OK 1 = bad year (before −4711) 2 = bad month 3 = bad day

NOTES :

1. The purpose of sla_CLYD is to support sla_EARTH, sla_MOON and sla_ECOR.
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.

SLA_COMBN — Next Combination — SLA_COMBN

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

CALL : `CALL sla_COMBN (NSEL, NCAND, LIST, J)`

GIVEN :

NSEL	**I**	number of items (subset size)
NCAND	**I**	number of candidates (set size)

GIVEN and RETURNED :

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

RETURNED :

J	**I**	status: −1 = illegal NSEL or NCAND 0 = OK +1 = no more combinations available

NOTES :

1. NSEL and NCAND must both be at least 1, and NSEL must be less than or equal to NCAND.
2. This routine returns, in the LIST array, a subset of NSEL integers chosen from the range 1 to NCAND inclusive, in ascending order. Before calling the routine 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:

   LIST(1)=1, LIST(2)=2, . . . , LIST(NSEL)=NSEL

This is also the combination returned for the "finished" (J=1) case. The final permutation to be generated is:

LIST(1)=NCAND, LIST(2)=NCAND−1, . . . ,
LIST(NSEL)=NCAND−NSEL+1

4. If the "finished" (J=1) status is ignored, the routine continues to deliver combinations, the pattern repeating every NCAND!/(NSEL!(NCAND−NSEL)!) calls.
5. The algorithm is by R. F. Warren-Smith (private communication).

SLA_CR2AF — Radians to Deg,Arcmin,Arcsec — SLA_CR2AF

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

CALL : `CALL sla_CR2AF (NDP, ANGLE, SIGN, IDMSF)`

GIVEN :

NDP	**I**	number of decimal places of arcseconds
ANGLE	**R**	angle in radians

RETURNED :

SIGN	**C**	'+' or '−'
IDMSF	**I(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 REAL floating-point numbers on the target machine, and the risk of overflowing IDMSF(4). For example, on a VAX computer, 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 the 32-bit integer IHMSF(4).
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(1)=360 and setting IDMSF(1-4) to zero.

SLA_CR2TF — Radians to Hour,Min,Sec — SLA_CR2TF

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

CALL : `CALL sla_CR2TF (NDP, ANGLE, SIGN, IHMSF)`

GIVEN :

NDP	**I**	number of decimal places of seconds
ANGLE	**R**	angle in radians

RETURNED :

SIGN	**C**	'+' or '−'
IHMSF	**I(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 REAL floating-point numbers on the target machine, and the risk of overflowing IHMSF(4). For example, on a VAX computer, 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 the 32-bit integer IHMSF(4).
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(1)=24 and setting IHMSF(1-4) to zero.

SLA_CS2C — Spherical to Cartesian — SLA_CS2C

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

CALL : `CALL sla_CS2C (A, B, V)`

GIVEN :

A,B	**R**	spherical coordinates in radians: $[\alpha, \delta]$ *etc.*

RETURNED :

V	**R(3)**	$[x, y, z]$ unit vector

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

SLA_CS2C6 — Spherical Pos/Vel to Cartesian — SLA_CS2C6

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

CALL : `CALL sla_CS2C6 (A, B, R, AD, BD, RD, V)`

GIVEN :

A	**R**	longitude (radians) – for example α
B	**R**	latitude (radians) – for example δ
R	**R**	radial coordinate
AD	**R**	longitude derivative (radians per unit time)
BD	**R**	latitude derivative (radians per unit time)
RD	**R**	radial derivative

RETURNED :

V	**R(6)**	$[\,x, y, z, \dot{x}, \dot{y}, \dot{z}\,]$

SLA_CTF2D — Hour,Min,Sec to Days — SLA_CTF2D

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

CALL : `CALL sla_CTF2D (IHOUR, IMIN, SEC, DAYS, J)`

GIVEN :

IHOUR	**I**	hours
IMIN	**I**	minutes
SEC	**R**	seconds

RETURNED :

DAYS	**R**	interval in days
J	**I**	status: 0 = OK 1 = IHOUR outside range 0-23 2 = IMIN outside range 0-59 3 = SEC outside range 0-59.999$\cdots$

NOTES :

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

SLA_CTF2R — Hour,Min,Sec to Radians — SLA_CTF2R

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

CALL : `CALL sla_CTF2R (IHOUR, IMIN, SEC, RAD, J)`

GIVEN :

IHOUR	**I**	hours
IMIN	**I**	minutes
SEC	**R**	seconds

RETURNED :

RAD	**R**	angle in radians
J	**I**	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 routine.

SLA_DAF2R Deg,Arcmin,Arcsec to Radians SLA_DAF2R

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

CALL : `CALL sla_DAF2R (IDEG, IAMIN, ASEC, RAD, J)`

GIVEN :

IDEG	**I**	degrees
IAMIN	**I**	arcminutes
ASEC	**D**	arcseconds

RETURNED :

RAD	**D**	angle in radians
J	**I**	status: 1 = IDEG outside range 0−359 2 = IAMIN outside range 0−59 3 = ASEC 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 routine.

SLA_DAFIN   Sexagesimal character string to angle   SLA_DAFIN

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

CALL : `CALL sla_DAFIN (STRING, NSTRT, DRESLT, JF)`

GIVEN :

STRING	**C*(*)**	string containing deg, arcmin, arcsec fields
NSTRT	**I**	pointer to start of decode (beginning of STRING = 1)

RETURNED :

NSTRT	**I**	advanced past the decoded angle
DRESLT	**D**	angle in radians
JF	**I**	status: 0 = OK +1 = default, DRESLT unchanged (note 2) −1 = bad degrees (note 3) −2 = bad arcminutes (note 3) −3 = bad arcseconds (note 3)

EXAMPLE :

argument	*before*	*after*
STRING	`'-57␣17␣44.806␣␣12␣34␣56.7'`	unchanged
NSTRT	1	16 (*i.e.* pointing to 12...)
RESLT	-	−1.00000D0
JF	-	0

A further call to sla_DAFIN, without adjustment of NSTRT, will decode the second angle, $12^\circ\ 34'\ 56''.7$.

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 sla_DFLTIN routine 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 DRESLT is unchanged. In all other cases DRESLT 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 pointer NSTRT is left in the correct state for further decoding with the present routine or with sla_DFLTIN *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:

for STRING	*to obtain*	*multiply DRESLT by*
$\circ\ \prime\ \prime\prime$	radians	1.0D0
$\circ\ \prime\ \prime\prime$	turns	$1/2\pi = 0.1591549430918953358$D0
h m s	radians	15.0D0
h m s	days	$15/2\pi = 2.3873241463784300365$D0

SLA_DAT — TAI−UTC — SLA_DAT

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

CALL : `D = sla_DAT (UTC)`

GIVEN :

UTC	**D**	UTC date as a modified JD (JD−2400000.5)

RETURNED :

sla_DAT	**D**	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. 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 $23^{\rm h}\,59^{\rm m}\,59^{\rm s}$.
2. UTC began at 1960 January 1. To specify a UTC prior to this date would be meaningless; in such cases the parameters for the year 1960 are used by default.
3. This routine 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.

SLA_DAV2M Rotation Matrix from Axial Vector SLA_DAV2M

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

CALL : `CALL sla_DAV2M (AXVEC, RMAT)`

GIVEN :

AXVEC	**D(3)**	axial vector (radians)

RETURNED :

RMAT	**D(3,3)**	rotation matrix

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis. The axis is called the *Euler axis*, and the angle through which the reference frame rotates is called the *Euler angle*. The axial vector supplied to this routine has the same direction as the Euler axis, and its magnitude is the Euler angle 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.

SLA_DBEAR Direction Between Points on a Sphere SLA_DBEAR

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

CALL : `D = sla_DBEAR (A1, B1, A2, B2)`

GIVEN :

A1,B1	**D**	spherical coordinates of one point
A2,B2	**D**	spherical coordinates of the other point

RETURNED :

sla_DBEAR	**D**	bearing from first point to second

NOTES :

1. The spherical coordinates are $[\alpha, \delta]$, $[\lambda, \phi]$ *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 routine sla_DPAV performs an equivalent function except that the points are specified in the form of Cartesian unit vectors.

SLA_DBJIN Decode String to B/J Epoch (DP) SLA_DBJIN

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

CALL : `CALL sla_DBJIN (STRING, NSTRT, DRESLT, J1, J2)`

GIVEN :

STRING	**C**	string containing field to be decoded
NSTRT	**I**	pointer to first character of field in string

RETURNED :

NSTRT	**I**	incremented past the decoded field
DRESLT	**D**	result
J1	**I**	DFLTIN status: $-1 = -$OK $0 = +$OK 1 = null field 2 = error
J2	**I**	syntax flag: 0 = normal DFLTIN 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 sla_DFLTIN, the following two extensions are recognized by sla_DBJIN:
   (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 sla_DFLTIN.

SLA_DC62S Cartesian 6-Vector to Spherical SLA_DC62S

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

CALL : `CALL sla_DC62S (V, A, B, R, AD, BD, RD)`

GIVEN :

V	**D(6)**	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$

RETURNED :

A	**D**	longitude (radians)
B	**D**	latitude (radians)
R	**D**	radial coordinate
AD	**D**	longitude derivative (radians per unit time)
BD	**D**	latitude derivative (radians per unit time)
RD	**D**	radial derivative

SLA_DCC2S Cartesian to Spherical SLA_DCC2S

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

CALL : `CALL sla_DCC2S (V, A, B)`

GIVEN :

V	**D(3)**	$[x, y, z]$ vector

RETURNED :

A,B	**D**	spherical coordinates in radians

NOTES :

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

SLA_DCMPF — Interpret Linear Fit — SLA_DCMPF

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

CALL : `CALL sla_DCMPF (COEFFS,XZ,YZ,XS,YS,PERP,ORIENT)`

GIVEN :

COEFFS	**D(6)**	transformation coefficients (see note)

RETURNED :

XZ	**D**	x zero point
YZ	**D**	y zero point
XS	**D**	x scale
YS	**D**	y scale
PERP	**D**	nonperpendicularity (radians)
ORIENT	**D**	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:

   $$x_2 = a + bx_1 + cy_1$$
   $$y_2 = d + ex_1 + fy_1$$

   The sla_DCMPF routine decomposes this transformation into four steps:

   (a) Zero points:

   $$x' = x_1 + \mathrm{XZ}$$
   $$y' = y_1 + \mathrm{YZ}$$

   (b) Scales:

   $$x'' = x'\mathrm{XS}$$
   $$y'' = y'\mathrm{YS}$$

   (c) Nonperpendicularity:

$$x''' = +x'' \cos \mathrm{PERP}/2 + y'' \sin \mathrm{PERP}/2$$
$$y''' = +x'' \sin \mathrm{PERP}/2 + y'' \cos \mathrm{PERP}/2$$

(d) Orientation:

$$x_2 = +x''' \cos \mathrm{ORIENT} + y''' \sin \mathrm{ORIENT}$$
$$y_2 = -x''' \sin \mathrm{ORIENT} + y''' \cos \mathrm{ORIENT}$$

2. See also sla_FITXY, sla_PXY, sla_INVF, sla_XY2XY.

SLA_DCS2C — Spherical to Cartesian — SLA_DCS2C

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

CALL : `CALL sla_DCS2C (A, B, V)`

GIVEN :

A,B	**D**	spherical coordinates in radians: $[\alpha, \delta]$ *etc.*

RETURNED :

V	**D(3)**	$[x, y, z]$ unit vector

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

SLA_DD2TF — Days to Hour,Min,Sec — SLA_DD2TF

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

CALL : `CALL sla_DD2TF (NDP, DAYS, SIGN, IHMSF)`

GIVEN :

NDP	**I**	number of decimal places of seconds
DAYS	**D**	interval in days

RETURNED :

SIGN	**C**	'+' or '−'
IHMSF	**I(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 PRECISION floating-point numbers on the target machine, and the risk of overflowing IHMSF(4). For example, on a VAX computer, for DAYS up to 1D0, the available floating-point precision corresponds roughly to NDP=12. However, the practical limit is NDP=9, set by the capacity of the 32-bit integer IHMSF(4).
3. The absolute value of DAYS may exceed 1D0. In cases where it does not, it is up to the caller to test for and handle the case where DAYS is very nearly 1D0 and rounds up to 24 hours, by testing for IHMSF(1)=24 and setting IHMSF(1-4) to zero.

SLA_DE2H — h, δ to Az,El — SLA_DE2H

ACTION : Equatorial to horizon coordinates (double precision).

CALL : `CALL sla_DE2H (HA, DEC, PHI, AZ, EL)`

GIVEN :

HA	**D**	hour angle (radians)
DEC	**D**	declination (radians)
PHI	**D**	latitude (radians)

RETURNED :

AZ	**D**	azimuth (radians)
EL	**D**	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 routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude, and (for tracking a star) sine and cosine of declination.

SLA_DEULER Euler Angles to Rotation Matrix SLA_DEULER

ACTION : Form a rotation matrix from the Euler angles – three successive rotations about specified Cartesian axes (double precision).

CALL : `CALL sla_DEULER (ORDER, PHI, THETA, PSI, RMAT)`

GIVEN :

ORDER	**C**	specifies about which axes the rotations occur
PHI	**D**	1st rotation (radians)
THETA	**D**	2nd rotation (radians)
PSI	**D**	3rd rotation (radians)

RETURNED :

RMAT	**D(3,3)**	rotation matrix

NOTES :

1. A rotation is positive when the reference frame rotates anticlockwise as seen looking towards the origin from the positive region of the specified axis.
2. 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.
3. The axis names can be any of the following, in any order or combination: X, Y, Z, uppercase or lowercase, 1, 2, 3. Normal axis labelling/numbering conventions apply; the xyz ($\equiv$ 123) triad is right-handed. Thus, the 'ZXZ' example given above could be written 'zxz' or '313' (or even 'ZxZ' or '3xZ'). ORDER is terminated by length or by the first unrecognized character. Fewer than three rotations are acceptable, in which case the later angle arguments are ignored. Zero rotations produces a unit RMAT.

SLA_DFLTIN Decode a Double Precision Number SLA_DFLTIN

ACTION : Convert free-format input into double precision floating point.

CALL : `CALL sla_DFLTIN (STRING, NSTRT, DRESLT, JFLAG)`

GIVEN :

STRING	**C**	string containing number to be decoded
NSTRT	**I**	pointer to where decoding is to commence
DRESLT	**D**	current value of result

RETURNED :

NSTRT	**I**	advanced to next number
DRESLT	**D**	result
JFLAG	**I**	status: -1 = $-$OK, 0 = +OK, 1 = null result, 2 = error

NOTES :

1. The reason sla_DFLTIN 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. *n.b.* The test for TAB is ASCII-specific.
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,.,D,E or space may be used to signal the end of the number and terminate decoding.
   - Comma is recognized by sla_DFLTIN as a special case; it is skipped, leaving the pointer 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,.,D or E, or consists entirely of spaces. When this condition is detected, JFLAG is set to 1 and DRESLT is left untouched.
12. NSTRT = 1 for the first character in the string.
13. On return from sla_DFLTIN, 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 sla_DFLTIN, 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, sla_DFLTIN does not detect that '1D999' is unacceptable (on a computer 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 sla_FLOTIN and sla_INTIN.

SLA_DH2E — Az,El to h, δ — SLA_DH2E

ACTION : Horizon to equatorial coordinates (double precision).

CALL : `CALL sla_DH2E (AZ, EL, PHI, HA, DEC)`

GIVEN :

AZ	**D**	azimuth (radians)
EL	**D**	elevation (radians)
PHI	**D**	latitude (radians)

RETURNED :

HA	**D**	hour angle (radians)
DEC	**D**	declination (radians)

NOTES :

1. The sign convention for azimuth is north zero, east $+\pi/2$.
2. HA 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 sla_POLMO).
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 routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude.

SLA_DIMXV Apply 3D Reverse Rotation SLA_DIMXV

ACTION : Multiply a 3-vector by the inverse of a rotation matrix (double precision).

CALL : `CALL sla_DIMXV (DM, VA, VB)`

GIVEN :

DM	**D(3,3)**	rotation matrix
VA	**D(3)**	vector to be rotated

RETURNED :

VB	**D(3)**	result vector

NOTES :

1. This routine performs the operation:

   $$\mathbf{b} = \mathbf{M}^T \cdot \mathbf{a}$$

   where **a** and **b** are the 3-vectors VA and VB respectively, and **M** is the 3×3 matrix DM.
2. The main function of this routine is apply an inverse rotation; under these circumstances, M is *orthogonal*, with its inverse the same as its transpose.
3. To comply with the ANSI Fortran 77 standard, VA and VB must **not** be the same array. The routine is, in fact, coded so as to work properly on the VAX and many other systems even if this rule is violated, something that is **not**, however, recommended.

SLA_DJCAL MJD to Gregorian for Output SLA_DJCAL

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 : `CALL sla_DJCAL (NDP, DJM, IYMDF, J)`

GIVEN :

NDP	**I**	number of decimal places of days in fraction
DJM	**D**	modified Julian Date (JD−2400000.5)

RETURNED :

IYMDF	**I(4)**	year, month, day, fraction in Gregorian calendar
J	**I**	status: nonzero = out of range

NOTES :

1. Any date after 4701BC March 1 is accepted.
2. NDP should be 4 or less to avoid overflow on machines which use 32-bit integers.

REFERENCE : The algorithm is derived from that of Hatcher, *Q. Jl. R. astr. Soc.* (1984) **25**, 53-55.

SLA_DJCL MJD to Year,Month,Day,Frac SLA_DJCL

ACTION : Modified Julian Date to Gregorian year, month, day, and fraction of a day.

CALL : `CALL sla_DJCL (DJM, IY, IM, ID, FD, J)`

GIVEN :

DJM	**D**	modified Julian Date (JD−2400000.5)

RETURNED :

IY	**I**	year
IM	**I**	month
ID	**I**	day
FD	**D**	fraction of day
J	**I**	status: 0 = OK −1 = unacceptable date (before 4701BC March 1)

REFERENCE : The algorithm is derived from that of Hatcher, *Q. Jl. R. astr. Soc.* (1984) **25**, 53-55.

SLA_DM2AV Rotation Matrix to Axial Vector SLA_DM2AV

ACTION : From a rotation matrix, determine the corresponding axial vector (double precision).

CALL : `CALL sla_DM2AV (RMAT, AXVEC)`

GIVEN :

RMAT	**D(3,3)**	rotation matrix

RETURNED :

AXVEC	**D(3)**	axial vector (radians)

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis. The axis is called the *Euler axis*, and the angle through which the reference frame rotates is called the *Euler angle*. The *axial vector* returned by this routine has the same direction as the Euler axis, and its magnitude is the Euler angle in radians.
2. The magnitude and direction of the axial vector can be separated by means of the routine sla_DVN.
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.

SLA_DMAT — Solve Simultaneous Equations — SLA_DMAT

ACTION : Matrix inversion and solution of simultaneous equations (double precision).

CALL : `CALL sla_DMAT (N, A, Y, D, JF, IW)`

GIVEN :

N	**I**	number of unknowns
A	**D(N,N)**	matrix
Y	**D(N)**	vector

RETURNED :

A	**D(N,N)**	matrix inverse
Y	**D(N)**	solution
D	**D**	determinant
JF	**I**	singularity flag: 0=OK
IW	**I(N)**	workspace

NOTES :

1. For the set of n simultaneous linear equations in n unknowns:

   $\mathbf{A}\cdot\mathbf{y} = \mathbf{x}$

   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,

   sla_DMAT computes:

   - the inverse of matrix $\mathbf{A}$,
   - the determinant of matrix $\mathbf{A}$, and
   - the vector of n unknowns $\mathbf{y}$.

   Argument N is the order n, A (given) is the matrix $\mathbf{A}$, Y (given) is the vector $\mathbf{x}$ and Y (returned) is the vector $\mathbf{y}$. The argument A (returned) is the inverse matrix $\mathbf{A}^{-1}$, and D is $det(\mathbf{A})$.

2. JF is the singularity flag. If the matrix is non-singular, JF=0 is returned. If the matrix is singular, JF=−1 and D=0D0 are returned. In the latter case, the contents of array 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.
4. This routine replaces the obsolete sla_DMATRX.

SLA_DMOON Approx Moon Pos/Vel SLA_DMOON

ACTION : Approximate geocentric position and velocity of the Moon (double precision).

CALL : `CALL sla_DMOON (DATE, PV)`

GIVEN :

DATE	**D**	TDB (loosely ET) as a Modified Julian Date (JD−2400000.5)

RETURNED :

PV	**D(6)**	Moon $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, mean equator and equinox of date (AU, AU s^{-1})

NOTES :

1. This routine 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 timescale *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.

SLA_DMXM — Multiply 3×3 Matrices — SLA_DMXM

ACTION : Product of two 3×3 matrices (double precision).

CALL : `CALL sla_DMXM (A, B, C)`

GIVEN :

A	**D(3,3)**	matrix **A**
B	**D(3,3)**	matrix **B**

RETURNED :

C	**D(3,3)**	matrix result: **A**×**B**

NOTE : To comply with the ANSI Fortran 77 standard, A, B and C must be different arrays. The routine is, in fact, coded so as to work properly on the VAX and many other systems even if this rule is violated, something that is **not**, however, recommended.

SLA_DMXV — Apply 3D Rotation — SLA_DMXV

ACTION : Multiply a 3-vector by a rotation matrix (double precision).

CALL : `CALL sla_DMXV (DM, VA, VB)`

GIVEN :

DM	**D(3,3)**	rotation matrix
VA	**D(3)**	vector to be rotated

RETURNED :

VB	**D(3)**	result vector

NOTES :

1. This routine performs the operation:

   $$\mathbf{b} = \mathbf{M}{\cdot}\mathbf{a}$$

   where **a** and **b** are the 3-vectors VA and VB respectively, and **M** is the 3×3 matrix DM.
2. The main function of this routine is apply a rotation; under these circumstances, **M** is a *proper real orthogonal* matrix.
3. To comply with the ANSI Fortran 77 standard, VA and VB must **not** be the same array. The routine is, in fact, coded so as to work properly on the VAX and many other systems even if this rule is violated, something that is **not**, however, recommended.

SLA_DPAV Position-Angle Between Two Directions SLA_DPAV

ACTION : Returns the bearing (position angle) of one celestial direction with respect to another (double precision).

CALL : `D = sla_DPAV (V1, V2)`

GIVEN :

V1	**D(3)**	direction cosines of one point
V2	**D(3)**	directions cosines of the other point

RETURNED :

sla_DPAV	**D**	position-angle of 2nd point with respect to 1st

NOTES :

1. The coordinate frames correspond to $[\,\alpha, \delta\,]$, $[\lambda, \phi]$ *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. The routine sla_DBEAR performs an equivalent function except that the points are specified in the form of spherical coordinates.

SLA_DR2AF Radians to Deg,Min,Sec,Frac SLA_DR2AF

ACTION : Convert an angle in radians to degrees, arcminutes, arcseconds, fraction (double precision).

CALL : `CALL sla_DR2AF (NDP, ANGLE, SIGN, IDMSF)`

GIVEN :

NDP	**I**	number of decimal places of arcseconds
ANGLE	**D**	angle in radians

RETURNED :

SIGN	**C**	'+' or '−'
IDMSF	**I(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 PRECISION floating-point numbers on the target machine, and the risk of overflowing IDMSF(4). For example, on a VAX computer, 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 the 32-bit integer IDMSF(4).
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(1)=360 and setting IDMSF(1-4) to zero.

SLA_DR2TF Radians to Hour,Min,Sec,Frac SLA_DR2TF

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

CALL : `CALL sla_DR2TF (NDP, ANGLE, SIGN, IHMSF)`

GIVEN :

NDP	**I**	number of decimal places of seconds
ANGLE	**D**	angle in radians

RETURNED :

SIGN	**C**	'+' or '−'
IHMSF	**I(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 PRECISION floating-point numbers on the target machine, and the risk of overflowing IHMSF(4). For example, on a VAX computer, 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 the 32-bit integer IHMSF(4).
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(1)=24 and setting IHMSF(1-4) to zero.

SLA_DRANGE Put Angle into Range $\pm\pi$ SLA_DRANGE

ACTION : Normalize an angle into the range $\pm\pi$ (double precision).

CALL : `D = sla_DRANGE (ANGLE)`

GIVEN :

ANGLE	**D**	angle in radians

RETURNED :

sla_DRANGE	**D**	ANGLE expressed in the range $\pm\pi$.

SLA_DRANRM Put Angle into Range $0\!-\!2\pi$ SLA_DRANRM

ACTION : Normalize an angle into the range $0\!-\!2\pi$ (double precision).

CALL : `D = sla_DRANRM (ANGLE)`

GIVEN :

ANGLE	**D**	angle in radians

RETURNED :

sla_DRANRM	**D**	ANGLE expressed in the range $0\!-\!2\pi$

SLA_DS2C6 — Spherical Pos/Vel to Cartesian — SLA_DS2C6

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

CALL : `CALL sla_DS2C6 (A, B, R, AD, BD, RD, V)`

GIVEN :

A	**D**	longitude (radians) – for example α
B	**D**	latitude (radians) – for example δ
R	**D**	radial coordinate
AD	**D**	longitude derivative (radians per unit time)
BD	**D**	latitude derivative (radians per unit time)
RD	**D**	radial derivative

RETURNED :

V	**D(6)**	$[\,x, y, z, \dot{x}, \dot{y}, \dot{z}\,]$

SLA_DS2TP — Spherical to Tangent Plane — SLA_DS2TP

ACTION : Projection of spherical coordinates onto the tangent plane (double precision).

CALL : `CALL sla_DS2TP (RA, DEC, RAZ, DECZ, XI, ETA, J)`

GIVEN :

RA,DEC	**D**	spherical coordinates of star (radians)
RAZ,DECZ	**D**	spherical coordinates of tangent point (radians)

RETURNED :

XI,ETA	**D**	tangent plane coordinates (radians)
J	**I**	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 routine sla_DV2TP is available.

SLA_DSEP — Angle Between 2 Points on Sphere — SLA_DSEP

ACTION : Angle between two points on a sphere (double precision).

CALL : `D = sla_DSEP (A1, B1, A2, B2)`

GIVEN :

A1,B1	**D**	spherical coordinates of one point (radians)
A2,B2	**D**	spherical coordinates of the other point (radians)

RETURNED :

sla_DSEP	**D**	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.

SLA_DT — Approximate ET minus UT — SLA_DT

ACTION : Estimate ΔT, the offset between dynamical time and Universal Time, for a given historical epoch.

CALL : `D = sla_DT (EPOCH)`

GIVEN :

EPOCH	**D**	(Julian) epoch (*e.g.* 1850D0)

RETURNED :

sla_DT	**D**	approximate ET−UT (after 1984, TT−UT1) 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 routines.

4. The models used are based on a lunar tidal acceleration value of $-26\overset{''}{.}00$ 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.

SLA_DTF2D — Hour,Min,Sec to Days — SLA_DTF2D

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

CALL : `CALL sla_DTF2D (IHOUR, IMIN, SEC, DAYS, J)`

GIVEN :

IHOUR	**I**	hours
IMIN	**I**	minutes
SEC	**D**	seconds

RETURNED :

DAYS	**D**	interval in days
J	**I**	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 routine.

SLA_DTF2R — Hour,Min,Sec to Radians — SLA_DTF2R

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

CALL : `CALL sla_DTF2R (IHOUR, IMIN, SEC, RAD, J)`

GIVEN :

IHOUR	**I**	hours
IMIN	**I**	minutes
SEC	**D**	seconds

RETURNED :

RAD	**D**	angle in radians
J	**I**	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 routine.

SLA_DTP2S — Tangent Plane to Spherical — SLA_DTP2S

ACTION : Transform tangent plane coordinates into spherical coordinates (double precision)

CALL : `CALL sla_DTP2S (XI, ETA, RAZ, DECZ, RA, DEC)`

GIVEN :

XI,ETA	**D**	tangent plane rectangular coordinates (radians)
RAZ,DECZ	**D**	spherical coordinates of tangent point (radians)

RETURNED :

RA,DEC	**D**	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 routine sla_DTP2V is available.

SLA_DTP2V Tangent Plane to Direction Cosines SLA_DTP2V

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 : `CALL sla_DTP2V (XI, ETA, V0, V)`

GIVEN :

XI,ETA	**D**	tangent plane coordinates of star (radians)
V0	**D(3)**	direction cosines of tangent point

RETURNED :

V	**D(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 routine is the Cartesian equivalent of the routine sla_DTP2S.

SLA_DTPS2C Plate centre from ξ,η and α,δ SLA_DTPS2C

ACTION : From the tangent plane coordinates of a star of known $[\,\alpha,\delta\,]$, determine the $[\,\alpha,\delta\,]$ of the tangent point (double precision)

CALL : `CALL sla_DTPS2C (XI, ETA, RA, DEC, RAZ1, DECZ1, RAZ2, DECZ2, N)`

GIVEN :

XI,ETA	**D**	tangent plane rectangular coordinates (radians)
RA,DEC	**D**	spherical coordinates (radians)

RETURNED :

RAZ1,DECZ1	**D**	spherical coordinates of tangent point, solution 1
RAZ2,DECZ2	**D**	spherical coordinates of tangent point, solution 2
N	**I**	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 "over-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 routine sla_DTPV2C is available.

SLA_DTPV2C — Plate centre from ξ,η and x,y,z — SLA_DTPV2C

ACTION : From the tangent plane coordinates of a star of known direction cosines, determine the direction cosines of the tangent point (double precision)

CALL : `CALL sla_DTPV2C (XI, ETA, V, V01, V02, N)`

GIVEN :

XI,ETA	**D**	tangent plane coordinates of star (radians)
V	**D(3)**	direction cosines of star

RETURNED :

V01	**D(3)**	direction cosines of tangent point, solution 1
V01	**D(3)**	direction cosines of tangent point, solution 2
N	**I**	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 "over-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 routine is the Cartesian equivalent of the routine sla_DTPS2C.

SLA_DTT — TT minus UTC — SLA_DTT

ACTION : Compute ΔTT, the increment to be applied to Coordinated Universal Time UTC to give Terrestrial Time TT.

CALL : `D = sla_DTT (DJU)`

GIVEN :

DJU	**D**	UTC date as a modified JD (JD−2400000.5)

RETURNED :

sla_DTT	**D**	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 $23^{\rm h}\,59^{\rm m}\,59^{\rm s}$.

2. Pre 1972 January 1 a fixed value of 10 + ET−TAI is returned.
3. TT is one interpretation of the defunct timescale *Ephemeris Time*, ET.
4. See also the routine sla_DT, which roughly estimates ET−UT for historical epochs.

SLA_DV2TP — Direction Cosines to Tangent Plane — SLA_DV2TP

ACTION : Given the direction cosines of a star and of the tangent point, determine the star's tangent-plane coordinates (double precision).

CALL : `CALL sla_DV2TP (V, V0, XI, ETA, J)`

GIVEN :

V	**D(3)**	direction cosines of star
V0	**D(3)**	direction cosines of tangent point

RETURNED :

XI,ETA	**D**	tangent plane coordinates (radians)
J	**I**	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 routine is the Cartesian equivalent of the routine sla_DS2TP.

SLA_DVDV — Scalar Product — SLA_DVDV

ACTION : Scalar product of two 3-vectors (double precision).

CALL : `D = sla_DVDV (VA, VB)`

GIVEN :

VA	**D(3)**	first vector
VB	**D(3)**	second vector

RETURNED :

sla_DVDV	**D**	scalar product VA.VB

SLA_DVN — Normalize Vector — SLA_DVN

ACTION : Normalize a 3-vector, also giving the modulus (double precision).

CALL : `CALL sla_DVN (V, UV, VM)`

GIVEN :

V	**D(3)**	vector

RETURNED :

UV	**D(3)**	unit vector in direction of V
VM	**D**	modulus of V

NOTE : If the modulus of V is zero, UV is set to zero as well.

SLA_DVXV — Vector Product — SLA_DVXV

ACTION : Vector product of two 3-vectors (double precision).

CALL : `CALL sla_DVXV (VA, VB, VC)`

GIVEN :

VA	**D(3)**	first vector
VB	**D(3)**	second vector

RETURNED :

VC	**D(3)**	vector product VA×VB

SLA_E2H — h, δ to Az,El — SLA_E2H

ACTION : Equatorial to horizon coordinates (single precision).

CALL : `CALL sla_DE2H (HA, DEC, PHI, AZ, EL)`

GIVEN :

HA	**R**	hour angle (radians)
DEC	**R**	declination (radians)
PHI	**R**	latitude (radians)

RETURNED :

AZ	**R**	azimuth (radians)
EL	**R**	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 routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude, and (for tracking a star) sine and cosine of declination.

SLA_EARTH — Approx Earth Pos/Vel — SLA_EARTH

ACTION : Approximate heliocentric position and velocity of the Earth (single precision).

CALL : `CALL sla_EARTH (IY, ID, FD, PV)`

GIVEN :

IY	**I**	year
ID	**I**	day in year (1 = Jan 1st)
FD	**R**	fraction of day

RETURNED :

PV	**R(6)**	Earth $[\,x,y,z,\dot{x},\dot{y},\dot{z}\,]$ (AU, AU s^{-1})

NOTES :

1. The date and time is TDB (loosely ET) 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 sla_CALYD or sla_CLYD.
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 routine sla_EVP.

SLA_ECLEQ — Ecliptic to Equatorial — SLA_ECLEQ

ACTION : Transformation from ecliptic longitude and latitude to J2000.0 $[\alpha, \delta]$.

CALL : `CALL sla_ECLEQ (DL, DB, DATE, DR, DD)`

GIVEN :

DL,DB	**D**	ecliptic longitude and latitude (mean of date, IAU 1980 theory, radians)
DATE	**D**	TDB (formerly ET) as Modified Julian Date (JD−2400000.5)

RETURNED :

DR,DD	**D**	J2000.0 mean $[\alpha, \delta]$ (radians)

SLA_ECMAT — Form $\alpha, \delta \rightarrow \lambda, \beta$ Matrix — SLA_ECMAT

ACTION : Form the equatorial to ecliptic rotation matrix (IAU 1980 theory).

CALL : `CALL sla_ECMAT (DATE, RMAT)`

GIVEN :

DATE	**D**	TDB (formerly ET) as Modified Julian Date (JD−2400000.5)

RETURNED :

RMAT	**D(3,3)**	rotation matrix

NOTES :

1. RMAT is matrix **M** in the expression $\mathbf{v}_{ecl} = \mathbf{M} \cdot \mathbf{v}_{equ}$.
2. The equator, equinox and ecliptic are mean of date.

REFERENCE : Murray, C.A., *Vectorial Astrometry*, section 4.3.

SLA_ECOR RV & Time Corrns to Sun SLA_ECOR

ACTION : Component of Earth orbit velocity and heliocentric light time in a given direction.

CALL : `CALL sla_ECOR (RM, DM, IY, ID, FD, RV, TL)`

GIVEN :

RM,DM	**R**	mean $[\,\alpha, \delta\,]$ of date (radians)
IY	**I**	year
ID	**I**	day in year (1 = Jan 1st)
FD	**R**	fraction of day

RETURNED :

RV	**R**	component of Earth orbital velocity (km s^{-1})
TL	**R**	component of heliocentric light time (s)

NOTES :

1. The date and time is TDB (loosely ET) 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 sla_CALYD or sla_CLYD.
2. Sign convention:
   - The velocity component is +ve when the Earth is receding from the given point on the sky.
   - The light time component is +ve 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 $0\overset{s}{.}03$ at worst, but is usually better than $0\overset{s}{.}01$.

   For applications requiring higher accuracy, see the sla_EVP routine.

SLA_EG50 B1950 α, δ to Galactic SLA_EG50

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

CALL : `CALL sla_EG50 (DR, DD, DL, DB)`

GIVEN :

DR,DD	**D**	B1950.0 $[\alpha, \delta]$ (radians)

RETURNED :

DL,DB	**D**	galactic longitude and latitude $[l^{II}, b^{II}]$ (radians)

NOTE : The equatorial coordinates are B1950.0 FK4. Use the routine sla_EQGAL if conversion from J2000.0 FK5 coordinates is required.

REFERENCE : Blaauw *et al.*, 1960, *Mon.Not.R.astr.Soc.*, **121**, 123.

SLA_EL2UE Conventional to Universal Elements SLA_EL2UE

ACTION : Transform conventional osculating orbital elements into "universal" form.

CALL :
```
CALL sla_EL2UE (DATE, JFORM, EPOCH, ORBINC, ANODE,
                PERIH, AORQ, E, AORL, DM,
                U, JSTAT)
```

GIVEN :

DATE	**D**	epoch (TT MJD) of osculation (Note 3)
JFORM	**I**	choice of element set (1-3; Note 6)
EPOCH	**D**	epoch of elements (t_0 or T, TT MJD)
ORBINC	**D**	inclination (i, radians)
ANODE	**D**	longitude of the ascending node (Ω, radians)
PERIH	**D**	longitude or argument of perihelion (ϖ or ω, radians)
AORQ	**D**	mean distance or perihelion distance (a or q, AU)
E	**D**	eccentricity (e)
AORL	**D**	mean anomaly or longitude (M or L, radians, JFORM=1,2 only)
DM	**D**	daily motion (n, radians, JFORM=1 only)

RETURNED :

U	**D(13)**	universal orbital elements (Note 1)
(1)		combined mass $(M + m)$
(2)		total energy of the orbit (α)
(3)		reference (osculating) epoch (t_0)
(4-6)		position at reference epoch $(\mathbf{r}_0)$
(7-9)		velocity at reference epoch $(\mathbf{v}_0)$
(10)		heliocentric distance at reference epoch
(11)		$\mathbf{r}_0.\mathbf{v}_0$
(12)		date (t)
(13)		universal eccentric anomaly (ψ) of date, approx
JSTAT	**I**	status: 0 = OK −1 = illegal JFORM −2 = illegal E −3 = illegal AORQ −4 = illegal DM −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 routine is sla_UE2PV. This takes the set of numbers that the present routine outputs and uses them to derive the object's position and velocity. A single prediction requires one call to the present routine followed by one call to sla_UE2PV; for convenience, the two calls are packaged as the routine sla_PLANEL. Multiple predictions may be made by again calling the present routine once, but then calling sla_UE2PV multiple times, which is faster than multiple calls to sla_PLANEL.
3. DATE is the epoch of osculation. It is in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD−2400000.5).
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:

EPOCH	=	epoch of elements t_0 (TT 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 (TT 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 (TT 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 $(0 \leq e \leq 10)$

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 program of D. H. P. Jones (private communication, 1996). The method is based on Stumpff's Universal Variables.

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

SLA_EPB — MJD to Besselian Epoch — SLA_EPB

ACTION : Conversion of Modified Julian Date to Besselian Epoch.

CALL : `D = sla_EPB (DATE)`

GIVEN :

DATE	**D**	Modified Julian Date (JD−2400000.5)

RETURNED :

sla_EPB	**D**	Besselian Epoch

REFERENCE : Lieske, J.H., 1979, *Astr.Astrophys.* **73**, 282.

SLA_EPB2D Besselian Epoch to MJD SLA_EPB2D

ACTION : Conversion of Besselian Epoch to Modified Julian Date.

CALL : `D = sla_EPB2D (EPB)`

GIVEN :

EPB	**D**	Besselian Epoch

RETURNED :

sla_EPB2D	**D**	Modified Julian Date (JD−2400000.5)

REFERENCE : Lieske, J.H., 1979. *Astr.Astrophys.* **73**, 282.

SLA_EPCO Convert Epoch to B or J SLA_EPCO

ACTION : Convert an epoch to Besselian or Julian to match another one.

CALL : `D = sla_EPCO (K0, K, E)`

GIVEN :

K0	**C**	form of result: 'B'=Besselian, 'J'=Julian
K	**C**	form of given epoch: 'B' or 'J'
E	**D**	epoch

RETURNED :

sla_EPCO	**D**	the given epoch converted as necessary

NOTES :

1. The result is always either equal to or very close to the given epoch E. The routine is required only in applications where punctilious treatment of heterogeneous mixtures of star positions is necessary.
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.

SLA_EPJ — MJD to Julian Epoch — SLA_EPJ

ACTION : Convert Modified Julian Date to Julian Epoch.

CALL : `D = sla_EPJ (DATE)`

GIVEN :

DATE	**D**	Modified Julian Date (JD−2400000.5)

RETURNED :

sla_EPJ	**D**	Julian Epoch

REFERENCE : Lieske, J.H., 1979. *Astr.Astrophys.*, **73**, 282.

SLA_EPJ2D — Julian Epoch to MJD — SLA_EPJ2D

ACTION : Convert Julian Epoch to Modified Julian Date.

CALL : `D = sla_EPJ2D (EPJ)`

GIVEN :

EPJ	**D**	Julian Epoch

RETURNED :

sla_EPJ2D	**D**	Modified Julian Date (JD−2400000.5)

REFERENCE : Lieske, J.H., 1979. *Astr.Astrophys.*, **73**, 282.

SLA_EQECL — J2000 α, δ to Ecliptic — SLA_EQECL

ACTION : Transformation from J2000.0 equatorial coordinates to ecliptic longitude and latitude.

CALL : `CALL sla_EQECL (DR, DD, DATE, DL, DB)`

GIVEN :

DR,DD	**D**	J2000.0 mean $[\,\alpha, \delta\,]$ (radians)
DATE	**D**	TDB (formerly ET) as Modified Julian Date (JD−2400000.5)

RETURNED :

DL,DB	**D**	ecliptic longitude and latitude (mean of date, IAU 1980 theory, radians)

SLA_EQEQX — Equation of the Equinoxes — SLA_EQEQX

ACTION : Equation of the equinoxes (IAU 1994).

CALL : `D = sla_EQEQX (DATE)`

GIVEN :

DATE	**D**	TDB (formerly ET) as Modified Julian Date (JD−2400000.5)

RETURNED :

sla_EQEQX	**D**	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 takes place on 1997 February 27.

REFERENCE : Capitaine, N. & Gontier, A.-M. (1993), *Astron. Astrophys.*, **275**, 645-650.

SLA_EQGAL — J2000 α, δ to Galactic — SLA_EQGAL

ACTION : Transformation from J2000.0 FK5 equatorial coordinates to IAU 1958 galactic coordinates.

CALL : `CALL sla_EQGAL (DR, DD, DL, DB)`

GIVEN :

DR,DD	**D**	J2000.0 $[\alpha, \delta]$ (radians)

RETURNED :

DL,DB	**D**	galactic longitude and latitude $[l^{II}, b^{II}]$ (radians)

NOTE : The equatorial coordinates are J2000.0 FK5. Use the routine sla_EG50 if conversion from B1950.0 FK4 coordinates is required.

REFERENCE : Blaauw *et al.*, 1960, *Mon.Not.R.astr.Soc.*, **121**, 123.

SLA_ETRMS — E-terms of Aberration — SLA_ETRMS

ACTION : Compute the E-terms vector – the part of the annual aberration which arises from the eccentricity of the Earth's orbit.

CALL : `CALL sla_ETRMS (EP, EV)`

GIVEN :

EP	**D**	Besselian epoch

RETURNED :

EV	**D(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.

SLA_EULER — Rotation Matrix from Euler Angles — SLA_EULER

ACTION : Form a rotation matrix from the Euler angles – three successive rotations about specified Cartesian axes (single precision).

CALL : `CALL sla_EULER (ORDER, PHI, THETA, PSI, RMAT)`

GIVEN :

ORDER	**C*(*)**	specifies about which axes the rotations occur
PHI	**R**	1st rotation (radians)
THETA	**R**	2nd rotation (radians)
PSI	**R**	3rd rotation (radians)

RETURNED :

RMAT	**R(3,3)**	rotation matrix

NOTES :

1. A rotation is positive when the reference frame rotates anticlockwise as seen looking towards the origin from the positive region of the specified axis.
2. 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. In detail:
   - The axis names can be any of the following, in any order or combination: X, Y, Z, uppercase or lowercase, 1, 2, 3. Normal axis labelling/numbering conventions apply; the xyz ($\equiv$ 123) triad is right-handed. Thus, the 'ZXZ' example given above could be written 'zxz' or '313' (or even 'ZxZ' or '3xZ').
   - ORDER is terminated by length or by the first unrecognized character.
   - Fewer than three rotations are acceptable, in which case the later angle arguments are ignored.
3. Zero rotations produces a unit RMAT.

SLA_EVP — Earth Position & Velocity — SLA_EVP

ACTION : Barycentric and heliocentric velocity and position of the Earth.

CALL : `CALL sla_EVP (DATE, DEQX, DVB, DPB, DVH, DPH)`

GIVEN :

DATE	**D**	TDB (formerly ET) as a Modified Julian Date (JD−2400000.5)
DEQX	**D**	Julian Epoch (*e.g.* 2000D0) 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 :

DVB	**D(3)**	barycentric $[\,\dot{x},\dot{y},\dot{z}\,]$, AU s^{-1}
DPB	**D(3)**	barycentric $[\,x,y,z\,]$, AU
DVH	**D(3)**	heliocentric $[\,\dot{x},\dot{y},\dot{z}\,]$, AU s^{-1}
DPH	**D(3)**	heliocentric $[\,x,y,z\,]$, AU

NOTES :

1. This routine is used when accuracy is more important than CPU time, yet the extra complication of reading a pre-computed ephemeris is not justified. 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 routine is an adaption of the BARVEL and BARCOR subroutines of P.Stumpff, which are described in *Astr.Astrophys.Suppl.Ser.* **41**, 1-8 (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 new (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 routines 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. Note also that one of Stumpff's precession constants differs by $0\overset{''}{.}001$ from the value given in the *Explanatory Supplement.*

SLA_FITXY — Fit Linear Model to Two $[\,x,y\,]$ Sets — SLA_FITXY

ACTION : Fit a linear model to relate two sets of $[\,x,y\,]$ coordinates.

CALL : `CALL sla_FITXY (ITYPE,NP,XYE,XYM,COEFFS,J)`

GIVEN :

ITYPE	**I**	type of model: 4 or 6 (note 1)
NP	**I**	number of samples (note 2)
XYE	**D(2,NP)**	expected $[\,x,y\,]$ for each sample
XYM	**D(2,NP)**	measured $[\,x,y\,]$ for each sample

RETURNED :

COEFFS	**D(6)**	coefficients of model (note 3)
J	**I**	status: 0 = OK −1 = illegal ITYPE −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. 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:

   $$x_e = a + bx_m + cy_m$$
   $$y_e = d + ex_m + fy_m$$

   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 sla_PXY, sla_INVF, sla_XY2XY, sla_DCMPF.

SLA_FK425 FK4 to FK5 SLA_FK425

ACTION : Convert B1950.0 FK4 star data to J2000.0 FK5. This routine 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 :
```
CALL sla_FK425 (R1950,D1950,DR1950,DD1950,P1950,V1950,
                R2000,D2000,DR2000,DD2000,P2000,V2000)
```

GIVEN :

R1950	**D**	B1950.0 α (radians)
D1950	**D**	B1950.0 δ (radians)
DR1950	**D**	B1950.0 proper motion in α (radians per tropical year)
DD1950	**D**	B1950.0 proper motion in δ (radians per tropical year)
P1950	**D**	B1950.0 parallax (arcsec)
V1950	**D**	B1950.0 radial velocity (km s^{-1}, +ve = moving away)

RETURNED :

R2000	**D**	J2000.0 α (radians)
D2000	**D**	J2000.0 δ (radians)
DR2000	**D**	J2000.0 proper motion in α (radians per Julian year)
DD2000	**D**	J2000.0 proper motion in δ (radians per Julian year)
P2000	**D**	J2000.0 parallax (arcsec)
V2000	**D**	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. Conversion from Besselian epoch 1950.0 to Julian epoch 2000.0 only is provided for. Conversions involving other epochs will require use of the appropriate precession, proper motion, and E-terms routines before and/or after FK425 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 routine 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 sla_FK45Z, sla_FK524, sla_FK54Z.

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.

SLA_FK45Z FK4 to FK5, no P.M. or Parallax SLA_FK45Z

ACTION : Convert B1950.0 FK4 star data to J2000.0 FK5 assuming zero proper motion in the FK5 frame. This routine 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 routine 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 : `CALL sla_FK45Z (R1950,D1950,BEPOCH,R2000,D2000)`

GIVEN :

R1950	**D**	B1950.0 FK4 α at epoch BEPOCH (radians)
D1950	**D**	B1950.0 FK4 δ at epoch BEPOCH (radians)
BEPOCH	**D**	Besselian epoch (*e.g.* 1979.3D0)

RETURNED :

R2000	**D**	J2000.0 FK5 α (radians)
D2000	**D**	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. Conversion from Besselian epoch 1950.0 to Julian epoch 2000.0 only is provided for. Conversions involving other epochs will require use of the appropriate precession, proper motion, and E-terms routines before and/or after FK45Z 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 routine to include the effect of differential E-terms on the proper motions for all stars, whether polar or not. At epoch 2000, and measuring on the sky rather than in terms of $\Delta\alpha$, the errors resulting from this simplification are less than 1 milliarcsecond in position and 1 milliarcsecond per century in proper motion.

4. See also sla_FK425, sla_FK524, sla_FK54Z.

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.

SLA_FK524 — FK5 to FK4 — SLA_FK524

ACTION : Convert J2000.0 FK5 star data to B1950.0 FK4. This routine 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 :
```
CALL sla_FK524 (R2000,D2000,DR2000,DD2000,P2000,V2000,
                R1950,D1950,DR1950,DD1950,P1950,V1950)
```

GIVEN :

R2000	**D**	J2000.0 α (radians)
D2000	**D**	J2000.0 δ (radians)
DR2000	**D**	J2000.0 proper motion in α (radians per Julian year)
DD2000	**D**	J2000.0 proper motion in δ (radians per Julian year)
P2000	**D**	J2000.0 parallax (arcsec)
V2000	**D**	J2000 radial velocity (km s^{-1}, +ve = moving away)

RETURNED :

R1950	**D**	B1950.0 α (radians)
D1950	**D**	B1950.0 δ (radians)
DR1950	**D**	B1950.0 proper motion in α (radians per tropical year)
DD1950	**D**	B1950.0 proper motion in δ (radians per tropical year)
P1950	**D**	B1950.0 parallax (arcsec)
V1950	**D**	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 conversion from Julian epoch 2000.0 to Besselian epoch 1950.0 only is provided for. Conversions involving other epochs will require use of the appropriate precession, proper motion, and E-terms routines before and/or after FK524 is called.
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 routine to include the effect of differential E-terms on the proper motions for all stars, whether polar or not. At epoch 2000, and measuring on the sky rather than in terms of $\Delta\alpha$, the errors resulting from this simplification are less than 1 milliarcsecond in position and 1 milliarcsecond per century in proper motion.
4. See also sla_FK425, sla_FK45Z, sla_FK54Z.

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.

SLA_FK52H — FK5 to Hipparcos — SLA_FK52H

ACTION : Transform an FK5 (J2000) position and proper motion into the frame of the Hipparcos catalogue.

CALL : `CALL sla_FK52H (R5,D5,DR5,DD5,RH,DH,DRH,DDH)`

GIVEN :

R5	**D**	J2000.0 FK5 α (radians)
D5	**D**	J2000.0 FK5 δ (radians)
DR5	**D**	J2000.0 FK5 proper motion in α (radians per Julian year)
DD5	**D**	J2000.0 FK5 proper motion in δ (radians per Julian year)

RETURNED :

RH	**D**	Hipparcos α (radians)
DH	**D**	Hipparcos δ (radians)
DRH	**D**	Hipparcos proper motion in α (radians per Julian year)
DDH	**D**	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):

	orientation	*spin*
x	-19.9	-0.30
y	-9.1	$+0.60$
z	$+22.9$	$+0.70$
	mas	*mas/y*

   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 sla_FK5HZ, sla_H2FK5, sla_HFK5Z.

REFERENCE : Feissel, M. & Mignard, F., 1998., *Astron.Astrophys.* **331**, L33-L36.

SLA_FK54Z — FK5 to FK4, no P.M. or Parallax — SLA_FK54Z

ACTION : Convert a J2000.0 FK5 star position to B1950.0 FK4 assuming FK5 zero proper motion and parallax. This routine converts star positions from the new, IAU 1976, FK5, Fricke system to the old, Bessel-Newcomb, FK4 system.

CALL : `CALL sla_FK54Z (R2000,D2000,BEPOCH,R1950,D1950,DR1950,DD1950)`

GIVEN :

R2000	**D**	J2000.0 FK5 α (radians)
D2000	**D**	J2000.0 FK5 δ (radians)
BEPOCH	**D**	Besselian epoch (*e.g.* 1950D0)

RETURNED :

R1950	**D**	B1950.0 FK4 α at epoch BEPOCH (radians)
D1950	**D**	B1950.0 FK4 δ at epoch BEPOCH (radians)
DR1950	**D**	B1950.0 FK4 proper motion in α (radians per tropical year)
DD1950	**D**	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. Conversion from Julian epoch 2000.0 to Besselian epoch 1950.0 only is provided for. Conversions involving other epochs will require use of the appropriate precession routines before and after this routine is called.
3. Unlike in the sla_FK524 routine, 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 routine returns those *fictitious proper motions*.
5. The position returned by this routine is in the B1950 reference frame but at Besselian epoch BEPOCH. For comparison with catalogues the BEPOCH argument will frequently be 1950D0.
6. See also sla_FK425, sla_FK45Z, sla_FK524.

SLA_FK5HZ FK5 to Hipparcos, no P.M. SLA_FK5HZ

ACTION : Transform an FK5 (J2000) star position into the frame of the Hipparcos catalogue, assuming zero Hipparcos proper motion.

CALL : `CALL sla_FK52H (R5,D5,EPOCH,RH,DH)`

GIVEN :

R5	**D**	J2000.0 FK5 α (radians)
D5	**D**	J2000.0 FK5 δ (radians)
EPOCH	**D**	Julian epoch (TDB)

RETURNED :

RH	**D**	Hipparcos α (radians)
DH	**D**	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):

	orientation	*spin*
x	-19.9	-0.30
y	-9.1	$+0.60$
z	$+22.9$	$+0.70$
	mas	*mas/y*

   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 sla_FK52H, sla_H2FK5, sla_HFK5Z.

REFERENCE : Feissel, M. & Mignard, F., 1998., *Astron.Astrophys.* **331**, L33-L36.

SLA_FLOTIN Decode a Real Number SLA_FLOTIN

ACTION : Convert free-format input into single precision floating point.

CALL : `CALL sla_FLOTIN (STRING, NSTRT, RESLT, JFLAG)`

GIVEN :

STRING	**C**	string containing number to be decoded
NSTRT	**I**	pointer to where decoding is to commence
RESLT	**R**	current value of result

RETURNED :

NSTRT	**I**	advanced to next number
RESLT	**R**	result
JFLAG	**I**	status: $-1 = -\text{OK}$, $0 = +\text{OK}$, $1 =$ null result, $2 =$ error

NOTES :

1. The reason sla_FLOTIN 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. *n.b.* The test for TAB is ASCII-specific.
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,.,D,E or space may be used to signal the end of the number and terminate decoding.
   - Comma is recognized by sla_FLOTIN as a special case; it is skipped, leaving the pointer 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,.,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 sla_FLOTIN, 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 sla_FLOTIN, 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, sla_FLOTIN does not detect that '1E999' is unacceptable (on a computer 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 sla_DFLTIN and sla_INTIN.

SLA_GALEQ — Galactic to J2000 α, δ — SLA_GALEQ

ACTION : Transformation from IAU 1958 galactic coordinates to J2000.0 FK5 equatorial coordinates.

CALL : `CALL sla_GALEQ (DL, DB, DR, DD)`

GIVEN :

DL,DB	**D**	galactic longitude and latitude [l^{II}, b^{II}]

RETURNED :

DR,DD	**D**	J2000.0 [α, δ]

NOTES :

1. All arguments are in radians.
2. The equatorial coordinates are J2000.0 FK5. Use the routine sla_GE50 if conversion to B1950.0 FK4 coordinates is required.

SLA_GALSUP — Galactic to Supergalactic — SLA_GALSUP

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

CALL : `CALL sla_GALSUP (DL, DB, DSL, DSB)`

GIVEN :

DL,DB	**D**	galactic longitude and latitude [l^{II}, b^{II}] (radians)

RETURNED :

DSL,DSB	**D**	supergalactic longitude and latitude (radians)

REFERENCES :

1. de Vaucouleurs, de Vaucouleurs, & Corwin, *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 $l^{II} = 137.37$.)

SLA_GE50 Galactic to B1950 α, δ SLA_GE50

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

CALL : `CALL sla_GE50 (DL, DB, DR, DD)`

GIVEN :

DL,DB	**D**	galactic longitude and latitude $[\, l^{II}, b^{II}\,]$

RETURNED :

DR,DD	**D**	B1950.0 $[\, \alpha, \delta\,]$

NOTES :

1. All arguments are in radians.
2. The equatorial coordinates are B1950.0 FK4. Use the routine sla_GALEQ if conversion to J2000.0 FK5 coordinates is required.

REFERENCE : Blaauw *et al.*, 1960, *Mon.Not.R.astr.Soc.*, **121**, 123.

SLA_GEOC Geodetic to Geocentric SLA_GEOC

ACTION : Convert geodetic position to geocentric.

CALL : `CALL sla_GEOC (P, H, R, Z)`

GIVEN :

P	**D**	latitude (geodetic, radians)
H	**D**	height above reference spheroid (geodetic, metres)

RETURNED :

R	**D**	distance from Earth axis (AU)
Z	**D**	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.

SLA_GMST — UT to GMST — SLA_GMST

ACTION : Conversion from universal time UT1 to Greenwich mean sidereal time.

CALL : `D = sla_GMST (UT1)`

GIVEN :

UT1	**D**	universal time (strictly UT1) expressed as modified Julian Date (JD−2400000.5)

RETURNED :

sla_GMST	**D**	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 $0^{\rm h}$UT; in fact, it gives the difference between the GMST and the UT, which happens to equal the GMST (modulo 24 hours) at $0^{\rm h}$UT each day. In sla_GMST, 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 $1.0027379\cdots$ does not appear.
2. See also the routine sla_GMSTA, which delivers better numerical precision by accepting the UT date and time as separate arguments.

SLA_GMSTA UT to GMST (extra precision) SLA_GMSTA

ACTION : Conversion from universal time UT1 to Greenwich Mean sidereal time, with rounding errors minimized.

CALL : `D = sla_GMSTA (DATE, UT1)`

GIVEN :

DATE	**D**	UT1 date as Modified Julian Date (integer part of JD−2400000.5)
UT1	**D**	UT1 time (fraction of a day)

RETURNED :

sla_GMST	**D**	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 routine 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 routine sla_GMST, which accepts the UT1 as a single argument. Compared with sla_GMST, the extra numerical precision delivered by the present routine 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.

SLA_GRESID Gaussian Residual SLA_GRESID

ACTION : Generate pseudo-random normal deviate or *Gaussian residual.*

CALL : `R = sla_GRESID (S)`

GIVEN :

S	**R**	standard deviation

NOTES :

1. The results of many calls to this routine will be normally distributed with mean zero and standard deviation S.
2. The Box-Muller algorithm is used.
3. The implementation is machine-dependent.

REFERENCE : Ahrens & Dieter, 1972. *Comm.A.C.M.* **15**, 873.

SLA_H2E — Az,El to h, δ — SLA_H2E

ACTION : Horizon to equatorial coordinates (single precision).

CALL : `CALL sla_H2E (AZ, EL, PHI, HA, DEC)`

GIVEN :

AZ	**R**	azimuth (radians)
EL	**R**	elevation (radians)
PHI	**R**	latitude (radians)

RETURNED :

HA	**R**	hour angle (radians)
DEC	**R**	declination (radians)

NOTES :

1. The sign convention for azimuth is north zero, east $+\pi/2$.
2. HA 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 sla_POLMO).
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 routine it will be more efficient to use inline code, having previously computed fixed terms such as sine and cosine of latitude.

SLA_H2FK5 Hipparcos to FK5 SLA_H2FK5

ACTION : Transform a Hipparcos star position and proper motion into the FK5 (J2000) frame.

CALL : `CALL sla_H2FK5 (RH,DH,DRH,DDH,R5,D5,DR5,DD5)`

GIVEN :

RH	**D**	Hipparcos α (radians)
DH	**D**	Hipparcos δ (radians)
DRH	**D**	Hipparcos proper motion in α (radians per Julian year)
DDH	**D**	Hipparcos proper motion in δ (radians per Julian year)

RETURNED :

R5	**D**	J2000.0 FK5 α (radians)
D5	**D**	J2000.0 FK5 δ (radians)
DR5	**D**	J2000.0 FK5 proper motion in α (radians per Julian year)
DD5	**D**	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):

	orientation	*spin*
x	-19.9	-0.30
y	-9.1	$+0.60$
z	$+22.9$	$+0.70$
	mas	*mas/y*

   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 sla_FK52H, sla_FK5HZ, sla_HFK5Z.

REFERENCE : Feissel, M. & Mignard, F., 1998., *Astron.Astrophys.* **331**, L33-L36.

SLA_HFK5Z Hipparcos to FK5, no P.M. SLA_HFK5Z

ACTION : Transform a Hipparcos star position into the FK5 (J2000) frame assuming zero Hipparcos proper motion.

CALL : `CALL sla_HFK5Z (RH,DH,EPOCH,R5,D5,DR5,DD5)`

GIVEN :

RH	**D**	Hipparcos α (radians)
DH	**D**	Hipparcos δ (radians)
EPOCH	**D**	Julian epoch (TDB)

RETURNED :

R5	**D**	J2000.0 FK5 α (radians)
D5	**D**	J2000.0 FK5 δ (radians)
DR5	**D**	J2000.0 FK5 proper motion in α (radians per Julian year)
DD5	**D**	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):

	orientation	*spin*
x	-19.9	-0.30
y	-9.1	$+0.60$
z	$+22.9$	$+0.70$
	mas	*mas/y*

   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. The order of the rotations, which are very small,

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 routine returns those *fictitious proper motions.*

5. The position returned by this routine is in the FK5 J2000 reference frame but at Julian epoch EPOCH.
6. See also sla_FK52H, sla_FK5HZ, sla_H2FK5.

REFERENCE : Feissel, M. & Mignard, F., 1998., *Astron.Astrophys.* **331**, L33-L36.

SLA_IMXV — Apply 3D Reverse Rotation — SLA_IMXV

ACTION : Multiply a 3-vector by the inverse of a rotation matrix (single precision).

CALL : `CALL sla_IMXV (RM, VA, VB)`

GIVEN :

RM	**R(3,3)**	rotation matrix
VA	**R(3)**	vector to be rotated

RETURNED :

VB	**R(3)**	result vector

NOTES :

1. This routine performs the operation:

   $\mathbf{b} = \mathbf{M}^T \cdot \mathbf{a}$

   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 routine is apply an inverse rotation; under these circumstances, **M** is *orthogonal*, with its inverse the same as its transpose.
3. To comply with the ANSI Fortran 77 standard, VA and VB must **not** be the same array. The routine is, in fact, coded so as to work properly on the VAX and many other systems even if this rule is violated, something that is **not**, however, recommended.

SLA_INTIN — Decode an Integer Number — SLA_INTIN

ACTION : Convert free-format input into an integer.

CALL : `CALL sla_INTIN (STRING, NSTRT, IRESLT, JFLAG)`

GIVEN :

STRING	**C**	string containing number to be decoded
NSTRT	**I**	pointer to where decoding is to commence
IRESLT	**I**	current value of result

RETURNED :

NSTRT	**I**	advanced to next number
IRESLT	**I**	result
JFLAG	**I**	status: -1 = $-$OK, 0 = +OK, 1 = null result, 2 = error

NOTES :

1. The reason sla_INTIN 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. *n.b.* The test for TAB is ASCII-specific.
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 sla_INTIN as a special case; it is skipped, leaving the pointer 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 sla_INTIN, 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 sla_INTIN, 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 sla_FLOTIN and sla_DFLTIN.

SLA_INVF — Invert Linear Model — SLA_INVF

ACTION : Invert a linear model of the type produced by the sla_FITXY routine.

CALL : `CALL sla_INVF (FWDS,BKWDS,J)`

GIVEN :

FWDS	**D(6)**	model coefficients

RETURNED :

BKWDS	**D(6)**	inverse model
J	**I**	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:

   $x_2 = a + bx_1 + cy_1$
   $y_2 = d + ex_1 + fy_1$

   The present routine generates a new set of coefficients p, q, r, s, t & u (the array BKWDS) such that:

   $x_1 = p + qx_2 + ry_2$
   $y_1 = s + tx_2 + uy_2$

2. Two successive calls to this routine will deliver a set of coefficients equal to the starting values.
3. To comply with the ANSI Fortran 77 standard, FWDS and BKWDS must **not** be the same array. The routine is, in fact, coded so as to work properly on the VAX and many other systems even if this rule is violated, something that is **not**, however, recommended.
4. See also sla_FITXY, sla_PXY, sla_XY2XY, sla_DCMPF.

SLA_KBJ — Select Epoch Prefix — SLA_KBJ

ACTION : Select epoch prefix 'B' or 'J'.

CALL : `CALL sla_KBJ (JB, E, K, J)`

GIVEN :

JB	**I**	sla_DBJIN prefix status: 0=none, 1='B', 2='J'
E	**D**	epoch – Besselian or Julian

RETURNED :

K	**C**	'B' or 'J'
J	**I**	status: 0=OK

NOTE : The routine is mainly intended for use in conjunction with the sla_DBJIN routine. If the value of JB indicates that an explicit B or J prefix was detected by sla_DBJIN, 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.

SLA_M2AV — Rotation Matrix to Axial Vector — SLA_M2AV

ACTION : From a rotation matrix, determine the corresponding axial vector (single precision).

CALL : `CALL sla_M2AV (RMAT, AXVEC)`

GIVEN :

RMAT	**R(3,3)**	rotation matrix

RETURNED :

AXVEC	**R(3)**	axial vector (radians)

NOTES :

1. A rotation matrix describes a rotation about some arbitrary axis. The axis is called the *Euler axis*, and the angle through which the reference frame rotates is called the *Euler angle*. The *axial vector* returned by this routine has the same direction as the Euler axis, and its magnitude is the Euler angle in radians.
2. The magnitude and direction of the axial vector can be separated by means of the routine sla_VN.
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.

SLA_MAP — Mean to Apparent — SLA_MAP

ACTION : Transform star $[\alpha, \delta]$ from mean place to geocentric apparent. The reference frames and timescales used are post IAU 1976.

CALL : `CALL sla_MAP (RM, DM, PR, PD, PX, RV, EQ, DATE, RA, DA)`

GIVEN :

RM,DM	**D**	mean $[\alpha, \delta]$ (radians)
PR,PD	**D**	proper motions: $[\alpha, \delta]$ changes per Julian year
PX	**D**	parallax (arcsec)
RV	**D**	radial velocity (km s^{-1}, +ve if receding)
EQ	**D**	epoch and equinox of star data (Julian)
DATE	**D**	TDB for apparent place (JD−2400000.5)

RETURNED :

RA,DA	**D**	apparent $[\alpha, \delta]$ (radians)

NOTES :

1. EQ is the Julian epoch specifying both the reference frame and the epoch of the position – usually 2000. For positions where the epoch and equinox are different, use the routine sla_PM to apply proper motion corrections before using this routine.
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 routine may be wasteful for some applications because it recomputes the Earth position/velocity and the precession/nutation matrix each time, and because it allows for parallax and proper motion. Where multiple transformations are to be carried out for one epoch, a faster method is to call the sla_MAPPA routine once and then either the sla_MAPQK routine (which includes parallax and proper motion) or sla_MAPQKZ (which assumes zero parallax and FK5 proper motion).

REFERENCES :

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

SLA_MAPPA Mean to Apparent Parameters SLA_MAPPA

ACTION : Compute star-independent parameters in preparation for conversions between mean place and geocentric apparent place. The parameters produced by this routine are required in the parallax, light deflection, aberration, and precession/nutation parts of the mean/apparent transformations. The reference frames and timescales used are post IAU 1976.

CALL : `CALL sla_MAPPA (EQ, DATE, AMPRMS)`

GIVEN :

EQ	**D**	epoch of mean equinox to be used (Julian)
DATE	**D**	TDB (JD−2400000.5)

RETURNED :

AMPRMS	**D(21)**	star-independent mean-to-apparent parameters:
(1)		time interval for proper motion (Julian years)
(2-4)		barycentric position of the Earth (AU)
(5-7)		heliocentric direction of the Earth (unit vector)
(8)		(gravitational radius of Sun)×2/(Sun-Earth distance)
(9-11)		$\mathbf{v}$: barycentric Earth velocity in units of c
(12)		$\sqrt{1-\lvert\mathbf{v}\rvert^2}$
(13-21)		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 accuracy of the routines using the parameters AMPRMS is limited by the routine sla_EVP, 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.
3. The vectors AMPRMS(2-4) and AMPRMS(5-7) are referred to the mean equinox and equator of epoch EQ.
4. The parameters produced by this routine are used by sla_MAPQK and sla_MAPQKZ.

REFERENCES :

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

SLA_MAPQK — Quick Mean to Apparent — SLA_MAPQK

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 frames and timescales used are post IAU 1976.

CALL : `CALL sla_MAPQK (RM, DM, PR, PD, PX, RV, AMPRMS, RA, DA)`

GIVEN :

RM,DM	**D**	mean $[\,\alpha, \delta\,]$ (radians)
PR,PD	**D**	proper motions: $[\,\alpha, \delta\,]$ changes per Julian year
PX	**D**	parallax (arcsec)
RV	**D**	radial velocity (km s^{-1}, +ve if receding)
AMPRMS	**D(21)**	star-independent mean-to-apparent parameters:
(1)		time interval for proper motion (Julian years)
(2-4)		barycentric position of the Earth (AU)
(5-7)		heliocentric direction of the Earth (unit vector)
(8)		(gravitational radius of Sun)$\times$2/(Sun-Earth distance)
(9-11)		**v**: barycentric Earth velocity in units of c
(12)		$\sqrt{1-|\mathbf{v}|^2}$
(13-21)		precession/nutation 3×3 matrix

RETURNED :

RA,DA	**D**	apparent $[\,\alpha, \delta\,]$ (radians)

NOTES :

1. Use of this routine is appropriate when efficiency is important and where many star positions, all referred to the same equator and equinox, are to be transformed for one epoch. The star-independent parameters can be obtained by calling the sla_MAPPA routine.
2. If the parallax and proper motions are zero the sla_MAPQKZ routine can be used instead.
3. The vectors AMPRMS(2-4) and AMPRMS(5-7) are referred to the mean equinox and equator of epoch EQ.
4. Strictly speaking, the routine is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about $920''$ 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.

SLA_MAPQKZ Quick Mean-Appt, no PM *etc.* SLA_MAPQKZ

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 frames and timescales used are post IAU 1976.

CALL : `CALL sla_MAPQKZ (RM, DM, AMPRMS, RA, DA)`

GIVEN :

RM,DM	**D**	mean $[\alpha, \delta]$ (radians)
AMPRMS	**D(21)**	star-independent mean-to-apparent parameters:
(1)		time interval for proper motion (Julian years)
(2-4)		barycentric position of the Earth (AU)
(5-7)		heliocentric direction of the Earth (unit vector)
(8)		(gravitational radius of Sun)×2/(Sun-Earth distance)
(9-11)		**v**: barycentric Earth velocity in units of c
(12)		$\sqrt{1 - \lvert\mathbf{v}\rvert^2}$
(13-21)		precession/nutation 3×3 matrix

RETURNED :

RA,DA	**D**	apparent $[\alpha, \delta]$ (radians)

NOTES :

1. Use of this routine 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 sla_MAPPA routine.
2. The corresponding routine for the case of non-zero parallax and FK5 proper motion is sla_MAPQK.
3. The vectors AMPRMS(2-4) and AMPRMS(5-7) are referred to the mean equinox and equator of epoch EQ.
4. Strictly speaking, the routine is not valid for solar-system sources, though the error will usually be extremely small. However, to prevent gross errors in the case where the position of the Sun is specified, the gravitational deflection term is restrained within about $920''$ 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.

SLA_MOON — Approx Moon Pos/Vel — SLA_MOON

ACTION : Approximate geocentric position and velocity of the Moon (single precision).

CALL : `CALL sla_MOON (IY, ID, FD, PV)`

GIVEN :

IY	**I**	year
ID	**I**	day in year (1 = Jan 1st)
FD	**R**	fraction of day

RETURNED :

PV	**R(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 (loosely ET) 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 sla_CALYD or sla_CLYD.
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 socond 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 routine is principally intended for computing topocentric direction.
3. This routine 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 sla_DMOON).

REFERENCE : Meeus, *l'Astronomie*, June 1984, p348.

SLA_MXM — Multiply 3×3 Matrices — SLA_MXM

ACTION : Product of two 3×3 matrices (single precision).

CALL : `CALL sla_MXM (A, B, C)`

GIVEN :

A	**R(3,3)**	matrix **A**
B	**R(3,3)**	matrix **B**

RETURNED :

C	**R(3,3)**	matrix result: $\mathbf{A}\times\mathbf{B}$

NOTE : To comply with the ANSI Fortran 77 standard, A, B and C must be different arrays. The routine is, in fact, coded so as to work properly on the VAX and many other systems even if this rule is violated, something that is **not**, however, recommended.

SLA_MXV — Apply 3D Rotation — SLA_MXV

ACTION : Multiply a 3-vector by a rotation matrix (single precision).

CALL : `CALL sla_MXV (RM, VA, VB)`

GIVEN :

RM	**R(3,3)**	rotation matrix
VA	**R(3)**	vector to be rotated

RETURNED :

VB	**R(3)**	result vector

NOTES :

1. This routine performs the operation:

   $\mathbf{b} = \mathbf{M}\cdot\mathbf{a}$

   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 routine is apply a rotation; under these circumstances, **M** is a *proper real orthogonal* matrix.
3. To comply with the ANSI Fortran 77 standard, VA and VB must **not** be the same array. The routine is, in fact, coded so as to work properly on the VAX and many other systems even if this rule is violated, something that is **not**, however, recommended.

SLA_NUT — Nutation Matrix — SLA_NUT

ACTION : Form the matrix of nutation (IAU 1980 theory) for a given date.

CALL : `CALL sla_NUT (DATE, RMATN)`

GIVEN :

DATE	**D**	TDB (formerly ET) as Modified Julian Date (JD−2400000.5)

RETURNED :

RMATN	**D(3,3)**	nutation matrix

NOTE : The matrix is in the sense:

$$\mathbf{v}_{true} = \mathbf{M}\cdot\mathbf{v}_{mean}$$

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.

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.

SLA_NUTC Nutation Components SLA_NUTC

ACTION : Nutation (IAU 1980 theory): longitude & obliquity components, and mean obliquity.

CALL : `CALL sla_NUTC (DATE, DPSI, DEPS, EPS0)`

GIVEN :

DATE	**D**	TDB (formerly ET) as Modified Julian Date (JD−2400000.5)

RETURNED :

DPSI,DEPS	**D**	nutation in longitude and obliquity (radians)
EPS0	**D**	mean obliquity (radians)

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.

SLA_OAP — Observed to Apparent — SLA_OAP

ACTION : Observed to apparent place.

CALL :

```
CALL sla_OAP (TYPE, OB1, OB2, DATE, DUT, ELONGM, PHIM,
              HM, XP, YP, TDK, PMB, RH, WL, TLR, RAP, DAP)
```

GIVEN :

TYPE	**C*(*)**	type of coordinates – 'R', 'H' or 'A' (see below)
OB1	**D**	observed Az, HA or RA (radians; Az is N=0, E=90°)
OB2	**D**	observed zenith distance or δ (radians)
DATE	**D**	UTC date/time (Modified Julian Date, JD−2400000.5)
DUT	**D**	ΔUT: UT1−UTC (UTC seconds)
ELONGM	**D**	observer's mean longitude (radians, east +ve)
PHIM	**D**	observer's mean geodetic latitude (radians)
HM	**D**	observer's height above sea level (metres)
XP,YP	**D**	polar motion $[x, y]$ coordinates (radians)
TDK	**D**	local ambient temperature (degrees K; std=273.155D0)
PMB	**D**	local atmospheric pressure (mB; std=1013.25D0)
RH	**D**	local relative humidity (in the range 0D0 – 1D0)
WL	**D**	effective wavelength (μm, *e.g.* 0.55D0)
TLR	**D**	tropospheric lapse rate (degrees K per metre, *e.g.* 0.0065D0)

RETURNED :

RAP,DAP	**D**	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 +ve) and Declination; anything else ('A' or 'a' is recommended) indicates that OBS1 and OBS2 are Azimuth (north zero, east is 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 be better than 1 arcminute; useful results are available for a further 3°, beyond which the sla_REFRO routine returns a fixed value of the refraction. The complementary routines sla_AOP (or sla_AOPQK) and sla_OAP (or sla_OAPQK) 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 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 sla_AMP *etc.* routines will convert the apparent $[\,\alpha, \delta\,]$ produced by the present routine into an FK5 J2000 mean place, by allowing for the Sun's gravitational lens effect, annual aberration, nutation and precession. Should FK4 B1950 coordinates be needed, the routines sla_FK524 *etc.* will also need 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. The star-independent apparent-to-observed-place parameters in AOPRMS may be computed by means of the sla_AOPPA routine. If nothing has changed significantly except the time, the sla_AOPPAT routine may be used to perform the requisite partial recomputation of AOPRMS.
8. The DATE 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 $\pm 0^{\rm s}.9$.
10. IMPORTANT – TAKE CARE WITH THE LONGITUDE SIGN CONVENTION. The longitude required by the present routine is **east-positive**, in accordance with

geographical convention (and right-handed). In particular, note that the longitudes returned by the sla_OBS routine are west-positive (as in the *Astronomical Almanac* before 1984) and must be reversed in sign before use in the present routine.

11. The polar coordinates XP,YP can be obtained from IERS circulars and equivalent publications. The maximum amplitude is about $0\overset{''}{.}3$. If XP,YP values are unavailable, use XP=YP=0D0. See page B60 of the 1988 *Astronomical Almanac* for a definition of the two angles.

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 routine sla_OBS. If P, the pressure in mB, is available, an adequate estimate of HM can be obtained from the following expression:

    ```
    HM=-29.3D0*TSL*LOG(P/1013.25D0)
    ```

    where TSL is the approximate sea-level air temperature in degrees 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=1013.25D0*EXP(-HM/(29.3D0*TSL))
    ```

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

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

SLA_OAPQK Quick Observed to Apparent SLA_OAPQK

ACTION : Quick observed to apparent place.

CALL : `CALL sla_OAPQK (TYPE, OB1, OB2, AOPRMS, RAP, DAP)`

GIVEN :

TYPE	**C*(*)**	type of coordinates – 'R', 'H' or 'A' (see below)
OB1	**D**	observed Az, HA or RA (radians; Az is N=0, E=90°)
OB2	**D**	observed zenith distance or δ (radians)
AOPRMS	**D(14)**	star-independent apparent-to-observed parameters:
(1)		geodetic latitude (radians)
(2,3)		sine and cosine of geodetic latitude
(4)		magnitude of diurnal aberration vector
(5)		height (HM)
(6)		ambient temperature (TDK)
(7)		pressure (PMB)
(8)		relative humidity (RH)
(9)		wavelength (WL)
(10)		lapse rate (TLR)
(11,12)		refraction constants A and B (radians)
(13)		longitude + eqn of equinoxes + "sidereal ΔUT" (radians)
(14)		local apparent sidereal time (radians)

RETURNED :

RAP,DAP	**D**	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 +ve) and Declination; anything else ('A' or 'a' is recommended) indicates that OBS1 and OBS2 are Azimuth (north zero, east is 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 $0\farcs1$ for $\zeta < 70°$. Even at a topocentric zenith distance of 90°, the accuracy in elevation should be better than 1 arcminute; useful results are available for a further 3°, beyond which the sla_REFRO routine returns a fixed value of the refraction. The complementary routines sla_AOP (or sla_AOPQK) and sla_OAP (or sla_OAPQK) 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 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 sla_AMP *etc.* routines will convert the apparent $[\alpha, \delta]$ produced by the present routine into an FK5 J2000 mean place, by allowing for the Sun's gravitational lens effect, annual aberration, nutation and precession. Should FK4 B1950 coordinates be needed, the routines sla_FK524 *etc.* will also need 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. The star-independent apparent-to-observed-place parameters in AOPRMS may be computed by means of the sla_AOPPA routine. If nothing has changed significantly except the time, the sla_AOPPAT routine may be used to perform the requisite partial recomputation of AOPRMS.
8. The azimuths *etc.* used by the present routine are with respect to the celestial pole. Corrections from the terrestrial pole can be computed using sla_POLMO.

SLA_OBS — Observatory Parameters — SLA_OBS

ACTION : Look up an entry in a standard list of groundbased observing stations parameters.

CALL : `CALL sla_OBS (N, C, NAME, W, P, H)`

GIVEN :

N	**I**	number specifying observing station

GIVEN or RETURNED :

C	**C*(*)**	identifier specifying observing station

RETURNED :

NAME	**C*(*)**	name of specified observing station
W	**D**	longitude (radians, west +ve)
P	**D**	geodetic latitude (radians, north +ve)
H	**D**	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 Nth 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 routine 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 – BEWARE OF THE LONGITUDE SIGN CONVENTION. The longitude returned by sla_OBS 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 sla_AOP, sla_AOPPA and sla_OAP the sign of the longitude must be reversed.
8. Users are urged to inform the author of any improvements they would like to see made. For example:
   - typographical corrections
   - more accurate parameters
   - better station identifiers or names
   - additional stations

Stations supported by sla_OBS at the time of writing:

ID	*NAME*
AAT	Anglo-Australian 3.9m Telescope
ANU2.3	Siding Spring 2.3 metre
APO3.5	Apache Point 3.5m
ARECIBO	Arecibo 1000 foot
ATCA	Australia Telescope Compact Array
BLOEMF	Bloemfontein 1.52 metre

BOSQALEGRE	Bosque Alegre 1.54 metre
CAMB1MILE	Cambridge 1 mile
CAMB5KM	Cambridge 5km
CATALINA61	Catalina 61 inch
CFHT	Canada-France-Hawaii 3.6m Telescope
CSO	Caltech Sub-mm Observatory, Mauna Kea
DAO72	DAO Victoria BC 1.85 metre
DUNLAP74	David Dunlap 74 inch
DUPONT	Du Pont 2.5m Telescope, Las Campanas
EFFELSBERG	Effelsberg 100 metre
ESO3.6	ESO 3.6 metre
ESONTT	ESO 3.5 metre NTT
ESOSCHM	ESO 1 metre 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 8-m telescope
HARVARD	Harvard College Observatory 1.55m
HPROV1.52	Haute Provence 1.52 metre
HPROV1.93	Haute Provence 1.93 metre
IRTF	NASA IR Telescope Facility, Mauna Kea
JCMT	JCMT 15 metre
JODRELL1	Jodrell Bank 250 foot
KECK1	Keck 10m Telescope 1
KECK2	Keck 10m Telescope 2
KISO	Kiso 1.05 metre Schmidt, Japan
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
LPO1	Jacobus Kapteyn 1m Telescope
LPO2.5	Isaac Newton 2.5m Telescope
LPO4.2	William Herschel 4.2m Telescope
MAUNAK88	Mauna Kea 88 inch
MCDONLD2.1	McDonald 2.1 metre
MCDONLD2.7	McDonald 2.7 metre
MMT	MMT, Mt Hopkins
MOPRA	ATNF Mopra Observatory
MTEKAR	Mt Ekar 1.82 metre
MTHOP1.5	Mt Hopkins 1.5 metre
MTLEMMON60	Mt Lemmon 60 inch
NOBEYAMA	Nobeyama 45 metre
OKAYAMA	Okayama 1.88 metre
PALOMAR200	Palomar 200 inch

PALOMAR48	Palomar 48-inch Schmidt
PALOMAR60	Palomar 60 inch
PARKES	Parkes 64 metre
QUEBEC1.6	Quebec 1.6 metre
SAAO74	Sutherland 74 inch
SANPM83	San Pedro Martir 83 inch
ST.ANDREWS	St Andrews University Observatory
STEWARD90	Steward 90 inch
STROMLO74	Mount Stromlo 74 inch
SUBARU	Subaru 8 metre
SUGARGROVE	Sugar Grove 150 foot
TAUTNBG	Tautenburg 2 metre
TAUTSCHM	Tautenberg 1.34 metre Schmidt
TIDBINBLA	Tidbinbilla 64 metre
TOLOLO1.5M	Cerro Tololo 1.5 metre
TOLOLO4M	Cerro Tololo 4 metre
UKIRT	UK Infra Red Telescope
UKST	UK 1.2 metre Schmidt, Siding Spring
USSR6	USSR 6 metre
USSR600	USSR 600 foot
VLA	Very Large Array

SLA_PA — h, δ to Parallactic Angle — SLA_PA

ACTION : Hour angle and declination to parallactic angle (double precision).

CALL : `D = sla_PA (HA, DEC, PHI)`

GIVEN :

HA	**D**	hour angle in radians (geocentric apparent)
DEC	**D**	declination in radians (geocentric apparent)
PHI	**D**	latitude in radians (geodetic)

RETURNED :

sla_PA	**D**	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.

SLA_PAV Position-Angle Between Two Directions SLA_PAV

ACTION : Returns the bearing (position angle) of one celestial direction with respect to another (single precision).

CALL : `R = sla_PAV (V1, V2)`

GIVEN :

V1	**R(3)**	direction cosines of one point
V2	**R(3)**	directions cosines of the other point

RETURNED :

sla_PAV	**R**	position-angle of 2nd point with respect to 1st

NOTES :

1. The coordinate frames correspond to $[\,\alpha,\delta\,]$, $[\lambda,\phi]$ *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. The routine sla_BEAR performs an equivalent function except that the points are specified in the form of spherical coordinates.

SLA_PCD Apply Radial Distortion SLA_PCD

ACTION : Apply pincushion/barrel distortion to a tangent-plane $[\,x,y\,]$.

CALL : `CALL sla_PCD (DISCO,X,Y)`

GIVEN :

DISCO	**D**	pincushion/barrel distortion coefficient
X,Y	**D**	tangent-plane $[\,x,y\,]$

RETURNED :

X,Y	**D**	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 +ve; for *barrel* distortion, C is −ve.
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 routine, sla_UNPCD, which performs an approximately inverse operation.

SLA_PDA2H H.A. for a Given Azimuth SLA_PDA2H

ACTION : Hour Angle corresponding to a given azimuth (double precision).

CALL : `CALL sla_PDA2H (P, D, A, H1, J1, H2, J2)`

GIVEN :

P	**D**	latitude
D	**D**	declination
A	**D**	azimuth

RETURNED :

H1	**D**	hour angle: first solution if any
J1	**I**	flag: 0 = solution 1 is valid
H2	**D**	hour angle: second solution if any
J2	**I**	flag: 0 = solution 2 is valid

SLA_PDQ2H H.A. for a Given P.A. SLA_PDQ2H

ACTION : Hour Angle corresponding to a given parallactic angle (double precision).

CALL : `CALL sla_PDQ2H (P, D, Q, H1, J1, H2, J2)`

GIVEN :

P	**D**	latitude
D	**D**	declination
Q	**D**	azimuth

RETURNED :

H1	**D**	hour angle: first solution if any
J1	**I**	flag: 0 = solution 1 is valid
H2	**D**	hour angle: second solution if any
J2	**I**	flag: 0 = solution 2 is valid

SLA_PERMUT Next Permutation SLA_PERMUT

ACTION : Generate the next permutation of a specified number of items.

CALL : `CALL sla_PERMUT (N, ISTATE, IORDER, J)`

GIVEN :

N	**I**	number of items: there will be N! permutations
ISTATE	**I(N)**	state, ISTATE(1)= −1 to initialize

RETURNED :

ISTATE	**I(N)**	state, updated ready for next time
IORDER	**I(N)**	next permutation of numbers 1,2,...,N
J	**I**	status: −1 = illegal N (zero or less is illegal) 0 = OK +1 = no more permutations available

NOTES :

1. This routine 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 routine 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:

   IORDER(1)=N, IORDER(2)=N-1, ..., IORDER(N)=1

   This is also the permutation returned for the "finished" (J=1) case. The final permutation to be generated is:

   IORDER(1)=1, IORDER(2)=2, ..., IORDER(N)=N

3. If the "finished" (J=1) status is ignored, the routine continues to deliver permutations, the pattern repeating every N! calls.

SLA_PERTEL Perturbed Orbital Elements SLA_PERTEL

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

CALL :
```
CALL sla_PERTEL (JFORM, DATE0, DATE1,
                 EPOCH0, ORBI0, ANODE0, PERIH0, AORQ0, E0, AM0,
                 EPOCH1, ORBI1, ANODE1, PERIH1, AORQ1, E1, AM1,
                 JSTAT)
```

GIVEN (format and dates) :

JFORM	**I**	choice of element set (2 or 3; Note 1)
DATE0	**D**	date of osculation (TT MJD) for the given elements
DATE1	**D**	date of osculation (TT MJD) for the updated elements

GIVEN (the unperturbed elements) :

EPOCH0	**D**	epoch of the given element set (t_0 or T, TT MJD; Note 2)
ORBI0	**D**	inclination (i, radians)
ANODE0	**D**	longitude of the ascending node (Ω, radians)
PERIH0	**D**	argument of perihelion (ω, radians)
AORQ0	**D**	mean distance or perihelion distance (a or q, AU)
E0	**D**	eccentricity (e)
AM0	**D**	mean anomaly (M, radians, JFORM=2 only)

RETURNED (the updated elements) :

EPOCH1	**D**	epoch of the updated element set (t_0 or T, TT MJD; Note 2)
ORBI1	**D**	inclination (i, radians)
ANODE1	**D**	longitude of the ascending node (Ω, radians)
PERIH1	**D**	argument of perihelion (ω, radians)
AORQ1	**D**	mean distance or perihelion distance (a or q, AU)
E1	**D**	eccentricity (e)
AM1	**D**	mean anomaly (M, radians, JFORM=2 only)

RETURNED (status flag) :

JSTAT	**I**	status:
		+102 = warning, distant epoch
		+101 = warning, large timespan (> 100 years)
		+1 to +8 = 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 (TT 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 (TT 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 TT timescale (formerly Ephemeris Time, ET), expressed as Modified Julian Dates (JD−2400000.5).
   - DATE0 is the instant at which the given (*i.e.* unperturbed) osculating elements are correct.
   - DATE1 is the specified instant at which the updated osculating elements are correct.
   - EPOCH0 and EPOCH1 will be the same as DATE0 and DATE1 (respectively) for the JFORM=2 case, normally used for minor planets. For the JFORM=3 case, the two epochs will refer to perihelion passage and so will not, in general, be the same as DATE0 and/or DATE1 though they may be similar to one another.
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 sla_PERTUE routine for details of the algorithm used.
6. This routine is not intended to be used for major planets, which is why JFORM=1 is not available and why there is no opportunity to specify either the longitude of perihelion or the daily motion. However, if JFORM=2 elements are somehow obtained for a major planet and supplied to the routine, sensible results will, in fact, be produced. This happens because the sla_PERTUE routine that is called to perform the calculations checks the separation between the body and each of the planets and interprets a suspiciously small value (0.001 AU) as an attempt to apply it to the planet concerned. If this condition is detected, the contribution from that planet is ignored, and the status is set to the planet number (Mercury=1,...,Neptune=8) as a warning.

REFERENCE : Sterne, Theodore E., *An Introduction to Celestial Mechanics,* Interscience Publishers, 1960. Section 6.7, p199.

SLA_PERTUE Perturbed Universal Elements SLA_PERTUE

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

CALL : `CALL sla_PERTUE (DATE, U, JSTAT)`

GIVEN :

DATE1	**D**	final epoch (TT MJD) for the updated elements

GIVEN and RETURNED :

U	**D(13)**	universal elements (updated in place)
(1)		combined mass ($M + m$)
(2)		total energy of the orbit (α)
(3)		reference (osculating) epoch (t_0)
(4-6)		position at reference epoch ($\mathbf{r}_0$)
(7-9)		velocity at reference epoch ($\mathbf{v}_0$)
(10)		heliocentric distance at reference epoch
(11)		$\mathbf{r}_0.\mathbf{v_0}$
(12)		date (t)
(13)		universal eccentric anomaly (ψ) of date, approx

RETURNED :

JSTAT	**I**	status:
		+102 = warning, distant epoch
		+101 = warning, large timespan (> 100 years)
		+1 to +8 = 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(3) and U(12) are all 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 and Pitkin (reference 2) suggest rectifying the orbit at each integration step by propagating the new perturbed position and velocity as the new universal variables. In the present routine the orbit is rectified less frequently than this, in order to gain a slight speed advantage. However, the rectification is done directly in terms of position and velocity, as suggested by Everhart and Pitkin, bypassing the use of conventional orbital elements.

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

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

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

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

   The numerical integration requires estimates of the major-planet motions. Approximate positions for the major planets (Pluto alone is omitted) are obtained from the routine sla_PLANET. Two levels of interpolation are used, to enhance speed without significantly degrading accuracy. At a low frequency, the routine sla_PLANET 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, relativistic effects are not taken into account and the Earth-Moon system is treated as a single body.

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.

5. This routine is not intended to be used for major planets. However, if major-planet elements are supplied, sensible results will, in fact, be produced. This happens because the routine checks the separation between the body and each of the planets and interprets a suspiciously small value (0.001 AU) as an attempt to apply the routine to the planet concerned. If this condition is detected, the contribution from that planet is ignored, and the status is set to the planet number (Mercury=1,...,Neptune=8) as a warning.

REFERENCES :

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

SLA_PLANEL Planet Position from Elements SLA_PLANEL

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

CALL :
```
CALL sla_PLANEL (DATE, JFORM, EPOCH, ORBINC, ANODE, PERIH,
                 AORQ, E, AORL, DM, PV, JSTAT)
```

GIVEN :

DATE	**D**	Modified Julian Date (JD−2400000.5)
JFORM	**I**	choice of element set (1-3, see Note 3, below)
EPOCH	**D**	epoch of elements (t_0 or T, TT MJD)
ORBINC	**D**	inclination (i, radians)
ANODE	**D**	longitude of the ascending node (Ω, radians)
PERIH	**D**	longitude or argument of perihelion (ϖ or ω, radians)
AORQ	**D**	mean distance or perihelion distance (a or q, AU)
E	**D**	eccentricity (e)
AORL	**D**	mean anomaly or longitude (M or L, radians, JFORM=1,2 only)
DM	**D**	daily motion (n, radians, JFORM=1 only)

RETURNED :

PV	**D(6)**	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s)
JSTAT	**I**	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 TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD−2400000.5).
2. The elements are with respect to the J2000 ecliptic and equinox.
3. Three different element-format options are available, as follows.

   JFORM=1, suitable for the major planets:

EPOCH	=	epoch of elements t_0 (TT 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 (TT 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 (TT 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 ($0 \leq e \leq 10$)

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

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

SLA_PLANET Planetary Ephemerides SLA_PLANET

ACTION : Approximate heliocentric position and velocity of a planet.

CALL : `CALL sla_PLANET (DATE, NP, PV, JSTAT)`

GIVEN :

DATE	**D**	Modified Julian Date (JD−2400000.5)
NP	**I**	planet: 1 = Mercury 2 = Venus 3 = Earth-Moon Barycentre 4 = Mars 5 = Jupiter 6 = Saturn 7 = Uranus 8 = Neptune 9 = Pluto

RETURNED :

PV	**D(6)**	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s)
JSTAT	**I**	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 timescale and is in the form of a Modified Julian Date (JD−2400000.5).
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 from 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 Pluto is by Meeus.
5. Comparisons of the present routine with the JPL DE200 ephemeris give the following RMS errors over the interval 1960-2025:

	position (km)	*speed (metre/sec)*
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 (MJD = (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. NP = 3 the result is for the Earth-Moon Barycentre. To obtain the heliocentric position and velocity of the Earth, either use the SLALIB routine sla_EVP or call sla_DMOON and subtract 0.012150581 times the geocentric Moon vector from the EMB vector produced by the present routine. (The Moon vector should be precessed to J2000 first, but this can be omitted for modern epochs without introducing significant inaccuracy.)

REFERENCES :

1. Simon *et al.,* Astron. Astrophys. **282**, 663 (1994).
2. Meeus, J., *Astronomical Algorithms,* Willmann-Bell (1991).

SLA_PLANTE $[\alpha, \delta]$ of Planet from Elements SLA_PLANTE

ACTION : Topocentric apparent $[\alpha, \delta]$ of a Solar-System object whose heliocentric orbital elements are known.

CALL :
```
CALL sla_PLANTE (DATE, ELONG, PHI, JFORM, EPOCH, ORBINC, ANODE, PERIH,
                 AORQ, E, AORL, DM, RA, DEC, R, JSTAT)
```

GIVEN :

DATE	**D**	MJD of observation (JD−2400000.5)
ELONG,PHI	**D**	observer's longitude (east +ve) and latitude radians)
JFORM	**I**	choice of element set (1-3, see Note 4, below)
EPOCH	**D**	epoch of elements (t_0 or T, TT MJD)
ORBINC	**D**	inclination (i, radians)
ANODE	**D**	longitude of the ascending node (Ω, radians)
PERIH	**D**	longitude or argument of perihelion (ϖ or ω, radians)
AORQ	**D**	mean distance or perihelion distance (a or q, AU)
E	**D**	eccentricity (e)
AORL	**D**	mean anomaly or longitude (M or L, radians, JFORM=1,2 only)
DM	**D**	daily motion (n, radians, JFORM=1 only)

RETURNED :

RA,DEC	**D**	topocentric apparent $[\,\alpha,\delta\,]$ (radians)
R	**D**	distance from observer (AU)
JSTAT	**I**	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 TT timescale (formerly Ephemeris Time, ET) 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 Earth-crossing asteroids. Geocentric positions can be generated by appropriate use of the routines sla_EVP and sla_PLANEL.
3. The elements are with respect to the J2000 ecliptic and equinox.
4. Three different element-format options are available, as follows.

   JFORM=1, suitable for the major planets:

EPOCH	=	epoch of elements t_0 (TT 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 (TT 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 (TT 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

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

SLA_PM — Proper Motion — SLA_PM

ACTION : Apply corrections for proper motion to a star [α, δ].

CALL : `CALL sla_PM (R0, D0, PR, PD, PX, RV, EP0, EP1, R1, D1)`

GIVEN :

R0,D0	**D**	[α, δ] at epoch EP0 (radians)
PR,PD	**D**	proper motions: rate of change of [α, δ] (radians per year)
PX	**D**	parallax (arcsec)
RV	**D**	radial velocity (km s^{-1}, +ve if receding)
EP0	**D**	start epoch in years (*e.g.* Julian epoch)
EP1	**D**	end epoch in years (same system as EP0)

RETURNED :

R1,D1	**D**	[α, δ] at epoch EP1 (radians)

NOTE : The α proper motions are $\dot{\alpha}$ rather than $\dot{\alpha}\cos\delta$, and are in the same coordinate system as R0,D0.

REFERENCES :

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

SLA_POLMO — Polar Motion — SLA_POLMO

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

CALL : `CALL sla_POLMO (ELONGM, PHIM, XP, YP, ELONG, PHI, DAZ)`

GIVEN :

ELONGM	**D**	mean longitude of the site (radians, east +ve)
PHIM	**D**	mean geodetic latitude of the site (radians)
XP	**D**	polar motion x-coordinate (radians)
YP	**D**	polar motion y-coordinate (radians)

RETURNED :

ELONG	**D**	true longitude of the site (radians, east +ve)
PHI	**D**	true geodetic latitude of the site (radians)
DAZ	**D**	azimuth correction (terrestrial−celestial, radians)

NOTES :

1. "Mean" longitude and latitude are the (fixed) values for the site's location with respect to the IERS terrestrial reference frame; the latitude is geodetic. TAKE CARE WITH THE LONGITUDE SIGN CONVENTION. The longitudes used by the present routine are east-positive, in accordance with geographical convention (and right-handed). In particular, note that the longitudes returned by the sla_OBS routine are west-positive, following astronomical usage, and must be reversed in sign before use in the present routine.
2. XP and YP are the (changing) coordinates of the Celestial Ephemeris Pole with respect to the IERS Reference Pole. XP is positive along the meridian at longitude 0°, 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 $0\rlap{.}''3$.
3. "True" longitude and latitude are the (moving) values for the site's location with respect to the celestial ephemeris pole and the meridian which corresponds to the Greenwich apparent sidereal time. The true longitude and latitude link the terrestrial coordinates with the standard celestial models (for precession, nutation, sidereal time *etc*).

4. The azimuths produced by sla_AOP and sla_AOPQK are with respect to due north as defined by the Celestial Ephemeris Pole, and can therefore be called "celestial azimuths". However, a telescope fixed to the Earth measures azimuth essentially with respect to due north as defined by the IERS Reference Pole, and can therefore be called "terrestrial azimuth". Uncorrected, this would manifest itself as a changing "azimuth zero-point error". The value DAZ is the correction to be added to a celestial azimuth to produce a terrestrial azimuth.
5. The present routine is rigorous. For most practical purposes, the following simplified formulae provide an adequate approximation:

```
ELONG  =  ELONGM+XP*COS(ELONGM)-YP*SIN(ELONGM)
PHI    =  PHIM+(XP*SIN(ELONGM)+YP*COS(ELONGM))*TAN(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)
```

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

SLA_PREBN — Precession Matrix (FK4) — SLA_PREBN

ACTION : Generate the matrix of precession between two epochs, using the old, pre IAU 1976, Bessel-Newcomb model, in Andoyer's formulation.

CALL : `CALL sla_PREBN (BEP0, BEP1, RMATP)`

GIVEN :

BEP0	**D**	beginning Besselian epoch
BEP1	**D**	ending Besselian epoch

RETURNED :

RMATP	**D(3,3)**	precession matrix

NOTE : The matrix is in the sense:

$$\mathbf{v}_1 = \mathbf{M}\cdot\mathbf{v}_0$$

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.

SLA_PREC — Precession Matrix (FK5) — SLA_PREC

ACTION : Form the matrix of precession between two epochs (IAU 1976, FK5).

CALL : `CALL sla_PREC (EP0, EP1, RMATP)`

GIVEN :

EP0	**D**	beginning epoch
EP1	**D**	ending epoch

RETURNED :

RMATP	**D(3,3)**	precession matrix

NOTES :

1. The epochs are TDB Julian epochs.
2. The matrix is in the sense:

   $\mathbf{v}_1 = \mathbf{M}\cdot\mathbf{v}_0$

   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 known errors in the IAU precession rate. The absolute accuracy of the present formulation is better than $0\farcs1$ 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 routine sla_PRECL 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.

SLA_PRECES — Precession — SLA_PRECES

ACTION : Precession – either the old "FK4" (Bessel-Newcomb, pre IAU 1976) or new "FK5" (Fricke, post IAU 1976) as required.

CALL : `CALL sla_PRECES (SYSTEM, EP0, EP1, RA, DC)`

GIVEN :

SYSTEM	**C**	precession to be applied: 'FK4' or 'FK5'
EP0,EP1	**D**	starting and ending epoch
RA,DC	**D**	$[\alpha,\delta]$, mean equator & equinox of epoch EP0

RETURNED :

RA,DC	**D**	$[\alpha,\delta]$, mean equator & equinox of epoch EP1

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:

   ```
   CALL sla_PRECES ('FK4', 1900D0, 1950D0, RA, DC)
   ```
3. This routine will **NOT** correctly convert between the old and the new systems – for example conversion from B1950 to J2000. For these purposes see sla_FK425, sla_FK524, sla_FK45Z and sla_FK54Z.
4. If an invalid SYSTEM is supplied, values of −99D0,−99D0 are returned for both RA and DC.

SLA_PRECL Precession Matrix (latest) SLA_PRECL

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 : `CALL sla_PRECL (EP0, EP1, RMATP)`

GIVEN :

EP0	**D**	beginning epoch
EP1	**D**	ending epoch

RETURNED :

RMATP	**D(3,3)**	precession matrix

NOTES :

1. The epochs are TDB Julian epochs.
2. The matrix is in the sense:

   $\mathbf{v}_1 = \mathbf{M}\cdot\mathbf{v}_0$

   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.

SLA_PRENUT Precession/Nutation Matrix SLA_PRENUT

ACTION : Form the matrix of precession and nutation (IAU 1976, FK5).

CALL : `CALL sla_PRENUT (EPOCH, DATE, RMATPN)`

GIVEN :

EPOCH	**D**	Julian Epoch for mean coordinates
DATE	**D**	Modified Julian Date (JD−2400000.5) for true coordinates

RETURNED :

RMATPN	**D(3,3)**	combined precession/nutation matrix

NOTES :

1. The epoch and date are TDB.
2. The matrix is in the sense:

   $\mathbf{v}_{true} = \mathbf{M}\cdot\mathbf{v}_{mean}$

   where $\mathbf{v}_{true}$ is the star vector relative to the true equator and equinox of epoch 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 EPOCH.

SLA_PV2EL Orbital Elements from Position/Velocity SLA_PV2EL

ACTION : Heliocentric osculating elements obtained from instantaneous position and velocity.

CALL :
```
CALL sla_PV2EL (PV, DATE, PMASS, JFORMR, JFORM, EPOCH, ORBINC,
                ANODE, PERIH, AORQ, E, AORL, DM, JSTAT)
```

GIVEN :

PV	**D(6)**	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s; Note 1)
DATE	**D**	date (TT Modified Julian Date = JD−2400000.5)
PMASS	**D**	mass of the planet (Sun = 1; Note 2)
JFORMR	**I**	requested element set (1-3; Note 3)

RETURNED :

JFORM	**I**	element set actually returned (1-3; Note 4)
EPOCH	**D**	epoch of elements (t_0 or T, TT MJD)
ORBINC	**D**	inclination (i, radians)
ANODE	**D**	longitude of the ascending node (Ω, radians)
PERIH	**D**	longitude or argument of perihelion (ϖ or ω, radians)
AORQ	**D**	mean distance or perihelion distance (a or q, AU)
E	**D**	eccentricity (e)
AORL	**D**	mean anomaly or longitude (M or L, radians, JFORM=1,2 only)
DM	**D**	daily motion (n, radians, JFORM=1 only)
JSTAT	**I**	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 0D0. 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 (TT 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 $(0 \le e < 1)$
AORL	=	mean longitude L (radians)
DM	=	daily motion n (radians)

   JFORM=2, suitable for minor planets:

EPOCH	=	epoch of elements t_0 (TT 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 (TT 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 $(0 \leq e \leq 10)$

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:

JFORMR	JFORM	meaning
1	1	OK: elements are in the requested format
1	2	never happens
1	3	orbit not elliptical
2	1	never happens
2	2	OK: elements are in the requested format
2	3	orbit not elliptical
3	1	never happens
3	2	never happens
3	3	OK: elements are in the requested format

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

JFORM	1	2	3
EPOCH	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.

REFERENCE : Sterne, Theodore E., *An Introduction to Celestial Mechanics,* Interscience Publishers, 1960.

SLA_PV2UE Position/Velocity to Universal Elements SLA_PV2UE

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

CALL : `CALL sla_PV2UE (PV, DATE, PMASS, U, JSTAT)`

GIVEN :

PV	**D(6)**	heliocentric $[x,y,z,\dot{x},\dot{y},\dot{z}]$, equatorial, J2000 (AU, AU/s; Note 1)
DATE	**D**	date (TT Modified Julian Date = JD−2400000.5)
PMASS	**D**	mass of the planet (Sun = 1; Note 2)

RETURNED :

U	**D(13)**	universal orbital elements (Note 3)
(1)		combined mass ($M+m$)
(2)		total energy of the orbit (α)
(3)		reference (osculating) epoch (t_0)
(4-6)		position at reference epoch ($\mathbf{r}_0$)
(7-9)		velocity at reference epoch ($\mathbf{v}_0$)
(10)		heliocentric distance at reference epoch
(11)		$\mathbf{r}_0.\mathbf{v}_0$
(12)		date (t)
(13)		universal eccentric anomaly (ψ) of date, approx
JSTAT	**I**	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 0D0. 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., Am. J. Phys. 51, 712, 1983.

SLA_PVOBS Observatory Position & Velocity SLA_PVOBS

ACTION : Position and velocity of an observing station.

CALL : `CALL sla_PVOBS (P, H, STL, PV)`

GIVEN :

P	**D**	latitude (geodetic, radians)
H	**D**	height above reference spheroid (geodetic, metres)
STL	**D**	local apparent sidereal time (radians)

RETURNED :

PV	**D(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.

SLA_PXY — Apply Linear Model — SLA_PXY

ACTION : Given arrays of *expected* and *measured* $[x, y]$ coordinates, and a linear model relating them (as produced by sla_FITXY), compute the array of *predicted* coordinates and the RMS residuals.

CALL : `CALL sla_PXY (NP,XYE,XYM,COEFFS,XYP,XRMS,YRMS,RRMS)`

GIVEN :

NP	**I**	number of samples
XYE	**D(2,NP)**	expected $[x, y]$ for each sample
XYM	**D(2,NP)**	measured $[x, y]$ for each sample
COEFFS	**D(6)**	coefficients of model (see below)

RETURNED :

XYP	**D(2,NP)**	predicted $[x, y]$ for each sample
XRMS	**D**	RMS in X
YRMS	**D**	RMS in Y
RRMS	**D**	total RMS (vector sum of XRMS and YRMS)

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 sla_FITXY, sla_INVF, sla_XY2XY, sla_DCMPF

SLA_RANDOM — Random Number — SLA_RANDOM

ACTION : Generate pseudo-random real number in the range $0 \leq x < 1$.

CALL : `R = sla_RANDOM (SEED)`

GIVEN :

SEED	**R**	an arbitrary real number

RETURNED :

SEED	**R**	a new arbitrary value
sla_RANDOM	**R**	Pseudo-random real number $0 \leq x < 1$.

NOTE : The implementation is machine-dependent.

SLA_RANGE — Put Angle into Range $\pm\pi$ — SLA_RANGE

ACTION : Normalize an angle into the range $\pm\pi$ (single precision).

CALL : `R = sla_RANGE (ANGLE)`

GIVEN :

ANGLE	**R**	angle in radians

RETURNED :

sla_RANGE	**R**	ANGLE expressed in the range $\pm\pi$.

SLA_RANORM — Put Angle into Range $0-2\pi$ — SLA_RANORM

ACTION : Normalize an angle into the range $0-2\pi$ (single precision).

CALL : `R = sla_RANORM (ANGLE)`

GIVEN :

ANGLE	**R**	angle in radians

RETURNED :

sla_RANORM	**R**	ANGLE expressed in the range $0-2\pi$

SLA_RCC Barycentric Coordinate Time SLA_RCC

CALL : `D = sla_RCC (TDB, UT1, WL, U, V)`

ACTION : The relativistic clock correction TDB−TT, the difference between *proper time* on Earth and *coordinate time* in the solar system barycentric space-time frame of reference. The proper time is TT; the coordinate time is *an implementation* of TDB.

GIVEN :

TDB	**D**	coordinate time (MJD: JD−2400000.5)
UT1	**D**	universal time (fraction of one day)
WL	**D**	clock longitude (radians west)
U	**D**	clock distance from Earth spin axis (km)
V	**D**	clock distance north of Earth equatorial plane (km)

RETURNED :

sla_RCC	**D**	TDB−TT (sec)

NOTES :

1. TDB may be considered to be the coordinate time in the solar system barycentre frame of reference, and TT is the proper time given by clocks at mean sea level on the Earth.
2. The result has a main (annual) sinusoidal term of amplitude approximately 1.66ms, plus planetary terms up to about 20μs, and lunar and diurnal terms up to 2μs. 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 proper time (TT) can in practice be used.
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 routine, to comply with Starlink programming

standards and to avoid compilation problems on certain machines. On the supported computer types, the numerical results are essentially unaffected by the changes. The topocentric model is from Moyer (1981) and Murray (1983). During the interval 1950-2050, the absolute accuracy of the geocentric model is better than ± 3 nanoseconds relative to direct numerical integrations using the JPL DE200/LE200 solar system ephemeris.

5. The IAU definition of TDB is that it must differ from TT only by periodic terms. Though practical, this is an imprecise definition which 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.

REFERENCES :

1. Fairhead, L. & Bretagnon, P., 1990. *Astr.Astrophys.* **229**, 240-247.
2. Moyer, T.D., 1981. *Cel.Mech.* **23**, 33.
3. Murray, C.A., 1983, *Vectorial Astrometry*, Adam Hilger.

SLA_RDPLAN — Apparent $[\alpha, \delta]$ of Planet — SLA_RDPLAN

ACTION : Approximate topocentric apparent $[\alpha, \delta]$ and angular size of a planet.

CALL : `CALL sla_RDPLAN (DATE, NP, ELONG, PHI, RA, DEC, DIAM)`

GIVEN :

DATE	**D**	MJD of observation (JD−2400000.5)
NP	**I**	planet: 1 = Mercury 2 = Venus 3 = Moon 4 = Mars 5 = Jupiter 6 = Saturn 7 = Uranus 8 = Neptune 9 = Pluto else = Sun
ELONG,PHI	**D**	observer's longitude (east +ve) and latitude (radians)

RETURNED :

RA,DEC	**D**	topocentric apparent $[\alpha, \delta]$ (radians)
DIAM	**D**	angular diameter (equatorial, radians)

NOTES :

1. The date is in a dynamical timescale (TDB, formerly ET) 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 UT 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 routine its effect on planetary positions is small (negligible for the outer planets). Geocentric positions can be generated by appropriate use of the routines sla_DMOON and sla_PLANET.
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 Sun and planets. For more information on accuracy, refer to the routines sla_PLANET and sla_DMOON, which the present routine uses.

SLA_REFCO Refraction Constants SLA_REFCO

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 : `CALL sla_REFCO (HM, TDK, PMB, RH, WL, PHI, TLR, EPS, REFA, REFB)`

GIVEN :

HM	**D**	height of the observer above sea level (metre)
TDK	**D**	ambient temperature at the observer (degrees K)
PMB	**D**	pressure at the observer (mB)
RH	**D**	relative humidity at the observer (range 0 – 1)
WL	**D**	effective wavelength of the source (μm)
PHI	**D**	latitude of the observer (radian, astronomical)
TLR	**D**	temperature lapse rate in the troposphere (degrees K per metre)
EPS	**D**	precision required to terminate iteration (radian)

RETURNED :

REFA	**D**	$\tan\zeta$ coefficient (radians)
REFB	**D**	$\tan^3\zeta$ coefficient (radians)

NOTES :

1. Suggested values for the TLR and EPS arguments are 0.0065D0 and 1D−8 respectively.
2. The radio refraction is chosen by specifying WL $>$ 100 μm.
3. The routine is a slower but more accurate alternative to the sla_REFCOQ routine. The constants it produces give perfect agreement with sla_REFRO at zenith distances $\tan^{-1} 1$ (45°) and $\tan^{-1} 4$ ($\sim$ 76°). At other zenith distances, the model achieves: $0\rlap{.}''5$ accuracy for $\zeta < 80°$, $0\rlap{.}''01$ accuracy for $\zeta < 60°$, and $0\rlap{.}''001$ accuracy for $\zeta < 45°$.

SLA_REFCOQ — Refraction Constants (fast) — SLA_REFCOQ

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 sla_REFCO routine – see notes.)

CALL : `CALL sla_REFCOQ (TDK, PMB, RH, WL, REFA, REFB)`

GIVEN :

TDK	**D**	ambient temperature at the observer (degrees K)
PMB	**D**	pressure at the observer (mB)
RH	**D**	relative humidity at the observer (range 0 – 1)
WL	**D**	effective wavelength of the source (μm)

RETURNED :

REFA	**D**	$\tan\zeta$ coefficient (radians)
REFB	**D**	$\tan^3\zeta$ coefficient (radians)

NOTES :

1. The radio refraction is chosen by specifying WL $>$ 100 μm.
2. The model is an approximation, for moderate zenith distances, to the predictions of the sla_REFRO routine. The approximation is maintained across a range of conditions, and applies to both optical/IR and radio.

3. The algorithm is a fast alternative to the sla_REFCO routine. The latter calls the sla_REFRO routine 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 $\sim 76°$) 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 routine is not as accurate, though it satisfies most practical requirements.

4. 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) and (iii) variations in tropospheric lapse rate.

5. The model has been tested using the following range of conditions:

   - lapse rates 0.0055, 0.0065, 0.0075 degrees 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 $+20°$ with respect to 280°K at sea level
   - relative humidity 0, 0.5, 1
   - wavelength 0.4, 0.6, ... 2μm, + radio
   - zenith distances 15°, 45°, 75°

   For the above conditions, the comparison with sla_REFRO was as follows:

	worst	*RMS*
optical/IR	62	8
radio	319	49
	mas	mas

   For this particular set of conditions:

   - lapse rate $6.5° Kkm^{-1}$
   - latitude 50°
   - sea level
   - pressure 1005 mB
   - temperature 7°C
   - humidity 80%
   - wavelength 5740 Å

   the results were as follows:

ζ	sla_REFRO	sla_REFCOQ	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 and Sinclair (1985).

The results from the much slower but more accurate sla_REFCO routine have not been included in the tabulation as they are identical to those in the sla_REFRO column to the $0\farcs01$ resolution used.

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 expressions 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), expression 2.5.5.
   - The refractivity of air is a function of temperature, total pressure, water-vapour pressure and, in the case of optical/IR but not radio, wavelength. The formulae for the two cases are developed from the Essen and Froome expressions adopted in Resolution 1 of the 12th International Geodesy Association General Assembly (1963).

   The above three items are as used in the sla_REFRO routine.

   - The formula for β ($= H_0/r_0$) is an adaption of expression 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), expression 4.31.

REFERENCES :

1. Crane, R.K., Meeks, M.L. (ed), "Refraction Effects in the Neutral Atmosphere", *Methods of Experimental Physics: Astrophysics 12B,* Academic Press, 1976.

2. Gill, Adrian E., *Atmosphere-Ocean Dynamics,* Academic Press, 1982.
3. Hohenkerk, C.Y., & Sinclair, A.T., NAO Technical Note No. 63, 1985.
4. International Geodesy Association General Assembly, Bulletin Géodésique **70** p390, 1963.
5. Stone, Ronald C., P.A.S.P. **108** 1051-1058, 1996.
6. Green, R.M., *Spherical Astronomy,* Cambridge University Press, 1987.

SLA_REFRO — Refraction — SLA_REFRO

ACTION : Atmospheric refraction, for radio or optical/IR wavelengths.

CALL : `CALL sla_REFRO (ZOBS, HM, TDK, PMB, RH, WL, PHI, TLR, EPS, REF)`

GIVEN :

ZOBS	**D**	observed zenith distance of the source (radians)
HM	**D**	height of the observer above sea level (metre)
TDK	**D**	ambient temperature at the observer (degrees K)
PMB	**D**	pressure at the observer (mB)
RH	**D**	relative humidity at the observer (range $0-1$)
WL	**D**	effective wavelength of the source (μm)
PHI	**D**	latitude of the observer (radian, astronomical)
TLR	**D**	temperature lapse rate in the troposphere (degrees K per metre)
EPS	**D**	precision required to terminate iteration (radian)

RETURNED :

REF	**D**	refraction: *in vacuo* ZD minus observed ZD (radians)

NOTES :

1. A suggested value for the TLR argument is 0.0065D0. The refraction is significantly affected by TLR, and if studies of the local atmosphere have been carried out a better TLR value may be available.
2. A suggested value for the EPS argument is 1D−8. The result is usually at least two orders of magnitude more computationally precise than the supplied EPS value.
3. The routine computes the refraction for zenith distances up to and a little beyond 90° using the method of Hohenkerk & Sinclair (NAO Technical Notes 59 and 63, subsequently adopted in the *Explanatory Supplement to the Astronomical Almanac,* 1992 – see section 3.281).
4. The code is based on the AREF optical/IR refraction subroutine of C. Hohenkerk (HMNAO, September 1984), with extensions to support the radio case. The modifications to the original HMNAO optical/IR refraction code which affect the results are:

- Murray's values for the gas constants have been used (*Vectorial Astrometry,* Adam Hilger, 1983).
- A better model for $P_s(T)$ has been adopted (taken from Gill, *Atmosphere-Ocean Dynamics,* Academic Press, 1982).
- More accurate expressions for Pw_o have been adopted (again from Gill 1982).
- Provision for radio wavelengths has been added using expressions devised by A. T. Sinclair, RGO (private communication 1989), based on the Essen & Froome refractivity formula adopted in Resolution 1 of the 12th International Geodesy Association General Assembly (Bulletin Géodésique **70** p390, 1963).

None of the changes significantly affects the optical/IR results with respect to the algorithm given in the 1992 *Explanatory Supplement.* For example, at 70° zenith distance the present routine agrees with the ES algorithm to better than $0\farcs05$ for any reasonable combination of parameters. However, the improved water-vapour expressions do make a significant difference in the radio band, at 70° zenith distance reaching almost $4''$ for a hot, humid, low-altitude site during a period of low pressure.

5. The radio refraction is chosen by specifying WL $> 100\ \mu$m. Because the algorithm takes no account of the ionosphere, the accuracy deteriorates at low frequencies, below about 30 MHz.
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°, a fixed REF value equal to the result for ZOBS = 93° is returned, appropriately signed.
7. As in the original Hohenkerk and Sinclair algorithm, fixed values of the water vapour polytrope exponent, the height of the tropopause, and the height at which refraction is negligible are used.
8. The radio refraction has been tested against work done by Iain Coulson, JACH, (private communication 1995) for the James Clerk Maxwell Telescope, Mauna Kea. For typical conditions, agreement at the $0\farcs1$ level is achieved for moderate ZD, worsening to perhaps $0\farcs5 - 1\farcs0$ at ZD 80°. At hot and humid sea-level sites the accuracy will 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.

SLA_REFV — Apply Refraction to Vector — SLA_REFV

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 : `CALL sla_REFV (VU, REFA, REFB, VR)`

GIVEN :

VU	**D**	unrefracted position of the source ([Az, El] 3-vector)
REFA	**D**	$\tan\zeta$ coefficient (radians)
REFB	**D**	$\tan^3\zeta$ coefficient (radians)

RETURNED :

VR	**D**	refracted position of the source ([Az, El] 3-vector)

NOTES :

1. This routine applies the adjustment for refraction in the opposite sense to the usual one – it takes an unrefracted (*in vacuo*) position and produces an observed (refracted) position, whereas the $\Delta\zeta = a\tan\zeta + b\tan^3\zeta$ 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 routine 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 $\zeta = 70°$. The inherent accuracy of the model is, of course, far worse than this – see the documentation for sla_REFCO for more information.
3. At low elevations (below about 3°) the refraction correction is held back to prevent arithmetic problems and wildly wrong results. Over a wide range of observer heights and corresponding temperatures and pressures, the following levels of accuracy are achieved, relative to numerical integration through a model atmosphere:

ζ_{obs}	*error*	
80°	0.″4	
81°	0.″8	
82°	1.″6	
83°	3″	
84°	7″	
85°	17″	
86°	45″	
87°	150″	
88°	340″	
89°	620″	
90°	1100″	
91°	1900″	< high-altitude
92°	3200″	< sites only

4. See also the routine sla_REFZ, which performs the adjustment to the zenith distance rather than in $[x, y, z]$. The present routine is faster than sla_REFZ and, except very low down, is equally accurate for all practical purposes. However, beyond about $\zeta = 84°$ sla_REFZ should be used, and for the utmost accuracy iterative use of sla_REFRO should be considered.

SLA_REFZ — Apply Refraction to ZD — SLA_REFZ

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 : `CALL sla_REFZ (ZU, REFA, REFB, ZR)`

GIVEN :

ZU	**D**	unrefracted zenith distance of the source (radians)
REFA	**D**	$\tan\zeta$ coefficient (radians)
REFB	**D**	$\tan^3\zeta$ coefficient (radians)

RETURNED :

ZR	**D**	refracted zenith distance (radians)

NOTES :

1. This routine applies the adjustment for refraction in the opposite sense to the usual one – it takes an unrefracted (*in vacuo*) position and produces an observed (refracted) position, whereas the $\Delta\zeta = a\tan\zeta + b\tan^3\zeta$ 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 $\zeta = 80°$. The inherent accuracy of the model is, of course, far worse than this – see the documentation for sla_REFCO for more information.
2. At $\zeta = 83°$, the rapidly-worsening $\Delta\zeta = a\tan\zeta + b\tan^3\zeta$ model is abandoned and an empirical formula takes over:

$$\Delta\zeta = F\left(\frac{0.^\circ55445 - 0.^\circ01133E + 0.^\circ00202E^2}{1 + 0.28385E + 0.02390E^2}\right)$$

where $E = 90° - \zeta_{true}$ and F is a factor chosen to meet the $\Delta\zeta = a\tan\zeta + b\tan^3\zeta$ formula at $\zeta = 83°$. Over a wide range of observer heights and corresponding temperatures and pressures, the following levels of accuracy are achieved, relative to numerical integration through a model atmosphere:

ζ_{obs}	*error*	
80°	$0''.4$	
81°	$0''.8$	
82°	$1''.5$	
83°	$3''.2$	
84°	$4''.9$	
85°	$5''.8$	
86°	$6''.1$	
87°	$7''.1$	
88°	$11''$	
89°	$21''$	
90°	$43''$	
91°	$92''$	< high-altitude
92°	$220''$	< sites only

3. See also the routine sla_REFV, which performs the adjustment in $[\,x, y, z\,]$, and with the emphasis on speed rather than numerical accuracy.

SLA_RVEROT RV Corrn to Earth Centre SLA_RVEROT

ACTION : Velocity component in a given direction due to Earth rotation.

CALL : `R = sla_RVEROT (PHI, RA, DA, ST)`

GIVEN :

PHI	**R**	geodetic latitude of observing station (radians)
RA,DA	**R**	apparent $[\,\alpha, \delta\,]$ (radians)
ST	**R**	local apparent sidereal time (radians)

RETURNED :

sla_RVEROT	**R**	Component of Earth rotation in direction RA,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 precision, use the routine sla_PVOBS.

SLA_RVGALC RV Corrn to Galactic Centre SLA_RVGALC

ACTION : Velocity component in a given direction due to the rotation of the Galaxy.

CALL : `R = sla_RVGALC (R2000, D2000)`

GIVEN :

R2000,D2000	**R**	J2000.0 mean $[\alpha, \delta]$ (radians)

RETURNED :

sla_RVGALC	**R**	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 $l^{II} = 90^\circ$, $b^{II} = 0$.

REFERENCE : Kerr & Lynden-Bell (1986), MNRAS, 221, p1023.

SLA_RVLG RV Corrn to Local Group SLA_RVLG

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 = sla_RVLG (R2000, D2000)`

GIVEN :

R2000,D2000	**R**	J2000.0 mean $[\alpha, \delta]$ (radians)

RETURNED :

sla_RVLG	**R**	Component of **solar** (*n.b.*) 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.

SLA_RVLSRD — RV Corrn to Dynamical LSR — SLA_RVLSRD

ACTION : Velocity component in a given direction due to the Sun's motion with respect to the "dynamical" Local Standard of Rest.

CALL : `R = sla_RVLSRD (R2000, D2000)`

GIVEN :

R2000,D2000	**R**	J2000.0 mean $[\,\alpha, \delta\,]$ (radians)

RETURNED :

sla_RVLSRD	**R**	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 routine sla_RVLSRK.
4. The peculiar solar motion is from Delhaye (1965), in *Stars and Stellar Systems*, vol 5, p73: in Galactic Cartesian coordinates (+9,+12,+7) km s^{-1}. This corresponds to about 16.6 km s^{-1} towards Galactic coordinates $l^{II} = 53^\circ, b^{II} = +25^\circ$.

SLA_RVLSRK — RV Corrn to Kinematical LSR — SLA_RVLSRK

ACTION : Velocity component in a given direction due to the Sun's motion with respect to a kinematical Local Standard of Rest.

CALL : `R = sla_RVLSRK (R2000, D2000)`

GIVEN :

R2000,D2000	**R**	J2000.0 mean $[\,\alpha,\delta\,]$ (radians)

RETURNED :

sla_RVLSRK	**R**	Component of *standard* 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 routine sla_RVLSRD.
4. The adopted standard solar motion is 20 km s^{-1} towards $\alpha = 18^{\mathrm{h}}, \delta = +30^{\circ}$ (1900).

REFERENCES :

1. Delhaye (1965), in *Stars and Stellar Systems*, vol 5, p73.
2. *Methods of Experimental Physics* (ed Meeks), vol 12, part C, sec 6.1.5.2, p281.

SLA_S2TP — Spherical to Tangent Plane — SLA_S2TP

ACTION : Projection of spherical coordinates onto the tangent plane (single precision).

CALL : `CALL sla_S2TP (RA, DEC, RAZ, DECZ, XI, ETA, J)`

GIVEN :

RA,DEC	**R**	spherical coordinates of star (radians)
RAZ,DECZ	**R**	spherical coordinates of tangent point (radians)

RETURNED :

XI,ETA	**R**	tangent plane coordinates (radians)
J	**I**	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 routine sla_V2TP is available.

SLA_SEP — Angle Between 2 Points on Sphere — SLA_SEP

ACTION : Angle between two points on a sphere (single precision).

CALL : `R = sla_SEP (A1, B1, A2, B2)`

GIVEN :

A1,B1	**R**	spherical coordinates of one point (radians)
A2,B2	**R**	spherical coordinates of the other point (radians)

RETURNED :

sla_SEP	**R**	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.

SLA_SMAT — Solve Simultaneous Equations — SLA_SMAT

ACTION : Matrix inversion and solution of simultaneous equations (single precision).

CALL : `CALL sla_SMAT (N, A, Y, D, JF, IW)`

GIVEN :

N	**I**	number of unknowns
A	**R(N,N)**	matrix
Y	**R(N)**	vector

RETURNED :

A	**R(N,N)**	matrix inverse
Y	**R(N)**	solution
D	**R**	determinant
JF	**I**	singularity flag: 0=OK
IW	**I(N)**	workspace

NOTES :

1. For the set of n simultaneous linear equations in n unknowns:

   $\mathbf{A}\cdot\mathbf{y} = \mathbf{x}$

   where:

   - **A** is a non-singular $n \times n$ matrix,
   - **y** is the vector of n unknowns, and
   - **x** is the known vector,

   sla_SMAT computes:

   - the inverse of matrix **A**,
   - the determinant of matrix **A**, and
   - the vector of n unknowns **y**.

   Argument N is the order n, A (given) is the matrix **A**, Y (given) is the vector **x** and Y (returned) is the vector **y**. The argument A (returned) is the inverse matrix $\mathbf{A}^{-1}$, and D is $det(\mathbf{A})$.
2. JF is the singularity flag. If the matrix is non-singular, JF=0 is returned. If the matrix is singular, JF=−1 and D=0.0 are returned. In the latter case, the contents of array 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.
4. This routine replaces the obsolete sla_SMATRX.

SLA_SUBET   Remove E-terms   SLA_SUBET

ACTION : Remove the E-terms (elliptic component of annual aberration) from a pre IAU 1976 catalogue $[\,\alpha,\delta\,]$ to give a mean place.

CALL : `CALL sla_SUBET (RC, DC, EQ, RM, DM)`

GIVEN :

RC,DC	**D**	$[\,\alpha,\delta\,]$ with E-terms included (radians)
EQ	**D**	Besselian epoch of mean equator and equinox

RETURNED :

RM,DM	**D**	$[\,\alpha,\delta\,]$ 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 routine converts such a position to a formal mean place (allowing, for example, comparison with a pulsar timing position).

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

SLA_SUPGAL — Supergalactic to Galactic — SLA_SUPGAL

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

CALL : `CALL sla_GALSUP (DL, DB, DSL, DSB)`

GIVEN :

DSL,DSB	**D**	supergalactic longitude and latitude (radians)

RETURNED :

DL,DB	**D**	galactic longitude and latitude $[\,l^{II}, b^{II}\,]$ (radians)

REFERENCES :

1. de Vaucouleurs, de Vaucouleurs, & Corwin, *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 $l^{II} = 137.37$.)

ACTION : Singular value decomposition. This routine expresses a given matrix **A** as the product of three matrices **U**, **W**, $\mathbf{V}^T$:

$$\mathbf{A} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T$$

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 $w_{ii} \geq 0$
$\mathbf{V}^T$	is the transpose of an $n \times n$ orthogonal matrix

CALL : `CALL sla_SVD (M, N, MP, NP, A, W, V, WORK, JSTAT)`

GIVEN :

M,N	**I**	m, n, the numbers of rows and columns in matrix **A**
MP,NP	**I**	physical dimensions of array containing matrix **A**
A	**D(MP,NP)**	array containing $m \times n$ matrix **A**

RETURNED :

A	**D(MP,NP)**	array containing $m \times n$ column-orthogonal 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** (*n.b.* not $\mathbf{V}^T$)
WORK	**D(N)**	workspace
JSTAT	**I**	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 routine 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 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.

REFERENCES : The algorithm is an adaptation of the routine SVD in the *EISPACK* library (Garbow *et al.* 1977, *EISPACK Guide Extension*, Springer Verlag), which is a FORTRAN 66 implementation of the Algol routine SVD of Wilkinson & Reinsch 1971 (*Handbook for Automatic Computation*, vol 2, ed Bauer *et al.*, Springer Verlag). 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 *Numerical Recipes* (Press *et al.* 1987, Cambridge University Press), which includes another variant of the EISPACK code.

SLA_SVDCOV Covariance Matrix from SVD SLA_SVDCOV

ACTION : From the **W** and **V** matrices from the SVD factorization of a matrix (as obtained from the sla_SVD routine), obtain the covariance matrix.

CALL : `CALL sla_SVDCOV (N, NP, NC, W, V, WORK, CVM)`

GIVEN :

N	**I**	n, the number of rows and columns in matrices **W** and **V**
NP	**I**	first dimension of array containing $n \times n$ matrix **V**
NC	**I**	first dimension of array CVM
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	**D(N)**	workspace
CVM	**D(NC,NC)**	array to receive covariance matrix

REFERENCE : *Numerical Recipes*, section 14.3.

SLA_SVDSOL Solution Vector from SVD SLA_SVDSOL

ACTION : From a given vector and the SVD of a matrix (as obtained from the sla_SVD routine), obtain the solution vector. This routine solves the equation:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

where:

$\mathbf{A}$	is a given m (rows) $\times n$ (columns) matrix, where $m \geq n$
$\mathbf{x}$	is the n-vector we wish to find, and
$\mathbf{b}$	is a given m-vector

by means of the *Singular Value Decomposition* method (SVD).

CALL : `CALL sla_SVDSOL (M, N, MP, NP, B, U, W, V, WORK, X)`

GIVEN :

M,N	**I**	m, n, the numbers of rows and columns in matrix **A**
MP,NP	**I**	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	**D(N)**	workspace
X	**D(N)**	unknown vector **x**

NOTES :

1. In the Singular Value Decomposition method (SVD), the matrix **A** is first factorized (for example by the routine sla_SVD) into the following components:

$$\mathbf{A} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T$$

where:

$\mathbf{A}$	is any m (rows) $\times n$ (columns) matrix, where $m > n$
$\mathbf{U}$	is an $m \times n$ column-orthogonal matrix
$\mathbf{W}$	is an $n \times n$ diagonal matrix with $w_{ii} \geq 0$
$\mathbf{V}^T$	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:

$$\mathbf{x} = \mathbf{V} \cdot [diag(1/\mathbf{W}_j)] \cdot (\mathbf{U}^T \cdot \mathbf{b})$$

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 routine 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.

REFERENCE : *Numerical Recipes*, section 2.9.

SLA_TP2S Tangent Plane to Spherical SLA_TP2S

ACTION : Transform tangent plane coordinates into spherical coordinates (single precision)

CALL : `CALL sla_TP2S (XI, ETA, RAZ, DECZ, RA, DEC)`

GIVEN :

XI,ETA	**R**	tangent plane rectangular coordinates (radians)
RAZ,DECZ	**R**	spherical coordinates of tangent point (radians)

RETURNED :

RA,DEC	**R**	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 routine sla_TP2V is available.

SLA_TP2V Tangent Plane to Direction Cosines SLA_TP2V

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 : `CALL sla_TP2V (XI, ETA, V0, V)`

GIVEN :

XI,ETA	**R**	tangent plane coordinates of star (radians)
V0	**R(3)**	direction cosines of tangent point

RETURNED :

V	**R(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 routine is the Cartesian equivalent of the routine sla_TP2S.

SLA_TPS2C — Plate centre from ξ, η and α, δ — SLA_TPS2C

ACTION : From the tangent plane coordinates of a star of known $[\,\alpha,\delta\,]$, determine the $[\,\alpha,\delta\,]$ of the tangent point (single precision)

CALL : `CALL sla_TPS2C (XI, ETA, RA, DEC, RAZ1, DECZ1, RAZ2, DECZ2, N)`

GIVEN :

XI,ETA	**R**	tangent plane rectangular coordinates (radians)
RA,DEC	**R**	spherical coordinates (radians)

RETURNED :

RAZ1,DECZ1	**R**	spherical coordinates of tangent point, solution 1
RAZ2,DECZ2	**R**	spherical coordinates of tangent point, solution 2
N	**I**	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. $\mathrm{N}=1$ indicates only one useful solution, the usual case; under these circumstances, the second solution corresponds to the "over-the-pole" case, and this is reflected in the values of RAZ2 and DECZ2 which are returned.
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 routine sla_TPV2C is available.

SLA_TPV2C Plate centre from ξ,η and x,y,z SLA_TPV2C

ACTION : From the tangent plane coordinates of a star of known direction cosines, determine the direction cosines of the tangent point (single precision)

CALL : `CALL sla_TPV2C (XI, ETA, V, V01, V02, N)`

GIVEN :

XI,ETA	**R**	tangent plane coordinates of star (radians)
V	**R(3)**	direction cosines of star

RETURNED :

V01	**R(3)**	direction cosines of tangent point, solution 1
V01	**R(3)**	direction cosines of tangent point, solution 2
N	**I**	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 "over-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 routine is the Cartesian equivalent of the routine sla_TPS2C.

SLA_UE2EL Universal to Conventional Elements SLA_UE2EL

ACTION : Transform universal elements into conventional heliocentric osculating elements.

CALL :
```
CALL sla_UE2EL (U, JFORMR,
                JFORM, EPOCH, ORBINC, ANODE, PERIH,
                AORQ, E, AORL, DM, JSTAT)
```

GIVEN :

U		**D(13)**	universal orbital elements (updated; Note 1)
	(1)		combined mass $(M + m)$
	(2)		total energy of the orbit (α)
	(3)		reference (osculating) epoch (t_0)
	(4-6)		position at reference epoch $(\mathbf{r}_0)$
	(7-9)		velocity at reference epoch $(\mathbf{v}_0)$
	(10)		heliocentric distance at reference epoch
	(11)		$\mathbf{r}_0.\mathbf{v}_0$
	(12)		date (t)
	(13)		universal eccentric anomaly (ψ) of date, approx
JFORMR		**I**	requested element set (1-3; Note 3)

RETURNED :

JFORM	**I**	element set actually returned (1-3; Note 4)
EPOCH	**D**	epoch of elements (t_0 or T, TT MJD)
ORBINC	**D**	inclination (i, radians)
ANODE	**D**	longitude of the ascending node (Ω, radians)
PERIH	**D**	longitude or argument of perihelion (ϖ or ω, radians)
AORQ	**D**	mean distance or perihelion distance (a or q, AU)
E	**D**	eccentricity (e)
AORL	**D**	mean anomaly or longitude (M or L, radians, JFORM=1,2 only)
DM	**D**	daily motion (n, radians, JFORM=1 only)
JSTAT	**I**	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:

EPOCH	=	epoch of elements t_0 (TT 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 (TT 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 ($0 \le e < 1$)
AORL	=	mean anomaly M (radians)

JFORM=3, suitable for comets:

EPOCH	=	epoch of perihelion T (TT 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 ($0 \le e \le 10$)

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:

JFORMR	JFORM	meaning
1	1	OK: elements are in the requested format
1	2	never happens
1	3	orbit not elliptical
2	1	never happens
2	2	OK: elements are in the requested format
2	3	orbit not elliptical
3	1	never happens
3	2	never happens
3	3	OK: elements are in the requested format

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

JFORM	1	2	3
EPOCH	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.

REFERENCES :

1. Sterne, Theodore E., *An Introduction to Celestial Mechanics,* Interscience Publishers, 1960. Section 6.7, p199.

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

SLA_UE2PV Pos/Vel from Universal Elements SLA_UE2PV

ACTION : Heliocentric position and velocity of a planet, asteroid or comet, starting from orbital elements in the "universal variables" form.

CALL : `CALL sla_UE2PV (DATE, U, PV, JSTAT)`

GIVEN :

DATE	**D**	date (TT Modified Julian Date = JD−2400000.5)

GIVEN and RETURNED :

U	**D(13)**	universal orbital elements (updated; Note 1)
(1)		combined mass $(M+m)$
(2)		total energy of the orbit (α)
(3)		reference (osculating) epoch (t_0)
(4-6)		position at reference epoch $(\mathbf{r}_0)$
(7-9)		velocity at reference epoch $(\mathbf{v}_0)$
(10)		heliocentric distance at reference epoch
(11)		$\mathbf{r}_0.\mathbf{v}_0$
(12)		date (t)
(13)		universal eccentric anomaly (ψ) of date, approx

RETURNED :

PV	**D(6)**	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$, equatorial, J2000 (AU, AU/s; Note 1)
$JSTAT$	**I**	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 routine is sla_EL2UE. This takes the conventional orbital elements and transforms them into the set of numbers needed by the present routine. A single prediction requires one one call to sla_EL2UE followed by one call to the present routine; for convenience, the two calls are packaged as the routine sla_PLANEL. Multiple predictions may be made by again calling sla_EL2UE once, but then calling the present routine multiple times, which is faster than multiple calls to sla_PLANEL.

   It is not obligatory to use sla_EL2UE to obtain the parameters. However, it should be noted that because sla_EL2UE performs its own validation, no checks on the contents of the array U are made by the present routine.

3. DATE is the instant for which the prediction is required. It is in the TT timescale (formerly Ephemeris Time, ET) and is a Modified Julian Date (JD−2400000.5).

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 sla_EL2UE routine in fact produces coordinates with respect to the J2000 equator and equinox.

5. The algorithm was originally adapted from the EPHSLA program of D. H. P. Jones (private communication, 1996). The method is based on Stumpff's Universal Variables.

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

SLA_UNPCD Remove Radial Distortion SLA_UNPCD

ACTION : Remove pincushion/barrel distortion from a distorted $[x, y]$ to give tangent-plane $[x, y]$.

CALL : `CALL sla_UNPCD (DISCO,X,Y)`

GIVEN :

DISCO	**D**	pincushion/barrel distortion coefficient
X, Y	**D**	distorted $[x, y]$

RETURNED :

X, Y	**D**	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 +ve; for *barrel* distortion, C is −ve.
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. The present routine is an approximate inverse to the companion routine sla_PCD, obtained from two iterations of Newton's method. The mismatch between the sla_PCD and sla_UNPCD is negligible for astrometric applications; to reach 1 milliarcsec at the edge of the AAT triplet or Schmidt field would require field diameters of $2^\circ\!.4$ and 42° respectively.

SLA_V2TP Direction Cosines to Tangent Plane SLA_V2TP

ACTION : Given the direction cosines of a star and of the tangent point, determine the star's tangent-plane coordinates (single precision).

CALL : `CALL sla_V2TP (V, V0, XI, ETA, J)`

GIVEN :

V	**R(3)**	direction cosines of star
V0	**R(3)**	direction cosines of tangent point

RETURNED :

XI,ETA	**R**	tangent plane coordinates (radians)
J	**I**	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 routine is the Cartesian equivalent of the routine sla_S2TP.

SLA_VDV Scalar Product SLA_VDV

ACTION : Scalar product of two 3-vectors (single precision).

CALL : `R = sla_VDV (VA, VB)`

GIVEN :

VA	**R(3)**	first vector
VB	**R(3)**	second vector

RETURNED :

sla_VDV	**R**	scalar product VA.VB

SLA_VN Normalize Vector SLA_VN

ACTION : Normalize a 3-vector, also giving the modulus (single precision).

CALL : `CALL sla_VN (V, UV, VM)`

GIVEN :

V	**R(3)**	vector

RETURNED :

UV	**R(3)**	unit vector in direction of V
VM	**R**	modulus of V

NOTE : If the modulus of V is zero, UV is set to zero as well.

SLA_VXV Vector Product SLA_VXV

ACTION : Vector product of two 3-vectors (single precision).

CALL : `CALL sla_VXV (VA, VB, VC)`

GIVEN :

VA	**R(3)**	first vector
VB	**R(3)**	second vector

RETURNED :

VC	**R(3)**	vector product VA×VB

SLA_WAIT Time Delay SLA_WAIT

ACTION : Wait for a specified interval.

CALL : `CALL sla_WAIT (DELAY)`

GIVEN :

DELAY	**R**	delay in seconds

NOTES :

1. The implementation is machine-specific.
2. The delay actually requested is restricted to the range 100ns-200s in the present implementation.
3. There is no guarantee of accuracy, though on almost all types of computer the program will certainly not resume execution *before* the stated interval has elapsed.

SLA_XY2XY — Apply Linear Model to an $[x, y]$ — SLA_XY2XY

ACTION : Transform one $[x, y]$ into another using a linear model of the type produced by the sla_FITXY routine.

CALL : `CALL sla_XY2XY (X1,Y1,COEFFS,X2,Y2)`

GIVEN :

X1,Y1	**D**	$[x, y]$ before transformation
COEFFS	**D(6)**	transformation coefficients (see note)

RETURNED :

X2,Y2	**D**	$[x, y]$ 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 routine performs the transformation:

   $x_2 = a + bx_1 + cy_1$
   $y_2 = d + ex_1 + fy_1$

2. See also sla_FITXY, sla_PXY, sla_INVF, sla_DCMPF.

SLA_ZD — h, δ to Zenith Distance — SLA_ZD

ACTION : Hour angle and declination to zenith distance (double precision).

CALL : `D = sla_ZD (HA, DEC, PHI)`

GIVEN :

HA	**D**	hour angle in radians
DEC	**D**	declination in radians
PHI	**D**	latitude in radians

RETURNED :

sla_ZD	**D**	zenith distance (radians, $0-\pi$)

NOTES :

1. The latitude must be geodetic. In critical applications, corrections for polar motion should be applied (see sla_POLMO).

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 routine 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.

4 EXPLANATION AND EXAMPLES

To guide the writer of positional-astronomy applications software, this final chapter puts the SLALIB routines 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 routines. 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.

Only brief details of individual SLALIB routines 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 routines (available in both Fortran and C) is also intended to be used as documentation.

4.1 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 line 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 $[\,l^{II},b^{II}\,]$) 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 an indication that a vector-related method might be preferable.

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

```
      IMPLICIT NONE

*  Degrees to radians
      REAL D2R
      PARAMETER (D2R=0.01745329252)

*  Longitudes and latitudes (radians) for London and Sydney
      REAL AL,BL,AS,BS
      PARAMETER (AL=-0.2*D2R,BL=51.5*D2R,AS=151.2*D2R,BS=-33.9*D2R)

*  Earth radius in km (spherical approximation)
      REAL RKM
      PARAMETER (RKM=6375.0)

      REAL sla_SEP,sla_BEAR


*  Distance and initial heading (N=0, E=90)
      WRITE (*,'(1X,I5,'' km,'',I4,'' deg'')')
     :    NINT(sla_SEP(AL,BL,AS,BS)*RKM),NINT(sla_BEAR(AL,BL,AS,BS)/D2R)

      END
```

(The result is 17011 km, 61°.)

The routines sla_PAV and sla_DPAV are equivalents of sla_BEAR and sla_DBEAR but starting from direction-cosines instead of spherical coordinates.

4.1.1 Formatting angles

SLALIB has routines 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 routines avoid.

There are five routines 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.

Three of the routines decode single numbers: sla_INTIN (integer), sla_FLOTIN (single precision floating point) and sla_DFLTIN (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 $+1^\circ$. Here is an example (in Fortran) where we wish to read two numbers, and integer `IX` and a real, `Y`, with `X` defaulting to zero and `Y` defaulting to `X`:

```
      DOUBLE PRECISION Y
      CHARACTER*80 A
      INTEGER IX,I,J

*  Input the string to be decoded
      READ (*,'(A)') A

*  Preset IX to its default value
      IX = 0

*  Point to the start of the string
      I = 1

*  Decode an integer
      CALL sla_INTIN(A,I,IX,J)
      IF (J.GT.1) GO TO ... (bad IX)

*  Preset Y to its default value
      Y = DBLE(IX)

*  Decode a double precision number
      CALL sla_DFLTIN(A,I,Y,J)
      IF (J.GT.1) GO TO ... (bad Y)
```

Two additional routines decode a 3-field sexagesimal number: sla_AFIN (degrees, arcminutes, arcseconds to single precision radians) and sla_DAFIN (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 Fortran program which uses sla_DAFIN was given earlier, in section 1.2.

SLALIB provides four routines for expressing an angle in radians in a preferred range. The function sla_RANGE expresses an angle in the range $\pm\pi$; sla_RANORM expresses an angle in the range $0-2\pi$. The functions sla_DRANGE and sla_DRANRM are double precision versions.

Several routines (sla_CTF2D, sla_CR2AF *etc.*) are provided to convert angles to and from sexagesimal form (hours, minute, seconds or degrees, arcminutes and arcseconds). They avoid the common "converting from integer to real at the wrong time" bug, which produces angles like $24^{\mathrm{h}}\,59^{\mathrm{m}}\,59\overset{\mathrm{s}}{.}999$. Here is a program which displays an hour angle stored in radians:

```
DOUBLE PRECISION HA
CHARACTER SIGN
INTEGER IHMSF(4)
:
CALL sla_DR2TF(3,HA,SIGN,IHMSF)
WRITE (*,'(1X,A,3I3.2,''.'',I3.3)') SIGN,IHMSF
```

4.2 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 usually 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 positive latitudes. 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 involving objects with 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 may 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.
- Accuracy 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.

4.2.1 Using vectors

SLALIB provides conversions between spherical and vector form (sla_CS2C, sla_CC2S *etc.*), plus an assortment of standard vector and matrix operations (sla_VDV, sla_MXV *etc.*). There are also routines (sla_EULER *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* (the pole of the rotation, and the amount of rotation), and routines are provided for this (sla_AV2M, sla_M2AV *etc.*).

Here is an example where spherical coordinates `P1` and `Q1` undergo a coordinate transformation and become `P2` and `Q2`; the transformation consists of a rotation of the coordinate system through angles `A`, `B` and `C` about the z, new y and new z axes respectively:

```
      REAL A,B,C,R(3,3),P1,Q1,V1(3),V2(3),P2,Q2
       :
*  Create rotation matrix
      CALL sla_EULER('ZYZ',A,B,C,R)

*  Transform position (P,Q) from spherical to Cartesian
      CALL sla_CS2C(P1,Q1,V1)

*  Multiply by rotation matrix
      CALL sla_MXV(R,V1,V2)

*  Back to spherical
      CALL sla_CC2S(V2,P2,Q2)
```

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 $\mathbf{V1}$, the resulting vector $\mathbf{V2}$ has direction $< \mathbf{V1} + \mathbf{D} >$ but is no longer of unit length. A better approximation is available if the result is multiplied by a scaling factor of $(1 - \mathbf{D} \cdot \mathbf{V1})$, where the dot means scalar product. In Fortran:

```
F = (1D0-(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 Fortran, for initial vector `V1X,V1Y,V1Z`, change in elevation `DEL` (+ve $\equiv$ upwards), and result vector `V2X,V2Y,V2Z`:

```
COSDEL = 1D0-DEL*DEL/2D0
R1 = SQRT(V1X*V1X+V1Y*V1Y)
F = COSDEL*(R1-DEL*V1Z)/R1
V2X = F*V1X
V2Y = F*V1Y
V2Z = COSDEL*(V1Z+DEL*R1)
```

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

SLALIB includes routines for the case where both a position and a velocity are involved. The routines sla_CS2C6 and sla_CC62S convert from $[\theta, \phi, \dot{\theta}, \dot{\phi}]$ to $[\,x, y, z, \dot{x}, \dot{y}, \dot{z}\,]$ and back; sla_DCS26 and sla_DC62S are double precision equivalents.

4.3 Celestial Coordinate Systems

SLALIB has routines to perform transformations of celestial positions between different spherical coordinate systems, including those shown in the following table:

system	*symbols*	*longitude*	*latitude*	*x-y plane*	*long. zero*	*RH/LH*
horizon	–	azimuth	elevation	horizontal	north	L
equatorial	α, δ	R.A.	Dec.	equator	equinox	R
local equ.	h, δ	H.A.	Dec.	equator	meridian	L
ecliptic	λ, β	ecl. long.	ecl. lat.	ecliptic	equinox	R
galactic	l^{II}, b^{II}	gal. long.	gal. lat.	gal. equator	gal. centre	R
supergalactic	SGL,SGB	SG long.	SG lat.	SG equator	node w. gal. equ.	R

Transformations between $[\,h, \delta\,]$ and $[\,Az, El\,]$ can be performed by calling sla_E2H and sla_H2E, or, in double precision, sla_DE2H and sla_DH2E. There is also a routine for obtaining zenith distance alone for a given $[\,h, \delta\,]$, sla_ZD, and one for determining the parallactic angle, sla_PA. Three routines are included which relate to altazimuth telescope mountings. For a given $[\,h, \delta\,]$ and latitude, sla_ALTAZ returns the azimuth, elevation and parallactic angle, plus velocities and accelerations for sidereal tracking. The routines sla_PDA2H and sla_PDQ2H predict at what hour angle a given azimuth or parallactic angle will be reached.

The routines sla_EQECL and sla_ECLEQ transform between ecliptic coordinates and $[\,\alpha, \delta\,]$; there is also a routine for generating the equatorial to ecliptic rotation matrix for a given date: sla_ECMAT.

For conversion between Galactic coordinates and $[\,\alpha, \delta\,]$ there are two sets of routines, depending on whether the $[\,\alpha, \delta\,]$ is old-style, B1950, or new-style, J2000; sla_EG50 and sla_GE50 are $[\,\alpha, \delta\,]$ to $[\,l^{II}, b^{II}\,]$ and *vice versa* for the B1950 case, while sla_EQGAL and sla_GALEQ are the J2000 equivalents.

Finally, the routines sla_GALSUP and sla_SUPGAL transform $[\,l^{II}, b^{II}\,]$ 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, orbital perturbations ...) and some have several interpretations, all subtly different. The various frames 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. Figure 1 shows how some of these systems are related, to one another and to the direction in which a celestial source actually appears in the sky. At the top of the diagram are the various sorts of *mean place* found in star catalogues and papers;[2] at the bottom is the *observed* $[\,Az, El\,]$, where a perfect theodolite would be pointed to see the source; and in the body of the diagram are the intermediate processing steps and coordinate systems. To help understand this diagram, and the SLALIB routines 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.

[2] One frame not included in Figure 1 is that of the Hipparcos catalogue. This is currently the best available implementation in the optical of the *International Celestial Reference System* (ICRS), which is based on extragalactic radio sources observed by VLBI. The distinction between FK5 J2000 and Hipparcos coordinates only becomes important when accuracies of 50 mas or better are required. More details are given in Section 4.14.

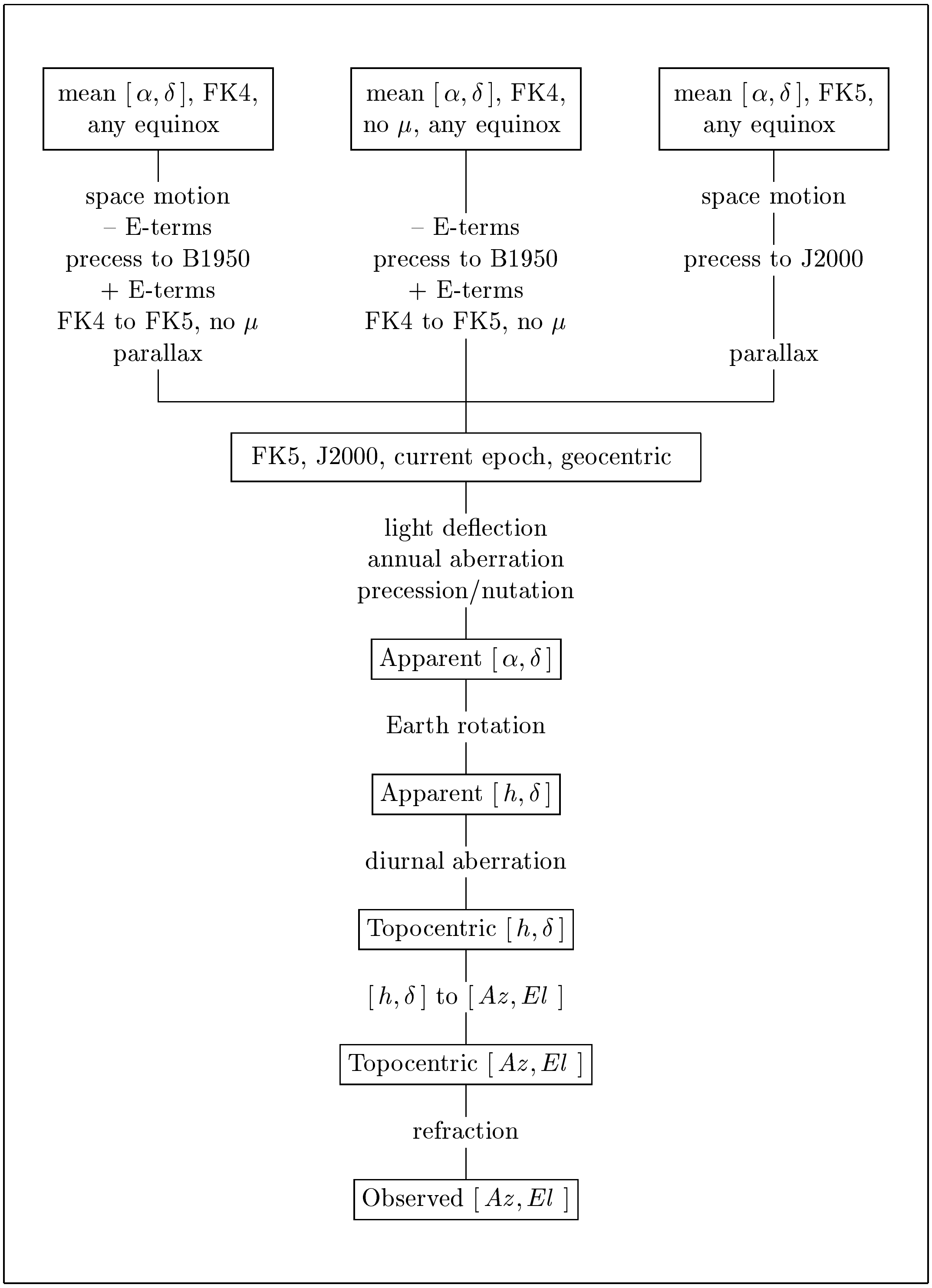

Figure 1: Relationship Between Celestial Coordinates

Star positions are published or catalogued using one of the mean [α, δ] systems shown at the top. The "FK4" systems were used before about 1980 and are usually equinox B1950. The "FK5" system, equinox J2000, is now preferred, or rather its modern equivalent, the International Celestial Reference Frame (in the optical, the Hipparcos catalogue). The figure relates a star's mean [α, δ] 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.

4.4 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 zero point of α is the point of 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*.[3]

This simple picture is unfortunately complicated by the difficulty of defining a suitable equator and equinox. One problem is that the Sun's apparent motion is not completely regular, due to the ellipticity of the Earth's orbit and its continuous disturbance by the Moon and planets. This is dealt with by separating the motion into (i) a smooth and steady *mean Sun* and (ii) a set of periodic corrections and perturbations; only the former is involved in establishing reference frames and timescales. A second, far larger problem, is that the celestial equator and the ecliptic are both moving with respect to the stars. These motions arise because of the gravitational interactions between the Earth and the other solar-system bodies.

By far the largest effect is the so-called "precession of the equinoxes", where the Earth's rotation axis sweeps out a cone centred on the ecliptic pole, completing one revolution in about 26,000 years. The cause of the motion is the torque exerted on the distorted and spinning Earth by the Sun and 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 precessional effect.

The complex interactions between the three bodies 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 $17''$ and a period of about 18.6 years. This difference of scale makes it convenient to treat these two components of the motion separately. The main 26,000-year effect is called *luni-solar precession*; the smaller, faster, periodic terms are called the *nutation*.

Note that precession and nutation are simply different frequency components of the same physical effect. It 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

[3] With the introduction of the International Celestial Reference System (ICRS), the connection between (i) star coordinates and (ii) the Earth's orientation and orbit has been broken. However, the orientation of the International Celestial Reference Frame (ICRF) axes was, for convenience, chosen to match J2000 FK5, and for most practical purposes ICRF coordinates (for example entries in the Hipparcos catalogue) can be regarded as synonymous with J2000 FK5. See Section 4.14 for further details.

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 is not fixed in orientation, a result of the attractions of the planets. This slow (about $0\rlap{.}''5$ per year) secular rotation of the ecliptic about a slowly-moving diameter is called, confusingly, *planetary precession* 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 frames.

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; SLALIB's nutation model (IAU 1980) involves 106 terms in each of ψ (longitude) and ϵ (obliquity), some as small as $0\rlap{.}''0001$.

4.4.1 SLALIB support for precession and nutation

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 (sla_PREBN).
- The new Fricke, IAU 1976, "FK5" model, used for J2000 star positions and other post-1984.0 purposes (sla_PREC).
- A model published by Simon *et al.* which is more accurate than the IAU 1976 model and which is suitable for long periods of time (sla_PRECL).

In each case, the named SLALIB routine generates the (3×3) *precession matrix* for a given start and finish time. For example, here is the Fortran code for generating the rotation matrix which describes the precession between the epochs J2000 and J1985.372 (IAU 1976 model):

```
DOUBLE PRECISION PMAT(3,3)
 :
CALL sla_PREC(2000D0,1985.372D0,PMAT)
```

It is instructive to examine the resulting matrix:

```
+0.9999936402  +0.0032709208  +0.0014214694
-0.0032709208  +0.9999946505  -0.0000023247
-0.0014214694  -0.0000023248  +0.9999989897
```

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 isn't going to move a position vector very far (in this case about $0^\circ\!.2$).

For convenience, a direct $[\,\alpha,\delta\,]$ to $[\,\alpha,\delta\,]$ precession routine is also provided (sla_PRECES), suitable for either the old or the new system (but not a mixture of the two).

SLALIB provides only one nutation model, the new, IAU 1980 model, implemented in the routine sla_NUTC. This returns the components of nutation in longitude and latitude (and also provides the obliquity) from which a nutation matrix can be generated by calling sla_DEULER (and from which the *equation of the equinoxes*, described later, can be found). Alternatively, the nutation matrix can be generated in a single call by using sla_NUT.

A rotation matrix for applying the entire precession/nutation transformation in one go can be generated by calling sla_PRENUT.

4.5 Mean Places

The main effect of the precession/nutation is a steady increase of about $50''$/year in the ecliptic longitudes of the stars. It is therefore essential, when reporting the position of an astronomical target, 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. [4] For simplicity, only the smooth and steady "general precession" part of the complete precession/nutation effect is included, thereby defining what is called the *mean* equator and equinox for the epoch concerned. We say a star has a mean place of (for example) $12^{\rm h}\,07^{\rm m}\,58^{\rm s}\!.09$ $-19^\circ\,44'\,37''\!.1$ "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).

4.6 Epoch

The word "epoch" just means a moment in time, and can be supplied in a variety of forms, using different calendar systems and timescales.

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*, is defined in such a way that its units are tropical years of about 365.2422 days and its timescale 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 timescale.

[4] An 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, 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. These details are supported by the SLALIB routines sla_DBJIN (decodes numbers from a character string, accepting an optional leading B or J), sla_KBJ (decides whether B or J depending on prefix or range) and sla_EPCO (converts one epoch to match another).

SLALIB has four routines for converting Besselian and Julian epochs into other forms. The functions sla_EPB2D and sla_EPJ2D convert Besselian and Julian epochs into MJD; the functions sla_EPB and sla_EPJ do the reverse. For example, to express B1950 as a Julian epoch:

```
DOUBLE PRECISION sla_EPJ,sla_EPB2D
 :
WRITE (*,'(1X,''J'',F10.5)') sla_EPJ(sla_EPB2D(1950D0))
```

(The answer is J1949.99979.)

4.7 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 $1''$/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 $[\,\alpha,\delta\,]$ and rate of change are not sufficient to pin down a star; 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 $[\,\alpha,\delta\,]$. The full specification for a star is $[\,\alpha,\delta\,]$, proper motions, equinox and epoch (plus something to identify which set of models for the precession *etc.* is being used – see the next section). For convenience, coordinates given in star catalogues are almost always adjusted to make the equinox and epoch the same – for example B1950 in the case of the SAO catalogue.

SLALIB provides one routine to handle proper motion on its own, sla_PM. Proper motion is also allowed for in various other routines as appropriate, for example sla_MAP and sla_FK425. Note that in all SLALIB routines 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 some catalogues the α component is in the form $\dot{\alpha}\cos\delta$ (*i.e.* angle on the sky).

4.8 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 3 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.

SLALIB does not provide facilities for handling parallax and radial-velocity on their own, but their contribution is allowed for in such routines as sla_PM, sla_MAP and sla_FK425. Catalogue 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.

4.9 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 Newtonian model is, in fact, highly 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 sla_SUBET or applied (for example to allow a pulsar timing-position to be plotted on a B1950 finding chart) by calling sla_ADDET; the E-terms vector itself can be obtained by calling sla_ETRMS. 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 $0''.2$.

No SLALIB routine is provided for calculating the aberration on its own, though the required velocity vectors can be generated using sla_EVP and sla_GEOC. Annual and diurnal aberration are allowed for where required, for example in sla_MAP *etc.* and sla_AOP *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.

4.10 Different Sorts of Mean Place

A particularly confusing aspect of published mean places 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 a mean place will involve relying on some existing star catalogue, for example the fundamental catalogues FK4 and FK5, and applying given 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, 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 grid 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.[5] 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, observational astronomers 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. FK4, in contrast, contains two important traps:

- The FK4 system rotates at about $0\overset{''}{.}5$ 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 frame 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).

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:

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

[5]This was true until the inception of the International Celestial Reference System, which is based on the idea of axes locked into the distant background. The coordinates of the extragalactic sources which realize these axes have no individual significance; there is a "no net rotation" condition which has to be satisfied each time any revisions take place.

4.11 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) 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 epoch, as might be required in a telescope-control program for example. Most of the steps can be carried out by calling a single SLALIB routines; there are other SLALIB routines which offer set-piece end-to-end transformation routines 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 SLALIB to transform mean places, here is a program which 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 $\alpha = 16^{\rm h}\,09^{\rm m}\,55\overset{\rm s}{.}13$, $\delta = -75^\circ\,59'\,27\overset{''}{.}2$, equinox 1900, epoch 1963.087, $\mu_\alpha = -0\overset{\rm s}{.}0312/y$, $\mu_\delta = +0\overset{''}{.}103/y$, parallax $= 0\overset{''}{.}062$, radial velocity $= -34.22$ km/s. The epoch of observation is 1994.35.

```
      IMPLICIT NONE
      DOUBLE PRECISION AS2R,S2R
      PARAMETER (AS2R=4.8481368110953599D-6,S2R=7.2722052166430399D-5)
      INTEGER J,I
      DOUBLE PRECISION 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),PXR,V(3)
      DOUBLE PRECISION sla_EPB,sla_EPJ2D

*  RA, Dec etc of example star
      CALL sla_DTF2R(16,09,55.13D0,R0,J)
      CALL sla_DAF2R(75,59,27.2D0,D0,J)
      D0=-D0
      EQ0=1900D0
      EP0=1963.087D0
      PR=-0.0312D0*S2R
      PD=+0.103D0*AS2R
      PX=0.062D0
      RV=-34.22D0
```

```
         EP1=1994.35D0

*  Epoch of observation as MJD and Besselian epoch
         EP1D=sla_EPJ2D(EP1)
         EP1B=sla_EPB(EP1D)

*  Space motion to the current epoch
         CALL sla_PM(R0,D0,PR,PD,PX,RV,EP0,EP1B,R1,D1)

*  Remove E-terms of aberration for the original equinox
         CALL sla_SUBET(R1,D1,EQ0,R2,D2)

*  Precess to B1950
         R3=R2
         D3=D2
         CALL sla_PRECES('FK4',EQ0,1950D0,R3,D3)

*  Add E-terms for the standard equinox B1950
         CALL sla_ADDET(R3,D3,1950D0,R4,D4)

*  Transform to J2000, no proper motion
         CALL sla_FK45Z(R4,D4,EP1B,R5,D5)

*  Parallax
         CALL sla_EVP(sla_EPJ2D(EP1),2000D0,W,EB,W,W)
         PXR=PX*AS2R
         CALL sla_DCS2C(R5,D5,V)
         DO I=1,3
            V(I)=V(I)-PXR*EB(I)
         END DO
         CALL sla_DCC2S(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`	*original equinox and epoch*
`16 09 54.155 -75 59 23.98`	*with space motion*
`16 09 54.229 -75 59 24.18`	*with old E-terms removed*
`16 16 28.213 -76 06 54.57`	*precessed to 1950.0*
`16 16 28.138 -76 06 54.37`	*with new E-terms*
`16 23 07.901 -76 13 58.87`	*J2000, current epoch*
`16 23 07.907 -76 13 58.92`	*including parallax*

Other remarks about the above (unusually 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 sla_FK425, after which a call to sla_PM 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, sla_FK425 and sla_FK45Z, 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 routines rather than just one is that the FK4 frame rotates relative to the background, whereas the FK5 frame 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 routines, sla_FK524 and sla_FK54Z, 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.

4.12 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 seldom significant. Its value at the limb of the Sun is about $1\overset{''}{.}74$; it falls off rapidly with distance from the Sun and has shrunk to about $0\overset{''}{.}02$ at an elongation of 20°.

As already described, the *annual aberration* is a function of the Earth's velocity relative to the solar system barycentre (available through the SLALIB routine sla_EVP) and produces shifts of up to about $20\overset{''}{.}5$.

The *precession/nutation*, from J2000 to the current epoch, is expressed by a rotation matrix which is available through the SLALIB routine sla_PRENUT.

The whole mean-to-apparent transformation can be done using the SLALIB routine sla_MAP. 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:

```
      IMPLICIT NONE
      DOUBLE PRECISION PI,PIBY2,D2R,S2R,AS2R
      PARAMETER (PI=3.141592653589793238462643D0)
      PARAMETER (D2R=PI/180D0,
     :           PIBY2=PI/2D0,
```

```
     :           S2R=PI/(12D0*3600D0),
     :           AS2R=PI/(180D0*3600D0))
      DOUBLE PRECISION RM,DM,PR,PD,DATE,RA,DA
      INTEGER J,IDS,IDE,ID,IYMDF(4),I
      DOUBLE PRECISION sla_EPJ2D

      CALL sla_DTF2R(02,31,49.8131D0,RM,J)
      CALL sla_DAF2R(89,15,50.661D0,DM,J)
      PR=+21.7272D0*S2R/100D0
      PD=-1.571D0*AS2R/100D0
      WRITE (*,'(1X,'//
     :            '''Polaris north polar distance (deg) 2096-2105''/)')
      WRITE (*,'(4X,''Date'',7X''NPD''/)')
      CALL sla_CLDJ(2096,1,1,DATE,J)
      IDS=NINT(DATE)
      CALL sla_CLDJ(2105,12,31,DATE,J)
      IDE=NINT(DATE)
      DO ID=IDS,IDE,10
         DATE=DBLE(ID)
         CALL sla_DJCAL(0,DATE,IYMDF,J)
         CALL sla_MAP(RM,DM,PR,PD,0D0,0D0,2000D0,DATE,RA,DA)
         WRITE (*,'(1X,I4,2I3.2,F9.5)') (IYMDF(I),I=1,3),(PIBY2-DA)/D2R
      END DO

      END
```

For cases where the transformation has to be repeated for different times or for more than one star, the straightforward sla_MAP 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 sla_MAPPA, and then to use either sla_MAPQKZ, when only the $[\,\alpha,\delta\,]$ is known, or sla_MAPQK, when full catalogue positions, including proper motion, parallax and radial velocity, are available. How frequently to call sla_MAPPA depends on the accuracy objectives; once per night will deliver sub-arcsecond accuracy for example.

The routines sla_AMP and sla_AMPQK allow the reverse transformation, from apparent to mean place.

4.13 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 apparent $[h, \delta]$ 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 sla_GMST);
4. add the observer's (east) longitude, giving the local mean sidereal time;
5. add the equation of the equinoxes (using sla_EQEQX).

The *equation of the equinoxes* ($= \Delta\psi \cos\epsilon$ plus small terms) is the effect of nutation on the sidereal time. Its value is typically a second or less. It is interesting to note that if the object of the exercise is to transform a mean place all the way into an observed place (very often the case), then the equation of the equinoxes and the longitude component of nutation can both be omitted, removing a great deal of computation. However, SLALIB follows the normal convention and works *via* the apparent place.

Note that for very precise work the observer's longitude should be corrected for *polar motion*. This can be done with sla_POLMO. The corrections are always less than about $0\overset{''}{.}3$, and are futile unless the position of the observer's telescope is known to better than a few metres.

Tables of observed and predicted UT1−UTC corrections and polar motion data are published every few weeks by the International Earth Rotation Service.

The transformation from apparent $[h, \delta]$ to *topocentric* $[h, \delta]$ consists of allowing for *diurnal aberration*. This effect, maximum amplitude $0\overset{''}{.}2$, was described earlier. There is no specific SLALIB routine for computing the diurnal aberration, though the routines sla_AOP *etc.* include it, and the required velocity vector can be determined by calling sla_GEOC.

The next stage is the major coordinate rotation from local equatorial coordinates $[h, \delta]$ into horizon coordinates. The SLALIB routines sla_E2H *etc.* can be used for this. For high-precision applications the mean geodetic latitude should be corrected for polar motion.

4.13.1 Refraction

The final correction is for atmospheric refraction. This effect, which depends on local meteorological conditions 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 $0\overset{\circ}{.}5$. The amount of refraction can by computed by calling the SLALIB routine sla_REFRO; 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 sla_REFRO 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 $-0\overset{''}{.}06$ respectively for a sea-level site. The two constants can be calculated for a given set of conditions by calling either sla_REFCO or sla_REFCOQ.

sla_REFCO works by calling sla_REFRO for two zenith distances and fitting A and B to match. The calculation is onerous, but delivers accurate results whatever the conditions. sla_REFCOQ uses a direct formulation of A and B and is much faster; it is slightly less accurate than sla_REFCO 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 $0\overset{''}{.}2$ 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 sla_REFZ or to an unrefracted $[\,x, y, z\,]$ by calling sla_REFV. Over most of the sky these two routines deliver almost identical results, but beyond $\zeta = 83°$ sla_REFV becomes unacceptably inaccurate while sla_REFZ remains usable. (However sla_REFV is significantly faster, which may be important in some applications.) SLALIB also provides a routine for computing the airmass, the function sla_AIRMAS.

The refraction "constants" returned by sla_REFCO and sla_REFCOQ 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 sla_REFCO several times it is more efficient to call sla_REFCO just once, for a selected "base" wavelength, and then to call sla_ATMDSP once for each wavelength of interest.

All the SLALIB refraction routines 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 no wavelength dependence, however. The choice of optical/IR or radio is made by specifying a wavelength greater than $100\mu m$ for the radio case.

4.13.2 Efficiency considerations

The complete apparent place to observed place transformation can be carried out by calling sla_AOP. 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 sla_AOPPA, the time can be updated by calling sla_AOPPAT, and sla_AOPQK can then be used to perform the apparent-to-observed transformation. The reverse transformation is available through sla_OAP and sla_OAPQK.

(*n.b.* These routines 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 UT1−UTC, omitting diurnal aberration and using sla_REFZ to apply the refraction.)

4.14 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 directions 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 frame was essentially arbitrary, and the close match to FK5 J2000 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 ICRF currently available is the Hipparcos catalogue. The extragalactic sources were not directly observable by the Hipparcos satellite and so the link from Hipparcos to ICRF was established through a variety of indirect techniques: VLBI and conventional interferometry of radio stars, photographic astrometry and so on. The Hipparcos frame is aligned to the ICRF 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 frame 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 frames 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 routines: sla_FK52H and sla_H2FK5, which transform FK5 positions and proper motions to the Hipparcos frame and *vice versa,* and sla_FK5HZ and sla_HFK5Z, 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).

4.15 Timescales

SLALIB provides for conversion between several timescales, and involves use of one or two others. The full list is as follows:

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

- UT: Universal Time
- GMST: Greenwich Mean Sidereal Time
- LAST: Local Apparent Sidereal Time
- TT: Terrestrial Time
- TDB: Barycentric Dynamical Time.

Three obsolete timescales 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.

4.15.1 Atomic Time: TAI

International Atomic Time TAI is a laboratory timescale. Its unit is the SI second, which is defined in terms of a defined 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 timescales are noticeably separated in epoch because of the build-up of leap seconds. At the time of writing, UTC lags about half a minute behind TAI.

For any given date, the difference TAI−UTC can be obtained by calling the SLALIB routine sla_DAT. Note, however, that an up-to-date copy of the routine 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 sla_DAT 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`.

4.15.2 Universal Time: UTC, UT1

Coordinated Universal Time UTC is the basis 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. The UTC second is the same as the SI second, as for TAI. In the long term, UTC keeps in step with the Sun. It does so even though the Earth's rotation is slightly variable (due to large scale movements of water and atmosphere among other things) by occasionally introducing a *leap second*.

Universal Time UT, or more specifically UT1, is in effect the mean solar time. It is continuous (*i.e.* there are no leap seconds) but has a variable rate because of the Earth's non-uniform rotation period. It is needed for computing the sidereal time, an essential part of pointing a telescope at a celestial source. To obtain UT1, you have to look up the value of UT1−UTC for the date concerned in tables published by the International Earth Rotation Service; this quantity, kept in the range $\pm 0\overset{s}{.}9$ by means of UTC leap seconds, is then added to the UTC. The quantity UT1−UTC, which typically changes by 1 or 2 ms per day, can only be obtained by observation, though seasonal trends are 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. On the average the solar day is slightly longer than the nominal 86,400 SI seconds and so 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

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. 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 $23^{\rm h}\,59^{\rm m}\,60\overset{s}{.}0$ and 1994 July 1 $00^{\rm h}\,00^{\rm m}\,00\overset{s}{.}0$ 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 routines sla_DAT and sla_DTT expect UTC in the form of an MJD, the meaning here is really a whole-number *date* rather than a time. Though the routines 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.

4.15.3 Sidereal Time: GMST, LAST

Sidereal Time is the "time of day" 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 routines sla_GMST and sla_GMSTA. There are, of course, no leap seconds in GMST, but the second 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 the timescale unsuitable for everything except predicting the apparent directions of celestial sources.

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 α. It can be obtained from the GMST by adding the east longitude (corrected for polar motion in precise work) 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 routine sla_EQEQX or, neglecting certain very small terms, by calling sla_NUTC and using the expression $\Delta\psi \cos\epsilon$.

4.15.4 Dynamical Time: TT, TDB

Dynamical time is the independent variable in the theories which describe the motions of bodies in the solar system. When you use published formulae which 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 you use must be in terms of one of the dynamical timescales. It is a common but understandable mistake to use UT directly, in which case the results will be about 1 minute out (in the present era).

It is not hard to see why such timescales 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 eventually the ephemeris and the ensemble of atomic clocks would drift apart. In effect, the ephemeris *is* a clock, with the bodies of the solar system the hands.

Only two of the dynamical timescales are of any great importance to observational astronomers, TT and TDB. (The obsolete timescale ET, ephemeris time, was more or less the same as TT.)

Terrestrial Time TT is the theoretical timescale of apparent geocentric ephemerides of solar system bodies. It applies, in principle, to an Earthbound clock, at sea-level, and for practical purposes it is tied to Atomic Time TAI through the formula TT = TAI + $32^{\rm s}\!.184$. In practice, therefore, the units of TT are ordinary SI seconds, and the offset of $32^{\rm s}\!.184$ with respect to TAI is fixed. The SLALIB routine sla_DTT returns TT−UTC for a given UTC (*n.b.* sla_DTT calls sla_DAT, and the latter must be an up-to-date version if recent leap seconds are to be taken into account).

Barycentric Dynamical Time TDB differs from TT by an amount which cycles back and forth by a millisecond or two due to relativistic effects. The variation is negligible for most purposes, but unless taken into account would swamp long-term analysis of pulse arrival times from the millisecond pulsars. It is a consequence of the TT clock being on the Earth rather than in empty space: the ellipticity of the Earth's orbit means that 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. By definition, TDB and TT differ only by periodic terms, and the main effect is a sinusoidal variation of amplitude $0^{\rm s}\!.0016$; the largest planetary terms are nearly two orders of magnitude smaller. The SLALIB routine sla_RCC provides a model of TDB-TT accurate to a few nanoseconds. There are other dynamical timescales, not supported by SLALIB routines, which include allowance also for the secular terms. These timescales gain on TT and TDB by about $0^{\rm s}\!.0013$/day.

For most purposes the more accessible TT is the timescale to use, for example when calling sla_PRENUT to generate a precession/nutation matrix or when calling sla_EVP to predict the Earth's position and velocity. For some purposes TDB is the correct timescale, for example when interrogating the JPL planetary ephemeris (see *Starlink User Note 87*), though in most cases TT will be near enough and will involve less computation.

Investigations of topocentric solar-system phenomena such as occultations and eclipses require solar time 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 ΔT = ET − UT is available via the routine sla_DT. For a given epoch (*e.g.* 1650) this returns an approximation to ΔT in seconds.

4.16 Calendars

The ordinary *Gregorian Calendar Date*, together with a time of day, can be used to express an epoch in any desired timescale. 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; 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 routines 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 timescales 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 routines for converting to and from dates in the Gregorian calendar. The routines sla_CLDJ and sla_CALDJ 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 routines sla_DJCL and sla_DJCAL both convert an MJD into calendar year, month, day and fraction of a day; the latter performs rounding to a specified precision, important to avoid dates like '`94 04 01.***`' appearing in messages. Some of SLALIB's low-precision ephemeris routines (sla_EARTH, sla_MOON and sla_ECOR) 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 sla_CALYD (which defaults years 0-99 into 1950-2049) or sla_CLYD (which covers the full range from prehistoric times).

4.17 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 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 routine sla_OBS 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 Fortran code returns the full name, longitude and latitude of a specified observatory:

```
CHARACTER IDENT*10,NAME*40
DOUBLE PRECISION W,P,H
 :
CALL sla_OBS(0,IDENT,NAME,W,P,H)
IF (NAME.EQ.'?') ...              (not recognized)
```

(Beware of the longitude sign convention, which is west +ve for historical reasons.) The following lists all the supported observatories:

```
  :
INTEGER N
 :
N=1
NAME=' '
DO WHILE (NAME.NE.'?')
   CALL sla_OBS(N,IDENT,NAME,W,P,H)
   IF (NAME.NE.'?') THEN
      WRITE (*,'(1X,I3,4X,A,4X,A)') N,IDENT,NAME
      N=N+1
   END IF
END DO
```

The routine sla_GEOC 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 be evaluating ATAN2 of the two numbers. A full 3-D vector description of the position and velocity of the observer is available through the routine sla_PVOBS. For a specified geodetic latitude, height above sea level, and local sidereal time, sla_PVOBS 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 $[\,\alpha,\delta\,]$ frame (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 routine sla_RVEROT, later.

4.18 Ephemerides

SLALIB includes routines 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 conversion routines are not strictly applicable to solar-system cases, though they are adequate for most practical purposes.

Earth/Sun ephemerides can be generated using the routine sla_EVP, which predicts Earth position and velocity with respect to both the solar-system barycentre and the Sun. Maximum velocity error is 0.42 metres per second; maximum heliocentric position error is 1600 km (about $2''$), with barycentric position errors about 4 times worse. (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 sla_DMOON, 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 sla_EARTH and sla_MOON. Both are single 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 sla_CLYD or sla_CALYD to get the required year and day). The sla_EARTH routine 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 sla_MOON. The positional accuracy is better than $30''$ in direction and 1000 km in distance.

Approximate ephemerides for all the major planets can be generated by calling sla_PLANET or sla_RDPLAN. These routines offer arcminute accuracy (much better for the inner planets and for Pluto) over a span of several millennia (but only ± 100 years for Pluto). The routine sla_PLANET 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 sla_PLANET can be used in a variety of ways according to the requirements of the application concerned. The routine sla_RDPLAN uses sla_PLANET and sla_DMOON 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 sla_PVOBS and subtracting the resulting 6-vector from the one produced by sla_DMOON, as is demonstrated by the following example:

```
*  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.

      IMPLICIT NONE
      CHARACTER NAME*40,SH,SD
      INTEGER J,I,IHMSF(4),IDMSF(4)
      DOUBLE PRECISION SLONGW,SLAT,H,DJUTC,FDUTC,DJUT1,DJTT,STL,
     :                 RMATN(3,3),PMM(6),PMT(6),RM,DM,PVO(6),TL
      DOUBLE PRECISION sla_DTT,sla_GMST,sla_EQEQX,sla_DRANRM

*  Get AAT longitude and latitude in radians and height in metres
      CALL sla_OBS(0,'AAT',NAME,SLONGW,SLAT,H)

*  UTC (1992 January 13, 11 13 59) to MJD
      CALL sla_CLDJ(1992,1,13,DJUTC,J)
      CALL sla_DTF2D(11,13,59.0D0,FDUTC,J)
      DJUTC=DJUTC+FDUTC

*  UT1 (UT1-UTC value of -0.152 sec is from IERS Bulletin B)
      DJUT1=DJUTC+(-0.152D0)/86400D0

*  TT
      DJTT=DJUTC+sla_DTT(DJUTC)/86400D0
```

```
*  Local apparent sidereal time
      STL=sla_GMST(DJUT1)-SLONGW+sla_EQEQX(DJTT)

*  Geocentric position/velocity of Moon (mean of date)
      CALL sla_DMOON(DJTT,PMM)

*  Nutation to true equinox of date
      CALL sla_NUT(DJTT,RMATN)
      CALL sla_DMXV(RMATN,PMM,PMT)
      CALL sla_DMXV(RMATN,PMM(4),PMT(4))

*  Report geocentric HA,Dec
      CALL sla_DCC2S(PMT,RM,DM)
      CALL sla_DR2TF(2,sla_DRANRM(STL-RM),SH,IHMSF)
      CALL sla_DR2AF(1,DM,SD,IDMSF)
      WRITE (*,'(1X,'' geocentric:'',2X,A,I2.2,2I3.2,''.'',I2.2,'//
     :                              '1X,A,I2.2,2I3.2,''.'',I1)')
     :                                                  SH,IHMSF,SD,IDMSF

*  Geocentric position of observer (true equator and equinox of date)
      CALL sla_PVOBS(SLAT,H,STL,PVO)

*  Place origin at observer
      DO I=1,6
         PMT(I)=PMT(I)-PVO(I)
      END DO

*  Allow for planetary aberration
      TL=499.004782D0*SQRT(PMT(1)**2+PMT(2)**2+PMT(3)**2)
      DO I=1,3
         PMT(I)=PMT(I)-TL*PMT(I+3)
      END DO

*  Report topocentric HA,Dec
      CALL sla_DCC2S(PMT,RM,DM)
      CALL sla_DR2TF(2,sla_DRANRM(STL-RM),SH,IHMSF)
      CALL sla_DR2AF(1,DM,SD,IDMSF)
      WRITE (*,'(1X,''topocentric:'',2X,A,I2.2,2I3.2,''.'',I2.2,'//
     :                              '1X,A,I2.2,2I3.2,''.'',I1)')
     :                                                  SH,IHMSF,SD,IDMSF
      END
```

The output produced is as follows:

```
 geocentric:  +03 06 55.59 +15 03 39.0
topocentric:  +03 09 23.79 +15 40 51.5
```

(An easier but less instructive method of estimating the topocentric apparent place of the Moon is to call the routine sla_RDPLAN.)

As an example of using sla_PLANET, 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:

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

      IMPLICIT NONE

      DOUBLE PRECISION SEPMIN,DJD0,FD,DJD,DJDM,DF,PV(6),RMATP(3,3),
     :                 PVM(6),PVE(6),TL,RV,DV,RJ,DJ,SEP
      INTEGER IHOUR,IMIN,J,I,IHMIN,IMMIN
      DOUBLE PRECISION sla_EPJ,sla_DSEP


*  Search for closest approach on the given day
      DJD0=1720859.5D0
      SEPMIN=1D10
      DO IHOUR=20,22
         DO IMIN=0,59
            CALL sla_DTF2D(IHOUR,IMIN,0D0,FD,J)

*        Julian date and MJD
            DJD=DJD0+FD
            DJDM=DJD-2400000.5D0

*        Earth to Moon (mean of date)
            CALL sla_DMOON(DJDM,PV)

*        Precess Moon position to J2000
            CALL sla_PRECL(sla_EPJ(DJDM),2000D0,RMATP)
            CALL sla_DMXV(RMATP,PV,PVM)

*        Sun to Earth-Moon Barycentre (mean J2000)
            CALL sla_PLANET(DJDM,3,PVE,J)

*        Correct from EMB to Earth
            DO I=1,3
               PV(I)=PVE(I)-0.012150581D0*PVM(I)
            END DO

*        Sun to Venus
            CALL sla_PLANET(DJDM,2,PV,J)

*        Earth to Venus
            DO I=1,6
```

```
               PV(I)=PV(I)-PVE(I)
            END DO

*        Light time to Venus (sec)
            TL=499.004782D0*SQRT((PV(1)-PVE(1))**2+
     :                            (PV(2)-PVE(2))**2+
     :                            (PV(3)-PVE(3))**2)

*        Extrapolate backwards in time by that much
            DO I=1,3
               PV(I)=PV(I)-TL*PV(I+3)
            END DO

*       To RA,Dec
            CALL sla_DCC2S(PV,RV,DV)

*        Same for Jupiter
            CALL sla_PLANET(DJDM,5,PV,J)
            DO I=1,6
               PV(I)=PV(I)-PVE(I)
            END DO
            TL=499.004782D0*SQRT((PV(1)-PVE(1))**2+
     :                            (PV(2)-PVE(2))**2+
     :                            (PV(3)-PVE(3))**2)
            DO I=1,3
               PV(I)=PV(I)-TL*PV(I+3)
            END DO
            CALL sla_DCC2S(PV,RJ,DJ)

*        Separation (arcsec)
            SEP=sla_DSEP(RV,DV,RJ,DJ)

*        Keep if smallest so far
            IF (SEP.LT.SEPMIN) THEN
               IHMIN=IHOUR
               IMMIN=IMIN
               SEPMIN=SEP
            END IF
         END DO
      END DO

*  Report
      WRITE (*,'(1X,I2.2,'':'',I2.2,F6.1)') IHMIN,IMMIN,
     :                                       206264.8062D0*SEPMIN

      END
```

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

```
21:19  33.7
```

For comparison, accurate predictions based on the JPL DE 102 ephemeris give a separation about $8''$ less than the above estimate, occurring about half an hour earlier (see *Sky and Telescope,* April 1987, p 357).

The following program demonstrates sla_RDPLAN.

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

      IMPLICIT NONE
      CHARACTER PNAMES(0:9)*7,B*80,S
      INTEGER I,NP,IY,J,IM,ID,IHMSF(4),IDMSF(4)
      DOUBLE PRECISION R2AS,FD,DJM,ELONG,PHI,RA,DEC,DIAM
      PARAMETER (R2AS=206264.80625D0)
      DATA PNAMES / 'Sun','Mercury','Venus','Moon','Mars','Jupiter',
     :              'Saturn','Uranus','Neptune', 'Pluto' /


*  Loop until 'end' typed
      B=' '
      DO WHILE (B.NE.'END'.AND.B.NE.'end')

*     Get date, time and observer's location
         PRINT *,'Date? (Y,M,D, Gregorian)'
         READ (*,'(A)') B
         IF (B.NE.'END'.AND.B.NE.'end') THEN
            I=1
            CALL sla_INTIN(B,I,IY,J)
            CALL sla_INTIN(B,I,IM,J)
            CALL sla_INTIN(B,I,ID,J)
            PRINT *,'Time? (H,M,S, dynamical)'
            READ (*,'(A)') B
            I=1
            CALL sla_DAFIN(B,I,FD,J)
            FD=FD*2.3873241463784300365D0
            CALL sla_CLDJ(IY,IM,ID,DJM,J)
            DJM=DJM+FD
            PRINT *,'Longitude? (D,M,S, east +ve)'
            READ (*,'(A)') B
            I=1
            CALL sla_DAFIN(B,I,ELONG,J)
            PRINT *,'Latitude? (D,M,S, (geodetic)'
            READ (*,'(A)') B
            I=1
```

```
            CALL sla_DAFIN(B,I,PHI,J)

*        Loop planet by planet
            DO NP=0,8

*           Get RA,Dec and diameter
               CALL sla_RDPLAN(DJM,NP,ELONG,PHI,RA,DEC,DIAM)

*           One line of report
               CALL sla_DR2TF(2,RA,S,IHMSF)
               CALL sla_DR2AF(1,DEC,S,IDMSF)
               WRITE (*,
     : '(1X,A,2X,3I3.2,''.'',I2.2,2X,A,I2.2,2I3.2,''.'',I1,F8.1)')
     :                           PNAMES(NP),IHMSF,S,IDMSF,R2AS*DIAM

*           Next planet
            END DO
            PRINT *,' '
         END IF

*     Next case
      END DO

      END
```

Entering the following data (for 1927 June 29 at $5^{\mathrm{h}}\,25^{\mathrm{m}}$ 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 17.5  1887.8
Mercury   08 08 58.62  +19 20 57.3     9.3
Venus     09 38 53.64  +15 35 32.9    22.8
Moon      06 28 18.30  +23 18 37.3  1903.9
Mars      09 06 49.34  +17 52 26.7     4.0
Jupiter   00 11 12.06  -00 10 57.5    41.1
Saturn    16 01 43.34  -18 36 55.9    18.2
Uranus    00 13 33.53  +00 39 36.0     3.5
Neptune   09 49 35.75  +13 38 40.8     2.2
Pluto     07 05 29.50  +21 25 04.2      .1
```

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 sla_RDPLAN and sla_PLANET.

The SLALIB planetary-prediction routines that work with orbital elements are sla_PLANTE (the orbital-elements equivalent of sla_RDPLAN), which predicts the topocentric $[\alpha, \delta]$, and sla_PLANEL (the orbital-elements equivalent of sla_PLANET), which predicts the heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ with respect to the J2000 equinox and equator. In addition, the routine sla_PV2EL does the inverse of sla_PLANEL, transforming $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ into *osculating elements.*

Osculating elements describe the unperturbed 2-body orbit. This is a good 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 even just 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 routine sla_PERTEL does. Once sla_PERTEL has been called, to provide osculating elements close to the required date, the elements can be passed to sla_PLANEL or sla_PLANTE 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 at which the elements were correct
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 sla_PLANEL the mass of the planet. If it is not available, it should be claculated from $n^2a^3 = k^2(1+m)$, where $k = 0.01720209895$ and m is the mass of the planet ($M_\odot = 1$); a is in AU.

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 routines 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 routines supplement these elements with certain redundant values in order to avoid unnecessary recomputation when the elements are next used.

The routines sla_EL2UE and sla_UE2EL transform conventional elements into the universal form and *vice versa.* The routine sla_PV2UE takes an $[\,x, y, z, \dot{x}, \dot{y}, \dot{z}\,]$ and forms the set of universal elements; sla_UE2PV takes a set of universal elements and predicts the $[\,x, y, z, \dot{x}, \dot{y}, \dot{z}\,]$ for a specified epoch. The routine sla_PERTUE provides updated universal elements, taking into account perturbations from the major planets.

4.19 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 routines 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 routine sla_RVEROT corrects for the diurnal rotation of the observer around the Earth's axis. This is always less than 0.5 km/s.

No specific routine is provided to correct a radial velocity from geocentric to heliocentric, but this can easily be done by calling sla_EVP as follows (array declarations *etc.* omitted):

```
         :
*  Star vector, J2000
      CALL sla_DCS2C(RM,DM,V)

*  Earth/Sun velocity and position, J2000
      CALL sla_EVP(TDB,2000D0,DVB,DPB,DVH,DPH)

*  Radial velocity correction due to Earth orbit (km/s)
      VCORB = -sla_DVDV(V,DVH)*149.597870D6
         :
```

The maximum value of this correction is the Earth's orbital speed of about 30 km/s. A related routine, sla_ECOR, computes the light-time correction with respect to the Sun. It would be used when reducing observations of a rapid variable-star for instance. Note, however, that the accuracy objectives for pulsar work are beyond the scope of these SLALIB routines, and even the superior sla_EVP routine is unsuitable for arrival-time calculations of better than 25 millisecond accuracy.

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 routine sla_RVLSRD and to the adopted kinematical LSR by sla_RVLSRK. See the individual specifications for these routines 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 sla_RVGALC. 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 routine for computing the radial velocity correction in this case is sla_RVLG. 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.

4.20 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 routines 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 sla_S2TP (single precision) or sla_DS2TP (double precision). The reverse transformation, where the $[\,x, y\,]$ is known and we wish to find the $[\,\alpha, \delta\,]$, is carried out by calling sla_TP2S or sla_DTP2S. 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 sla_TPS2C or sla_DTPS2C. (All of these transformations apply not just to

$[\alpha, \delta]$ but to other spherical coordinate systems, of course.) Equivalent (and faster) routines are provided which work directly in $[x, y, z]$ instead of spherical coordinates: sla_V2TP and sla_DV2TP, sla_TP2V and sla_DTP2V, sla_TPV2C and sla_DTPV2C.

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 routine sla_PCD applies the distortion to an $[x, y]$ and sla_UNPCD 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 routines which can be put together to build a simple plate-reduction program. The heart of the group is sla_FITXY, 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:

$$x_p = a + bx_m + cy_m$$

$$y_p = d + ex_m + fy_m$$

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–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, sla_PXY 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 routine sla_XY2XY transforms one $[x, y]$ into another using the linear model. A model can be inverted by calling sla_INVF, and decomposed into zero points, scales, x/y nonperpendicularity and orientation by calling sla_DCMPF.

4.21 Numerical Methods

SLALIB contains a small number of simple, general-purpose numerical-methods routines. They have no specific connection with positional astronomy but have proved useful in applications to do with simulation and fitting.

At the heart of many simulation programs is the generation of pseudo-random numbers, evenly distributed in a given range: sla_RANDOM does this. Pseudo-random normal deviates, or "Gaussian residuals", are often required to simulate noise and can be generated by means of the function sla_GRESID. Neither routine will pass super-sophisticated statistical tests, but they work adequately for most practical purposes and avoid the need to call non-standard library routines peculiar to one sort of computer.

Applications which perform a least-squares fit using a traditional normal-equations methods can accomplish the required matrix-inversion by calling either sla_SMAT (single precision) or sla_DMAT (double). A generally better way to perform such fits is to use singular value decomposition. SLALIB provides a routine to do the decomposition itself, sla_SVD, and two routines to use the results: sla_SVDSOL generates the solution, and sla_SVDCOV produces the covariance matrix. A simple demonstration of the use of the SLALIB SVD routines is given below. It generates 500 simulated data points and fits them to a model which has 4 unknown coefficients. (The arrays in the example are sized to accept up to 1000 points and 20 unknowns.) The model is:

$$y = C_1 + C_2 x + C_3 sin x + C_4 cos x$$

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, $\sigma = 5.0$, is added to each "observation".

```
      IMPLICIT NONE

*  Sizes of arrays, physical and logical
      INTEGER MP,NP,NC,M,N
      PARAMETER (MP=1000,NP=10,NC=20,M=500,N=4)

*  The unknowns we are going to solve for
      DOUBLE PRECISION C1,C2,C3,C4
      PARAMETER (C1=50D0,C2=-2D0,C3=-10D0,C4=25D0)

*  Arrays
      DOUBLE PRECISION A(MP,NP),W(NP),V(NP,NP),
     :                 WORK(NP),B(MP),X(NP),CVM(NC,NC)

      DOUBLE PRECISION VAL,BF1,BF2,BF3,BF4,SD2,D,VAR
      REAL sla_GRESID
      INTEGER I,J

*  Fill the design matrix
      DO I=1,M

*     Dummy independent variable
         VAL=DBLE(I)/10D0

*     The basis functions
         BF1=1D0
         BF2=VAL
         BF3=SIN(VAL)
         BF4=COS(VAL)

*     The observed value, including deliberate Gaussian noise
         B(I)=C1*BF1+C2*BF2+C3*BF3+C4*BF4+DBLE(sla_GRESID(5.0))
```

```
*     Fill one row of the design matrix
         A(I,1)=BF1
         A(I,2)=BF2
         A(I,3)=BF3
         A(I,4)=BF4
      END DO

*  Factorize the design matrix, solve and generate covariance matrix
      CALL sla_SVD(M,N,MP,NP,A,W,V,WORK,J)
      CALL sla_SVDSOL(M,N,MP,NP,B,A,W,V,WORK,X)
      CALL sla_SVDCOV(N,NP,NC,W,V,WORK,CVM)

*  Compute the variance
      SD2=0D0
      DO I=1,M
         VAL=DBLE(I)/10D0
         BF1=1D0
         BF2=VAL
         BF3=SIN(VAL)
         BF4=COS(VAL)
         D=B(I)-(X(1)*BF1+X(2)*BF2+X(3)*BF3+X(4)*BF4)
         SD2=SD2+D*D
      END DO
      VAR=SD2/DBLE(M)

*  Report the RMS and the solution
      WRITE (*,'(1X,''RMS ='',F5.2/)') SQRT(VAR)
      DO I=1,N
         WRITE (*,'(1X,''C'',I1,'' ='',F7.3,'' +/-'',F6.3)')
     :                                    I,X(I),SQRT(VAR*CVM(I,I))
      END DO
      END
```

The program produces the following output:

```
RMS = 4.88

C1 = 50.192 +/- 0.439
C2 = -2.002 +/- 0.015
C3 = -9.771 +/- 0.310
C4 = 25.275 +/- 0.310
```

In this above example, essentially identical results would be obtained if the more commonplace normal-equations method had been used, and the large 1000×20 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 Jth 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 routines sla_COMBN and sla_PERMUT 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 routine is called repeatedly.

5 SUMMARY OF CALLS

The basic trigonometrical and numerical facilities are supplied in both single and double precision versions. Most of the more esoteric position and time routines use double precision arguments only, even in cases where single precision would normally be adequate in practice. Certain routines 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 routines uses a mixture of single and double precision arguments; each routine is either wholly single precision or wholly double precision.

In the classified list, below, *subroutine* subprograms are those whose names and argument lists are preceded by 'CALL', whereas *function* subprograms are those beginning 'R=' (when the result is REAL) or 'D=' (when the result is DOUBLE PRECISION).

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 works on one sort of machine but not on another.

String Decoding

CALL sla_INTIN (STRING, NSTRT, IRESLT, JFLAG)
  Convert free-format string into integer

CALL sla_FLOTIN (STRING, NSTRT, RESLT, JFLAG)
CALL sla_DFLTIN (STRING, NSTRT, DRESLT, JFLAG)
  Convert free-format string into floating-point number

CALL sla_AFIN (STRING, NSTRT, RESLT, JFLAG)
CALL sla_DAFIN (STRING, NSTRT, DRESLT, JFLAG)
  Convert free-format string from deg,arcmin,arcsec to radians

Sexagesimal Conversions

CALL sla_CTF2D (IHOUR, IMIN, SEC, DAYS, J)
CALL sla_DTF2D (IHOUR, IMIN, SEC, DAYS, J)
  Hours, minutes, seconds to days

CALL sla_CD2TF (NDP, DAYS, SIGN, IHMSF)
CALL sla_DD2TF (NDP, DAYS, SIGN, IHMSF)
  Days to hours, minutes, seconds

CALL sla_CTF2R (IHOUR, IMIN, SEC, RAD, J)
CALL sla_DTF2R (IHOUR, IMIN, SEC, RAD, J)
  Hours, minutes, seconds to radians

CALL sla_CR2TF (NDP, ANGLE, SIGN, IHMSF)
CALL sla_DR2TF (NDP, ANGLE, SIGN, IHMSF)
  Radians to hours, minutes, seconds

CALL sla_CAF2R (IDEG, IAMIN, ASEC, RAD, J)
CALL sla_DAF2R (IDEG, IAMIN, ASEC, RAD, J)
    Degrees, arcminutes, arcseconds to radians

CALL sla_CR2AF (NDP, ANGLE, SIGN, IDMSF)
CALL sla_DR2AF (NDP, ANGLE, SIGN, IDMSF)
    Radians to degrees, arcminutes, arcseconds

Angles, Vectors and Rotation Matrices

R = sla_RANGE (ANGLE)
D = sla_DRANGE (ANGLE)
    Normalize angle into range $\pm\pi$

R = sla_RANORM (ANGLE)
D = sla_DRANRM (ANGLE)
    Normalize angle into range $0-2\pi$

CALL sla_CS2C (A, B, V)
CALL sla_DCS2C (A, B, V)
    Spherical coordinates to $[\,x, y, z\,]$

CALL sla_CC2S (V, A, B)
CALL sla_DCC2S (V, A, B)
    $[\,x, y, z\,]$ to spherical coordinates

R = sla_VDV (VA, VB)
D = sla_DVDV (VA, VB)
    Scalar product of two 3-vectors

CALL sla_VXV (VA, VB, VC)
CALL sla_DVXV (VA, VB, VC)
    Vector product of two 3-vectors

CALL sla_VN (V, UV, VM)
CALL sla_DVN (V, UV, VM)
    Normalize a 3-vector also giving the modulus

R = sla_SEP (A1, B1, A2, B2)
D = sla_DSEP (A1, B1, A2, B2)
    Angle between two points on a sphere

R = sla_BEAR (A1, B1, A2, B2)
D = sla_DBEAR (A1, B1, A2, B2)
    Direction of one point on a sphere seen from another

R = sla_PAV (V1, V2)
D = sla_DPAV (V1, V2)
    Position-angle of one $[\,x, y, z\,]$ with respect to another

CALL sla_EULER (ORDER, PHI, THETA, PSI, RMAT)
CALL sla_DEULER (ORDER, PHI, THETA, PSI, RMAT)
    Form rotation matrix from three Euler angles

CALL sla_AV2M (AXVEC, RMAT)
CALL sla_DAV2M (AXVEC, RMAT)
Form rotation matrix from axial vector

CALL sla_M2AV (RMAT, AXVEC)
CALL sla_DM2AV (RMAT, AXVEC)
Determine axial vector from rotation matrix

CALL sla_MXV (RM, VA, VB)
CALL sla_DMXV (DM, VA, VB)
Rotate vector forwards

CALL sla_IMXV (RM, VA, VB)
CALL sla_DIMXV (DM, VA, VB)
Rotate vector backwards

CALL sla_MXM (A, B, C)
CALL sla_DMXM (A, B, C)
Product of two 3x3 matrices

CALL sla_CS2C6 (A, B, R, AD, BD, RD, V)
CALL sla_DS2C6 (A, B, R, AD, BD, RD, V)
Conversion of position and velocity in spherical coordinates to Cartesian coordinates

CALL sla_CC62S (V, A, B, R, AD, BD, RD)
CALL sla_DC62S (V, A, B, R, AD, BD, RD)
Conversion of position and velocity in Cartesian coordinates to spherical coordinates

Calendars

CALL sla_CLDJ (IY, IM, ID, DJM, J)
Gregorian Calendar to Modified Julian Date

CALL sla_CALDJ (IY, IM, ID, DJM, J)
Gregorian Calendar to Modified Julian Date, permitting century default

CALL sla_DJCAL (NDP, DJM, IYMDF, J)
Modified Julian Date to Gregorian Calendar, in a form convenient for formatted output

CALL sla_DJCL (DJM, IY, IM, ID, FD, J)
Modified Julian Date to Gregorian Year, Month, Day, Fraction

CALL sla_CALYD (IY, IM, ID, NY, ND, J)
Calendar to year and day in year, permitting century default

CALL sla_CLYD (IY, IM, ID, NY, ND, J)
Calendar to year and day in year

D = sla_EPB (DATE)
Modified Julian Date to Besselian Epoch

D = sla_EPB2D (EPB)
Besselian Epoch to Modified Julian Date

D = sla_EPJ (DATE)
Modified Julian Date to Julian Epoch

D = sla_EPJ2D (EPJ)
Julian Epoch to Modified Julian Date

Timescales

D = sla_GMST (UT1)
Conversion from Universal Time to sidereal time

D = sla_GMSTA (DATE, UT1)
Conversion from Universal Time to sidereal time, rounding errors minimized

D = sla_EQEQX (DATE)
Equation of the equinoxes

D = sla_DAT (DJU)
Offset of Atomic Time from Coordinated Universal Time: TAI−UTC

D = sla_DT (EPOCH)
Approximate offset between dynamical time and universal time

D = sla_DTT (DJU)
Offset of Terrestrial Time from Coordinated Universal Time: TT−UTC

D = sla_RCC (TDB, UT1, WL, U, V)
Relativistic clock correction: TDB−TT

Precession and Nutation

CALL sla_NUT (DATE, RMATN)
Nutation matrix

CALL sla_NUTC (DATE, DPSI, DEPS, EPS0)
Longitude and obliquity components of nutation, and mean obliquity

CALL sla_PREC (EP0, EP1, RMATP)
Precession matrix (IAU)

CALL sla_PRECL (EP0, EP1, RMATP)
Precession matrix (suitable for long periods)

CALL sla_PRENUT (EPOCH, DATE, RMATPN)
Combined precession/nutation matrix

CALL sla_PREBN (BEP0, BEP1, RMATP)
Precession matrix, old system

CALL sla_PRECES (SYSTEM, EP0, EP1, RA, DC)
Precession, in either the old or the new system

Proper Motion

CALL sla_PM (R0, D0, PR, PD, PX, RV, EP0, EP1, R1, D1)
Adjust for proper motion

FK4/FK5/Hipparcos Conversions

CALL sla_FK425 (R1950, D1950, DR1950, DD1950, P1950, V1950,
R2000, D2000, DR2000, DD2000, P2000, V2000)
Convert B1950.0 FK4 star data to J2000.0 FK5

CALL sla_FK45Z (R1950, D1950, EPOCH, R2000, D2000)
Convert B1950.0 FK4 position to J2000.0 FK5 assuming zero FK5 proper motion and no parallax

CALL sla_FK524 (R2000, D2000, DR2000, DD2000, P2000, V2000,
R1950, D1950, DR1950, DD1950, P1950, V1950)
Convert J2000.0 FK5 star data to B1950.0 FK4

CALL sla_FK54Z (R2000, D2000, BEPOCH, R1950, D1950, DR1950, DD1950)
Convert J2000.0 FK5 position to B1950.0 FK4 assuming zero FK5 proper motion and no parallax

CALL sla_FK52H (R5, D5, DR5, DD5, RH, DH, DRH, DDH)
Convert J2000.0 FK5 star data to Hipparcos

CALL sla_FK5HZ (R5, D5, EPOCH, RH, DH)
Convert J2000.0 FK5 position to Hipparcos assuming zero Hipparcos proper motion

CALL sla_H2FK5 (RH, DH, DRH, DDH, R5, D5, DR5, DD5)
Convert Hipparcos star data to J2000.0 FK5

CALL sla_HFK5Z (RH, DH, EPOCH, R5, D5, DR5, DD5)
Convert Hipparcos position to J2000.0 FK5 assuming zero Hipparcos proper motion

CALL sla_DBJIN (STRING, NSTRT, DRESLT, J1, J2)
Like sla_DFLTIN but with extensions to accept leading 'B' and 'J'

CALL sla_KBJ (JB, E, K, J)
Select epoch prefix 'B' or 'J'

D = sla_EPCO (K0, K, E)
Convert an epoch into the appropriate form – 'B' or 'J'

Elliptic Aberration

CALL sla_ETRMS (EP, EV)
E-terms

CALL sla_SUBET (RC, DC, EQ, RM, DM)
Remove the E-terms

CALL sla_ADDET (RM, DM, EQ, RC, DC)
Add the E-terms

Geographical and Geocentric Coordinates

CALL sla_OBS (NUMBER, ID, NAME, WLONG, PHI, HEIGHT)
Interrogate list of observatory parameters

CALL sla_GEOC (P, H, R, Z)
Convert geodetic position to geocentric

CALL sla_POLMO (ELONGM, PHIM, XP, YP, ELONG, PHI, DAZ)
Polar motion

CALL sla_PVOBS (P, H, STL, PV)
Position and velocity of observatory

Apparent and Observed Place

CALL sla_MAP (RM, DM, PR, PD, PX, RV, EQ, DATE, RA, DA)
Mean place to geocentric apparent place

CALL sla_MAPPA (EQ, DATE, AMPRMS)
Precompute mean to apparent parameters

CALL sla_MAPQK (RM, DM, PR, PD, PX, RV, AMPRMS, RA, DA)
Mean to apparent using precomputed parameters

CALL sla_MAPQKZ (RM, DM, AMPRMS, RA, DA)
Mean to apparent using precomputed parameters, for zero proper motion, parallax and radial velocity

CALL sla_AMP (RA, DA, DATE, EQ, RM, DM)
Geocentric apparent place to mean place

CALL sla_AMPQK (RA, DA, AOPRMS, RM, DM)
Apparent to mean using precomputed parameters

CALL sla_AOP (RAP, DAP, UTC, DUT, ELONGM, PHIM, HM, XP, YP,
TDK, PMB, RH, WL, TLR, AOB, ZOB, HOB, DOB, ROB)
Apparent place to observed place

CALL sla_AOPPA (UTC, DUT, ELONGM, PHIM, HM, XP, YP,
TDK, PMB, RH, WL, TLR, AOPRMS)
Precompute apparent to observed parameters

CALL sla_AOPPAT (UTC, AOPRMS)
Update sidereal time in apparent to observed parameters

CALL sla_AOPQK (RAP, DAP, AOPRMS, AOB, ZOB, HOB, DOB, ROB)
Apparent to observed using precomputed parameters

CALL sla_OAP (TYPE, OB1, OB2, UTC, DUT, ELONGM, PHIM, HM, XP, YP,
TDK, PMB, RH, WL, TLR, RAP, DAP)
Observed to apparent

CALL sla_OAPQK (TYPE, OB1, OB2, AOPRMS, RA, DA)
Observed to apparent using precomputed parameters

Azimuth and Elevation

CALL sla_ALTAZ (HA, DEC, PHI,
AZ, AZD, AZDD, EL, ELD, ELDD, PA, PAD, PADD)
Positions, velocities *etc.* for an altazimuth mount

CALL sla_E2H (HA, DEC, PHI, AZ, EL)
CALL sla_DE2H (HA, DEC, PHI, AZ, EL)
$[\, h, \delta \,]$ to $[\, Az, El \,]$

CALL sla_H2E (AZ, EL, PHI, HA, DEC)
CALL sla_DH2E (AZ, EL, PHI, HA, DEC)
$[\, Az, El \,]$ to $[\, h, \delta \,]$

CALL sla_PDA2H (P, D, A, H1, J1, H2, J2)
Hour Angle corresponding to a given azimuth

CALL sla_PDQ2H (P, D, Q, H1, J1, H2, J2)
Hour Angle corresponding to a given parallactic angle

D = sla_PA (HA, DEC, PHI)
$[\, h, \delta \,]$ to parallactic angle

D = sla_ZD (HA, DEC, PHI)
$[\, h, \delta \,]$ to zenith distance

Refraction and Air Mass

CALL sla_REFRO (ZOBS, HM, TDK, PMB, RH, WL, PHI, TLR, EPS, REF)
Change in zenith distance due to refraction

CALL sla_REFCO (HM, TDK, PMB, RH, WL, PHI, TLR, EPS, REFA, REFB)
Constants for simple refraction model (accurate)

CALL sla_REFCOQ (TDK, PMB, RH, WL, REFA, REFB)
Constants for simple refraction model (fast)

CALL sla_ATMDSP (TDK, PMB, RH, WL1, REFA1, REFB1, WL2, REFA2, REFB2)
Adjust refraction constants for colour

CALL sla_REFZ (ZU, REFA, REFB, ZR)
Unrefracted to refracted ZD, simple model

CALL sla_REFV (VU, REFA, REFB, VR)
Unrefracted to refracted $[\, Az, El \,]$ vector, simple model

D = sla_AIRMAS (ZD)
Air mass

Ecliptic Coordinates

CALL sla_ECMAT (DATE, RMAT)
Equatorial to ecliptic rotation matrix

CALL sla_EQECL (DR, DD, DATE, DL, DB)
J2000.0 'FK5' to ecliptic coordinates

CALL sla_ECLEQ (DL, DB, DATE, DR, DD)
Ecliptic coordinates to J2000.0 'FK5'

Galactic Coordinates

CALL sla_EG50 (DR, DD, DL, DB)
B1950.0 'FK4' to galactic

CALL sla_GE50 (DL, DB, DR, DD)
Galactic to B1950.0 'FK4'

CALL sla_EQGAL (DR, DD, DL, DB)
J2000.0 'FK5' to galactic

CALL sla_GALEQ (DL, DB, DR, DD)
Galactic to J2000.0 'FK5'

Supergalactic Coordinates

CALL sla_GALSUP (DL, DB, DSL, DSB)
Galactic to supergalactic

CALL sla_SUPGAL (DSL, DSB, DL, DB)
Supergalactic to galactic

Ephemerides

CALL sla_DMOON (DATE, PV)
Approximate geocentric position and velocity of the Moon

CALL sla_EARTH (IY, ID, FD, PV)
Approximate heliocentric position and velocity of the Earth

CALL sla_EVP (DATE, DEQX, DVB, DPB, DVH, DPH)
Barycentric and heliocentric velocity and position of the Earth

CALL sla_MOON (IY, ID, FD, PV)
Approximate geocentric position and velocity of the Moon

CALL sla_PLANET (DATE, NP, PV, JSTAT)
Approximate heliocentric position and velocity of a planet

CALL sla_RDPLAN (DATE, NP, ELONG, PHI, RA, DEC, DIAM)
Approximate topocentric apparent place of a planet

CALL sla_PLANEL (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

CALL sla_PLANTE (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 elements are known

CALL sla_PV2EL (PV, DATE, PMASS, JFORMR, JFORM, EPOCH, ORBINC,
ANODE, PERIH, AORQ, E, AORL, DM, JSTAT)
Orbital elements of a planet from instantaneous position and velocity

CALL sla_PERTEL (JFORM, DATE0, DATE1,
EPOCH0, ORBI0, ANODE0, PERIH0, AORQ0, E0, AM0,
EPOCH1, ORBI1, ANODE1, PERIH1, AORQ1, E1, AM1,
JSTAT)
Update elements by applying perturbations

CALL sla_EL2UE (DATE, JFORM, EPOCH, ORBINC, ANODE,
PERIH, AORQ, E, AORL, DM,
U, JSTAT)
Transform conventional elements to universal elements

CALL sla_UE2EL (U, JFORMR,
JFORM, EPOCH, ORBINC, ANODE, PERIH,
AORQ, E, AORL, DM, JSTAT)
Transform universal elements to conventional elements

CALL sla_PV2UE (PV, DATE, PMASS, U, JSTAT)
Package a position and velocity for use as universal elements

CALL sla_UE2PV (DATE, U, PV, JSTAT)
Extract the position and velocity from universal elements

CALL sla_PERTUE (DATE, U, JSTAT)
Update universal elements by applying perturbations

R = sla_RVEROT (PHI, RA, DA, ST)
Velocity component due to rotation of the Earth

CALL sla_ECOR (RM, DM, IY, ID, FD, RV, TL)
Components of velocity and light time due to Earth orbital motion

R = sla_RVLSRD (R2000, D2000)
Velocity component due to solar motion wrt dynamical LSR

R = sla_RVLSRK (R2000, D2000)
Velocity component due to solar motion wrt kinematical LSR

R = sla_RVGALC (R2000, D2000)
Velocity component due to rotation of the Galaxy

R = sla_RVLG (R2000, D2000)
Velocity component due to rotation and translation of the Galaxy, relative to the mean motion of the local group

Astrometry

CALL sla_S2TP (RA, DEC, RAZ, DECZ, XI, ETA, J)
CALL sla_DS2TP (RA, DEC, RAZ, DECZ, XI, ETA, J)
Transform spherical coordinates into tangent plane

CALL sla_V2TP (V, V0, XI, ETA, J)
CALL sla_DV2TP (V, V0, XI, ETA, J)
Transform $[\,x, y, z\,]$ into tangent plane coordinates

CALL sla_DTP2S (XI, ETA, RAZ, DECZ, RA, DEC)
CALL sla_TP2S (XI, ETA, RAZ, DECZ, RA, DEC)
Transform tangent plane coordinates into spherical coordinates

CALL sla_DTP2V (XI, ETA, V0, V)
CALL sla_TP2V (XI, ETA, V0, V)
Transform tangent plane coordinates into $[\,x, y, z\,]$

CALL sla_DTPS2C (XI, ETA, RA, DEC, RAZ1, DECZ1, RAZ2, DECZ2, N)
CALL sla_TPS2C (XI, ETA, RA, DEC, RAZ1, DECZ1, RAZ2, DECZ2, N)
Get plate centre from star $[\,\alpha, \delta\,]$ and tangent plane coordinates

CALL sla_DTPV2C (XI, ETA, V, V01, V02, N)
CALL sla_TPV2C (XI, ETA, V, V01, V02, N)
Get plate centre from star $[\,x, y, z\,]$ and tangent plane coordinates

CALL sla_PCD (DISCO, X, Y)
Apply pincushion/barrel distortion

CALL sla_UNPCD (DISCO, X, Y)
Remove pincushion/barrel distortion

CALL sla_FITXY (ITYPE, NP, XYE, XYM, COEFFS, J)
Fit a linear model to relate two sets of $[\,x, y\,]$ coordinates

CALL sla_PXY (NP, XYE, XYM, COEFFS, XYP, XRMS, YRMS, RRMS)
Compute predicted coordinates and residuals

CALL sla_INVF (FWDS, BKWDS, J)
Invert a linear model

CALL sla_XY2XY (X1, Y1, COEFFS, X2, Y2)
Transform one $[\,x, y\,]$

CALL sla_DCMPF (COEFFS, XZ, YZ, XS, YS, PERP, ORIENT)
Decompose a linear fit into scales *etc.*

Numerical Methods

CALL sla_COMBN (NSEL, NCAND, LIST, J)
Next combination (subset from a specified number of items)

CALL sla_PERMUT (N, ISTATE, IORDER, J)
Next permutation of a specified number of items

CALL sla_SMAT (N, A, Y, D, JF, IW)
CALL sla_DMAT (N, A, Y, D, JF, IW)
Matrix inversion and solution of simultaneous equations

CALL sla_SVD (M, N, MP, NP, A, W, V, WORK, JSTAT)
Singular value decomposition of a matrix

CALL sla_SVDSOL (M, N, MP, NP, B, U, W, V, WORK, X)
Solution from given vector plus SVD

CALL sla_SVDCOV (N, NP, NC, W, V, WORK, CVM)
Covariance matrix from SVD

R = sla_RANDOM (SEED)
Generate pseudo-random real number in the range $0 \leq x < 1$

R = sla_GRESID (S)
Generate pseudo-random normal deviate (≡ 'Gaussian residual')

Real-time

CALL sla_WAIT (DELAY)
Interval wait