

# SLALIB — Positional Astronomy Library

2.4-12

## Programmer's Manual

### Abstract

SLALIB is a library used by writers of positional-astronomy applications. Most of the 187 routines are concerned with astronomical position and time, but a number have wider trigonometrical, numerical or general applications.

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Purpose . . . . .	1
1.2	Example Application . . . . .	1
1.3	Scope . . . . .	2
1.4	Objectives . . . . .	3
1.5	Fortran Version . . . . .	4
1.6	C Version . . . . .	4
1.7	Future Versions . . . . .	4
1.8	New Functions . . . . .	5
1.9	Acknowledgements . . . . .	5
<b>2</b>	<b>LINKING</b>	<b>5</b>
<b>3</b>	<b>SUBPROGRAM SPECIFICATIONS</b>	<b>6</b>
<b>4</b>	<b>EXPLANATION AND EXAMPLES</b>	<b>169</b>
4.1	Spherical Trigonometry . . . . .	169
4.2	Vectors and Matrices . . . . .	172
4.3	Celestial Coordinate Systems . . . . .	174
4.4	Precession and Nutation . . . . .	177
4.5	Mean Places . . . . .	179
4.6	Epoch . . . . .	179
4.7	Proper Motion . . . . .	180
4.8	Parallax and Radial Velocity . . . . .	180
4.9	Aberration . . . . .	181
4.10	Different Sorts of Mean Place . . . . .	182
4.11	Mean Place Transformations . . . . .	183
4.12	Mean Place to Apparent Place . . . . .	185
4.13	Apparent Place to Observed Place . . . . .	186
4.14	The Hipparcos Catalogue and the ICRS . . . . .	189
4.15	Timescales . . . . .	189
4.16	Calendars . . . . .	193
4.17	Geocentric Coordinates . . . . .	193
4.18	Ephemerides . . . . .	194
4.19	Radial Velocity and Light-Time Corrections . . . . .	203
4.20	Focal-Plane Astrometry . . . . .	204
4.21	Numerical Methods . . . . .	205
<b>5</b>	<b>SUMMARY OF CALLS</b>	<b>208</b>

# 1 INTRODUCTION

## 1.1 Purpose

SLALIB<sup>1</sup> 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 proprietary 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

```

---

<sup>1</sup>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  $0^\circ$  and  $-1^\circ$  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 187 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 187 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 (Ultron), 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` (the one exception being `slaIntin`, which returns a `long` and is supplemented by an additional routine, not present in the Fortran SLALIB, called `slaInt2in`, which returns an `int`). 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 object-oriented techniques, in particular through the C++ and Java languages. For example “celestial position” could be a class and many of the transformations could happen automatically. This requires further study and would result in a complete redesign. Various attempts have been made to do this, but none as yet has the author’s seal of approval. Furthermore, the impact of Fortran 90 has yet to be assessed. Should 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 (Linux, 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.)

### 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** :

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

**RETURNED** :

<i>RC,DC</i>	<b>D</b>	$[\alpha, \delta]$ 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, RESULT, JF)

**GIVEN** :

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

**RETURNED :**

<i>NSTRT</i>	<b>I</b>	advanced past the decoded angle
<i>RESLT</i>	<b>R</b>	angle in radians
<i>JF</i>	<b>I</b>	status:
		0 = OK
		+1 = default, <i>RESLT</i> unchanged (note 2)
		-1 = bad degrees (note 3)
		-2 = bad arcminutes (note 3)
		-3 = bad arcseconds (note 3)

**EXAMPLE :**

<i>argument</i>	<i>before</i>	<i>after</i>
STRING	'-57_17_44.806_12_34_56.7'	unchanged
NSTRT	1	16 ( <i>i.e.</i> pointing to 12...)
RESLT	-	-1.00000
JF	-	0

A further call to `sla_AFIN`, without adjustment of `NSTRT`, will decode the second angle, 12° 34' 56".7.

**NOTES :**

- The first three "fields" in `STRING` are degrees, arcminutes, arcseconds, separated by spaces or commas. The degrees field may be signed, but not the others. The decoding is carried out by the `sla_DFLTIN` routine and is free-format.
- Successive fields may be absent, defaulting to zero. For zero status, the only combinations allowed are degrees alone, degrees and arcminutes, and all three fields present. If all three fields are omitted, a status of +1 is returned and `RESLT` is unchanged. In all other cases `RESLT` is changed.
- Range checking:
  - The degrees field is not range checked. However, it is expected to be integral unless the other two fields are absent.
  - The arcminutes field is expected to be 0-59, and integral if the arcseconds field is present. If the arcseconds field is absent, the arcminutes is expected to be 0-59.9999...
  - The arcseconds field is expected to be 0-59.9999...
  - Decoding continues even when a check has failed. Under these circumstances the field takes the supplied value, defaulting to zero, and the result `RESLT` is computed and returned.
- 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 RESULT should be multiplied as follows:

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

---

**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** :

<i>HA</i>	<b>D</b>	hour angle
<i>DEC</i>	<b>D</b>	declination
<i>PHI</i>	<b>D</b>	observatory latitude

**RETURNED** :

<i>AZ</i>	<b>D</b>	azimuth
<i>AZD</i>	<b>D</b>	azimuth velocity
<i>AZDD</i>	<b>D</b>	azimuth acceleration
<i>EL</i>	<b>D</b>	elevation
<i>ELD</i>	<b>D</b>	elevation velocity
<i>ELDD</i>	<b>D</b>	elevation acceleration
<i>PA</i>	<b>D</b>	parallactic angle
<i>PAD</i>	<b>D</b>	parallactic angle velocity
<i>PADD</i>	<b>D</b>	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:

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

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  $\pm\pi$ . 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.

---

<b>SLA_AMP</b>	Apparent to Mean	<b>SLA_AMP</b>
----------------	------------------	----------------

**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** :

<i>RA, DA</i>	<b>D</b>	apparent $[\alpha, \delta]$ (radians)
<i>DATE</i>	<b>D</b>	TDB for apparent place (JD-2400000.5)
<i>EQ</i>	<b>D</b>	equinox: Julian epoch of mean place

**RETURNED** :

<i>RM, DM</i>	<b>D</b>	mean $[\alpha, \delta]$ (radians)
---------------	----------	-----------------------------------

**NOTES** :

1. The distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
2. Iterative techniques are used for the aberration and light deflection corrections so that the 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.
3. Where multiple apparent places are to be converted to mean places, for a fixed date and equinox, it is more efficient to use the sla\_MAPPA routine to compute the required parameters once, followed by one call to sla\_AMPQK per star.
4. The accuracy is limited by imperfections in the IAU 1976 and 1980 models for precession and nutation. Corrections are tabulated in IERS Bulletin B and at the present epoch are of order 50 milliarcsecond. An improved precession-nutation model can be introduced by using sla\_MAPPA and sla\_AMPQK (see the previous note) and replacing the precession-nutation matrix into the parameter array directly.
5. The accuracy is further 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.

**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 :**

<i>RA,DA</i>	<b>D</b>	apparent $[\alpha, \delta]$ (radians)
<i>AMPRMS</i>	<b>D(21)</b>	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) $\times 2 / (\text{Sun-Earth distance})$
(9-11)		<b>v</b> : barycentric Earth velocity in units of $c$
(12)		$\sqrt{1 -  \mathbf{v} ^2}$
(13-21)		precession-nutation $3 \times 3$ matrix

**RETURNED :**

<i>RM,DM</i>	<b>D</b>	mean $[\alpha, \delta]$ (radians)
--------------	----------	-----------------------------------

**NOTES :**

1. 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** :

<i>RAP,DAP</i>	D	geocentric apparent [ $\alpha, \delta$ ] (radians)
<i>DATE</i>	D	UTC date/time (Modified Julian Date, JD-2400000.5)
<i>DUT</i>	D	$\Delta$ UT: UT1-UTC (UTC seconds)
<i>ELONGM</i>	D	observer's mean longitude (radians, east +ve)
<i>PHIM</i>	D	observer's mean geodetic latitude (radians)
<i>HM</i>	D	observer's height above sea level (metres)
<i>XP,YP</i>	D	polar motion [ $x, y$ ] coordinates (radians)
<i>TDK</i>	D	local ambient temperature (degrees K; std=273.15D0)
<i>PMB</i>	D	local atmospheric pressure (mB; std=1013.25D0)
<i>RH</i>	D	local relative humidity (in the range 0D0-1D0)
<i>WL</i>	D	effective wavelength ( $\mu$ m, e.g. 0.55D0)
<i>TLR</i>	D	tropospheric lapse rate (degrees K per metre, e.g. 0.0065D0)

**RETURNED** :

<i>AOB</i>	D	observed azimuth (radians: N=0, E=90°)
<i>ZOB</i>	D	observed zenith distance (radians)
<i>HOB</i>	D	observed Hour Angle (radians)
<i>DOB</i>	D	observed $\delta$ (radians)
<i>ROB</i>	D	observed $\alpha$ (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  $90^\circ$ , the accuracy in elevation should be better than 1 arcminute; useful results are available for a further  $3^\circ$ , 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  $\alpha$  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  $\Delta UT$  and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap  $\Delta UT$  should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the  $\Delta UT$  changes by one second to its post-leap new value.
9. The  $\Delta UT$  (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep  $\Delta UT$  within  $\pm 0^s.9$ .
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  $0^{\circ}.3$ . 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** :

<i>DATE</i>	D	UTC date/time (Modified Julian Date, JD-2400000.5)
<i>DUT</i>	D	$\Delta UT$ : UT1-UTC (UTC seconds)
<i>ELONGM</i>	D	observer's mean longitude (radians, east +ve)
<i>PHIM</i>	D	observer's mean geodetic latitude (radians)
<i>HM</i>	D	observer's height above sea level (metres)
<i>XP, YP</i>	D	polar motion $[x, y]$ coordinates (radians)
<i>TDK</i>	D	local ambient temperature (degrees K; std=273.15D0)
<i>PMB</i>	D	local atmospheric pressure (mB; std=1013.25D0)
<i>RH</i>	D	local relative humidity (in the range 0D0-1D0)
<i>WL</i>	D	effective wavelength ( $\mu m$ , <i>e.g.</i> 0.55D0)
<i>TLR</i>	D	tropospheric lapse rate (degrees K per metre, <i>e.g.</i> 0.0065D0)

**RETURNED** :

<i>AOPRMS</i>	<b>D(14)</b>	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 $\Delta$ UT” (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  $\Delta$ UT and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap  $\Delta$ UT should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the  $\Delta$ UT changes by one second to its post-leap new value.
3. The  $\Delta$ UT (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep  $\Delta$ UT within  $\pm 0^s.9$ . The “sidereal  $\Delta$ UT” 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  $0^s.3$ . 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** :

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

**RETURNED** :

<i>AOPRMS</i>	<b>D(14)</b>	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 :**

<i>RAP,DAP</i>	<b>D</b>	geocentric apparent $[\alpha, \delta]$ (radians)
<i>AOPRMS</i>	<b>D(14)</b>	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 ( $\mu\text{m}$ )
(10)		lapse rate (degrees K per metre)
(11,12)		refraction constants A and B (radians)
(13)		longitude + eqn of equinoxes + “sidereal $\Delta\text{UT}$ ” (radians)
(14)		local apparent sidereal time (radians)

**RETURNED :**

<i>AOB</i>	<b>D</b>	observed azimuth (radians: N=0, E=90°)
<i>ZOB</i>	<b>D</b>	observed zenith distance (radians)
<i>HOB</i>	<b>D</b>	observed Hour Angle (radians)
<i>DOB</i>	<b>D</b>	observed Declination (radians)
<i>ROB</i>	<b>D</b>	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  $90^\circ$ , the accuracy in elevation should be better than 1 arcminute; useful results are available for a further  $3^\circ$ , 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  $\alpha$  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  $\Delta UT$ ” 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  $76^\circ$ , 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** :

<i>TDK</i>	D	ambient temperature at the observer (degrees K)
<i>PMB</i>	D	pressure at the observer (mB)
<i>RH</i>	D	relative humidity at the observer (range 0–1)
<i>WL1</i>	D	base wavelength ( $\mu\text{m}$ )
<i>A1</i>	D	refraction coefficient A for wavelength WL1 (radians)
<i>B1</i>	D	refraction coefficient B for wavelength WL1 (radians)
<i>WL2</i>	D	wavelength for which adjusted A,B required ( $\mu\text{m}$ )

**RETURNED :**

<i>A2</i>	D	refraction coefficient A for wavelength WL2 (radians)
<i>B2</i>	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  $0.7 \mu\text{m}$  and the UV atmospheric cutoff, and the effect increases strongly towards the UV end. For this reason a blue reference wavelength is recommended, for example  $0.4 \mu\text{m}$ .
3. The accuracy, for this set of conditions:

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

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

4. If either wavelength exceeds  $100 \mu\text{m}$ , the radio case is assumed and the returned refraction coefficients are the same as the given ones. Note that radio refraction coefficients cannot be turned into optical values using this routine, nor vice versa.

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<sup>h</sup>  
*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  $n$ BC use  $IY = -(n - 1)$ .
3. When an invalid year or month is supplied (status  $J = 1$  or  $2$ ) the MJD is **not** computed. When an invalid day is supplied (status  $J = 3$ ) the MJD is computed.

---

**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.

---

<b>SLA_CC2S</b>	Cartesian to Spherical	<b>SLA_CC2S</b>
-----------------	------------------------	-----------------

---

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

**CALL :** CALL sla\_CC2S (V, A, B)

**GIVEN :**

$V$	<b>R(3)</b>	$[x, y, z]$ vector
-----	-------------	--------------------

**RETURNED :**

$A, B$	<b>R</b>	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.

---

<b>SLA_CC62S</b>	Cartesian 6-Vector to Spherical	<b>SLA_CC62S</b>
------------------	---------------------------------	------------------

---

**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 :**

<i>V</i>	<b>R(6)</b>	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$
----------	-------------	--

**RETURNED :**

<i>A</i>	<b>R</b>	longitude (radians) – for example $\alpha$
<i>B</i>	<b>R</b>	latitude (radians) – for example $\delta$
<i>R</i>	<b>R</b>	radial coordinate
<i>AD</i>	<b>R</b>	longitude derivative (radians per unit time)
<i>BD</i>	<b>R</b>	latitude derivative (radians per unit time)
<i>RD</i>	<b>R</b>	radial derivative

---

<b>SLA_CD2TF</b>	Days to Hour,Min,Sec	<b>SLA_CD2TF</b>
------------------	----------------------	------------------

---

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

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

**GIVEN :**

<i>NDP</i>	<b>I</b>	number of decimal places of seconds
<i>DAYS</i>	<b>R</b>	interval in days

**RETURNED :**

<i>SIGN</i>	<b>C</b>	'+' or '-'
<i>IHMSF</i>	<b>I(4)</b>	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.

---

<b>SLA_CLDJ</b>	Calendar to MJD	<b>SLA_CLDJ</b>
-----------------	-----------------	-----------------

---

**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<sup>h</sup>  
*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 *n*BC 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.

---

<b>SLA_COMBN</b>	Next Combination	<b>SLA_COMBN</b>
------------------	------------------	------------------

---

**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 :**

<i>NSEL</i>	<b>I</b>	number of items (subset size)
<i>NCAND</i>	<b>I</b>	number of candidates (set size)

**GIVEN and RETURNED :**

<i>LIST</i>	<b>I(NSEL)</b>	latest combination, LIST(1)=0 to initialize
-------------	----------------	---

**RETURNED :**

<i>J</i>	<b>I</b>	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:

$$\begin{aligned} \text{LIST}(1) &= \text{NCAND}, \text{LIST}(2) = \text{NCAND} - 1, \dots, \\ &\text{LIST}(\text{NSEL}) = \text{NCAND} - \text{NSEL} + 1 \end{aligned}$$

4. If the “finished” (J=1) status is ignored, the routine continues to deliver combinations, the pattern repeating every  $\text{NCAND!}/(\text{NSEL}!(\text{NCAND}-\text{NSEL})!)$  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** :

<i>NDP</i>	<b>I</b>	number of decimal places of arcseconds
<i>ANGLE</i>	<b>R</b>	angle in radians

**RETURNED** :

<i>SIGN</i>	<b>C</b>	‘+’ or ‘-’
<i>IDMSF</i>	<b>I(4)</b>	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  $2\pi$ , 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  $2\pi$ . In cases where it does not, it is up to the caller to test for and handle the case where ANGLE is very nearly  $2\pi$  and rounds up to  $360^\circ$ , 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 :**

<i>NDP</i>	<b>I</b>	number of decimal places of seconds
<i>ANGLE</i>	<b>R</b>	angle in radians

**RETURNED :**

<i>SIGN</i>	<b>C</b>	'+' or '-'
<i>IHMSF</i>	<b>I(4)</b>	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  $2\pi$ , 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  $2\pi$ . In cases where it does not, it is up to the caller to test for and handle the case where ANGLE is very nearly  $2\pi$  and rounds up to 24 hours, by testing for IHMSF(1)=24 and setting IHMSF(1-4) to zero.

---

<b>SLA_CS2C</b>	Spherical to Cartesian	<b>SLA_CS2C</b>
-----------------	------------------------	-----------------

**ACTION :** Spherical coordinates to Cartesian coordinates (single precision).**CALL :** CALL sla\_CS2C (A, B, V)**GIVEN :**

<i>A,B</i>	<b>R</b>	spherical coordinates in radians: $[\alpha, \delta]$ etc.
------------	----------	---

**RETURNED :**

<i>V</i>	<b>R(3)</b>	$[x, y, z]$ 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.

---

<b>SLA_CS2C6</b>	Spherical Pos/Vel to Cartesian	<b>SLA_CS2C6</b>
------------------	--------------------------------	------------------

**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 :**

<i>A</i>	<b>R</b>	longitude (radians) – for example $\alpha$
<i>B</i>	<b>R</b>	latitude (radians) – for example $\delta$
<i>R</i>	<b>R</b>	radial coordinate
<i>AD</i>	<b>R</b>	longitude derivative (radians per unit time)
<i>BD</i>	<b>R</b>	latitude derivative (radians per unit time)
<i>RD</i>	<b>R</b>	radial derivative

**RETURNED :**

<i>V</i>	<b>R(6)</b>	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$
----------	-------------	--

---

<b>SLA_CTF2D</b>	Hour,Min,Sec to Days	<b>SLA_CTF2D</b>
------------------	----------------------	------------------

**ACTION :** Convert hours, minutes, seconds to days (single precision).**CALL :** CALL sla\_CTF2D (Ihour, IMIN, SEC, DAYS, J)**GIVEN :**

<i>Ihour</i>	<b>I</b>	hours
<i>IMIN</i>	<b>I</b>	minutes
<i>SEC</i>	<b>R</b>	seconds

**RETURNED :**

<i>DAYS</i>	<b>R</b>	interval in days
<i>J</i>	<b>I</b>	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.

---

<b>SLA_CTF2R</b>	Hour,Min,Sec to Radians	<b>SLA_CTF2R</b>
------------------	-------------------------	------------------

**ACTION :** Convert hours, minutes, seconds to radians (single precision).**CALL :** CALL sla\_CTF2R (Ihour, IMIN, SEC, RAD, J)

**GIVEN :**

<i>I</i> <b>HO</b> <i>UR</i>	<b>I</b>	hours
<i>I</i> <b>M</b> <i>IN</i>	<b>I</b>	minutes
<i>S</i> <b>E</b> <i>C</i>	<b>R</b>	seconds

**RETURNED :**

<i>R</i> <b>A</b> <i>D</i>	<b>R</b>	angle in radians
<i>J</i>	<b>I</b>	status:
		0 = OK
		1 = <i>I</i> <b>H</b> <i>O</i> <b>U</b> <i>R</i> outside range 0-23
		2 = <i>I</i> <b>M</b> <i>I</i> <b>N</b> outside range 0-59
		3 = <i>S</i> <b>E</b> <i>C</i> 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 :**

<i>I</i> <b>D</b> <i>E</i> <b>G</b>	<b>I</b>	degrees
<i>I</i> <b>A</b> <i>M<i>I</i><b>N</b></i>	<b>I</b>	arcminutes
<i>A<i>S</i><b>E</b><i>C</i></i>	<b>D</b>	arcseconds

**RETURNED :**

<i>R</i> <b>A</b> <i>D</i>	<b>D</b>	angle in radians
<i>J</i>	<b>I</b>	status:
		1 = <i>I</i> <b>D</b> <i>E</i> <b>G</b> outside range 0-359
		2 = <i>I</i> <b>A</b> <i>M<i>I<b>N</b> outside range 0-59</i></i>
		3 = <i>A<i>S</i><b>E</b><i>C</i> outside range 0-59.999...</i>

**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 :**

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

**RETURNED :**

<i>NSTRT</i>	<b>I</b>	advanced past the decoded angle
<i>DRESLT</i>	<b>D</b>	angle in radians
<i>JF</i>	<b>I</b>	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 :**

<i>argument</i>	<i>before</i>	<i>after</i>
STRING	'-57_17_44.806_12_34_56.7'	unchanged
NSTRT	1	16 ( <i>i.e.</i> pointing to 12...)
RESLT	-	-1.00000D0
JF	-	0

A further call to sla\_DAFIN, without adjustment of NSTRT, will decode the second angle, 12° 34' 56".7.

**NOTES :**

1. The first three "fields" in STRING are degrees, arcminutes, arcseconds, separated by spaces or commas. The degrees field may be signed, but not the others. The decoding is carried out by the sla\_DFLTIN routine and is free-format.



**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<sup>h</sup> 59<sup>m</sup> 59<sup>s</sup>.
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 :**

<i>A1,B1</i>	<b>D</b>	spherical coordinates of one point
<i>A2,B2</i>	<b>D</b>	spherical coordinates of the other point

**RETURNED :**

<i>sla_DBEAR</i>	<b>D</b>	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 vectors.

---

<b>SLA_DBJIN</b>	Decode String to B/J Epoch (DP)	<b>SLA_DBJIN</b>
------------------	---------------------------------	------------------

**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 :**

<i>STRING</i>	<b>C</b>	string containing field to be decoded
<i>NSTRT</i>	<b>I</b>	pointer to first character of field in string

**RETURNED :**

<i>NSTRT</i>	<b>I</b>	incremented past the decoded field
<i>DRESLT</i>	<b>D</b>	result
<i>J1</i>	<b>I</b>	DFLTIN status: -1 = -OK 0 = +OK 1 = null field 2 = error
<i>J2</i>	<b>I</b>	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.

---

<b>SLA_DC62S</b>	Cartesian 6-Vector to Spherical	<b>SLA_DC62S</b>
------------------	---------------------------------	------------------

---

**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 :**

$V$	<b>D(6)</b>	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$
-----	-------------	--

**RETURNED :**

$A$	<b>D</b>	longitude (radians)
$B$	<b>D</b>	latitude (radians)
$R$	<b>D</b>	radial coordinate
$AD$	<b>D</b>	longitude derivative (radians per unit time)
$BD$	<b>D</b>	latitude derivative (radians per unit time)
$RD$	<b>D</b>	radial derivative

---

<b>SLA_DCC2S</b>	Cartesian to Spherical	<b>SLA_DCC2S</b>
------------------	------------------------	------------------

---

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

**CALL :** CALL sla\_DCC2S (V, A, B)

**GIVEN :**

$V$	<b>D(3)</b>	$[x, y, z]$ vector
-----	-------------	--------------------

**RETURNED :**

$A, B$	<b>D</b>	spherical coordinates in radians
--------	----------	----------------------------------



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

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

(d) Orientation:

$$x_2 = +x''' \cos \text{ORIENT} + y''' \sin \text{ORIENT}$$

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

2. See also sla\_FITXY, sla\_PXY, sla\_INVF, sla\_XY2XY.

---

<b>SLA_DCS2C</b>	Spherical to Cartesian	<b>SLA_DCS2C</b>
------------------	------------------------	------------------

---

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

**CALL** : CALL sla\_DCS2C (A, B, V)

**GIVEN** :

<i>A, B</i>	<b>D</b>	spherical coordinates in radians: $[\alpha, \delta]$ etc.
-------------	----------	---

**RETURNED** :

<i>V</i>	<b>D(3)</b>	$[x, y, z]$ 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.

---

<b>SLA_DD2TF</b>	Days to Hour,Min,Sec	<b>SLA_DD2TF</b>
------------------	----------------------	------------------

---

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

**CALL** : CALL sla\_DD2TF (NDP, DAYS, SIGN, IHMSF)

**GIVEN** :

<i>NDP</i>	<b>I</b>	number of decimal places of seconds
<i>DAYS</i>	<b>D</b>	interval in days

**RETURNED** :

<i>SIGN</i>	<b>C</b>	'+' or '-'
<i>IHMSF</i>	<b>I(4)</b>	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 :**

<i>HA</i>	<b>D</b>	hour angle (radians)
<i>DEC</i>	<b>D</b>	declination (radians)
<i>PHI</i>	<b>D</b>	latitude (radians)

**RETURNED :**

<i>AZ</i>	<b>D</b>	azimuth (radians)
<i>EL</i>	<b>D</b>	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** :

<i>ORDER</i>	<b>C</b>	specifies about which axes the rotations occur
<i>PHI</i>	<b>D</b>	1st rotation (radians)
<i>THETA</i>	<b>D</b>	2nd rotation (radians)
<i>PSI</i>	<b>D</b>	3rd rotation (radians)

**RETURNED** :

<i>RMAT</i>	<b>D(3,3)</b>	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 'zxx' 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** :

<i>STRING</i>	<b>C</b>	string containing number to be decoded
<i>NSTRT</i>	<b>I</b>	pointer to where decoding is to commence
<i>DRESLT</i>	<b>D</b>	current value of result

**RETURNED :**

<i>NSTRT</i>	<b>I</b>	advanced to next number
<i>DRESLT</i>	<b>D</b>	result
<i>JFLAG</i>	<b>I</b>	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  $h, \delta$ **SLA\_DH2E**

**ACTION** : Horizon to equatorial coordinates (double precision).

**CALL** : CALL sla\_DH2E (AZ, EL, PHI, HA, DEC)

**GIVEN** :

<i>AZ</i>	<b>D</b>	azimuth (radians)
<i>EL</i>	<b>D</b>	elevation (radians)
<i>PHI</i>	<b>D</b>	latitude (radians)

**RETURNED** :

<i>HA</i>	<b>D</b>	hour angle (radians)
<i>DEC</i>	<b>D</b>	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** :

<i>DM</i>	<b>D(3,3)</b>	rotation matrix
<i>VA</i>	<b>D(3)</b>	vector to be rotated

**RETURNED** :

<i>VB</i>	<b>D(3)</b>	result vector
-----------	-------------	---------------

**NOTES** :

1. This routine performs the operation:

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

where  $\mathbf{a}$  and  $\mathbf{b}$  are the 3-vectors VA and VB respectively, and  $\mathbf{M}$  is the  $3 \times 3$  matrix DM.

2. The main function of this routine is apply an inverse rotation; under these circumstances,  $\mathbf{M}$  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, IY MDF, J)

**GIVEN** :

<i>NDP</i>	<b>I</b>	number of decimal places of days in fraction
<i>DJM</i>	<b>D</b>	modified Julian Date (JD-2400000.5)

**RETURNED :**

<i>IY MDF</i>	<b>I(4)</b>	year, month, day, fraction in Gregorian calendar
<i>J</i>	<b>I</b>	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 :**

<i>DJM</i>	<b>D</b>	modified Julian Date (JD-2400000.5)
------------	----------	-------------------------------------

**RETURNED :**

<i>IY</i>	<b>I</b>	year
<i>IM</i>	<b>I</b>	month
<i>ID</i>	<b>I</b>	day
<i>FD</i>	<b>D</b>	fraction of day
<i>J</i>	<b>I</b>	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 :**

<i>RMAT</i>	<b>D(3,3)</b>	rotation matrix
-------------	---------------	-----------------

**RETURNED :**

<i>AXVEC</i>	<b>D(3)</b>	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.

---

<b>SLA_DMAT</b>	<b>Solve Simultaneous Equations</b>	<b>SLA_DMAT</b>
-----------------	-------------------------------------	-----------------

---

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

**CALL :** CALL *sla\_DMAT* (N, A, Y, D, JF, IW)

**GIVEN :**

<i>N</i>	<b>I</b>	number of unknowns
<i>A</i>	<b>D(N,N)</b>	matrix
<i>Y</i>	<b>D(N)</b>	vector

**RETURNED :**

<i>A</i>	<b>D(N,N)</b>	matrix inverse
<i>Y</i>	<b>D(N)</b>	solution
<i>D</i>	<b>D</b>	determinant
<i>JF</i>	<b>I</b>	singularity flag: 0=OK
<i>IW</i>	<b>I(N)</b>	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\_DMATRIX.

---

<b>SLA_DMOON</b>	Approx Moon Pos/Vel	<b>SLA_DMOON</b>
------------------	---------------------	------------------

---

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

**CALL :** CALL sla\_DMOON (DATE, PV)

**GIVEN :**

<i>DATE</i>	<b>D</b>	TDB (loosely ET) as a Modified Julian Date (JD-2400000.5)
-------------	----------	--

**RETURNED :**

<i>PV</i>	<b>D(6)</b>	Moon [ $x, y, z, \dot{x}, \dot{y}, \dot{z}$ ], mean equator and equinox of date (AU, AU s <sup>-1</sup> )
-----------	-------------	--

**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 :**

<i>A</i>	<b>D(3,3)</b>	matrix <b>A</b>
<i>B</i>	<b>D(3,3)</b>	matrix <b>B</b>

**RETURNED :**

<i>C</i>	<b>D(3,3)</b>	matrix result: <b>A</b> × <b>B</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\_DM XV**                      Apply 3D Rotation                      **SLA\_DM XV**

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

**CALL :** CALL sla\_DM XV (DM, VA, VB)

**GIVEN :**

<i>DM</i>	<b>D(3,3)</b>	rotation matrix
<i>VA</i>	<b>D(3)</b>	vector to be rotated

**RETURNED :**

<i>VB</i>	<b>D(3)</b>	result vector
-----------	-------------	---------------

**NOTES :**

1. This routine performs the operation:

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

where  $\mathbf{a}$  and  $\mathbf{b}$  are the 3-vectors *VA* and *VB* respectively, and  $\mathbf{M}$  is the  $3 \times 3$  matrix *DM*.

2. The main function of this routine is apply a rotation; under these circumstances,  $\mathbf{M}$  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 :**

<i>V1</i>	<b>D(3)</b>	vector to one point
<i>V2</i>	<b>D(3)</b>	vector to the other point

**RETURNED :**

<i>sla_DPAV</i>	<b>D</b>	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. There is no requirement for either vector to be of unit length.
4. 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** :

<i>NDP</i>	<b>I</b>	number of decimal places of arcseconds
<i>ANGLE</i>	<b>D</b>	angle in radians

**RETURNED** :

<i>SIGN</i>	<b>C</b>	'+' or '-'
<i>IDMSF</i>	<b>I(4)</b>	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  $2\pi$ , 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  $2\pi$ . In cases where it does not, it is up to the caller to test for and handle the case where ANGLE is very nearly  $2\pi$  and rounds up to  $360^\circ$ , 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** :

<i>NDP</i>	<b>I</b>	number of decimal places of seconds
<i>ANGLE</i>	<b>D</b>	angle in radians

**RETURNED :**

<i>SIGN</i>	<b>C</b>	'+' or '-'
<i>IHMSF</i>	<b>I(4)</b>	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  $2\pi$ , 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  $2\pi$ . In cases where it does not, it is up to the caller to test for and handle the case where ANGLE is very nearly  $2\pi$  and rounds up to 24 hours, by testing for IHMSF(1)=24 and setting IHMSF(1-4) to zero.

---

<b>SLA_DRANGE</b>	Put Angle into Range $\pm\pi$	<b>SLA_DRANGE</b>
-------------------	-------------------------------	-------------------

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

**CALL :** D = sla\_DRANGE (ANGLE)

**GIVEN :**

<i>ANGLE</i>	<b>D</b>	angle in radians
--------------	----------	------------------

**RETURNED :**

<i>sla_DRANGE</i>	<b>D</b>	ANGLE expressed in the range $\pm\pi$ .
-------------------	----------	---

---

<b>SLA_DRANRM</b>	Put Angle into Range $0-2\pi$	<b>SLA_DRANRM</b>
-------------------	-------------------------------	-------------------

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

**CALL :** D = sla\_DRANRM (ANGLE)

**GIVEN :**

<i>ANGLE</i>	<b>D</b>	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 :**

<i>A</i>	<b>D</b>	longitude (radians) – for example $\alpha$
<i>B</i>	<b>D</b>	latitude (radians) – for example $\delta$
<i>R</i>	<b>D</b>	radial coordinate
<i>AD</i>	<b>D</b>	longitude derivative (radians per unit time)
<i>BD</i>	<b>D</b>	latitude derivative (radians per unit time)
<i>RD</i>	<b>D</b>	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 :**

<i>RA,DEC</i>	<b>D</b>	spherical coordinates of star (radians)
<i>RAZ,DECZ</i>	<b>D</b>	spherical coordinates of tangent point (radians)

**RETURNED :**

<i>XI,ETA</i>	<b>D</b>	tangent plane coordinates (radians)
<i>J</i>	<b>I</b>	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.

---

<b>SLA_DSEP</b>	Angle Between 2 Points on Sphere	<b>SLA_DSEP</b>
-----------------	----------------------------------	-----------------

---

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

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

**GIVEN :**

<i>A1, B1</i>	<b>D</b>	spherical coordinates of one point (radians)
<i>A2, B2</i>	<b>D</b>	spherical coordinates of the other point (radians)

**RETURNED :**

<i>sla_DSEP</i>	<b>D</b>	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.

---

<b>SLA_DSEPV</b>	Angle Between 2 Vectors	<b>SLA_DSEPV</b>
------------------	-------------------------	------------------

---

**ACTION :** Angle between two vectors (double precision).

**CALL :** `D = sla_DSEPV (V1, V2)`

**GIVEN :**

<i>V1</i>	<b>D(3)</b>	first vector
<i>V2</i>	<b>D(3)</b>	second vector

**RETURNED :**

<i>sla_DSEPV</i>	<b>D</b>	angle between $V1$ and $V2$ in radians
------------------	----------	--

**NOTES :**

1. There is no requirement for either vector to be of unit length.
2. If either vector is null, zero is returned.
3. The result is always positive.

---

<b>SLA_DT</b>	Approximate ET minus UT	<b>SLA_DT</b>
---------------	-------------------------	---------------

**ACTION :** Estimate  $\Delta T$ , the offset between dynamical time and Universal Time, for a given historical epoch.

**CALL :** D = sla\_DT (EPOCH)

**GIVEN :**

<i>EPOCH</i>	<b>D</b>	(Julian) epoch ( <i>e.g.</i> 1850D0)
--------------	----------	--------------------------------------

**RETURNED :**

<i>sla_DT</i>	<b>D</b>	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<sup>s</sup> during the interval since 1650, rising to perhaps 30<sup>m</sup> 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  $-26''00$  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.

---

<b>SLA_DTF2D</b>	Hour,Min,Sec to Days	<b>SLA_DTF2D</b>
------------------	----------------------	------------------

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

**CALL :** CALL sla\_DTF2D (Ihour, Imin, Sec, Days, J)

**GIVEN :**

<i>I</i> HOOR	<b>I</b>	hours
<i>I</i> MIN	<b>I</b>	minutes
<i>S</i> EC	<b>D</b>	seconds

**RETURNED :**

<i>D</i> AYS	<b>D</b>	interval in days
<i>J</i>	<b>I</b>	status:
		0 = OK
		1 = IHOOR 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.

---

<b>SLA_DTF2R</b>	Hour,Min,Sec to Radians	<b>SLA_DTF2R</b>
------------------	-------------------------	------------------

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

**CALL :** CALL sla\_DTF2R (IHOOR, IMIN, SEC, RAD, J)

**GIVEN :**

<i>I</i> HOOR	<b>I</b>	hours
<i>I</i> MIN	<b>I</b>	minutes
<i>S</i> EC	<b>D</b>	seconds

**RETURNED :**

<i>R</i> AD	<b>D</b>	angle in radians
<i>J</i>	<b>I</b>	status:
		0 = OK
		1 = IHOOR 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 :**

<i>XI,ETA</i>	<b>D</b>	tangent plane rectangular coordinates (radians)
<i>RAZ,DECZ</i>	<b>D</b>	spherical coordinates of tangent point (radians)

**RETURNED :**

<i>RA,DEC</i>	<b>D</b>	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 :**

<i>XI,ETA</i>	<b>D</b>	tangent plane coordinates of star (radians)
<i>V0</i>	<b>D(3)</b>	direction cosines of tangent point

**RETURNED :**

<i>V</i>	<b>D(3)</b>	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\_DTPTS2C**      Plate centre from  $\xi, \eta$  and  $\alpha, \delta$       **SLA\_DTPTS2C**

**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\_DTPTS2C (XI, ETA, RA, DEC, RAZ1, DECZ1, RAZ2, DECZ2, N)

**GIVEN :**

<i>XI,ETA</i>	<b>D</b>	tangent plane rectangular coordinates (radians)
<i>RA,DEC</i>	<b>D</b>	spherical coordinates (radians)

**RETURNED :**

<i>RAZ1,DECZ1</i>	<b>D</b>	spherical coordinates of tangent point, solution 1
<i>RAZ2,DECZ2</i>	<b>D</b>	spherical coordinates of tangent point, solution 2
<i>N</i>	<b>I</b>	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  $\xi$  value, and hence it is meaningless to ask where the tangent point would have to be to bring about this combination of  $\xi$  and  $\delta$ .
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  $\xi, \eta$  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** :

$XI, ETA$	<b>D</b>	tangent plane coordinates of star (radians)
$V$	<b>D(3)</b>	direction cosines of star

**RETURNED** :

$V01$	<b>D(3)</b>	direction cosines of tangent point, solution 1
$V02$	<b>D(3)</b>	direction cosines of tangent point, solution 2
$N$	<b>I</b>	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\_DTSP2C.



**NOTES :**

1. If vector  $V_0$  is not of unit length, or if vector  $V$  is of zero length, the results will be wrong.
2. If  $V_0$  points at a pole, the returned  $\xi, \eta$  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.

---

<b>SLA_DVDV</b>	Scalar Product	<b>SLA_DVDV</b>
-----------------	----------------	-----------------

---

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

**CALL :** D = sla\_DVDV (VA, VB)

**GIVEN :**

$VA$	<b>D(3)</b>	first vector
$VB$	<b>D(3)</b>	second vector

**RETURNED :**

$sla\_DVDV$	<b>D</b>	scalar product VA.VB
-------------	----------	----------------------

---

<b>SLA_DVN</b>	Normalize Vector	<b>SLA_DVN</b>
----------------	------------------	----------------

---

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

**CALL :** CALL sla\_DVN (V, UV, VM)

**GIVEN :**

$V$	<b>D(3)</b>	vector
-----	-------------	--------

**RETURNED :**

$UV$	<b>D(3)</b>	unit vector in direction of V
$VM$	<b>D</b>	modulus of V

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

---

<b>SLA_DVXV</b>	Vector Product	<b>SLA_DVXV</b>
-----------------	----------------	-----------------

---

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

**CALL** : CALL sla\_DVXV (VA, VB, VC)

**GIVEN** :

<i>VA</i>	<b>D(3)</b>	first vector
<i>VB</i>	<b>D(3)</b>	second vector

**RETURNED** :

<i>VC</i>	<b>D(3)</b>	vector product VA×VB
-----------	-------------	----------------------

---

<b>SLA_E2H</b>	<i>h, δ</i> to Az,El	<b>SLA_E2H</b>
----------------	----------------------	----------------

---

**ACTION** : Equatorial to horizon coordinates (single precision).

**CALL** : CALL sla\_DE2H (HA, DEC, PHI, AZ, EL)

**GIVEN** :

<i>HA</i>	<b>R</b>	hour angle (radians)
<i>DEC</i>	<b>R</b>	declination (radians)
<i>PHI</i>	<b>R</b>	latitude (radians)

**RETURNED** :

<i>AZ</i>	<b>R</b>	azimuth (radians)
<i>EL</i>	<b>R</b>	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** :

<i>IY</i>	<b>I</b>	year
<i>ID</i>	<b>I</b>	day in year (1 = Jan 1st)
<i>FD</i>	<b>R</b>	fraction of day

**RETURNED** :

<i>PV</i>	<b>R(6)</b>	Earth [ $x, y, z, \dot{x}, \dot{y}, \dot{z}$ ] (AU, AU s <sup>-1</sup> )
-----------	-------------	--

**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<sup>-1</sup> = 0.0070/0.0039 km s<sup>-1</sup> 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** :

<i>DL, DB</i>	<b>D</b>	ecliptic longitude and latitude (mean of date, IAU 1980 theory, radians)
<i>DATE</i>	<b>D</b>	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 Corrn's 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 ( $\text{km s}^{-1}$ )  
*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  $0.004 \text{ km s}^{-1}$  of the correct value and is never in error by more than  $0.007 \text{ km s}^{-1}$ .
  - The error in light time correction is about  $0^{\text{s}}.03$  at worst, but is usually better than  $0^{\text{s}}.01$ .

For applications requiring higher accuracy, see the sla\_EVP routine.

---

<b>SLA_EG50</b>	B1950 $\alpha, \delta$ to Galactic	<b>SLA_EG50</b>
-----------------	------------------------------------	-----------------

---

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

**CALL :** CALL sla\_EG50 (DR, DD, DL, DB)

**GIVEN :**

<i>DR, DD</i>	<b>D</b>	B1950.0 [ $\alpha, \delta$ ] (radians)
---------------	----------	--

**RETURNED :**

<i>DL, DB</i>	<b>D</b>	galactic longitude and latitude [ $l^{\text{II}}, b^{\text{II}}$ ] (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.

---

<b>SLA_EL2UE</b>	Conventional to Universal Elements	<b>SLA_EL2UE</b>
------------------	------------------------------------	------------------

---

**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 :**

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

**RETURNED :**

<i>U</i>	<b>D(13)</b>	universal orbital elements (Note 1)
(1)		combined mass ( $M + m$ )
(2)		total energy of the orbit ( $\alpha$ )
(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)		$\mathbf{r}_0 \cdot \mathbf{v}_0$
(12)		date ( $t$ )
(13)		universal eccentric anomaly ( $\psi$ ) of date, approx
<i>JSTAT</i>	<b>I</b>	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)  $\alpha$ , which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of  $\psi$ , the “universal eccentric anomaly” at a given date and (v) that date.

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 ( $\text{JD}-2400000.5$ ).
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  $\Omega$  (radians)  
`PERIH` = longitude of perihelion  $\varpi$  (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  $\Omega$  (radians)  
`PERIH` = argument of perihelion  $\omega$  (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  $\Omega$  (radians)  
`PERIH` = argument of perihelion  $\omega$  (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.



**GIVEN :**

<i>K0</i>	<b>C</b>	form of result: 'B'=Besselian, 'J'=Julian
<i>K</i>	<b>C</b>	form of given epoch: 'B' or 'J'
<i>E</i>	<b>D</b>	epoch

**RETURNED :**

<i>sla_EPJO</i>	<b>D</b>	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.

---

<b>SLA_EPJ</b>	<b>MJD to Julian Epoch</b>	<b>SLA_EPJ</b>
----------------	----------------------------	----------------

**ACTION :** Convert Modified Julian Date to Julian Epoch.**CALL :** D = sla\_EPJ (DATE)**GIVEN :**

<i>DATE</i>	<b>D</b>	Modified Julian Date (JD-2400000.5)
-------------	----------	-------------------------------------

**RETURNED :**

<i>sla_EPJ</i>	<b>D</b>	Julian Epoch
----------------	----------	--------------

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

---

<b>SLA_EPJ2D</b>	<b>Julian Epoch to MJD</b>	<b>SLA_EPJ2D</b>
------------------	----------------------------	------------------

**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  $\alpha, \delta$  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  $\alpha, \delta$  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 :**

<i>ORDER</i>	<b>C*(*)</b>	specifies about which axes the rotations occur
<i>PHI</i>	<b>R</b>	1st rotation (radians)
<i>THETA</i>	<b>R</b>	2nd rotation (radians)
<i>PSI</i>	<b>R</b>	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** :

<i>DATE</i>	<b>D</b>	TDB (formerly ET) as a Modified Julian Date (JD−2400000.5)
<i>DEQX</i>	<b>D</b>	Julian Epoch ( <i>e.g.</i> 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** :

<i>DVB</i>	<b>D(3)</b>	barycentric $[\dot{x}, \dot{y}, \dot{z}]$ , AU s <sup>−1</sup>
<i>DPB</i>	<b>D(3)</b>	barycentric $[x, y, z]$ , AU
<i>DVH</i>	<b>D(3)</b>	heliocentric $[\dot{x}, \dot{y}, \dot{z}]$ , AU s <sup>−1</sup>
<i>DPH</i>	<b>D(3)</b>	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<sup>−1</sup>
  - 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 0’001 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** :

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

**RETURNED** :

<i>COEFFS</i>	<b>D(6)</b>	coefficients of model (note 3)
<i>J</i>	<b>I</b>	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:

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

For the *solid body rotation* option (ITYPE=4), the magnitudes of  $b$  and  $f$ , and of  $c$  and  $e$ , are equal. The signs of these coefficients depend on whether there is a sign reversal between  $[x_e, y_e]$  and  $[x_m, y_m]$ ; fits are performed with and without a sign reversal and the best one chosen.

4. Error status values  $J=-1$  and  $-2$  leave COEFFS unchanged; if  $J=-3$  COEFFS may have been changed.

5. See also 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** :

<i>R1950</i>	<b>D</b>	B1950.0 $\alpha$ (radians)
<i>D1950</i>	<b>D</b>	B1950.0 $\delta$ (radians)
<i>DR1950</i>	<b>D</b>	B1950.0 proper motion in $\alpha$ (radians per tropical year)
<i>DD1950</i>	<b>D</b>	B1950.0 proper motion in $\delta$ (radians per tropical year)
<i>P1950</i>	<b>D</b>	B1950.0 parallax (arcsec)
<i>V1950</i>	<b>D</b>	B1950.0 radial velocity (km s <sup>-1</sup> , +ve = moving away)

**RETURNED** :

<i>R2000</i>	<b>D</b>	J2000.0 $\alpha$ (radians)
<i>D2000</i>	<b>D</b>	J2000.0 $\delta$ (radians)
<i>DR2000</i>	<b>D</b>	J2000.0 proper motion in $\alpha$ (radians per Julian year)
<i>DD2000</i>	<b>D</b>	J2000.0 proper motion in $\delta$ (radians per Julian year)
<i>P2000</i>	<b>D</b>	J2000.0 parallax (arcsec)
<i>V2000</i>	<b>D</b>	J2000.0 radial velocity (km s <sup>-1</sup> , +ve = moving away)

**NOTES** :

1. The  $\alpha$  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  $10^\circ$  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 :

<i>R1950</i>	<b>D</b>	B1950.0 FK4 $\alpha$ at epoch BEPOCH (radians)
<i>D1950</i>	<b>D</b>	B1950.0 FK4 $\delta$ at epoch BEPOCH (radians)
<i>BEPOCH</i>	<b>D</b>	Besselian epoch ( <i>e.g.</i> 1979.3D0)

#### RETURNED :

<i>R2000</i>	<b>D</b>	J2000.0 FK5 $\alpha$ (radians)
<i>D2000</i>	<b>D</b>	J2000.0 FK5 $\delta$ (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  $10^\circ$  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 :**

<i>R2000</i>	D	J2000.0 $\alpha$ (radians)
<i>D2000</i>	D	J2000.0 $\delta$ (radians)
<i>DR2000</i>	D	J2000.0 proper motion in $\alpha$ (radians per Julian year)
<i>DD2000</i>	D	J2000.0 proper motion in $\delta$ (radians per Julian year)
<i>P2000</i>	D	J2000.0 parallax (arcsec)
<i>V2000</i>	D	J2000 radial velocity ( $\text{km s}^{-1}$ , +ve = moving away)

**RETURNED :**

<i>R1950</i>	D	B1950.0 $\alpha$ (radians)
<i>D1950</i>	D	B1950.0 $\delta$ (radians)
<i>DR1950</i>	D	B1950.0 proper motion in $\alpha$ (radians per tropical year)
<i>DD1950</i>	D	B1950.0 proper motion in $\delta$ (radians per tropical year)
<i>P1950</i>	D	B1950.0 parallax (arcsec)
<i>V1950</i>	D	radial velocity ( $\text{km s}^{-1}$ , +ve = moving away)

**NOTES :**

1. The  $\alpha$  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  $10^\circ$  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** :

<i>R5</i>	<b>D</b>	J2000.0 FK5 $\alpha$ (radians)
<i>D5</i>	<b>D</b>	J2000.0 FK5 $\delta$ (radians)
<i>DR5</i>	<b>D</b>	J2000.0 FK5 proper motion in $\alpha$ (radians per Julian year)
<i>DD5</i>	<b>D</b>	J2000.0 FK5 proper motion in $\delta$ (radians per Julian year)

**RETURNED** :

<i>RH</i>	<b>D</b>	Hipparcos $\alpha$ (radians)
<i>DH</i>	<b>D</b>	Hipparcos $\delta$ (radians)
<i>DRH</i>	<b>D</b>	Hipparcos proper motion in $\alpha$ (radians per Julian year)
<i>DDH</i>	<b>D</b>	Hipparcos proper motion in $\delta$ (radians per Julian year)

**NOTES** :

1. The  $\alpha$  proper motions are  $\dot{\alpha}$  rather than  $\dot{\alpha} \cos \delta$ , and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	-19.9	-0.30
<i>y</i>	-9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians.

4. See also 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** :

<i>R2000</i>	<b>D</b>	J2000.0 FK5 $\alpha$ (radians)
<i>D2000</i>	<b>D</b>	J2000.0 FK5 $\delta$ (radians)
<i>BEPOCH</i>	<b>D</b>	Besselian epoch ( <i>e.g.</i> 1950D0)

**RETURNED** :

<i>R1950</i>	<b>D</b>	B1950.0 FK4 $\alpha$ at epoch BEPOCH (radians)
<i>D1950</i>	<b>D</b>	B1950.0 FK4 $\delta$ at epoch BEPOCH (radians)
<i>DR1950</i>	<b>D</b>	B1950.0 FK4 proper motion in $\alpha$ (radians per tropical year)
<i>DD1950</i>	<b>D</b>	B1950.0 FK4 proper motion in $\delta$ (radians per tropical year)

**NOTES** :

1. The  $\alpha$  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\_FK5HZ (R5,D5,EPOCH,RH,DH)

**GIVEN** :

<i>R5</i>	<b>D</b>	J2000.0 FK5 $\alpha$ (radians)
<i>D5</i>	<b>D</b>	J2000.0 FK5 $\delta$ (radians)
<i>EPOCH</i>	<b>D</b>	Julian epoch (TDB)

**RETURNED** :

<i>RH</i>	<b>D</b>	Hipparcos $\alpha$ (radians)
<i>DH</i>	<b>D</b>	Hipparcos $\delta$ (radians)

**NOTES** :

1. The  $\alpha$  proper motions are  $\dot{\alpha}$  rather than  $\dot{\alpha} \cos \delta$ , and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	-19.9	-0.30
<i>y</i>	-9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians.

4. See also 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, RESULT, JFLAG)

**GIVEN :**

<i>STRING</i>	<b>C</b>	string containing number to be decoded
<i>NSTRT</i>	<b>I</b>	pointer to where decoding is to commence
<i>RESLT</i>	<b>R</b>	current value of result

**RETURNED :**

<i>NSTRT</i>	<b>I</b>	advanced to next number
<i>RESLT</i>	<b>R</b>	result
<i>JFLAG</i>	<b>I</b>	status: -1 = -OK, 0 = +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  $\alpha, \delta$ **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 [ $\alpha, \delta$ ]

**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.



**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.

---

<b>SLA_GEOC</b>	Geodetic to Geocentric	<b>SLA_GEOC</b>
-----------------	------------------------	-----------------

---

**ACTION :** Convert geodetic position to geocentric.

**CALL :** CALL sla\_GEOC (P, H, R, Z)

**GIVEN :**

<i>P</i>	<b>D</b>	latitude (geodetic, radians)
<i>H</i>	<b>D</b>	height above reference spheroid (geodetic, metres)

**RETURNED :**

<i>R</i>	<b>D</b>	distance from Earth axis (AU)
<i>Z</i>	<b>D</b>	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.

---

<b>SLA_GMST</b>	UT to GMST	<b>SLA_GMST</b>
-----------------	------------	-----------------

---

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

**CALL :** D = sla\_GMST (UT1)

**GIVEN :**

<i>UT1</i>	<b>D</b>	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<sup>h</sup>UT; in fact, it gives the difference between the GMST and the UT, which happens to equal the GMST (modulo 24 hours) at 0<sup>h</sup>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... 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.

---

<b>SLA_GRESID</b>	Gaussian Residual	<b>SLA_GRESID</b>
-------------------	-------------------	-------------------

---

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

**CALL** : R = sla\_GRESID (S)

**GIVEN** :

<i>S</i>	<b>R</b>	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.

---

<b>SLA_H2E</b>	Az,El to $h, \delta$	<b>SLA_H2E</b>
----------------	----------------------	----------------

---

**ACTION** : Horizon to equatorial coordinates (single precision).

**CALL** : CALL sla\_H2E (AZ, EL, PHI, HA, DEC)

**GIVEN** :

<i>AZ</i>	<b>R</b>	azimuth (radians)
<i>EL</i>	<b>R</b>	elevation (radians)
<i>PHI</i>	<b>R</b>	latitude (radians)

**RETURNED** :

<i>HA</i>	<b>R</b>	hour angle (radians)
<i>DEC</i>	<b>R</b>	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 :**

<i>RH</i>	<b>D</b>	Hipparcos $\alpha$ (radians)
<i>DH</i>	<b>D</b>	Hipparcos $\delta$ (radians)
<i>DRH</i>	<b>D</b>	Hipparcos proper motion in $\alpha$ (radians per Julian year)
<i>DDH</i>	<b>D</b>	Hipparcos proper motion in $\delta$ (radians per Julian year)

**RETURNED :**

<i>R5</i>	<b>D</b>	J2000.0 FK5 $\alpha$ (radians)
<i>D5</i>	<b>D</b>	J2000.0 FK5 $\delta$ (radians)
<i>DR5</i>	<b>D</b>	J2000.0 FK5 proper motion in $\alpha$ (radians per Julian year)
<i>DD5</i>	<b>D</b>	FK5 J2000.0 proper motion in $\delta$ (radians per Julian year)

**NOTES :**

1. The  $\alpha$  proper motions are  $\dot{\alpha}$  rather than  $\dot{\alpha} \cos \delta$ , and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	-19.9	-0.30
<i>y</i>	-9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians.

4. See also 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 :**

<i>RH</i>	<b>D</b>	Hipparcos $\alpha$ (radians)
<i>DH</i>	<b>D</b>	Hipparcos $\delta$ (radians)
<i>EPOCH</i>	<b>D</b>	Julian epoch (TDB)

**RETURNED :**

<i>R5</i>	<b>D</b>	J2000.0 FK5 $\alpha$ (radians)
<i>D5</i>	<b>D</b>	J2000.0 FK5 $\delta$ (radians)
<i>DR5</i>	<b>D</b>	J2000.0 FK5 proper motion in $\alpha$ (radians per Julian year)
<i>DD5</i>	<b>D</b>	FK5 J2000.0 proper motion in $\delta$ (radians per Julian year)

**NOTES :**

1. The  $\alpha$  proper motions are  $\dot{\alpha}$  rather than  $\dot{\alpha} \cos \delta$ , and are per year rather than per century.
2. The FK5 to Hipparcos transformation consists of a pure rotation and spin; zonal errors in the FK5 catalogue are not taken into account.
3. The adopted epoch J2000.0 FK5 to Hipparcos orientation and spin values are as follows (see reference):

	<i>orientation</i>	<i>spin</i>
<i>x</i>	-19.9	-0.30
<i>y</i>	-9.1	+0.60
<i>z</i>	+22.9	+0.70
	<i>mas</i>	<i>mas/y</i>

These orientation and spin components are interpreted as *axial vectors*. An axial vector points at the pole of the rotation and its length is the amount of rotation in radians. 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** :

<i>RM</i>	<b>R(3,3)</b>	rotation matrix
<i>VA</i>	<b>R(3)</b>	vector to be rotated

**RETURNED** :

<i>VB</i>	<b>R(3)</b>	result vector
-----------	-------------	---------------

**NOTES :**

1. This routine performs the operation:

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

where  $\mathbf{a}$  and  $\mathbf{b}$  are the 3-vectors VA and VB respectively, and  $\mathbf{M}$  is the  $3 \times 3$  matrix RM.

2. The main function of this routine is apply an inverse rotation; under these circumstances,  $\mathbf{M}$  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 :**

<i>STRING</i>	<b>C</b>	string containing number to be decoded
<i>NSTRT</i>	<b>I</b>	pointer to where decoding is to commence
<i>IRESLT</i>	<b>I</b>	current value of result

**RETURNED :**

<i>NSTRT</i>	<b>I</b>	advanced to next number
<i>IRESLT</i>	<b>I</b>	result
<i>JFLAG</i>	<b>I</b>	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.

---

<b>SLA_INVF</b>	Invert Linear Model	<b>SLA_INVF</b>
-----------------	---------------------	-----------------

**ACTION** : Invert a linear model of the type produced by the sla\_FITXY routine.

**CALL** : CALL sla\_INVF (FWDS,BKWDS,J)

**GIVEN** :

<i>FWDS</i>	<b>D(6)</b>	model coefficients
-------------	-------------	--------------------

**RETURNED** :

<i>BKWDS</i>	<b>D(6)</b>	inverse model
<i>J</i>	<b>I</b>	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\_DCMPPF.

---

**SLA\_KBJ**

Select Epoch Prefix

**SLA\_KBJ****ACTION :** Select epoch prefix 'B' or 'J'.**CALL :** CALL sla\_KBJ (JB, E, K, J)**GIVEN :**

<i>JB</i>	<b>I</b>	sla_DBJIN prefix status: 0=none, 1='B', 2='J'
<i>E</i>	<b>D</b>	epoch – Besselian or Julian

**RETURNED :**

<i>K</i>	<b>C</b>	'B' or 'J'
<i>J</i>	<b>I</b>	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 ( $\text{km s}^{-1}$ , +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  $\alpha$  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).
5. The accuracy is limited by imperfections in the IAU 1976 and 1980 models for precession and nutation. Corrections are tabulated in IERS Bulletin B and at the present epoch are of order 50 milliarcsecond. An improved precession-nutation model can be introduced by using sla\_MAPPA and sla\_MAPQK (see the previous note) and replacing the precession-nutation matrix into the parameter array directly.
6. The accuracy is further 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.

**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 IAU 1976/80.

**CALL :** CALL sla\_MAPPA (EQ, DATE, AMPRMS)

**GIVEN :**

<i>EQ</i>	<b>D</b>	epoch of mean equinox to be used (Julian)
<i>DATE</i>	<b>D</b>	TDB (JD-2400000.5)

**RETURNED :**

<i>AMPRMS</i>	<b>D(21)</b>	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) $\times 2 / (\text{Sun-Earth distance})$
(9-11)		<b>v</b> : barycentric Earth velocity in units of $c$
(12)		$\sqrt{1 -  \mathbf{v} ^2}$
(13-21)		precession-nutation $3 \times 3$ matrix

**NOTES :**

1. For DATE, the distinction between the required TDB and TT is always negligible. Moreover, for all but the most critical applications UTC is adequate.
2. The vectors AMPRMS(2-4) and AMPRMS(5-7) are referred to the mean equinox and equator of epoch EQ.
3. The parameters produced by this routine are used by sla\_MAPQK, sla\_MAPQKZ and sla\_AMPQK.
4. The accuracy is limited by imperfections in the IAU 1976 and 1980 models for precession and nutation. Corrections are tabulated in IERS Bulletin B and at the present epoch are of order 50 mas. An improved precession-nutation model can be introduced by first calling the present routine and then replacing the precession-nutation matrix into the AMPRMS array directly.
5. A further limit to the accuracy of routines using the parameter array AMPRMS is imposed 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.

**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 :**

<i>RM,DM</i>	<b>D</b>	mean $[\alpha, \delta]$ (radians)
<i>PR,PD</i>	<b>D</b>	proper motions: $[\alpha, \delta]$ changes per Julian year
<i>PX</i>	<b>D</b>	parallax (arcsec)
<i>RV</i>	<b>D</b>	radial velocity ( $\text{km s}^{-1}$ , +ve if receding)
<i>AMPRMS</i>	<b>D(21)</b>	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) $\times 2 / (\text{Sun-Earth distance})$
(9-11)		<b>v</b> : barycentric Earth velocity in units of $c$
(12)		$\sqrt{1 -  \mathbf{v} ^2}$
(13-21)		precession-nutation $3 \times 3$ matrix

**RETURNED :**

<i>RA,DA</i>	<b>D</b>	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** :

<i>RM, DM</i>	<b>D</b>	mean $[\alpha, \delta]$ (radians)
<i>AMPRMS</i>	<b>D(21)</b>	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) $\times 2 / (\text{Sun-Earth distance})$
(9-11)		<b>v</b> : barycentric Earth velocity in units of $c$
(12)		$\sqrt{1 -  \mathbf{v} ^2}$
(13-21)		precession-nutation $3 \times 3$ matrix

**RETURNED** :

<i>RA, DA</i>	<b>D</b>	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.

---

<b>SLA_MOON</b>	Approx Moon Pos/Vel	<b>SLA_MOON</b>
-----------------	---------------------	-----------------

---

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

**CALL :** CALL sla\_MOON (IY, ID, FD, PV)

**GIVEN :**

<i>IY</i>	<b>I</b>	year
<i>ID</i>	<b>I</b>	day in year (1 = Jan 1st)
<i>FD</i>	<b>R</b>	fraction of day

**RETURNED :**

<i>PV</i>	<b>R(6)</b>	Moon [ $x, y, z, \dot{x}, \dot{y}, \dot{z}$ ], mean equator and equinox of date (AU, AU s <sup>-1</sup> )
-----------	-------------	---

**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 second in distance. (RMS figures with respect to JPL DE200 for the interval 1960-2025 are 14" and 0'.2 per hour in longitude, 9" and 0'.2 per hour in latitude, 350 km and 2 metres per second in distance.) Note that the distance accuracy is comparatively poor because this 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.

---

<b>SLA_MXM</b>	Multiply 3 × 3 Matrices	<b>SLA_MXM</b>
----------------	-------------------------	----------------

---

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

**CALL :** CALL sla\_MXM (A, B, C)

**GIVEN :**

$A$	<b>R(3,3)</b>	matrix <b>A</b>
$B$	<b>R(3,3)</b>	matrix <b>B</b>

**RETURNED :**

$C$	<b>R(3,3)</b>	matrix result: <b>A</b> × <b>B</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.

---

<b>SLA_MXV</b>	Apply 3D Rotation	<b>SLA_MXV</b>
----------------	-------------------	----------------

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

**CALL :** CALL sla\_MXV (RM, VA, VB)

**GIVEN :**

$RM$	<b>R(3,3)</b>	rotation matrix
$VA$	<b>R(3)</b>	vector to be rotated

**RETURNED :**

$VB$	<b>R(3)</b>	result vector
------	-------------	---------------

**NOTES :**

1. This routine performs the operation:  

$$\mathbf{b} = \mathbf{M} \cdot \mathbf{a}$$
where  $\mathbf{a}$  and  $\mathbf{b}$  are the 3-vectors VA and VB respectively, and  $\mathbf{M}$  is the  $3 \times 3$  matrix RM.
2. The main function of this routine is apply a rotation; under these circumstances,  $\mathbf{M}$  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.

---

<b>SLA_NUT</b>	Nutation Matrix	<b>SLA_NUT</b>
----------------	-----------------	----------------

**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  $3 \times 3$  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)



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  $90^\circ$ , the accuracy in elevation should be better than 1 arcminute; useful results are available for a further  $3^\circ$ , 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, δ*] via the standard rotation, using the geodetic latitude (corrected for polar motion), while the observed HA and RA are related simply through the local apparent ST. *Observed* [ $\alpha, \delta$ ] or [*h, δ*] thus means the position that would be seen by a perfect equatorial located at the observer and with its polar axis aligned to the Earth's axis of rotation (*n.b.* not to the refracted pole). By removing from the observed place the effects of atmospheric refraction and diurnal aberration, the geocentric apparent [ $\alpha, \delta$ ] 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  $\Delta UT$  and the UTC are consistent there are no difficulties, except during a leap second. In this case, the start of the 61st second of the final minute should begin a new MJD day and the old pre-leap  $\Delta UT$  should continue to be used. As the 61st second completes, the MJD should revert to the start of the day as, simultaneously, the  $\Delta UT$  changes by one second to its post-leap new value.
9. The  $\Delta UT$  (UT1–UTC) is tabulated in IERS circulars and elsewhere. It increases by exactly one second at the end of each UTC leap second, introduced in order to keep  $\Delta UT$  within  $\pm 0^s.9$ .
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 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** :

<i>TYPE</i>	<b>C*(* )</b>	type of coordinates – ‘R’, ‘H’ or ‘A’ (see below)
<i>OB1</i>	<b>D</b>	observed Az, HA or RA (radians; Az is N=0, E=90°)
<i>OB2</i>	<b>D</b>	observed zenith distance or $\delta$ (radians)
<i>AOPRMS</i>	<b>D(14)</b>	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 $\Delta UT$ ” (radians)
(14)		local apparent sidereal time (radians)

**RETURNED :**

<i>RAP,DAP</i>	<b>D</b>	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 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`.

---

<b>SLA_OBS</b>	Observatory Parameters	<b>SLA_OBS</b>
----------------	------------------------	----------------

---

**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** :

<i>N</i>	<b>I</b>	number specifying observing station
----------	----------	-------------------------------------

**GIVEN or RETURNED** :

<i>C</i>	<b>C*(* )</b>	identifier specifying observing station
----------	---------------	---

**RETURNED** :

<i>NAME</i>	<b>C*(* )</b>	name of specified observing station
<i>W</i>	<b>D</b>	longitude (radians, west +ve)
<i>P</i>	<b>D</b>	geodetic latitude (radians, north +ve)
<i>H</i>	<b>D</b>	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:

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

BOSQALEGRE	Bosque Alegre 1.54m
CAMB1MILE	Cambridge 1 mile
CAMB5KM	Cambridge 5 km
CATALINA61	Catalina 61 inch
CFHT	Canada-France-Hawaii 3.6m Telescope
CSO	Caltech Sub-mm Observatory, Mauna Kea
DAO72	DAO Victoria BC 1.85m
DUNLAP74	David Dunlap 74 inch
DUPONT	Du Pont 2.5m Telescope, Las Campanas
EFFELSBURG	Effelsberg 100m
ESO3.6	ESO 3.6m
ESONTT	ESO 3.5m NTT
ESOSCHM	ESO 1m Schmidt, La Silla
FCRAO	Five College Radio Astronomy Obs
FLAGSTF61	USNO 61 inch astrograph, Flagstaff
GBVA140	Greenbank 140 foot
GBVA300	Greenbank 300 foot
GEMININ	Gemini North 8m
GEMINIS	Gemini South 8m
HARVARD	Harvard College Observatory 1.55m
HPROV1.52	Haute Provence 1.52m
HPROV1.93	Haute Provence 1.93m
IRTF	NASA IR Telescope Facility, Mauna Kea
JCMT	JCMT 15m
JODRELL1	Jodrell Bank 250 foot
KECK1	Keck 10m Telescope 1
KECK2	Keck 10m Telescope 2
KISO	Kiso 1.05m Schmidt, Japan
KOSMA3M	Cologne Submillimeter Observatory 3m
KOTTAMIA	Kottamia 74 inch
KPNO158	Kitt Peak 158 inch
KPNO36FT	Kitt Peak 36 foot
KPNO84	Kitt Peak 84 inch
KPNO90	Kitt Peak 90 inch
LICK120	Lick 120 inch
LOWELL72	Perkins 72 inch, Lowell
LPO1	Jacobus Kapteyn 1m Telescope
LPO2.5	Isaac Newton 2.5m Telescope
LPO4.2	William Herschel 4.2m Telescope
MAGELLAN1	Magellan 1, 6.5m, Las Campanas
MAGELLAN2	Magellan 2, 6.5m, Las Campanas
MAUNAK88	Mauna Kea 88 inch
MCDONLD2.1	McDonald 2.1m
MCDONLD2.7	McDonald 2.7m
MMT	MMT, Mt Hopkins
MOPRA	ATNF Mopra Observatory
MTEKAR	Mt Ekar 1.82m
MTHOP1.5	Mt Hopkins 1.5m

MTLEMMON60	Mt Lemmon 60 inch
NOBEYAMA	Nobeyama 45m
OKAYAMA	Okayama 1.88m
PALOMAR200	Palomar 200 inch
PALOMAR48	Palomar 48-inch Schmidt
PALOMAR60	Palomar 60 inch
PARKES	Parkes 64m
QUEBEC1.6	Quebec 1.6m
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 8m
SUGARGROVE	Sugar Grove 150 foot
TAUTNBG	Tautenburg 2m
TAUTSCHM	Tautenberg 1.34m Schmidt
TIDBINBLA	Tidbinbilla 64m
TOLOLO1.5M	Cerro Tololo 1.5m
TOLOLO4M	Cerro Tololo 4m
UKIRT	UK Infra Red Telescope
UKST	UK 1.2m Schmidt, Siding Spring
USSR6	USSR 6m
USSR600	USSR 600 foot
VLA	Very Large Array
VLT1	ESO VLT 8m, UT1
VLT2	ESO VLT 8m, UT2
VLT3	ESO VLT 8m, UT3
VLT4	ESO VLT 8m, UT4

---

<b>SLA_PA</b>	$h, \delta$ to Parallaxial Angle	<b>SLA_PA</b>
---------------	----------------------------------	---------------

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

**CALL** : D = sla\_PA (HA, DEC, PHI)

**GIVEN** :

<i>HA</i>	<b>D</b>	hour angle in radians (geocentric apparent)
<i>DEC</i>	<b>D</b>	declination in radians (geocentric apparent)
<i>PHI</i>	<b>D</b>	latitude in radians (geodetic)

**RETURNED** :

<i>sla_PA</i>	<b>D</b>	parallaxial 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	<b>R(3)</b>	vector to one point
V2	<b>R(3)</b>	vector to the other point

**RETURNED :**

sla_PAV	<b>R</b>	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. There is no requirement for either vector to be of unit length.
4. 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	<b>D</b>	pincushion/barrel distortion coefficient
X,Y	<b>D</b>	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  $\rho$  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 the 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 :**

<i>P</i>	<b>D</b>	latitude
<i>D</i>	<b>D</b>	declination
<i>Q</i>	<b>D</b>	azimuth

**RETURNED :**

<i>H1</i>	<b>D</b>	hour angle: first solution if any
<i>J1</i>	<b>I</b>	flag: 0 = solution 1 is valid
<i>H2</i>	<b>D</b>	hour angle: second solution if any
<i>J2</i>	<b>I</b>	flag: 0 = solution 2 is valid

---

<b>SLA_PERMUT</b>	Next Permutation	<b>SLA_PERMUT</b>
-------------------	------------------	-------------------

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

**CALL :** CALL sla\_PERMUT (N, ISTATE, IORDER, J)

**GIVEN :**

<i>N</i>	<b>I</b>	number of items: there will be N! permutations
<i>ISTATE</i>	<b>I(N)</b>	state, ISTATE(1)= -1 to initialize

**RETURNED :**

<i>ISTATE</i>	<b>I(N)</b>	state, updated ready for next time
<i>IORDER</i>	<b>I(N)</b>	next permutation of numbers 1,2,...,N
<i>J</i>	<b>I</b>	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:

$$\text{IORDER}(1)=N, \text{IORDER}(2)=N-1, \dots, \text{IORDER}(N)=1$$

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

$$\text{IORDER}(1)=1, \text{IORDER}(2)=2, \dots, \text{IORDER}(N)=N$$

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) :**

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

**GIVEN (the unperturbed elements) :**

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

**RETURNED (the updated elements) :**

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

**RETURNED (status flag) :**

<i>JSTAT</i>	<b>I</b>	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 :**

- 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 $\Omega$ (radians)
PERIH	=	argument of perihelion $\omega$ (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 $\Omega$ (radians)
PERIH	=	argument of perihelion $\omega$ (radians)
AORQ	=	perihelion distance $q$ (AU)
E	=	eccentricity $e$ ( $0 \leq e \leq 10$ )

- DATE0, DATE1, EPOCH0 and EPOCH1 are all instants of time in the TT timescale (formerly Ephemeris Time, ET), expressed as Modified Julian Dates (JD-2400000.5).
  - DATE0 is the instant at which the given (*i.e.* unperturbed) osculating elements are correct.
  - DATE1 is the specified instant at which the updated osculating elements are correct.
  - EPOCH0 and EPOCH1 will be the same as DATE0 and DATE1 (respectively) for the JFORM=2 case, normally used for minor planets. For the JFORM=3 case, the two epochs will refer to perihelion passage and so will not, in general, be the same as DATE0 and/or DATE1 though they may be similar to one another.
- The elements are with respect to the J2000 ecliptic and mean equinox.
- 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** :

<i>U</i>	<b>D(13)</b>	universal elements (updated in place)
(1)		combined mass ( $M + m$ )
(2)		total energy of the orbit ( $\alpha$ )
(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)		$\mathbf{r}_0 \cdot \mathbf{v}_0$
(12)		date ( $t$ )
(13)		universal eccentric anomaly ( $\psi$ ) of date, approx

**RETURNED** :

<i>JSTAT</i>	<b>I</b>	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)  $\alpha$ , which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of  $\psi$ , the “universal eccentric anomaly” at a given date and (v) that date.
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 and relativistic effects are not taken into account.

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

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

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** :

<i>DATE</i>	<b>D</b>	TT MJD of observation (JD–2400000.5, Note 1)
<i>JFORM</i>	<b>I</b>	choice of element set (1-3, Note 3)
<i>EPOCH</i>	<b>D</b>	epoch of elements ( $t_0$ or $T$ , TT MJD, Note 4)
<i>ORBINC</i>	<b>D</b>	inclination ( $i$ , radians)
<i>ANODE</i>	<b>D</b>	longitude of the ascending node ( $\Omega$ , radians)
<i>PERIH</i>	<b>D</b>	longitude or argument of perihelion ( $\varpi$ or $\omega$ , radians)
<i>AORQ</i>	<b>D</b>	mean distance or perihelion distance ( $a$ or $q$ , AU)
<i>E</i>	<b>D</b>	eccentricity ( $e$ )
<i>AORL</i>	<b>D</b>	mean anomaly or longitude ( $M$ or $L$ , radians, JFORM=1,2 only)
<i>DM</i>	<b>D</b>	daily motion ( $n$ , radians, JFORM=1 only)

**RETURNED :**

<i>PV</i>	<b>D(6)</b>	heliocentric [ $x, y, z, \dot{x}, \dot{y}, \dot{z}$ ], equatorial, J2000 (AU, AU/s)
<i>JSTAT</i>	<b>I</b>	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. A choice of three different element-format options is available, as follows.

JFORM=1, suitable for the major planets:

EPOCH	=	epoch of elements $t_0$ (TT MJD)
ORBINC	=	inclination $i$ (radians)
ANODE	=	longitude of the ascending node $\Omega$ (radians)
PERIH	=	longitude of perihelion $\varpi$ (radians)
AORQ	=	mean distance $a$ (AU)
E	=	eccentricity $e$
AORL	=	mean longitude $L$ (radians)
DM	=	daily motion $n$ (radians)

JFORM=2, suitable for minor planets:

EPOCH = epoch of elements  $t_0$  (TT MJD)  
 ORBINC= inclination  $i$  (radians)  
 ANODE = longitude of the ascending node  $\Omega$  (radians)  
 PERIH = argument of perihelion  $\omega$  (radians)  
 AORQ = mean distance  $a$  (AU)  
 E = eccentricity  $e$   
 AORL = mean anomaly  $M$  (radians)

JFORM=3, suitable for comets:

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

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

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

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

- The epoch of observation: the moment in time for which the position of the body is to be predicted.
- The epoch defining the position of the body: the moment in time at which, in the absence of perturbations, the specified position—mean longitude, mean anomaly, or perihelion—is reached.
- The osculating epoch: the moment in time at which the given elements are correct.

For the major-planet and minor-planet cases it is usual to make the epoch that defines the position of the body the same as the epoch of osculation. Thus, only two different epochs are involved: the epoch of the elements and the epoch of observation. For comets, the epoch of perihelion fixes the position in the orbit and in general a different epoch of osculation will be chosen. Thus, all three types of epoch are involved.

For the present routine:

- The epoch of observation is the argument DATE.
- The epoch defining the position of the body is the argument EPOCH.
- The osculating epoch is not used and is assumed to be close enough to the epoch of observation to deliver adequate accuracy. If not, a preliminary call to

sla\_PERTEL may be used to update the element-set (and its associated osculating epoch) by applying planetary perturbations.

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** :

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

**RETURNED** :

<i>PV</i>	<b>D(6)</b>	heliocentric $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ , equatorial, J2000 (AU, AU/s)
<i>JSTAT</i>	<b>I</b>	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:

	<i>position (km)</i>	<i>speed (metre/sec)</i>
Mercury	334	0.437
Venus	1060	0.855
EMB	2010	0.815
Mars	7690	1.98
Jupiter	71700	7.70
Saturn	199000	19.4
Uranus	564000	16.4
Neptune	158000	14.4
Pluto	36400	0.137

From comparisons with DE102, Simon *et al.* quote the following longitude accuracies over the interval 1800-2200:

Mercury	4''
Venus	5''
EMB	6''
Mars	17''
Jupiter	71''
Saturn	81''
Uranus	86''
Neptune	11''

In the case of Pluto, Meeus quotes an accuracy of 0'6 in longitude and 0'2 in latitude for the period 1885-2099.

For all except Pluto, over the period 1000-3000, the accuracy is better than 1.5 times that over 1800-2200. Outside the interval 1000-3000 the accuracy declines. For Pluto the accuracy declines rapidly outside the period 1885-2099. Outside these ranges (1885-2099 for Pluto, 1000-3000 for the rest) a "date out of range" warning status (JSTAT=+1) is returned.

6. The algorithms for (i) Mercury through Neptune and (ii) Pluto are completely independent. In the Mercury through Neptune case, the present SLALIB implementation differs from the original Simon *et al.* Fortran code in the following respects:

- The date is supplied as a Modified Julian Date rather a Julian Date (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 processed 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** :

<i>DATE</i>	<b>D</b>	TT MJD of observation (JD–2400000.5, Notes 1,5)
<i>ELONG,PHI</i>	<b>D</b>	observer's longitude (east +ve) and latitude (radians, Note 2)
<i>JFORM</i>	<b>I</b>	choice of element set (1-3, Notes 3-6)
<i>EPOCH</i>	<b>D</b>	epoch of elements ( $t_0$ or $T$ , TT MJD, Note 5)
<i>ORBINC</i>	<b>D</b>	inclination ( $i$ , radians)
<i>ANODE</i>	<b>D</b>	longitude of the ascending node ( $\Omega$ , radians)
<i>PERIH</i>	<b>D</b>	longitude or argument of perihelion ( $\varpi$ or $\omega$ , radians)
<i>AORQ</i>	<b>D</b>	mean distance or perihelion distance ( $a$ or $q$ , AU)
<i>E</i>	<b>D</b>	eccentricity ( $e$ )
<i>AORL</i>	<b>D</b>	mean anomaly or longitude ( $M$ or $L$ , radians, JFORM=1,2 only)
<i>DM</i>	<b>D</b>	daily motion ( $n$ , radians, JFORM=1 only)

**RETURNED :**

<i>RA,DEC</i>	<b>D</b>	topocentric apparent [ $\alpha, \delta$ ] (radians)
<i>R</i>	<b>D</b>	distance from observer (AU)
<i>JSTAT</i>	<b>I</b>	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 near-Earth 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. A choice of three different element-format options is available, as follows.

JFORM=1, suitable for the major planets:

EPOCH	=	epoch of elements $t_0$ (TT MJD)
ORBINC	=	inclination $i$ (radians)
ANODE	=	longitude of the ascending node $\Omega$ (radians)
PERIH	=	longitude of perihelion $\varpi$ (radians)
AORQ	=	mean distance $a$ (AU)
E	=	eccentricity $e$
AORL	=	mean longitude $L$ (radians)
DM	=	daily motion $n$ (radians)

JFORM=2, suitable for minor planets:

EPOCH = epoch of elements  $t_0$  (TT MJD)  
 ORBINC= inclination  $i$  (radians)  
 ANODE = longitude of the ascending node  $\Omega$  (radians)  
 PERIH = argument of perihelion  $\omega$  (radians)  
 AORQ = mean distance  $a$  (AU)  
 E = eccentricity  $e$   
 AORL = mean anomaly  $M$  (radians)

JFORM=3, suitable for comets:

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

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

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

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

- The epoch of observation: the moment in time for which the position of the body is to be predicted.
- The epoch defining the position of the body: the moment in time at which, in the absence of perturbations, the specified position—mean longitude, mean anomaly, or perihelion—is reached.
- The osculating epoch: the moment in time at which the given elements are correct.

For the major-planet and minor-planet cases it is usual to make the epoch that defines the position of the body the same as the epoch of osculation. Thus, only two different epochs are involved: the epoch of the elements and the epoch of observation. For comets, the epoch of perihelion fixes the position in the orbit and in general a different epoch of osculation will be chosen. Thus, all three types of epoch are involved.

For the present routine:

- The epoch of observation is the argument DATE.
- The epoch defining the position of the body is the argument EPOCH.

- The osculating epoch is not used and is assumed to be close enough to the epoch of observation to deliver adequate accuracy. If not, a preliminary call to `sla_PERTEL` may be used to update the element-set (and its associated osculating epoch) by applying planetary perturbations.
6. Two important sources for orbital elements are *Horizons*, operated by the Jet Propulsion Laboratory, Pasadena, and the *Minor Planet Center*, operated by the Center for Astrophysics, Harvard. For further details, see Section 4.17.

---

**SLA\_PLANTU**     $[\alpha, \delta]$  from Universal Elements    **SLA\_PLANTU**

---

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

**CALL** : CALL `sla_PLANTU` (DATE, ELONG, PHI, U, RA, DEC, R, JSTAT)

**GIVEN** :

<i>DATE</i>	<b>D</b>	TT MJD of observation (JD−2400000.5)
<i>ELONG, PHI</i>	<b>D</b>	observer's longitude (east +ve) and latitude radians)
<i>U</i>	<b>D(13)</b>	universal elements (updated in place)
(1)		combined mass ( $M + m$ )
(2)		total energy of the orbit ( $\alpha$ )
(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)		$\mathbf{r}_0 \cdot \mathbf{v}_0$
(12)		date ( $t$ )
(13)		universal eccentric anomaly ( $\psi$ ) of date, approx

**RETURNED** :

<i>RA, DEC</i>	<b>D</b>	topocentric apparent $[\alpha, \delta]$ (radians)
<i>R</i>	<b>D</b>	distance from observer (AU)
<i>JSTAT</i>	<b>I</b>	status:
		0 = OK
		−1 = radius vector zero
		−2 = failed to converge

**NOTES** :

1. DATE is the instant for which the prediction is required. It is in the 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 near-Earth asteroids. Geocentric positions can be generated by appropriate use of the routines `sla_EVP` and `sla_UE2PV`.

3. The “universal” elements are those which define the orbit for the purposes of the method of universal variables (see reference 2). They consist of the combined mass of the two bodies, an epoch, and the position and velocity vectors (arbitrary reference frame) at that epoch. The parameter set used here includes also various quantities that can, in fact, be derived from the other information. This approach is taken to avoiding unnecessary computation and loss of accuracy. The supplementary quantities are (i)  $\alpha$ , which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of  $\psi$ , the “universal eccentric anomaly” at a given date and (v) that date.
4. The universal elements are with respect to the J2000 ecliptic and equinox.

#### 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_PM	Proper Motion	SLA_PM
--------	---------------	--------

---

**ACTION :** Apply corrections for proper motion to a star  $[\alpha, \delta]$ .

**CALL :** CALL sla\_pm (R0, D0, PR, PD, PX, RV, EP0, EP1, R1, D1)

#### GIVEN :

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

#### RETURNED :

<i>R1,D1</i>	D	[ $\alpha, \delta$ ] at epoch EP1 (radians)
--------------	---	---

#### NOTES :

1. The  $\alpha$  proper motions are  $\dot{\alpha}$  rather than  $\dot{\alpha} \cos \delta$ , and are in the same coordinate system as R0,D0.
2. If the available proper motions are pre-FK5 they will be per tropical year rather than per Julian year, and so the epochs must both be Besselian rather than Julian. In such cases, a scaling factor of 365.2422D0/365.25D0 should be applied to the radial velocity before use also.

**REFERENCES :**

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

---

**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 :**

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

**RETURNED :**

<i>ELONG</i>	<b>D</b>	true longitude of the site (radians, east +ve)
<i>PHI</i>	<b>D</b>	true geodetic latitude of the site (radians)
<i>DAZ</i>	<b>D</b>	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 0'3.
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:

$$\begin{aligned} \text{ELONG} &= \text{ELONGM} + \text{XP} * \text{COS}(\text{ELONGM}) - \text{YP} * \text{SIN}(\text{ELONGM}) \\ \text{PHI} &= \text{PHIM} + (\text{XP} * \text{SIN}(\text{ELONGM}) + \text{YP} * \text{COS}(\text{ELONGM})) * \text{TAN}(\text{PHIM}) \\ \text{DAZ} &= -\text{SQRT}(\text{XP} * \text{XP} + \text{YP} * \text{YP}) * \text{COS}(\text{ELONGM} - \text{ATAN2}(\text{XP}, \text{YP})) / \text{COS}(\text{PHIM}) \end{aligned}$$

An alternative formulation for DAZ is:

$$\begin{aligned} \text{X} &= \text{COS}(\text{ELONGM}) * \text{COS}(\text{PHIM}) \\ \text{Y} &= \text{SIN}(\text{ELONGM}) * \text{COS}(\text{PHIM}) \\ \text{DAZ} &= \text{ATAN2}(-\text{X} * \text{YP} - \text{Y} * \text{XP}, \text{X} * \text{X} + \text{Y} * \text{Y}) \end{aligned}$$

**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** :

<i>BEP0</i>	<b>D</b>	beginning Besselian epoch
<i>BEP1</i>	<b>D</b>	ending Besselian epoch

**RETURNED** :

<i>RMATP</i>	<b>D(3,3)</b>	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  $3 \times 3$  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** :

<i>EP0</i>	<b>D</b>	beginning epoch
<i>EP1</i>	<b>D</b>	ending epoch

**RETURNED** :

<i>RMATP</i>	<b>D(3,3)</b>	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  $3 \times 3$  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  $0''.1$  from 1960 AD to 2040 AD, better than  $1''$  from 1640 AD to 2360 AD, and remains below  $3''$  for the whole of the period 500 BC to 3000 AD. The errors exceed  $10''$  outside the range 1200 BC to 3900 AD, exceed  $100''$  outside 4200 BC to 5600 AD and exceed  $1000''$  outside 6800 BC to 8200 AD. The SLALIB 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 :**

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

**RETURNED :**

<i>RA,DC</i>	<b>D</b>	$[\alpha, \delta]$ , 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.

---

<b>SLA_PRECL</b>	Precession Matrix (latest)	<b>SLA_PRECL</b>
------------------	----------------------------	------------------

**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 :**

<i>EP0</i>	<b>D</b>	beginning epoch
<i>EP1</i>	<b>D</b>	ending epoch

**RETURNED :**

<i>RMATP</i>	<b>D(3,3)</b>	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  $3 \times 3$  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 :**

<i>EPOCH</i>	<b>D</b>	Julian Epoch for mean coordinates
<i>DATE</i>	<b>D</b>	Modified Julian Date (JD−2400000.5) for true coordinates

**RETURNED :**

<i>RMATPN</i>	<b>D(3,3)</b>	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  $3 \times 3$  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 :**

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

**RETURNED :**

<i>JFORM</i>	<b>I</b>	element set actually returned (1-3; Note 4)
<i>EPOCH</i>	<b>D</b>	epoch of elements ( $t_0$ or $T$ , TT MJD)
<i>ORBINC</i>	<b>D</b>	inclination ( $i$ , radians)
<i>ANODE</i>	<b>D</b>	longitude of the ascending node ( $\Omega$ , radians)
<i>PERIH</i>	<b>D</b>	longitude or argument of perihelion ( $\varpi$ or $\omega$ , radians)
<i>AORQ</i>	<b>D</b>	mean distance or perihelion distance ( $a$ or $q$ , AU)
<i>E</i>	<b>D</b>	eccentricity ( $e$ )
<i>AORL</i>	<b>D</b>	mean anomaly or longitude ( $M$ or $L$ , radians, JFORM=1,2 only)
<i>DM</i>	<b>D</b>	daily motion ( $n$ , radians, JFORM=1 only)
<i>JSTAT</i>	<b>I</b>	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 $\Omega$ (radians)
PERIH	=	longitude of perihelion $\varpi$ (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  $\Omega$  (radians)  
 PERIH = argument of perihelion  $\omega$  (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  $\Omega$  (radians)  
 PERIH = argument of perihelion  $\omega$  (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	$\Omega$	$\Omega$	$\Omega$
PERIH	$\varpi$	$\omega$	$\omega$
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)
$\Omega$	is the longitude of the ascending node (radians)
$\varpi$	is the longitude of perihelion (radians)
$\omega$	is the argument of perihelion (radians)
$a$	is the mean distance (AU)
$q$	is the perihelion distance (AU)
$e$	is the eccentricity
$L$	is the longitude (radians, $0 - 2\pi$ )
$M$	is the mean anomaly (radians, $0 - 2\pi$ )
$n$	is the daily motion (radians)
-	means no value is set

6. At very small inclinations, the longitude of the ascending node ANODE becomes indeterminate and under some circumstances may be set arbitrarily to zero. Similarly, if the orbit is close to circular, the true anomaly becomes indeterminate and under some circumstances may be set arbitrarily to zero. In such cases, the other elements are automatically adjusted to compensate, and so the elements remain a valid description of the orbit.

7. The osculating epoch for the returned elements is the argument DATE.

**REFERENCE** : Sterne, Theodore E., *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** :

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

**RETURNED** :

<i>U</i>	<b>D(13)</b>	universal orbital elements (Note 3)
(1)		combined mass ( $M + m$ )
(2)		total energy of the orbit ( $\alpha$ )
(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)		$\mathbf{r}_0 \cdot \mathbf{v}_0$
(12)		date ( $t$ )
(13)		universal eccentric anomaly ( $\psi$ ) of date, approx
<i>JSTAT</i>	<b>I</b>	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)  $\alpha$ , which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of  $\psi$ , the “universal eccentric anomaly” at a given date and (v) that date.

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

---

**SLA\_PVOBS**      Observatory Position & Velocity      **SLA\_PVOBS**

**ACTION** : Position and velocity of an observing station.

**CALL** : CALL sla\_PVOBS (P, H, STL, PV)

**GIVEN :**

$P$	<b>D</b>	latitude (geodetic, radians)
$H$	<b>D</b>	height above reference spheroid (geodetic, metres)
$STL$	<b>D</b>	local apparent sidereal time (radians)

**RETURNED :**

$PV$	<b>D(6)</b>	$[x, y, z, \dot{x}, \dot{y}, \dot{z}]$ (AU, AU s <sup>-1</sup> , true equator and equinox of date)
------	-------------	--

**NOTE :** IAU 1976 constants are used.

---

<b>SLA_PXY</b>	Apply Linear Model	<b>SLA_PXY</b>
----------------	--------------------	----------------

---

**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 :**

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

**RETURNED :**

$XYP$	<b>D(2,NP)</b>	predicted $[x, y]$ for each sample
$XRMS$	<b>D</b>	RMS in X
$YRMS$	<b>D</b>	RMS in Y
$RRMS$	<b>D</b>	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:

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

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: the difference between *proper time* at a point on the Earth and *coordinate time* in the solar system barycentric space-time frame of reference. The proper time is Terrestrial Time, TT; the coordinate time is an implementation of Barycentric Dynamical Time, TDB.

**GIVEN :**

*TDB*                    **D**            TDB (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; Note 1)

**NOTES :**

1. TDB is coordinate time in the solar system barycentre frame of reference, in units chosen to eliminate the scale difference with respect to terrestrial time. TT is the proper time for clocks at mean sea level on the Earth.
2. The number returned by sla\_RCC comprises a main (annual) sinusoidal term of amplitude approximately 1.66ms, plus lunar and planetary terms up to about  $20\mu\text{s}$ , and diurnal terms up to  $2\mu\text{s}$ . The variation arises from the transverse Doppler effect and the gravitational red-shift as the observer varies in speed and moves through different gravitational potentials.
3. The argument TDB is, strictly, the barycentric coordinate time; however, the terrestrial time (TT) can in practice be used without significant loss of accuracy.

4. The geocentric model is that of Fairhead & Bretagnon (1990), in its full form. It was supplied by Fairhead (private communication) as a Fortran subroutine. A number of coding changes were made to this subroutine in order match the calling sequence of previous versions of the present routine, to comply with Starlink programming standards and to avoid compilation problems on certain machines. The numerical results are essentially unaffected by the changes.
5. The topocentric model is from Moyer (1981) and Murray (1983). It is an approximation to the expression

$$\frac{\mathbf{v}_e \cdot (\mathbf{x} - \mathbf{x}_e)}{c^2}$$

where  $\mathbf{v}_e$  is the barycentric velocity of the Earth,  $\mathbf{x}$  and  $\mathbf{x}_e$  are the barycentric positions of the observer and the Earth respectively, and  $c$  is the speed of light. It can be disabled, if necessary, by setting the arguments U and V to zero.

6. During the interval 1950-2050, the absolute accuracy is better than  $\pm 3$  nanoseconds relative to direct numerical integrations using the JPL DE200/LE200 solar system ephemeris.
7. The IAU 1976 definition of TDB was that it must differ from TT only by periodic terms. Though practical, this is an imprecise definition 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. In 1991 the IAU introduced new timescales designed to overcome these objections: geocentric coordinate time, TCG, and barycentric coordinate time, TCB. In principle, therefore, TDB is obsolete. However, sla\_RCC can be used to implement the periodic part of TCB–TCG.

## REFERENCES :

1. Fairhead, L., & Bretagnon, P., *Astron. Astrophys.*, **229**, 240-247 (1990).
2. Moyer, T.D., *Cel. Mech.*, **23**, 33 (1981).
3. Murray, C.A., *Vectorial Astrometry*, Adam Hilger (1983).
4. Seidelmann, P.K. *et al*, *Explanatory Supplement to the Astronomical Almanac*, Chapter 2, University Science Books (1992).
5. Simon, J.L., Bretagnon, P., Chapront, J., Chapront-Touze, M., Francou, G. & Laskar, J., *Astron. Astrophys.*, **282**, 663-683 (1994).

---

**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 :**

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

**RETURNED :**

<i>RA,DEC</i>	<b>D</b>	topocentric apparent [ $\alpha, \delta$ ] (radians)
<i>DIAM</i>	<b>D</b>	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  $\zeta$  is the *observed* zenith distance (*i.e.* affected by refraction) and  $\Delta\zeta$  is what to add to  $\zeta$  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** :

<i>HM</i>	D	height of the observer above sea level (metre)
<i>TDK</i>	D	ambient temperature at the observer (degrees K)
<i>PMB</i>	D	pressure at the observer (mB)
<i>RH</i>	D	relative humidity at the observer (range 0–1)
<i>WL</i>	D	effective wavelength of the source ( $\mu\text{m}$ )
<i>PHI</i>	D	latitude of the observer (radian, astronomical)
<i>TLR</i>	D	temperature lapse rate in the troposphere (degrees K per metre)
<i>EPS</i>	D	precision required to terminate iteration (radian)

**RETURNED** :

<i>REFA</i>	D	$\tan \zeta$ coefficient (radians)
<i>REFB</i>	D	$\tan^3 \zeta$ coefficient (radians)

**NOTES** :

1. Suggested values for the TLR and EPS arguments are 0.0065D0 and 1D–8 respectively. The signs of both are immaterial.
2. The radio refraction is chosen by specifying  $WL > 100 \mu\text{m}$ .
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  $\tan^{-1} 1$  ( $45^\circ$ ) and  $\tan^{-1} 4$  ( $\sim 76^\circ$ ). At other zenith distances, the model achieves: 0'5 accuracy for  $\zeta < 80^\circ$ , 0'01 accuracy for  $\zeta < 60^\circ$ , and 0'001 accuracy for  $\zeta < 45^\circ$ .

---

**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  $\zeta$  is the *observed* zenith distance (*i.e.* affected by refraction) and  $\Delta\zeta$  is what to add to  $\zeta$  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 :**

<i>TDK</i>	<b>D</b>	ambient temperature at the observer (degrees K)
<i>PMB</i>	<b>D</b>	pressure at the observer (mB)
<i>RH</i>	<b>D</b>	relative humidity at the observer (range 0–1)
<i>WL</i>	<b>D</b>	effective wavelength of the source ( $\mu\text{m}$ )

**RETURNED :**

<i>REFA</i>	<b>D</b>	$\tan \zeta$ coefficient (radians)
<i>REFB</i>	<b>D</b>	$\tan^3 \zeta$ coefficient (radians)

**NOTES :**

1. The radio refraction is chosen by specifying  $WL > 100 \mu\text{m}$ .
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_REFRO` 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 ( $45^\circ$  and  $\sim 76^\circ$ ) and at other zenith distances is limited in accuracy only by the  $\Delta\zeta = a \tan \zeta + b \tan^3 \zeta$  formulation itself. The present 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  $0^\circ$ ,  $25^\circ$ ,  $50^\circ$ ,  $75^\circ$
  - heights 0, 2500, 5000 metres above sea level
  - pressures mean for height  $-10\%$  to  $+5\%$  in steps of 5%
  - temperatures  $-10^\circ$  to  $+20^\circ$  with respect to  $280^\circ\text{K}$  at sea level
  - relative humidity 0, 0.5, 1
  - wavelength 0.4, 0.6, ...  $2\mu\text{m}$ , + radio
  - zenith distances  $15^\circ$ ,  $45^\circ$ ,  $75^\circ$

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

	<i>worst</i>	<i>RMS</i>
optical/IR	62	8
radio	319	49
	mas	mas

For this particular set of conditions:

- lapse rate  $6.5^\circ \text{K km}^{-1}$

- latitude 50°
- sea level
- pressure 1005 mB
- temperature 7° C
- humidity 80%
- wavelength 5740 Å

the results were as follows:

$\zeta$	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\_REFRO routine have not been included in the tabulation as they are identical to those in the sla\_REFRO column to the 0.01 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 Hohenkerk & Sinclair (1985) and Rueger (2002).

The above three items are as used in the sla\_REFRO routine.

- The formula for  $\beta$  ( $= H_0/r_0$ ) 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  $\beta$  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. Green, R.M., *Spherical Astronomy*, Cambridge University Press, 1987.
4. Hohenkerk, C.Y., & Sinclair, A.T., NAO Technical Note No. 63, 1985.
5. Rueger, J.M., *Refractive Index Formulae for Electronic Distance Measurement with Radio and Millimetre Waves*, in Unisurv Report S-68, School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia, 2002.
6. Stone, Ronald C., P.A.S.P. **108** 1051-1058, 1996.

---

<b>SLA_REFRO</b>	Refraction	<b>SLA_REFRO</b>
------------------	------------	------------------

---

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

**CALL** : CALL sla\_REFRO (Z OBS, HM, TDK, PMB, RH, WL, PHI, TLR, EPS, REF)

## GIVEN :

<i>Z OBS</i>	<b>D</b>	observed zenith distance of the source (radians)
<i>HM</i>	<b>D</b>	height of the observer above sea level (metre)
<i>TDK</i>	<b>D</b>	ambient temperature at the observer (degrees K)
<i>PMB</i>	<b>D</b>	pressure at the observer (mB)
<i>RH</i>	<b>D</b>	relative humidity at the observer (range 0–1)
<i>WL</i>	<b>D</b>	effective wavelength of the source ( $\mu\text{m}$ )
<i>PHI</i>	<b>D</b>	latitude of the observer (radian, astronomical)
<i>TLR</i>	<b>D</b>	temperature lapse rate in the troposphere (degrees K per metre)
<i>EPS</i>	<b>D</b>	precision required to terminate iteration (radian)

## RETURNED :

<i>REF</i>	<b>D</b>	refraction: <i>in vacuo</i> ZD minus observed ZD (radians)
------------	----------	--

## NOTES :

1. A suggested value for the TLR argument is 0.0065D0 (sign immaterial). 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  $90^\circ$  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). The refractivity model currently used is from J.M. Rueger, *Refractive Index Formulae for Electronic Distance Measurement with Radio and Millimetre Waves*, in Unisurv Report S-68 (2002), School of Surveying and Spatial Information Systems, University of New South Wales, Sydney, Australia.

None of the changes significantly affects the optical/IR results with respect to the algorithm given in the 1992 *Explanatory Supplement*. For example, at  $70^\circ$  zenith distance the present routine agrees with the ES algorithm to better than  $0''.05$  for any reasonable combination of parameters. (The improved water-vapour expressions do make a significant difference in the radio band, though, at  $70^\circ$  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  $WL > 100 \mu\text{m}$ . 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  $93^\circ$ , a fixed REF value equal to the result for  $ZOBS = 93^\circ$  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  $0''.1$  level is achieved for moderate ZD, worsening

to perhaps 0'5 – 1'0 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** :

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

**RETURNED** :

<i>VR</i>	<b>D</b>	refracted position of the source ([ <i>Az</i> , <i>El</i> ] 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  $\zeta = 70^\circ$ . The inherent accuracy of the model is, of course, far worse than this – see the documentation for sla\_REFV 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:

$\zeta_{obs}$	<i>error</i>
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  $\zeta = 84^\circ$  sla\_REFZ should be used, and for the utmost accuracy iterative use of sla\_REFRO should be considered.

---

<b>SLA_REFZ</b>	Apply Refraction to ZD	<b>SLA_REFZ</b>
-----------------	------------------------	-----------------

---

**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** :

<i>ZU</i>	<b>D</b>	unrefracted zenith distance of the source (radians)
<i>REFA</i>	<b>D</b>	$\tan \zeta$ coefficient (radians)
<i>REFB</i>	<b>D</b>	$\tan^3 \zeta$ coefficient (radians)

**RETURNED** :

<i>ZR</i>	<b>D</b>	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  $\zeta = 80^\circ$ . The inherent accuracy of the model is, of course, far worse than this – see the documentation for sla\_REFCO for more information.

- At  $\zeta = 83^\circ$ , the rapidly-worsening  $\Delta\zeta = a \tan \zeta + b \tan^3 \zeta$  model is abandoned and an empirical formula takes over:

$$\Delta\zeta = F \left( \frac{0^\circ 55' 44.5'' - 0^\circ 01' 13.3'' E + 0^\circ 00' 20.2'' E^2}{1 + 0.28385 E + 0.02390 E^2} \right)$$

where  $E = 90^\circ - \zeta_{true}$  and  $F$  is a factor chosen to meet the  $\Delta\zeta = a \tan \zeta + b \tan^3 \zeta$  formula at  $\zeta = 83^\circ$ . 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:

$\zeta_{obs}$	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

- 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 Corr 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** :

<i>PHI</i>	<b>R</b>	geodetic latitude of observing station (radians)
<i>RA, DA</i>	<b>R</b>	apparent $[\alpha, \delta]$ (radians)
<i>ST</i>	<b>R</b>	local apparent sidereal time (radians)

**RETURNED :**

*sla\_RVEROT*     **R**                    Component of Earth rotation in direction RA,DA  
(km s<sup>-1</sup>)

**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<sup>-1</sup>. 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<sup>-1</sup>)

**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<sup>-1</sup> towards  $l^{\text{II}} = 90^\circ$ ,  $b^{\text{II}} = 0$ .

**REFERENCE** : Kerr & Lynden-Bell (1986), MNRAS, 221, p1023.

---

**SLA\_RVLG**                      RV Corr 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<sup>-1</sup>)

**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 Corr 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<sup>-1</sup>)

**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<sup>-1</sup>. This corresponds to about 16.6 km s<sup>-1</sup> towards Galactic coordinates  $l^{II} = 53^\circ, b^{II} = +25^\circ$ .

---

**SLA\_RVLSRK**      RV Corr 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<sup>-1</sup>)

**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<sup>-1</sup> towards  $\alpha = 18^h, \delta = +30^\circ$  (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 :**

<i>RA,DEC</i>	<b>R</b>	spherical coordinates of star (radians)
<i>RAZ,DECZ</i>	<b>R</b>	spherical coordinates of tangent point (radians)

**RETURNED :**

<i>XI,ETA</i>	<b>R</b>	tangent plane coordinates (radians)
<i>J</i>	<b>I</b>	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 :**

<i>A1,B1</i>	<b>R</b>	spherical coordinates of one point (radians)
<i>A2,B2</i>	<b>R</b>	spherical coordinates of the other point (radians)

**RETURNED :**

*sla\_SEPV*            **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\_SEPV**            Angle Between 2 Vectors            **SLA\_SEPV**

**ACTION :** Angle between two vectors (single precision).

**CALL :** R = sla\_SEPV (V1, V2)

**GIVEN :**

*V1*                    **R(3)**            first vector  
*V2*                    **R(3)**            second vector

**RETURNED :**

*sla\_SEPV*            **R**            angle between V1 and V2 in radians

**NOTES :**

1. There is no requirement for either vector to be of unit length.
2. If either vector is null, zero is returned.
3. The result is always positive.

---

**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 :**

<i>A</i>	<b>R(N,N)</b>	matrix inverse
<i>Y</i>	<b>R(N)</b>	solution
<i>D</i>	<b>R</b>	determinant
<i>JF</i>	<b>I</b>	singularity flag: 0=OK
<i>IW</i>	<b>I(N)</b>	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\_SMAT 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=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\_SMATRIX.

---

**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 :**

<i>RC,DC</i>	<b>D</b>	$[\alpha, \delta]$ with E-terms included (radians)
<i>EQ</i>	<b>D</b>	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\_SUPGAL (DSL, DSB, DL, DB)

**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  $l^II = 137.37$ .)

---

**SLA\_SVD**                      Singular Value Decomposition                      **SLA\_SVD**


---

**ACTION** : Singular value decomposition. This routine expresses a given matrix **A** as the product of three matrices **U**, **W**, **V<sup>T</sup>**:

$$\mathbf{A} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T$$

where:

**A**        is any  $m$  (rows)  $\times n$  (columns) matrix, where  $m \geq n$   
**U**        is an  $m \times n$  column-orthogonal matrix  
**W**        is an  $n \times n$  diagonal matrix with  $w_{ii} \geq 0$   
**V<sup>T</sup>**      is the transpose of an  $n \times n$  orthogonal matrix

**CALL** : CALL sla\_SVD (M, N, MP, NP, A, W, V, WORK, JSTAT)

**GIVEN** :

<i>M,N</i>	<b>I</b>	$m, n$ , the numbers of rows and columns in matrix <b>A</b>
<i>MP,NP</i>	<b>I</b>	physical dimensions of array containing matrix <b>A</b>
<i>A</i>	<b>D(MP,NP)</b>	array containing $m \times n$ matrix <b>A</b>

**RETURNED** :

<i>A</i>	<b>D(MP,NP)</b>	array containing $m \times n$ column-orthogonal matrix <b>U</b>
<i>W</i>	<b>D(N)</b>	$n \times n$ diagonal matrix <b>W</b> (diagonal elements only)
<i>V</i>	<b>D(NP,NP)</b>	array containing $n \times n$ orthogonal matrix <b>V</b> ( <i>n.b.</i> not <b>V<sup>T</sup></b> )
<i>WORK</i>	<b>D(N)</b>	workspace
<i>JSTAT</i>	<b>I</b>	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  $\mathbf{W}$  have not been correctly found. However, in practice the results are likely to be trustworthy. Applications should report the condition as a warning, but then proceed normally.

**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  $\mathbf{W}$  and  $\mathbf{V}$  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** :

$N$	<b>I</b>	$n$ , the number of rows and columns in matrices $\mathbf{W}$ and $\mathbf{V}$
$NP$	<b>I</b>	first dimension of array containing $n \times n$ matrix $\mathbf{V}$
$NC$	<b>I</b>	first dimension of array CVM
$W$	<b>D(N)</b>	$n \times n$ diagonal matrix $\mathbf{W}$ (diagonal elements only)
$V$	<b>D(NP, NP)</b>	array containing $n \times n$ orthogonal matrix $\mathbf{V}$

**RETURNED** :

$WORK$	<b>D(N)</b>	workspace
$CVM$	<b>D(NC, NC)</b>	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:

**A**        is a given  $m$  (rows)  $\times n$  (columns) matrix, where  $m \geq n$   
**x**        is the  $n$ -vector we wish to find, and  
**b**        is a given  $m$ -vector

by means of the *Singular Value Decomposition* method (SVD).

**CALL** : CALL sla\_SVDSOL (M, N, MP, NP, B, U, W, V, WORK, X)

**GIVEN** :

$M, N$	<b>I</b>	$m, n$ , the numbers of rows and columns in matrix <b>A</b>
$MP, NP$	<b>I</b>	physical dimensions of array containing matrix <b>A</b>
$B$	<b>D(M)</b>	known vector <b>b</b>
$U$	<b>D(MP, NP)</b>	array containing $m \times n$ matrix <b>U</b>
$W$	<b>D(N)</b>	$n \times n$ diagonal matrix <b>W</b> (diagonal elements only)
$V$	<b>D(NP, NP)</b>	array containing $n \times n$ orthogonal matrix <b>V</b>

**RETURNED** :

$WORK$	<b>D(N)</b>	workspace
$X$	<b>D(N)</b>	unknown vector <b>x</b>

**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:

**A**        is any  $m$  (rows)  $\times n$  (columns) matrix, where  $m > n$   
**U**        is an  $m \times n$  column-orthogonal matrix  
**W**        is an  $n \times n$  diagonal matrix with  $w_{ii} \geq 0$   
**V<sup>T</sup>**      is the transpose of an  $n \times n$  orthogonal matrix

Note that  $m$  and  $n$  are the *logical* dimensions of the matrices and vectors concerned, which can be located in arrays of larger *physical* dimensions MP and NP. The solution is then found from the expression:

$$\mathbf{x} = \mathbf{V} \cdot [\text{diag}(1/\mathbf{W}_j)] \cdot (\mathbf{U}^T \cdot \mathbf{b})$$

2. If matrix  $\mathbf{A}$  is square, and if the diagonal matrix  $\mathbf{W}$  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** :

<i>XI,ETA</i>	<b>R</b>	tangent plane rectangular coordinates (radians)
<i>RAZ,DECZ</i>	<b>R</b>	spherical coordinates of tangent point (radians)

**RETURNED** :

<i>RA,DEC</i>	<b>R</b>	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 :**

$XI, ETA$	<b>R</b>	tangent plane coordinates of star (radians)
$V0$	<b>R(3)</b>	direction cosines of tangent point

**RETURNED :**

$V$	<b>R(3)</b>	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`.

---

<b>SLA_TPS2C</b>	Plate centre from $\xi, \eta$ and $\alpha, \delta$	<b>SLA_TPS2C</b>
------------------	--	------------------

**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 :**

$XI, ETA$	<b>R</b>	tangent plane rectangular coordinates (radians)
$RA, DEC$	<b>R</b>	spherical coordinates (radians)

**RETURNED :**

$RAZ1, DECZ1$	<b>R</b>	spherical coordinates of tangent point, solution 1
$RAZ2, DECZ2$	<b>R</b>	spherical coordinates of tangent point, solution 2
$N$	<b>I</b>	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  $\xi$  value, and hence it is meaningless to ask where the tangent point would have to be to bring about this combination of  $\xi$  and  $\delta$ .
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\_TPV2C is available.

---

**SLA\_TPV2C**      Plate centre from  $\xi, \eta$  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 :**

$XI, ETA$	<b>R</b>	tangent plane coordinates of star (radians)
$V$	<b>R(3)</b>	direction cosines of star

**RETURNED :**

$V01$	<b>R(3)</b>	direction cosines of tangent point, solution 1
$V01$	<b>R(3)</b>	direction cosines of tangent point, solution 2
$N$	<b>I</b>	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 :**

$U$	<b>D(13)</b>	universal orbital elements (updated; Note 1)
(1)		combined mass ( $M + m$ )
(2)		total energy of the orbit ( $\alpha$ )
(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)		$\mathbf{r}_0 \cdot \mathbf{v}_0$
(12)		date ( $t$ )
(13)		universal eccentric anomaly ( $\psi$ ) of date, approx
$JFORMR$	<b>I</b>	requested element set (1-3; Note 3)

**RETURNED :**

<i>JFORM</i>	I	element set actually returned (1-3; Note 4)
<i>EPOCH</i>	D	epoch of elements ( $t_0$ or $T$ , TT MJD)
<i>ORBINC</i>	D	inclination ( $i$ , radians)
<i>ANODE</i>	D	longitude of the ascending node ( $\Omega$ , radians)
<i>PERIH</i>	D	longitude or argument of perihelion ( $\varpi$ or $\omega$ , radians)
<i>AORQ</i>	D	mean distance or perihelion distance ( $a$ or $q$ , AU)
<i>E</i>	D	eccentricity ( $e$ )
<i>AORL</i>	D	mean anomaly or longitude ( $M$ or $L$ , radians, JFORM=1,2 only)
<i>DM</i>	D	daily motion ( $n$ , radians, JFORM=1 only)
<i>JSTAT</i>	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)  $\alpha$ , which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of  $\psi$ , the “universal eccentric anomaly” at a given date and (v) that date.
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 $\Omega$ (radians)
PERIH	=	longitude of perihelion $\varpi$ (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  $\Omega$  (radians)  
 PERIH = argument of perihelion  $\omega$  (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  $\Omega$  (radians)  
 PERIH = argument of perihelion  $\omega$  (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	$\Omega$	$\Omega$	$\Omega$
PERIH	$\varpi$	$\omega$	$\omega$
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)
$\Omega$	is the longitude of the ascending node (radians)
$\varpi$	is the longitude of perihelion (radians)
$\omega$	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** :

<i>U</i>	<b>D(13)</b>	universal orbital elements (updated; Note 1)
(1)		combined mass ( $M + m$ )
(2)		total energy of the orbit ( $\alpha$ )
(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)		$\mathbf{r}_0 \cdot \mathbf{v}_0$
(12)		date ( $t$ )
(13)		universal eccentric anomaly ( $\psi$ ) of date, approx

**RETURNED :**

<i>PV</i>	<b>D(6)</b>	heliocentric [ $x, y, z, \dot{x}, \dot{y}, \dot{z}$ ], equatorial, J2000 (AU, AU/s; Note 1)
<i>JSTAT</i>	<b>I</b>	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)  $\alpha$ , which is proportional to the total energy of the orbit, (ii) the heliocentric distance at epoch, (iii) the outwards component of the velocity at the given epoch, (iv) an estimate of  $\psi$ , the “universal eccentric anomaly” at a given date and (v) that date.
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** :

<i>DISCO</i>	<b>D</b>	pincushion/barrel distortion coefficient
<i>X, Y</i>	<b>D</b>	distorted $[x, y]$

**RETURNED** :

<i>X, Y</i>	<b>D</b>	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  $\rho$  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 a rigorous inverse of the companion routine sla\_PCD. The expression for  $\rho$  in Note 1 is rewritten in the form  $x^3 + ax + b = 0$  and solved by standard techniques.

5. Cases where the cubic has multiple real roots can sometimes occur, corresponding to extreme instances of barrel distortion where up to three different undistorted  $[x, y]$ s all produce the same distorted  $[x, y]$ . However, only one solution is returned, the one that produces the smallest change in  $[x, y]$ .

---

**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	<b>R(3)</b>	direction cosines of star
V0	<b>R(3)</b>	direction cosines of tangent point

**RETURNED** :

XI,ETA	<b>R</b>	tangent plane coordinates (radians)
J	<b>I</b>	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  $\xi, \eta$  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 :**

$VA$	<b>R(3)</b>	first vector
$VB$	<b>R(3)</b>	second vector

**RETURNED :**

$sla\_VDV$	<b>R</b>	scalar product $VA.VB$
------------	----------	------------------------

---

<b>SLA_VN</b>	Normalize Vector	<b>SLA_VN</b>
---------------	------------------	---------------

**ACTION :** Normalize a 3-vector, also giving the modulus (single precision).**CALL :** CALL `sla_VN (V, UV, VM)`**GIVEN :**

$V$	<b>R(3)</b>	vector
-----	-------------	--------

**RETURNED :**

$UV$	<b>R(3)</b>	unit vector in direction of $V$
$VM$	<b>R</b>	modulus of $V$

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

---

<b>SLA_VXV</b>	Vector Product	<b>SLA_VXV</b>
----------------	----------------	----------------

**ACTION :** Vector product of two 3-vectors (single precision).**CALL :** CALL `sla_VXV (VA, VB, VC)`**GIVEN :**

$VA$	<b>R(3)</b>	first vector
$VB$	<b>R(3)</b>	second vector

**RETURNED :**

$VC$	<b>R(3)</b>	vector product $VA \times VB$
------	-------------	-------------------------------

---

<b>SLA_WAIT</b>	Time Delay	<b>SLA_WAIT</b>
-----------------	------------	-----------------

**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:

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

2. See also sla\_FITXY, sla\_PXY, sla\_INVF, sla\_DCMPPF.

---

**SLA\_ZD**                     $h, \delta$  to Zenith Distance                    **SLA\_ZD**

**ACTION :** Hour angle and declination to zenith distance (double precision).

**CALL :** D = sla\_ZD (HA, DEC, PHI)

**GIVEN :**

<i>HA</i>	<b>D</b>	hour angle in radians
<i>DEC</i>	<b>D</b>	declination in radians
<i>PHI</i>	<b>D</b>	latitude in radians

**RETURNED :**

<i>sla_ZD</i>	<b>D</b>	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\_SEPV, sla\_DSEPV, sla\_PAV, sla\_DPAV are equivalents of sla\_SEP, sla\_DSEP, sla\_BEAR and sla\_DBEAR but starting from vectors 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  $0^\circ$  to  $-1^\circ$  mysteriously migrate to the range  $0^\circ$  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  $24^{\text{h}} 59^{\text{m}} 59.999$ . 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  $3 \times 3$  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  $\mathbf{v}_1$ , the resulting vector  $\mathbf{v}_2$  has direction  $\langle \mathbf{v}_1 + \mathbf{d} \rangle$  but is no longer of unit length. A better approximation is available if the result is multiplied by a scaling factor of  $(1 - \mathbf{d} \cdot \mathbf{v}_1)$ , where the dot means scalar product. In 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  $\delta E$  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  $\delta E$  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  $\delta E$ . 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  $\delta E$  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:

<i>system</i>	<i>symbols</i>	<i>longitude</i>	<i>latitude</i>	<i>x-y plane</i>	<i>long. zero</i>	<i>RH/LH</i>
horizon	–	azimuth	elevation	horizontal	north	L
equatorial	$\alpha, \delta$	R.A.	Dec.	equator	equinox	R
local equ.	$h, \delta$	H.A.	Dec.	equator	meridian	L
ecliptic	$\lambda, \beta$	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;<sup>2</sup> 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.

<sup>2</sup>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.

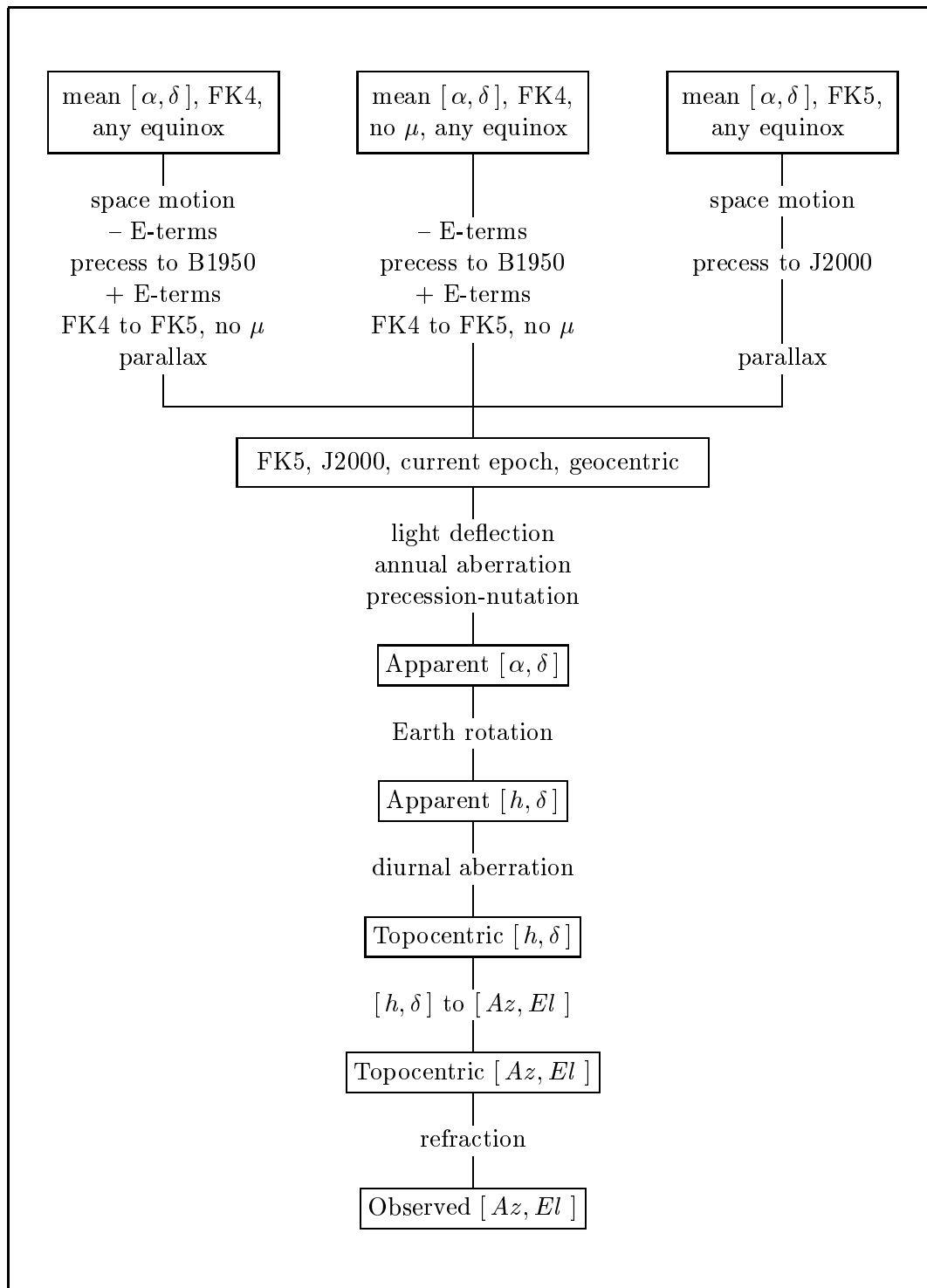


Figure 1: Relationship Between Celestial Coordinates

Star positions are published or catalogued using one of the mean  $[\alpha, \delta]$  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  $[\alpha, \delta]$  to the actual line-of-sight to the star. Note that for the conventional choices of equinox, namely B1950 or J2000, all of the precession and E-terms corrections are superfluous.

#### 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  $\alpha$  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*.<sup>3</sup>

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  $11''$  and a period of about 18.6 years. This difference of scale makes it convenient to treat these two components of the motion separately. 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

---

<sup>3</sup>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".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  $\psi$  (longitude) and  $\epsilon$  (obliquity), some as small as 0".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  $(3 \times 3)$  *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  $0.2^\circ$ ).

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.<sup>4</sup> 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)  $12^{\text{h}} 07^{\text{m}} 58^{\text{s}}.09 - 19^\circ 44' 37''.1$  “with respect to the mean equator and equinox of epoch J2000”. The short way of saying this is “[ $\alpha, \delta$ ] equinox J2000” (**not** “[ $\alpha, \delta$ ] epoch J2000”, which means something different to do with proper motion).

## 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  $280^\circ$ ; this happens near the start of the calendar year (which is why  $280^\circ$  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.

---

<sup>4</sup>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  $1''/\text{year}$ , and most are far below this level. A star observed as part of normal astronomy research will, as a rule, have a proper motion which is unknown.

Mean  $[\alpha, \delta]$  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  $\alpha$  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  $\alpha$  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 0.3.

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.<sup>5</sup> 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<sup>ʹ</sup>.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:

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

---

<sup>5</sup>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  $\alpha = 16^{\text{h}} 09^{\text{m}} 55^{\text{s}}.13$ ,  $\delta = -75^{\circ} 59' 27''.2$ , equinox 1900, epoch 1963.087,  $\mu_{\alpha} = -0^{\text{s}}.0312/y$ ,  $\mu_{\delta} = +0''.103/y$ , parallax =  $0''.062$ , 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 RO,DO,EQO,EPO,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)
DO=-D0
EQO=1900D0
EPO=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  $1''.74$ ; it falls off rapidly with distance from the Sun and has shrunk to about  $0''.02$  at an elongation of  $20^\circ$ .

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  $20''.5$ .

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*360D0),
:           AS2R=PI/(180D0*360D0)
DOUBLE PRECISION RM,DM,PR,PD,DATE,RA,DA
INTEGER J,IDS,IDE,ID,IY MDF(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,IY MDF,J)
    CALL sla_MAP(RM,DM,PR,PD,ODO,ODO,2000D0,DATE,RA,DA)
    WRITE (*,'(1X,I4,2I3.2,F9.5)') (IY MDF(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*,  $\theta$ :

$$h = \theta - \alpha$$

For this equation to work,  $\alpha$  must be the apparent right ascension for the time of observation, and  $\theta$  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  $0''.3$ , 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  $0''.2$ , 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  $0''.5$ . The amount of refraction can be 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,  $75^\circ$  the following model can be used instead:

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

where  $\zeta_{vac}$  is the topocentric zenith distance (i.e. *in vacuo*),  $\zeta_{obs}$  is the observed zenith distance (i.e. affected by refraction), and  $A$  and  $B$  are constants, about  $60''$  and  $-0''.06$  respectively for a sea-level site. The two constants can be calculated for a given set of conditions by calling either sla\_REFRO or sla\_REFROQ.

sla\_REFRO 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\_REFROQ uses a direct formulation of  $A$  and  $B$  and is much faster; it is slightly less accurate than sla\_REFRO 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  $\zeta_{vac}$  and  $\zeta_{obs}$  and reversing the signs, though approximately correct, gives avoidable errors which are just significant in some applications; for example about  $0''.2$  at  $70^\circ$  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  $\zeta = 83^\circ$  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\_REFRO and sla\_REFROQ 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\_REFRO several times it is more efficient to call sla\_REFRO 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  $100\mu\text{m}$  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  $76^\circ$ . 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  $\pm 0^{\circ}9$  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<sup>h</sup> 59<sup>m</sup> 60<sup>s</sup>:0 and 1994 July 1 00<sup>h</sup> 00<sup>m</sup> 00<sup>s</sup>: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  $\alpha$ . 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 to clocks at sea-level, and for practical purposes it is tied to Atomic Time TAI through the formula  $TT = TAI + 32^s.184$ . In practice, therefore, the units of TT are ordinary SI seconds, and the offset of  $32^s.184$  with respect to TAI is fixed. The SLALIB 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 between 1 and 2 milliseconds 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  $0^s.0017$ ; the largest lunar and 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 (TCG and TCB, not supported by SLALIB routines at present), which include allowance also for the secular terms. These timescales gain on TT and TDB by about  $0^s.0013/\text{day}$ .

For most purposes, the distinction between TT and TDB is of no practical importance. 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, the time argument is strictly TDB, but TT is near enough and will require 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  $\Delta T = ET - UT$  is available via the routine `sla_DT`. For a given epoch (*e.g.* 1650) this returns an approximation to  $\Delta T$  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; for example Julian Day Number 2449444 began at noon on 1994 April 1. *Julian Date* is the same system but with a fractional part appended; Julian Date 2449443.5 was the midnight on which 1994 April 1 commenced. Because of the unwieldy size of Julian Dates and the awkwardness of the half-day offset, it is accepted practice to remove the leading ‘24’ and the trailing ‘.5’, producing what is called the *Modified Julian Date*:  $MJD = JD - 2400000.5$ . SLALIB 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 by evaluating `ATAN2` of the two numbers. A full 3-D vector description of the position and velocity of the observer is available through the 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  $\pm 100$  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  $1^\circ$ . 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,DJDO,FD,DJD,DJDM,DF,PV(6),RMATP(3,3),
:          PVM(6),PVE(6),TL,RV,DV,RJ,DJ,SEP
      INTEGER IHOURL,IMIN,J,I,IHMIN,IMMIN
      DOUBLE PRECISION sla_EPJ,sla_DSEP

* Search for closest approach on the given day
      DJDO=1720859.5D0
      SEPMIN=1D10
      DO IHOURL=20,22
        DO IMIN=0,59
          CALL sla_DTF2D(IHOURL,IMIN,ODO,FD,J)

*          Julian date and MJD
          DJD=DJDO+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_DMV(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=IHOOR
          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, p357).

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<sup>h</sup> 25<sup>m</sup> 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. Depending on accuracy requirements, this unperturbed orbit is an adequate approximation to the actual orbit for a few weeks either side of the specified epoch, outside which perturbations due to the other bodies of the Solar System lead to increasing errors. Given a minor planet's osculating elements for a particular date, predictions for a date only 100 days earlier or later are likely to be in error by several arcseconds. These errors can be reduced if new elements are generated which take account of the perturbations of the major planets, and this is what the 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
$t_0$	$t_0$	$T$
$i$	$i$	$i$
$\Omega$	$\Omega$	$\Omega$
$\varpi$	$\omega$	$\omega$
$a$	$a$	$q$
$e$	$e$	$e$
$L$	$M$	
$n$		

The symbols have the following meanings:

$t_0$	epoch of osculation
$T$	epoch of perihelion passage
$i$	inclination of the orbit
$\Omega$	longitude of the ascending node
$\varpi$	longitude of perihelion ( $\varpi = \Omega + \omega$ )
$\omega$	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 calculated from  $n^2 a^3 = k^2(1 + m)$ , where  $k = 0.01720209895$  and  $m$  is the mass of the planet ( $M_\odot = 1$ );  $a$  is in AU.

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

- The epoch of observation: the moment in time for which the position of the body is to be predicted.
- The epoch defining the position of the body: the moment in time at which, in the absence of perturbations, the specified position—mean longitude, mean anomaly, or perihelion—is reached.
- The epoch of osculation: the moment in time at which the given elements precisely specify the body’s position and velocity.

For the major-planet and minor-planet cases it is usual to make the epoch that defines the position of the body the same as the epoch of osculation. Thus, for planets (major and minor) only two different epochs are involved: the epoch of the elements and the epoch of observation. For comets, the epoch of perihelion fixes the position in the orbit and in general a different epoch of osculation will be chosen. Thus, for comets all three types of epoch are involved. How many of the three elements are present in a given SLALIB argument list depends on the routine concerned.

Two important sources for orbital elements are the *Horizons* service, operated by the Jet Propulsion Laboratory, Pasadena, and the Minor Planet Center, operated by the Center for Astrophysics, Harvard. The JPL elements (heliocentric, J2000 ecliptic and equinox) and MPC elements correspond to SLALIB arguments as shown in the following table, where “(rad)” means conversion from degrees to radians, and “(MJD)” means “subtract 2400000.5D0”:

SLALIB argument	JPL			MPC	
	major planet	minor planet	comet	minor planet	comet
JFORM	1	2	3	2	3
EPOCH	JDCT (MJD)	JDCT (MJD)	Tp (MJD)	Epoch (MJD)	T (MJD)
ORBINC	IN (rad)	IN (rad)	IN (rad)	Incl. (rad)	Incl. (rad)
ANODE	OM (rad)	OM (rad)	OM (rad)	Node (rad)	Node. (rad)
PERIH	OM+W (rad)	W (rad)	W (rad)	Perih. (rad)	Perih. (rad)
AORQ	A	A	QR	a	q
E	EC	EC	EC	e	e
AORL	MA+OM+W (rad)	MA (rad)		M (rad)	
DM	N (rad)				
epoch of osculation	JDCT (MJD)	JDCT (MJD)	JDCT (MJD)	Epoch (MJD)	Epoch (MJD)

Conventional elements are not the only way of specifying an orbit. The  $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$  state vector is an equally valid specification, and the so-called *method of universal variables* allows orbital calculations to be made directly, bypassing angular quantities and avoiding Kepler’s Equation. The universal-variables approach has various advantages, including better handling of near-parabolic cases and greater efficiency. SLALIB uses universal variables for its internal calculations and also offers a number of 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. Starting with universal elements, the routine sla\_PLANTU (the universal elements equivalent of sla\_PLANTE) predicts topocentric  $[\alpha, \delta]$ .

#### 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,2000DO,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  $\sim 20$  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  $\alpha$ ). 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 ( $\theta$  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:

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

where the  $p$  subscript indicates “predicted” coordinates (the model’s approximation to the ideal “expected” coordinates) and the  $m$  subscript indicates “measured coordinates”. The six coefficients  $a$ – $f$  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_DCMPIF`.

#### 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:

$$y = C_1 + C_2x + C_3\sin x + C_4\cos x$$

The test values for the four coefficients are  $C_1 = +50.0$ ,  $C_2 = -2.0$ ,  $C_3 = -10.0$  and  $C_4 = +25.0$ . Gaussian noise,  $\sigma = 5.0$ , 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 common-place normal-equations method had been used, and the large  $1000 \times 20$  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  $J$ th column of the  $V$ -matrix identify the pair of solution coefficients that are dependent.

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

The 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 work 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\_SEPV (V1, V2)  
 D = sla\_DSEPV (V1, V2)  
 Angle between two  $[x, y, z]$  vectors

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\_PLANTU (DATE, ELONG, PHI, U, RA, DEC, R, JSTAT)  
 Topocentric apparent place of a Solar-System object whose heliocentric universal 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 ( $\equiv$  'Gaussian residual')

## Real-time

CALL sla\_WAIT (DELAY)  
Interval wait