SOL: functions for pointing a solar telescope

Patrick Wallace

2025 August 18

Contents

1 Introduction

This document[1] describes the SOL library, a set of ANSI C functions for use in the pointing control system of a solar telescope. The functions implement various specialized coordinate transformations that ordinary nighttime astronomical telescopes do not require. The document describes the functions themselves and suggests ways they might be used in a telescope control system (TCS). Its main sections are as follows:

- Section 2 describes pointing a solar telescope in general terms, and the role of SOL in extending a telescope control system for solar observing.
- Section 3 reviews the supported coordinate systems, which are the dominating influence on the observer's perception of the TCS's pointing controls for solar observing.
- Section 4 discusses methods for generating the required solar ephemerides, both orbital and physical.
- Section 5 describes the SOL functions in detail and how to use them.

1.1 Quick start

Compile and run the supplied `demo.c` program, which is self-contained apart from needing to be linked with the IAU SOFA library (`http://www.iausofa.org`). It writes a report to `stdout`, setting out the test case that is being demonstrated, the corresponding circumstances, and examples of transformations that can be performed by calling the SOL functions. A TCS developer can readily adapt and expand the program to cover other test cases.

Apart from `demo.c`, SOL consists of two header files and 19 ANSI C functions, ready to take their place alongside the application's own source code. There are no SOL-specific "build" facilities, not even a `make` file.

[1] See also Wallace (2002) and Wallace (2016).

2 The Pointing Control System for a Solar Telescope

Expressed at the most fundamental level, the job of a solar telescope's control system is to manage the alignment between the image of a solar target and the observing instrument. This has two complementary aspects, which involve the same chain of transformations and differ only in the direction in which the chain is traversed:

1. Given the coordinates of a solar surface or coronal feature (the "target") we want to control the moving parts of the mount and optics so that the image of the feature appears in the right place in the focal plane (pointing) and stays there (tracking).

2. Given the demanded settings of the mount and optics, we want to predict the solar coordinates imaged at a given place in the focal plane. This can provide displayed coordinates, and for 2D images a "world coordinate system".

The ends of the transformation chain are (i) the heliographic longitude and latitude (L,B)[2] and (ii) the demanded encoder coordinates that the telescope control system sends to the mount and instrument rotator. Along the chain are a number of intermediate coordinate systems, some of which have uses in their own right.

Again at the fundamental level, the software designer's task is to decide how the telescope control system should implement the above transformations, in order to command and interrogate the telescope hardware appropriately.

For some solar telescopes, for example ones that use a heliostat rather than a conventional equatorial or altazimuth mount, the control system will be so specialized that designing the TCS from scratch can be justified. But in the cases that are the focus of the present document it will be more efficient to regard the solar telescope as an ordinary astronomical telescope that happens to come with enhancements for observing solar features. It then becomes natural to use an existing TCS as the starting point, and to treat the purely solar capabilities as an extension. The SOL library is designed to help construct that extension.

[2] The symbols L and B mean the heliographic longitude and latitude respectively. It should be noted that not only are other symbols in use (λ and Φ for longitude, Ψ and Θ for latitude, in various pairings) but solar observers generally use the opposite order: latitude then longitude. Unfortunately, the latter practice is at odds with the normal mathematical convention, namely longitude first. Here, for consistency with positional-astronomy software such as IAU SOFA, we adopt the longitude then latitude order; the designer of the TCS's user interface may take a different view. Beware!

2.1 TCS architecture

Modern TCSs contain a component designed to shield the higher level controls (and the observer) from a multitude of internal complexities concerning coordinate systems and mount control. We will assume that such a *pointing kernel* exists and that it provides certain interfaces and capabilities to be described in due course.

The pointing kernel hides the complexities and inaccuracies of the actual machinery and simulates a telescope that is nominally perfect and has convenient, rigorously defined and safe control interfaces. An example of convenience is that it should not be necessary for the observing program to know whether the mount is equatorial or altazimuth: simply asserting the science target's celestial coordinates, where in the focal plane the image is to appear, and at what orientation, will cause an appropriate stream of demands to be sent to the moving parts. A more subtle example is where the observing program wishes to nod left and right of the current target to acquire sky samples. On an altazimuth telescope this must not be commanded by injecting offsets into the stream of azimuth demands; that such an effect would occur is merely a *consequence* of the fortuitous alignment between the mount axes and the demanded maneuver. On an equatorial telescope the resulting demand streams would be more elaborate but would achieve the same observational results.

In order to achieve the highest possible pointing and tracking accuracy, the pointing kernel will include a *pointing model*, a set of mathematical expressions that relates the direction of the incoming light (the so-called "observed place") with the encoder demands that need to be sent to the mount servos in order to achieve target acquisition. To create and maintain the pointing model involves carrying out pointing tests that gather samples of observed place and encoder demands (or the equivalent) all over the patrol area of the telescope. Of course, for science purposes a solar telescope needs to observe only one object and over a limited band of declinations. However, to calibrate the pointing it is much better to sample a larger area of sky, using stars as reference points. The main benefit is that the fitted pointing model is then dominated by interpolation, with no tendency for the residuals to worsen as the edges of the observing patrol area are approached, a particular danger if using polynomials as the correction terms. Another benefit is the ability to fit, with confidence, terms that although small on their own may in combination with others reach significant levels.

2.2 Pointing at solar targets

We will assume that the interface to the pointing kernel is *topocentric apparent place*, continuously recalculated in order to generate the requisite non-sidereal tracking rates. The starting point for this calculation will usually be *heliographic* coordinates (L,B), *e.g.* for photospheric targets, but

sometimes will be (*x,y,z*) *heliocentric* coordinates, suitable for coronal features, or *helioprojective* coordinates (*x,y*), referring to places on the solar disk. The pointing calculation can afford to be quite rigorous, taking properly into account planetary aberration (*i.e.* light time) and diurnal parallax. Light deflection (*i.e.* the Sun's gravitational lens effect) can be neglected, as for solar features it is always below 4 mas, though it could be up to 1.75″ for a coronal feature distant from the Sun and seen against the limb.[3]

Working in apparent (α,δ) has the disadvantage that the coordinates of the main object of interest, namely the Sun (or a solar feature), are continuously changing. The motion in latitude could be reduced were the kernel to support ecliptic coordinates, and the longitude drift could be reduced by working in terms of the mean Sun. However, they would never quite go away—diurnal, monthly, synodic and annual terms would always be present—making such strategies less advantageous than they might seem at first sight. Moreover, the rates in (α,δ) change so smoothly that keeping the solar coordinates and their rates of change up to date to the required accuracy is straightforward.

2.3 Pointing-kernel capabilities

It might be thought that all the pointing kernel has to do is use standard positional-astronomy transformations to compute mount encoder demands that correspond to the solar feature to be studied. However, there is more to it.

- As well as pointing and tracking the mount in order to follow the sky coordinates of a specified celestial target, a more complete pointing kernel design will go further by requiring the image position in the focal plane to be specified as well. Given this explicit control over image (*x,y*), the control system can deliver, as a matter of course, rapid transfer of images from acquisition device to instrument, fast dithering, precise blind positioning on slits and fibers and so on. In an unsophisticated TCS, such image shifts would be brought about by making *ad hoc* changes to the pointing, either by perturbing the target (α,δ) or (L,B) coordinates or by introducing spurious pointing-model offsets. A cascade of further adjustments will then be required, for example to let an autoguider know what is going on or to allow for differential refraction. In contrast, the modern pointing kernel that the present document assumes is available possesses all of these capabilities simply as part of the way the system works: the observing program merely declares that it wishes the image to be at a different (*x,y*), and everything necessary just happens automatically.

[3] As with other ephemeris minutiae, consistency with other solar observers will often be a more important consideration than rigor.

- The target image should remain centered on an off-axis instrument "hot-spot" even when the rotator is turned.

- The telescope's various moving optical elements should be treated not as free-standing devices but as components of an integrated pointing system. Limb-guiding while roaming or scanning should happen naturally, and things like bandwidth splitting between the slow mount and a fast tip/tilt mirror should be easy to implement.

- The pointing kernel should have inbuilt handling of field orientation at all foci: Nasmyth and coudé as well as Cassegrain *etc*. The fact that a given experiment is being done at, say, a coudé focus will then not (apart from any cable-wrap issues) be something the observer needs to be particularly aware of.

- For limb guiding, the pointing kernel should as a matter of course take care of differential refraction and atmospheric dispersion.

- The pointing kernel should provide for the accurate logging of target positions, together with support for World Coordinate System mapping.

- Control of instrument rotator angle should be handled rigorously, ensuring accurate results even near the zenith and in the presence of pointing corrections and differential refraction. In particular the calculation must **not** be based upon the parallactic angle: instead, projecting fiducial points in the focal plane back onto the sky will determine the current field orientation and allow steering corrections to be introduced. Furthermore, the field orientation can be reckoned at the hot-spot (which is in general not at the rotator axis) and in solar coordinates, enabling precise alignment between solar and instrument coordinates.

- The basic pointing transformations that the pointing kernel uses should be rigorous and glitch-free even near the zenith and for any large mechanical misalignments.

- When developing the pointing kernel software, it is wise to keep computational efficiency in mind. If hundreds or even thousands of highly accurate astrometric transformations per second are available, the TCS designer can count on using sledgehammer tactics, leaving the sophistication to the kernel.

- The design should allow for continuous development of the pointing model, rather than simply embedding in the kernel whatever model successfully passed the final commissioning tests. The best plan is for the offline pointing analysis procedures and the real-time control system to use the identical pointing-model software, retaining the full repertoire of terms. One advantage is that revising the pointing model (*i.e.* the list of correction terms) can then take place without

even relinking, let alone introducing code changes. Another advantage is to provide a *tautological link* between the analysis and operational phases, completely avoiding the sign errors and subtle distortions that would otherwise degrade performance without it being suspected that there might be a software cause.

- During operation, the TCS need not be limited to a single pointing model. This supports the use of multiple foci while allowing for geometrical misalignments and flexures peculiar to different components of the telescope, such as acquisition devices and instruments.

2.4 Architectural considerations

Inside the TCS, a simple way to manage the interface to the pointing kernel is to maintain a *data context*, reacting to events by introducing appropriate changes to it. Such a scheme is essentially *modeless*, producing different effects as a consequence of the passing of time and the changing data context, **not** because set-piece choreographed maneuvers are being carried out. The importance of modeless operation is that observing patterns can be introduced that were never thought of before, with the full rigor of the system in play.

The design has to handle the following, among other things:

- Access to TAI date and time must be provided. Things like the 19s GPS offset and (especially) UTC leap seconds, not to mention local time, must be handled outside the kernel.

- Among many other tasks, the user interface must provide ways for the operator to change the kernel's context, in particular by announcing a new target.

- Depending on the adopted pointing kernel, the TCS may have the option of refreshing different parts of the context at different rates. For example there might be a call to the kernel's "slow update" function once every minute and a call to the "medium update" function once every 5s.

- The system must also call a "fast update" routine at, say, 20 Hz, to generate the tracking demands. For kernels that time-stamp the demands, the timing should not be especially time-critical. However, a fixed and reliable update rate, with no samples omitted or in the wrong order, will make for easier fault diagnosis and generally smoother results. Either way, the software should be designed so that any hiccups are temporary and that faults heal themselves.

- In addition to satisfying the pointing kernel's housekeeping needs, the TCS's slow, medium and fast updates will perform the solar

coordinate transformations necessary to supply the updated topocentric apparent places and differential rates that drive the kernel proper. A cascaded series of ephemeris calculations (for example, the heliocentric Earth position and velocity might be refreshed once a minute, the solar rotation perhaps every five seconds) will ensure that the CPU load remains small.

Once running, the pointing kernel will generate a stream of time-stamped mount and rotator encoder demands for tracking the specified target, taking into account both the complex and changing astrometric transformations and the pointing model. The precise interface to the mount control system (MCS) will dictate how this information will be used. Ideally the MCS will accept the time-stamped positions without restriction, and even during the abrupt change of demand when a slew begins will automatically apply the required signal shaping to catch up smoothly. However, MCS designers usually expect velocities as well as positions; this can be accommodated by using the kernel to make two position predictions and taking differences. In the inevitable horse-trading between the TCS and MCS designers, the point that must **never** be conceded is that the MCS must be prepared to accept essentially random numbers from the TCS, and not expect precalculated trajectories. A telescope is not like a CNC milling machine!

2.5 Differential rotation

Solar targets in general will have intrinsic motions, and surface features will share in the latitude-dependent differential rotation. Both of these effects can be accommodated by permitting solar targets, in whatever coordinate system, to have optional timestamps and differential rates.

Thus the user interface could supply a target as heliographic coordinates (L,B) at time t_0 and differential rates ($\partial L/\partial t$, $\partial B/\partial t$). The rates could come from a nominated standard model (in which case the latitude rate $\partial B/\partial t$ would always be zero), or obtained numerically by differencing positions recorded some time apart. A day or two later, at time t, the same target could be recalled, and the pointing kernel would as a matter of course set the telescope to the coordinates ($L + \Delta t\ \partial L/\partial t$, $B + \Delta t\ \partial B/\partial t$), where $\Delta t = t - t_0$. The telescope would then point to the target, with its differential motion having been taken into account.

2.6 Zenith blind spot (altazimuth only)

All altazimuth telescopes are limited in their ability to observe the zenith region. On a nighttime telescope, this is usually not an operational problem, because as one science target goes through the “blind spot”, a different one can be observed instead. However, a solar telescope has only one science target, namely the Sun, and at tropical sites there will be days during the summer when for a few minutes the Sun cannot be observed as it passes

through the blind spot. It is thus essential to understand what causes the blind spot and to deal with it efficiently.

Figure 1 shows the four natural limits to tracking in the zenith region:

- **Azimuth tracking speed.** The inaccessible region lies inside a figure-of-eight. The "waist" corresponds to the case of tracking exactly through the zenith, where it is possible to track right up to or away from the zenith, but there is a sudden (and time-consuming) 180° azimuth change between the two.

- **Azimuth slew speed.** In the case where the track does not go exactly through the zenith, a point is reached where, although neither the velocity nor the acceleration has reached the maximum allowed, unless a slew is begun immediately the target cannot be picked up again until after it has passed the point of symmetry on the other side of the meridian. This region is lenticular in shape, because it is the overlap of two regions, respectively north and south of the zenith.

- **Azimuth acceleration.** The inaccessible region in this case is a four-lobed shape. As a rule, the available velocity and acceleration performance means that the four-lobed shape lies almost entirely inside the figure-of-eight set by the maximum tracking speed. In other words, in practice the available acceleration is usually not the limiting factor.

- **Geometrical inaccessibility.** Mechanical non-perpendicularity between either (i) the azimuth and altitude axes of the mount or (ii) the telescope and the altitude axis will cause a circular hole in the sky coverage around the zenith: no combination of azimuth and altitude encoder readings will point the telescope within this region. For a typical solar telescope this impossible area is unlikely to be more than a few tens of arcseconds in radius and hence will lie entirely within the other blind spots.

The Figure shows the blind spots for the Daniel K. Inouye Solar Telescope, sited in Hawaii. For illustrative purposes it has been assumed that the full azimuth slew speed of 3°/s is available for the required 180° change, whilst for the maximum rate at which accurate tracking is available a lower figure of 0.75°/s has been used. The maximum azimuth acceleration available during tracking has been assumed to be 0.1°/s/s. The large circle is a nominal 0.5° radius blind spot, suitable for planning purposes

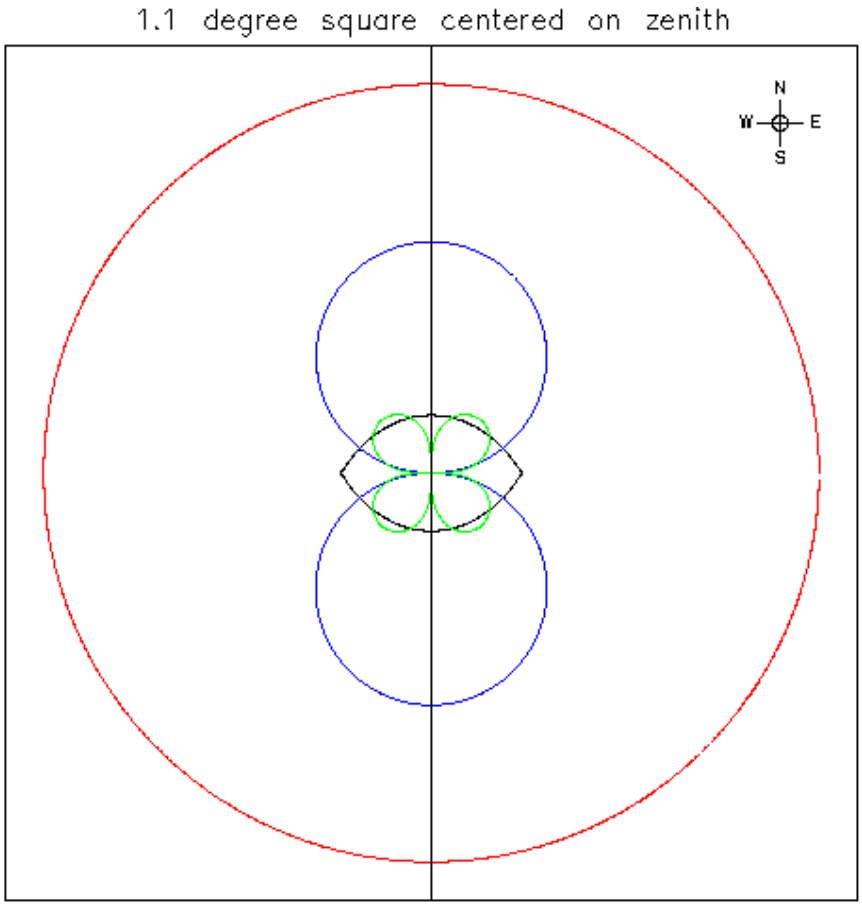

Figure 1: DKIST zenith blind spot. The figure-of-eight region is set by the maximum available tracking speed. The four-lobed region is set by the acceleration limit. The lenticular region indicates where slewing would begin and end for a blind spot symmetrically disposed about the meridian. The outer circle is for a nominal $r = 0.5°$.

In practice, it is not possible to give a hard-and-fast rule that dictates the optimum way to pass through the region. The symmetry of the lenticular-shaped slew-limited region proves that it must be the "minimum lost time" blind spot. However, one way or another the telescope and enclosure have to complete a 180° azimuth change, and this alone will dominate how much observing time will be lost and when. Moreover, any seconds shaved off the lost time by abandoning tracking the moment the lenticular region is reached may well be at the cost of prematurely terminating an observation. A pragmatic solution is for the TCS to warn when the speed limit will be reached and to leave it up to the user (or the automated observing sequence) to elect when to break off and get the 180° slew over with.

Some existing pointing kernels deal with zenith avoidance by executing a circle track around the zenith at a nominated distance. For a particular solar telescope something a little more elaborate may be needed. If so, this can either be implemented in the kernel itself or, probably easier, in the sequencer that is running the observations. Once the final pre-transit

observation has completed, overriding the demanded (α,δ) with one on the edge of the blind spot would be one option.

However, there are potential difficulties that extend well outside the area of the blind spot itself. As the zenith is approached, the pointing corrections that are being applied in order to track the source will call for large azimuth adjustments that will in turn cause complementary field-rotation effects. This will not in itself be a problem if the adopted pointing kernel is one that uses rigorous techniques both to track in (az,alt) and to control the rotator. But even then to accomplish this without continuous guiding requires very precise knowledge of the pointing coefficients and, moreover, the correct allocation of the total pointing corrections into the various geometrical contributions: zero points, non-perpendicularities and azimuth-axis tilts. Figure 2 demonstrates the effect. The test circumstances are as follows:

- The site is DKIST.
- We are about to make a 1-minute track touching the southern edge of the 0.5° diameter zenith blind spot at its mid-point.
- The operational pointing model includes a value for the northwards tilt of the azimuth axis that differs by 5" from the (unknown) true value. On the meridian, the mid-point of our simulated track, there is consequently a pointing error of 5".
- We correct the mid-exposure pointing by introducing an *ad hoc* 5" adjustment to the altitude encoder zero-point.
- The simulation records the changing positions of eight points equally spaced around a 5′ field of view during the 60-second exposure.

The resulting trails are over 2" long. Although 5" uncertainties in individual pointing coefficients, as used in the simulation, are rather larger than would be expected for a modern solar telescope, such errors are by no means unknown. There are two conclusions to be drawn:

1. Despite a solar telescope being a “one-target telescope”, thorough and frequent all-sky night-time pointing analyses will nonetheless be needed in order to characterize the sources of error. (See also Section 2.1.)
2. For fields near the zenith (even several degrees away), the adaptive optics system and/or limb guider will play an important role in maintaining accurate tracking.

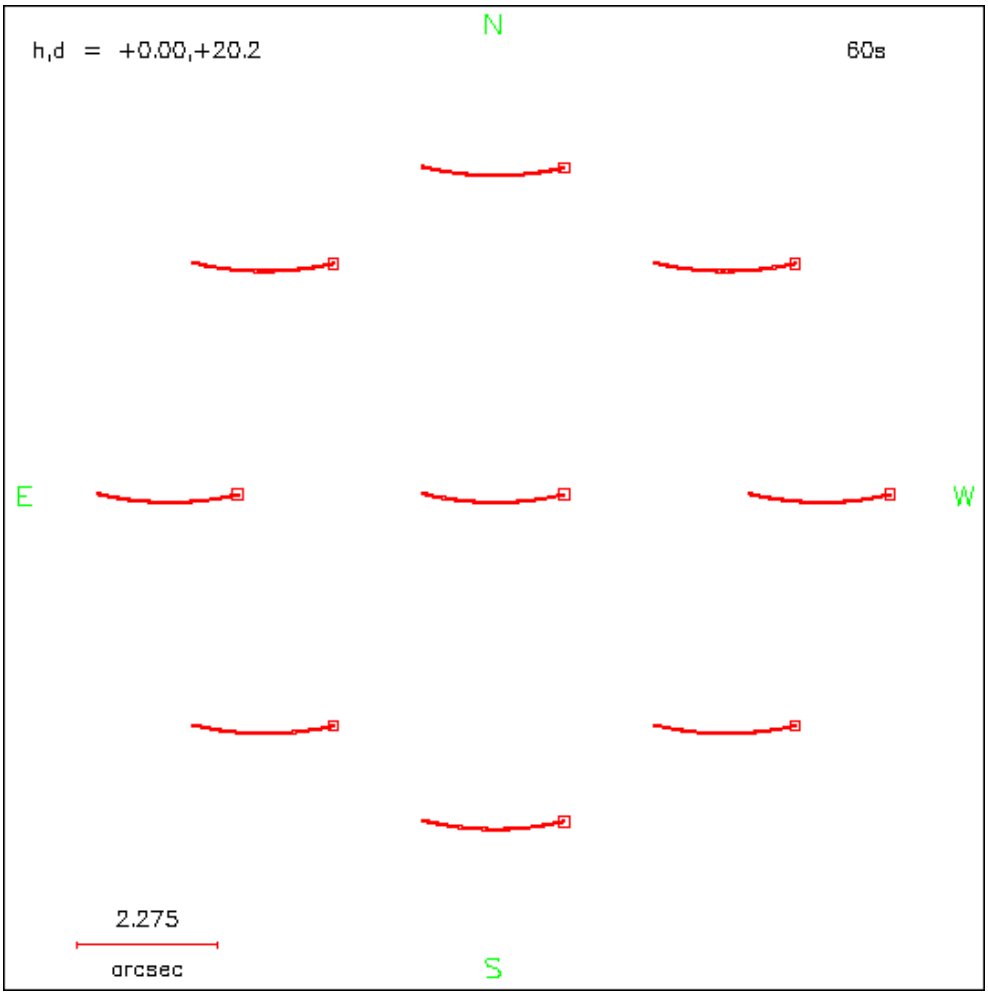

Figure 2: Tracking errors in 1 minute for a field on the southern edge of DKIST's nominal $r = 0.5°$ zenith blind spot, in the case where the pointing effect of a 5″ northwards tilt of the azimuth axis has been compensated by an *ad hoc* 5″ adjustment to the altitude zero-point.

Errors in the telescope/altitude and altitude/azimuth non-perpendicularities produce trails of similar size but at right-angles to those in Figure 2. Fortunately, the two types of non-perpendicularity cause similar effects near the zenith, where they have the largest effects on tracking; hence it is not necessary to determine the proportions of the two types of non-perpendicularity, as long as the total amount is accurately known.

3 Coordinate Systems, Solar and Celestial

Solar coordinate systems are discussed from a space physics point of view by Hapgood (1992) and by Fränz & Harper (2002). Thompson (2002) addresses the solar image aspects. There are also many web pages concerning sunspot observing and solar space observatories that contain useful material. In these references, there is a limited amount of sharing of identifiers (HEEQ, HCI *etc.*), but on the whole the nomenclature is somewhat inconsistent and precise definitions are elusive. There is of course a danger that the SOL library could itself add to this unsatisfactory state of affairs! To guard against this, the following approach has been taken here:

- The total number of SOL coordinate systems has been kept to a bare minimum. Only four systems are proposed, compared with several times that number in the cited references.
- Duplication is avoided by not separately naming the Cartesian and polar representations of each system.
- Similarly, units in most cases are not stipulated, and applications are free to choose (for example) solar radii or meters. The natural home for much of this flexibility is the user interface rather than the TCS itself.
- Where standard transformations exist, in common with other branches of astronomy, these are taken for granted and not provided in special solar forms.

These measures mean that the list of supported coordinate systems omits most of those discussed in the references. Apart from the desire for economy, there are two reasons for this. The first is that the overlap between the needs of (i) solar and solar-terrestrial space observatories and (ii) ground-based solar observatories is limited, so that an exhaustive list might confuse solar observers without adding useful capabilities. The second is that many of the omitted coordinate systems are based on the ecliptic.

Use of the ecliptic is problematical in modern high-accuracy work. Not only does the ecliptic slowly rotate relative to the stars (the so-called “planetary precession”), but at a sub-arcsecond level the concept has no rigorous basis. At least two distinct types of ecliptic are in use: inertial and geometric. Recent IAU resolutions on reference frames (IAU 2000, B1.1-9) in effect bypassed the ecliptic and equinox, and since then the trend has been towards use of the International Celestial Reference System alone.

The supported coordinate systems are described in the following sections. Each system available for target specification has a 2-character identifier. Others have incidental purposes.

3.1 Heliographic (HG)

Heliographic coordinates (L,B) are based on the solar equator, with a prime meridian that rotates relative to the Earth and stars. In a solid body this meridian would be defined by surface features, but in the case of the Sun it is provided by a conventional ephemeris. The heliographic longitude and latitude of a sunspot is approximately constant, though its apparent position on the Sun's disk changes from day to day.

HG coordinates should be the primary method for specifying the solar surface feature that is to be observed. Conversely, when the position of an object is to be recorded, HG coordinates would be the usual choice. If the radial distance *r* is unspecified, the default value is that for the photosphere. Outside the TCS, the user interface may support other options, for example height above the photosphere, or wavelength.

3.2 Stonyhurst

Stonyhurst coordinates should not, as a rule, be needed to specify targets. They differ from heliographic coordinates in that longitudes are reckoned from the center of the Sun's disk. A given solar feature thus has a Stonyhurst longitude that changes at about 13° per day, corresponding to the synodic period of 27.2753 days.

3.3 Helioprojective (HP)

Helioprojective coordinates specify the position of a feature seen in a solar image that has been projected onto a plane. Only one projection is supported here, namely gnomonic (*i.e.* tangent-plane), with the origin at the center of the Sun's disk. The default units are the projected radius of the solar disk.

Helioprojective coordinates can be used when displaying or storing solar image data.

3.4 Heliocentric (HC)

Heliocentric coordinates are Sun-centered but are tied neither to the Sun's equator nor to the solar rotation. They could be used when specifying a target in the corona. A good choice of units would be meters, though for some applications au might be more convenient (and is what the SOL library uses – see Section 5).

The only directly supported orientation is that of the International Celestial Reference System. For most solar observers this can be regarded as the same as the J2000.0 mean equator and equinox (within 25 mas) and FK5 (within 0.1″). Other orientations, such as the true equator and equinox of

date, and various ecliptic coordinates, are available through calls to functions in the SOFA library.

3.5 Topocentric apparent place (AP)

Topocentric apparent right ascension and declination (α,δ) is assumed to be the interface to the telescope control system. The pole is the precessing-nutating celestial intermediate pole (CIP); the zero point of right ascension is either the true equinox of date (the classical choice) or the celestial intermediate origin (CIO) for a TCS supporting the newer method (in which case the coordinates strictly should be called "topocentric celestial intermediate place").

Notes:

- This is not a spatial coordinate system, but one specifically used to describe a pointing direction; hence there is no radial coordinate.

- Topocentric apparent place differs from geocentric apparent place by including the effects of diurnal parallax and diurnal planetary aberration. It is thus not suitable for record-keeping, as it is peculiar to the observing site of the specific solar telescope.

The coordinates are unaffected by refraction - the direction corresponds to the case where the telescope is *in vacuo*; the TCS takes care of the effect as a normal part of controlling the telescope.

3.6 Geocentric apparent place

Geocentric apparent right ascension and declination (α,δ) should as a matter of course be available to assist in recording observations but would not be the link between solar coordinates and the telescope control system. As for the topocentric variety, the pole is the CIP and the zero point of right ascension is the equinox or the CIO.

Notes:

- This is not a spatial coordinate system, but one specifically used to describe a pointing direction; hence there is no radial coordinate.

- Geocentric apparent place differs from topocentric apparent place by being free from diurnal parallax and diurnal planetary aberration. Thus it is a convenient starting point from which a different solar observatory, *i.e.* somewhere else on the Earth, could begin the pointing calculation.

4 Solar Ephemerides

There are two ephemeris aspects to consider: the heliocentric Earth ephemeris, which provides the geometrical direction to the Sun and hence the solar telescope's pointing target, and the solar physical ephemeris, which allows points on the solar disk to be assigned heliographic coordinates.

4.1 Heliocentric Earth ephemeris

The telescope's pointing predictions begin with the heliocentric (x,y,z) coordinates of the Earth and their derivatives. There are two ways these can be obtained:

- By interrogating a JPL numerical ephemeris such as DE440.
- By using a mathematical model.

The first method has the advantage that the utmost accuracy can be achieved, moreover without a particularly heavy computing load. On the other hand using the interpolation software that comes with the JPL ephemeris is itself a support commitment, and the (binary) ephemeris file must be present and in the correct version.

For these reasons, the proposal here is to use the mathematical model approach, in the form of a simplified version of the VSOP2000 series (Moisson & Bretagnon 2001), as implemented as function iauEpv00 in the SOFA library. This 2500-line C function delivers pointing predictions that agree with JPL DE406 to within 15 mas throughout the 21st century. On a vintage 2 GHz Pentium processor each prediction consumes a non-trivial 4 ms of CPU time. However, the Earth position and velocity will be among the information recomputed at a low rate and then interpolated: the consequent deterioration in pointing accuracy even for hourly updates is well under 1 mas.

4.2 Solar physical ephemeris

Here we can adopt the model used by the *Astronomical Almanac* (see Seidelmann 1992):

- The solar north pole is fixed at $\alpha_{2000} = 286.1300°$, $\delta_{2000} = +63.8700°$.
- The rotation angle $W = 84.10° + 14.1844000° \times$(days since J2000.0 TDB).
- The radius of the photosphere is 696 000 000 meters.

Note that:

1. The original Carrington expressions are not quite rigorous. They do not describe a pole that is fixed in space and moreover are consistent with neither a fixed nor a moving ecliptic. Carrington wrote:

$$I = 7°15′$$
$$\Omega = 73°40′ + 50″.25\, t$$

   but should in fact have written:

$$I = 7°15′ + 0″.08\, t$$
$$\Omega = 73°40′ + 46″.60\, t$$

   to take account of planetary precession and thereby fix his solar rotation axis with respect to the stars.

2. Various subtly different interpretations are in use, the 1992 *Explanatory Supplement* formulation proposed here being just one of them.

3. The 1992 ES rotation period (360/14.1844 days) and Carrington's value (25.38 days) differ by about 0.4 seconds, causing a heliographic longitude drift of 0.1° per century.

4. The precise Carrington epoch with respect to modern time scales is not available, a further obstacle to truly canonical use. The software will be arranged so that although use of the ES methods is most convenient it will also be possible to supply different basic ephemerides should this be needed.

5. B_0, L_0 and P (the latitude and longitude of the center of the disk and the position angle of the north pole) will not be calculated from the usual formulas but will instead use the transformation procedures.

6. Carrington rotation number will be a user interface issue and not provided by the TCS.

7. The distinction between TDB and TT can be neglected.

5 The SOL Library

5.1 The library

The SOL library is an ANSI C implementation of the algorithms described in Wallace (2016). It comprises two header files, **`sollib.h`** and **`solsys.h`**, fifteen callable functions and four functions used internally. The IAU SOFA fundamental-astronomy library is a prerequisite. The functions are as follows:

SET UP SOLAR EPHEMERIS PARAMETERS

- **`solSolp`** bespoke
- **`solSolpci`** CIO/ERA based
- **`solSolpeq`** equinox/GST based
- **`solSolpt`** update for new time

BASIC SOLAR EPHEMERIDES

- **`solSoleph`** basic ephemerides

SOLAR COORDINATE TRANSFORMATIONS

- **`solAp2hg`** apparent to heliographic
- **`solAp2hp`** apparent to helioprojective
- **`solHc2ap`** heliocentric to apparent
- **`solHc2hg`** heliocentric to heliographic
- **`solHc2hp`** heliocentric to helioprojective
- **`solHg2ap`** heliographic to apparent
- **`solHg2hc`** heliographic to heliocentric
- **`solHg2hp`** heliographic to helioprojective
- **`solHp2ap`** helioprojective to apparent
- **`solHp2hg`** helioprojective to heliographic

INTERNAL

- **`sol_c2hc`** heliocentric target in given pointing direction
- **`sol_c2hp`** pointing direction to helioprojective *x,y*
- **`sol_hc2c`** pointing direction for given heliocentric target
- **`sol_hp2c`** helioprojective *x,y* to pointing direction

Notes:

1. The principle behind the SOL library is that there is a set of parameters that can be computed once and then used repeatedly to carry out multiple solar coordinate transformations economically. The scheme offers further opportunities for efficiency by allowing the parameters to be updated periodically rather than being recalculated from scratch every time.

2. The ten coordinate transformation functions are all based upon Cartesian 3-vectors. The application can accomplish conversions between between spherical coordinates (*e.g.* α,δ) and Cartesian coordinates (*x,y,z*) by using inline code and/or calls to SOFA functions.

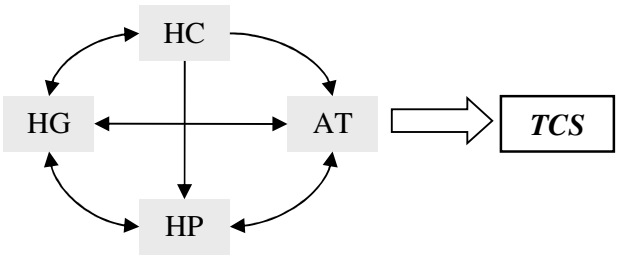

Figure 3: The four coordinate types available to the solar telescope control application, and the supported transformations between them. The interface to the core TCS is always via topocentric apparent place, but this is not something the user needs to be aware of.

3. With four coordinate systems available to applications, there is a maximum of 12 transformations. However, two have been omitted (see Figure 3, above) as unlikely to be useful: topocentric apparent place to heliocentric, and helioprojective to heliocentric. Should either of these be required, the application can simply perform the required transformation in two steps.

4. The four internal functions deal with the pointing direction to the target with respect to ICRS axes. (These functions are not part of the API and their prototypes are not found in the **`sollib.h`** header file that applications include. The **`solsys.h`** header file that the SOL functions themselves use contains the additional prototypes, as well as the **`sollib.h`** header file.)

5.2 Strategies for using the SOL library in a TCS application

The SOL library design presents the TCS implementor with a choice of architectures, offering different trade-offs between simplicity and efficiency. Here are three examples:

1. A no-CPU-expense-spared TCS could simply call **`solSolpeq`** (or one of the other functions that sets up the parameter set) every time a transformation was required – at say 20 Hz for a typical TCS, or even at servo rates.

2. A somewhat less prodigal TCS could call **`solSolpeq`** every minute (say) and then perform an update by calling **`solSolpt`** at 20 Hz, followed by transformations as required.

3. An economical and pragmatic approach is (i) to call **`solSolpeq`** occasionally, (ii) to perform **`solSolpt`** updates more frequently and (iii) to rely on the TCS's normal differential tracking facilities to deal with the Sun's non-sidereal motion.

Some existing pointing kernels (the proprietary TCSpk is an example) already have a pointing context that is refreshed at several different rates, with slowly-changing items such as the precession-nutation matrix recomputed every minute or so (in itself overkill), and more rapidly-changing things, such as the pointing corrections, updated every few seconds. This context then supports fast and numerous pointing transformations, repeated perhaps 20 times per second and controlling guiders and choppers as well as the telescope mount itself. It is therefore natural to adopt solution (3) from the above list. In detail:

- Call **`solSolpeq`** at the TCS *slow* rate (say once a minute) to refresh the heliocentric Earth ephemeris and precession-nutation matrix components of the solar ephemeris parameter set. (Even every hour would be ample, or just once per new target, but the additional savings would be negligible.)

- Call **`solSolpt`** at the TCS *medium* rate (say every 5s), followed by a call to **`solSoleph`**. This will produce a topocentric apparent α,δ for the center of the Sun and—by using the value from last time—differential track rates $d\alpha/dt$, $d\delta/dt$.

- At the TCS *fast* rate, perhaps 20Hz, call the appropriate transformation functions, to get α,δ for the solar target (heliographic for photospheric features, heliocentric for coronal features, helioprojective for points on the solar disk). Such an α,δ is for the time of the *medium* update. By retaining the differential rates (and their corresponding reference times), the target will be correctly tracked.

With this approach, the only significant compromise that is being made is that the solar rotation since the last *medium* update time is being ignored. Because it takes about two weeks for a solar feature to rotate from one limb of the Sun to the other, the maximum error in 5s is about 0.02 arcsec.

It should be noted that the transformation parameters include, among other things, rotation matrices, and hence the transformation functions are computationally efficient. This means that maneuvers such as offsets and scans can confidently be carried out in heliographic or helioprojective coordinates in just the same way that they apply to α,δ.

5.3 Transformation paths

The coordinate transformations are set out as shown in Figure 4, below. The shaded boxes are the supported coordinate systems. The unshaded box is the "celestial pointing direction" coordinate system used internally.

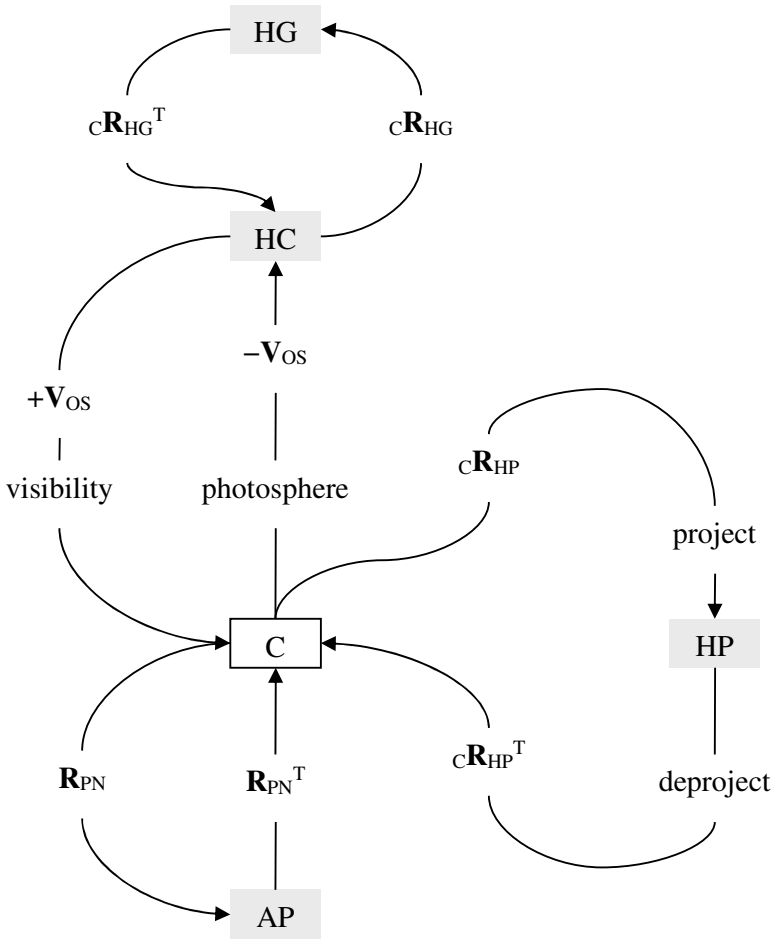

Figure 4: Details of the Figure 3 transformation paths. The various algorithmic components are described in Section 5.5.

Notes:

1. The **R** symbols represent coordinate rotations, with the subscripts representing the before and after coordinates. A superscript ^{T} means transpose, indicating that the rotation is backwards. In the solar ephemeris parameters data structure, **`Sunp`**, the matrix $_C\mathbf{R}_{HG}$ is the member **`rc2hg`**, the matrix $_C\mathbf{R}_{HP}$ is the member **`rc2hp`**, and the matrix $\mathbf{R}_{PN}$ is the member **`rpn`**.

2. The $\pm\mathbf{V}_{OS}$ symbols represent addition or subtraction of the observer-to-Sun vector, which in the solar ephemeris parameters data structure, **`Sunp,`** is the member **`vos.`**

3. The "visible?" items means testing whether the target point is occulted by the Sun (see 5.5.6).

4. The "photosphere" item means solving for the intersection of the line of sight with the photosphere (see 5.5.7).

5. The "project" and "deproject" items refer to the relationship between the line of sight and the projection plane (see 5.5.8 and 5.5.9).

5.4 The data context

The SOL library coordinate transformations use a set of precomputed parameters that include scalars, 3-vectors and 3×3 matrices. These reside in a data structure of type **`Sunp`** that is defined in the **`sollib.h`** header file. The SOL library source code uses the name **`solpars`** for instances of such structures.

The data context is built from scratch whenever the application calls the function **`solSolp`**, or more typically one of two front-end functions, **`solSolpci`** and **`solSolpeq`**. Subsequently, calls to the function **`solSolpt`** update the context to reflect the rotation of the Sun and Earth and the orbital motion of the Earth.

An application can, if required, set up and maintain more than one such data context at once, for example to allow an observation planned for later in the day to be previewed.

The following table lists the data members that comprise the **`Sunp`** structure, together with the symbolic names that will be used when explaining the component algorithms (Section 5.5).

member	*meaning*	*symbol*
`tau`	light time for 1 au (day)	τ
`rau`	radius of photosphere (au)	ρ
`w0`	J2000.0 position of prime meridian (radian)	w_0
`wdot`	rotation rate (radian/day)	$\partial w/\partial t$
`srnp`	functions of ICRS α,δ of solar north pole…	sin α
`crnp`	...	cos α
`cdnp`	...	cos δ
`srsdnp`	...	sin α sin δ
`crsdnp`	...	cos α sin δ
`ttref`	reference time (TT MJD)	t_0
`rpn`	bias-precession-nutation matrix	$\mathbf{R}_{PN}$

ehp	Earth position (BCRS, au) at time $t_0-\tau$	$\mathbf{P}_E(t_0)$
ehv	Earth velocity (BCRS, au/day)	$\mathbf{V}_E$
tt	current time (TT MJD)	t
vobs	geocentric vector to observer (au)	$\mathbf{V}_{OBS}$
vos	observer to Sun vector (ICRS, au)	$\mathbf{V}_{OS}$
d	apparent distance (au)	d
dl2	distance squared, observer to limb	d_L^2
tsd	tangent of semi-diameter	$\tan\sigma$
rc2hg	rotation matrix, ICRS to heliographic	${}_C\mathbf{R}_{HG}$
rc2hp	rotation matrix, ICRS to helioprojective	${}_C\mathbf{R}_{HP}$

The shaded portion identifies the items that are updated by calls to the function **solSolpt**. Changing the other items, in particular the Earth ephemeris and the precession-nutation matrix, requires a fresh call to the **solSolp** function.

5.5 Component algorithms

This section deals with (i) how the data context is set up and maintained, (ii) how the basic solar ephemerides are obtained, and (iii) how the processing steps in Figure 4 are carried out. Not discussed in detail are the SOFA functions used.

5.5.1 Initializing the data context

The data context is initialized by the function **solSolp**, which accepts as arguments various quantities that are either constant or that do not need to be refreshed frequently:

- the light time for 1 au, τ,
- the ICRS α,δ of the Sun's north pole (SNP),
- the coefficients w_0 and dw/dt for predicting the position of the heliographic meridian,
- the radius of the photosphere ρ,
- the reference time t_0 (a Modified Julian Date in the TT time scale),
- the heliocentric position and velocity of the Earth at time $t_0-\tau$,
- the precession-nutation matrix (see 5.5.10, below),
- the observatory coordinates,
- the Earth rotation measure, which is either the Earth rotation angle ERA or the Greenwich apparent sidereal time) depending on whether the precession-nutation matrix is CIO-based or equinox-based.

The "reference" time t_0 is probably close to the time when **solSolp** is called, but might be slightly in the future to minimize extrapolation errors during the lifetime of the set of parameters (though an unnecessary refinement). Note that the Earth orbital ephemeris data are for 1 au of

light time prior to the reference time, corresponding approximately to when the light currently seen left the Sun.

Most of these are stored in the **solpars** data structure as they are. In order to avoid wasteful recomputation later on, various trigonometrical functions of the SNP's coordinates, and the terrestrial coordinates of the observer, are stored also. The vector (ICRS, au) to the SNP, which we will refer to as $\mathbf{V}_{SNP}$, is in fact stored as the z-axis of the ${}_{C}\mathbf{R}_{HG}$ matrix: see 5.5.4, below.

The functions **solSolpci** and **solSolpeq** are convenient front ends to **solSolp** that make specific choices for the various items. In both cases, the constant items are defined in the header file **solsys.h** and the heliocentric Earth position and velocity vectors are obtained using the SOFA function **iauEpv00** (which is accurate to about 10 km and 5 mm/s). The two functions make different choices for the precession-nutation matrix and Earth rotation measure, allowing the application to select whether the interface to the TCS will be CIO-based celestial intermediate place or equinox-based topocentric apparent place.

5.5.2 Updating the data context

The **solSolp** function described in the previous section ends with a call to the **solSolpt** function for the reference time t_0. The **solSolpt** function, the topic of the present section, is called by the application in order to bring up to date the various transformation items shown in Figure 4. A typical application will call **solSolpci** or **solSolpeq** say once a minute (which is perhaps two orders of magnitude more often than necessary but costs very little) and **solSolpt** once every few seconds to update the transformation quantities that depend on solar rotation and Earth orbital motion, including the differential tracking rates for the Sun's center.

The updating procedure begins with extrapolating the Earth rotation measure and orbital position from their original epochs (t_0 and $t_0 - \tau$ respectively) to the time t specified in the **solSolpt** call. The Earth rotation measure is extrapolated by adding $(t - t_0) \times 6.3003878$ radians, where the numerical factor is the Earth rotation rate in radians per UT day; the value is a compromise between the ERA and GST values, the former being pure Earth rotation and the latter containing a small contribution from precession. This updated angle then allows the geocentric coordinates of the observer, with respect to celestial axes, to be determined. This vector (with precession-nutation allowed for) is combined with the heliocentric Earth position vector to give a Sun-to-observer vector, and from this the light time can be estimated and hence the required ephemeris lookup time. The difference between this and the actual lookup time (namely $t_0 - \tau$) provides the extrapolation interval; the product of this interval with the Earth velocity vector $\mathbf{V}_E$ is the correction to be added to the Earth position vector $\mathbf{P}_E(t_0)$ to form the revised Sun-to-Earth vector.

5.5.3 The observer-to-Sun vector $\mathbf{V}_{OS}$ *etc.*

The **`solSolpt`** function adds the updated Sun-to-Earth vector and the Earth-to-observer vector to give the observer-to-Sun vector $\mathbf{V}_{OS}$ used in Figure 4. The magnitude of this vector is the apparent distance d, the distance from the observer to the solar limb is $d_L = (d^2 - \rho^2)^{½}$, and the tangent of the semidiameter, $\tan \sigma = d_L / \rho$.

5.5.4 The "ICRS to heliographic" matrix ${}_C\mathbf{R}_{HG}$

The rotation matrix ${}_C\mathbf{R}_{HG}$ changes the axes of a heliocentric vector from ICRS to heliographic. It is formed by the function **`solSolpt`**. The heliographic triad has its z-axis pointing at the SNP and the x-axis pointing at the intersection between the Sun's equator and heliographic prime meridian, with the y-axis completing the right-handed system. The angle from the ascending node of the Sun's equator on the ICRS equator to the heliographic origin is:

- $w = w_0 + (t - d\tau - \text{J2000})\ \partial w/\partial t$;

and the x and y axes of ${}_C\mathbf{R}_{HG}$ are the following unit vectors:

- $\mathbf{V}_{HGx} = [\ -\sin w \cos\alpha \sin\delta - \cos w \sin\alpha\ ,$
  $\quad -\sin w \sin\alpha \sin\delta + \cos w \cos\alpha\ ,$
  $\quad +\sin w \cos\delta\ ]$
- $\mathbf{V}_{HGy} = [\ -\cos w \cos\alpha \sin\delta + \sin w \sin\alpha\ ,$
  $\quad -\cos w \sin\alpha \sin\delta - \sin w \cos\alpha\ ,$
  $\quad +\cos w \cos\delta\ ]$

where α,δ are the ICRS coordinates of the SNP. The z-axis $\mathbf{V}_{HGz}$ (which is the SNP) is fixed and has already been supplied by **`solSolp`**.

5.5.5 The "pointing direction to helioprojective" matrix ${}_C\mathbf{R}_{HP}$

The helioprojective coordinate system (see Figure 7, below) has a z-axis formed by the Sun-to-observer vector, with solar features projected on the xy plane, located somewhere behind the Sun. The y-axis is oriented so that the projection of the SNP falls on it, and the limb is a circle of unit radius centered on the origin. The **`solSolpt`** function forms the rotation matrix, called ${}_C\mathbf{R}_{HP}$ in Figure 4, that transforms the line of sight into a direction with respect to helioprojective axes, by treating each row as an individual unit vector:

- The helioprojective z-axis unit vector $\mathbf{V}_{HPz}$ is formed simply by reversing the sign of the observer-to-Sun vector $\mathbf{V}_{OS}$ and normalizing.
- The x-axis $\mathbf{V}_{HPx}$, which is the vector pointing west from the Sun's center that is normal to both the SNP and the observer-to-Sun

vector $\mathbf{V}_{OS}$, is formed by taking the vector product $\mathbf{V}_{HGz} \times \mathbf{V}_{HPz}$ and normalizing.

- The *y*-axis is normal to the other two and is formed by taking the vector product $\mathbf{V}_{HPz} \times \mathbf{V}_{HPx}$.

5.5.6 Visibility: check for target behind the Sun

Heliocentric and heliographic targets are located in 3-space rather than simply being pointing directions. Whilst in practice most targets will be points on the visible photosphere, this will not always be so. Apart from actual non-photospheric science targets, there is also the case where an arbitrary heliographic coordinate—perhaps a point on an (L,B) grid in a graphical display being plotted in helioprojective (*x,y*) —is being considered. It will often be important to know whether the target is obscured by the Sun: in the case of the graphical display, for example, this would decide whether the point was to be plotted or not. A visibility check is built into the internal function **`sol_hc2c`**, and passed back to the application through a status return by the functions **`solHc2ap`**, **`solHc2hp`**, **`solHg2ap`** and **`solHg2hp`**.

The visibility check comprises a sequence of two tests; only if the target fails both is it considered to be occulted by the Sun. The geometry is shown in Figure 5:

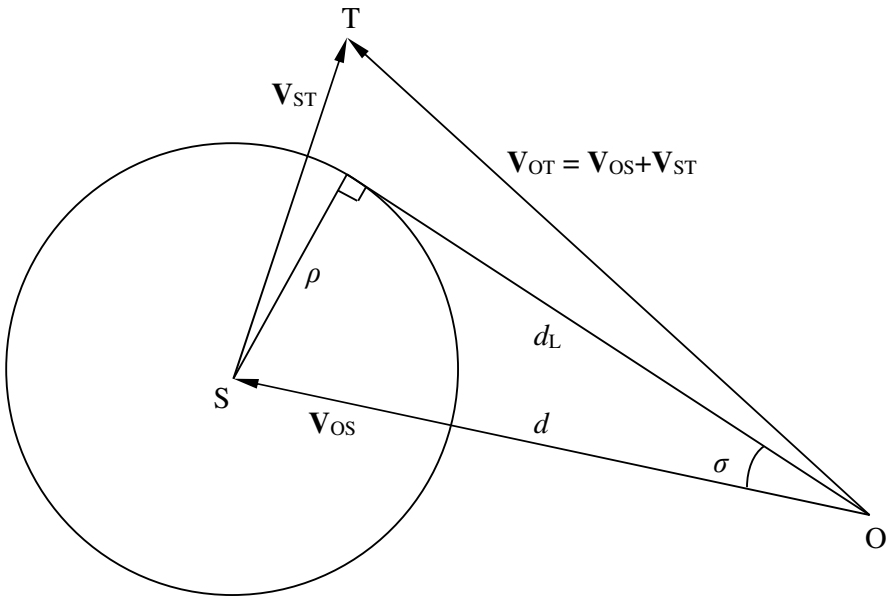

Figure 5: Visibility check. Points O, S and T are the observer, Sun center and target respectively. The circle is the photosphere.

The tests are (in order):

1. If the target is closer than the limb, *i.e.* $|\mathbf{V}_{OT}| < d_L$, it is visible. Note that in the unlikely case where the target lies beneath the photosphere the target would be deemed visible.
2. Test 1 having failed, if the target is outside the apparent disk of the Sun, it is visible, *i.e.* $\mathbf{U}_{OS} \cdot \mathbf{U}_{OT} < \cos \sigma$, where unit vectors $\mathbf{U}_{OS}$ and $\mathbf{U}_{OT}$ are the normalized versions of $\mathbf{V}_{OS}$ and $\mathbf{V}_{OT}$.

Test 2 is in fact carried out in a different way, for efficiency reasons. If $\mathbf{V}_{OS} \cdot \mathbf{V}_{OT} < 0$, the target is more than 90° from the Sun, admittedly not a practical case, and hence is visible. Otherwise, if $(\mathbf{V}_{OS} \cdot \mathbf{V}_{OT})^2 < |\mathbf{V}_{OT}|^2 \, d_L^2$, the target is outside the disk and is visible.

5.5.7 Photosphere: heliographic coordinates for pointing direction

In any of the transformation paths that involve expressing a pointing direction as a heliocentric target, the question arises of where along the line of sight to place the target. The scheme adopted by SOL (and implemented in the internal function **`sol_c2hc`**) is to use the intersection of the line of sight with the (near) photosphere. This is shown in Figure 6.

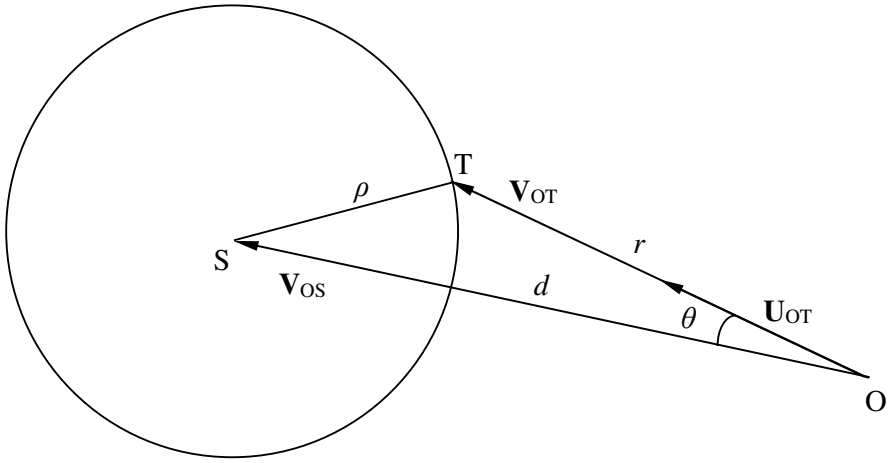

Figure 6: The photospheric target that corresponds to the pointing direction, given by the unit vector $\mathbf{U}_{OT}$. The nomenclature is the same as for **Figure 5**.

The line of sight is the unit vector $\mathbf{U}_{OT}$. The observer-to-target vector is $\mathbf{V}_{OT}$; r is the distance to the yet-to-be-chosen target, so that $\mathbf{V}_{OT} = r\mathbf{U}_{OT}$.

The cosine rule for plane triangles gives us:

$$\rho^2 = r^2 + d^2 - 2\, r\, d \cos\theta$$

Writing $p = d \cos\theta = \mathbf{V}_{OS} \cdot \mathbf{U}_{OT}$ and solving for r :

$$r = p - (\, p^2 - d^2 + \rho^2\,)^{½}$$

For pointing directions that lie outside the Sun's disk, the discriminant is negative. By substituting zero, the target is placed on an arc at distance $r = d_L$, which is a serviceable default.

5.5.8 Projection from pointing direction into helioprojective *x,y*

Figure 7 illustrates helioprojective coordinates. The common case of a target on the photosphere is shown, but the scheme relates to pointing direction only and therefore works for a target at any distance from the observer.

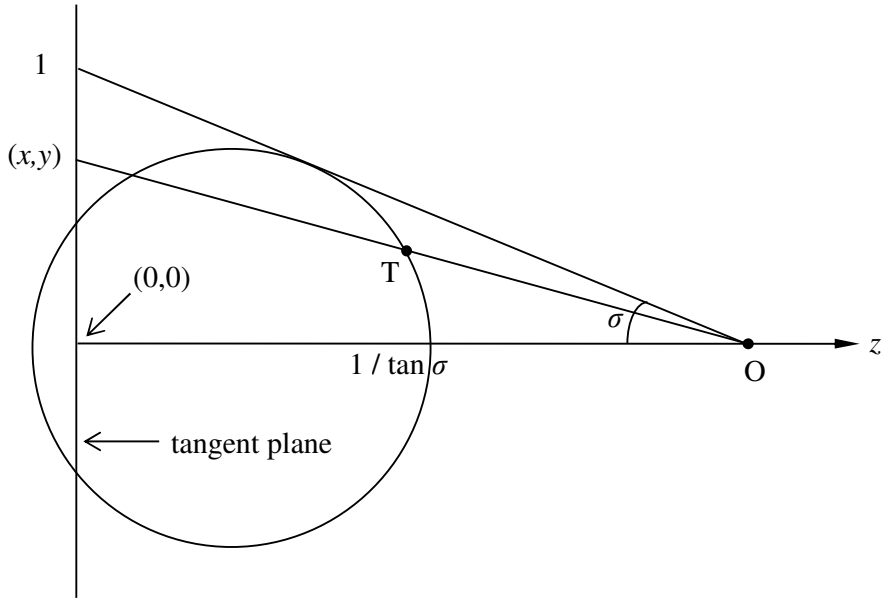

Figure 7: Helioprojective coordinates. The target T (which in this case happens to be on the photosphere) is seen projected onto the tangent plane, on which the limb appears as a circle of unit radius. The solar north pole lies on the helioprojective *y*-axis. The tangent plane is at an arbitrary distance from the observer.

Projection consists of two steps:

1. Refer the ICRS pointing direction $\mathbf{U}_{OT}$ to helioprojective axes by taking the matrix product $\mathbf{U}_{HP} = {}_{C}\mathbf{R}_{HP} \times \mathbf{U}_{OT}$; call the components of the resulting unit vector [$\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$].

2. The helioprojective coordinates are:

$$x = \boldsymbol{i} f$$
$$y = \boldsymbol{j} f$$

where the scale factor $f = k \tan \sigma$ corresponds to $r = 1$ for the limb.

5.5.9 Deprojection from helioprojective *x,y* into pointing direction

See Figure 7. As for projection (previous section), deprojection consists of two steps:

1. Supplement the helioprojective coordinates x,y with a matching z value:

   $$z = -1 / \tan \sigma$$

   Call the resulting vector $\mathbf{v}_{HP} = [\, x, y, z \,]$.

2. Refer the vector to ICRS axes by taking the matrix product $\mathbf{v}_{OT} = {}_{C}\mathbf{R}_{HP}{}^{T} \times \mathbf{v}_{HP}$.

It should be noted that the resulting pointing direction, $\mathbf{v}_{OT}$, is not a unit vector, and may need normalizing before it is used.

5.5.10 The "precession-nutation" matrix $\mathbf{R}_{PN}$

The 3×3 matrix $\mathbf{R}_{PN}$ expresses the rotation from ICRS axes into the α,δ axes of date, taking care of frame bias, precession and nutation. It is supplied through the call to the function **`solSolp`**, either directly for complete flexibility or, more conveniently, through one of the standard front-ends **`solSolpeq`** or **`solSolpci`**.

Interpreting its rows as three unit vectors:

- The bottom row of the matrix, $\mathbf{v}_{CIP}$, is the ICRS vector for the celestial intermediate pole (CIP) of date.
- The top row of the matrix, $\mathbf{v}_{\alpha 0}$, is the vector for the adopted zero point of right ascension, either the celestial intermediate origin (CIO) or the equinox.
- The middle row completes the right-handed triad and is the vector product $\mathbf{v}_{CIP} \times \mathbf{v}_{\alpha 0}$.

The choice between the CIO (introduced by IAU resolutions in 2000) and the equinox as the zero point of right ascension is up to the application. Having made this choice, the application must then supply the corresponding Earth rotation measure. For α_0 = CIO, the Earth rotation angle is required; for an application based on classical methods α_0 = equinox and sidereal time (the latter strictly apparent, but mean will suffice) is required.

5.6 Programmer's reference

Detailed specifications for each of the callable SOL functions can be found in the following pages. The functions are presented in alphabetical order.

solAp2hg	apparent to heliographic	solAp2hg

ACTION: Topocentric apparent place vector to heliographic vector.

GIVEN:

vap	double[3]	topocentric apparent place vector
solpars	Sunp*	solar ephemeris parameters

RETURNED:

vhg	double[3]	heliographic vector (au)

NOTES:

1. The topocentric apparent place vector **vap** is normally of unit length, but the results do not depend on this being the case.

2. The nomenclature "topocentric apparent place" strictly applies only to the equinox/GST case: in the CIO/ERA case, the correct term is "topocentric celestial intermediate place".

3. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

4. Heliographic coordinates are right-handed; their origin is the center of the Sun; the *z*-axis is through the solar north pole and the *x*-axis is through zero Carrington latitude and longitude; the units are au.

solAp2hp	apparent to helioprojective	solAp2hp

ACTION: Topocentric apparent place vector to helioprojective *x,y*.

GIVEN:

vap	double[3]	topocentric apparent place vector
solpars	Sunp*	solar ephemeris parameters

RETURNED:

hpx,hpy	double*	helioprojective coordinates

NOTES:

1. The topocentric apparent place vector **vap** is normally of unit length, but the results do not depend on this being the case.

2. The nomenclature "topocentric apparent place" strictly applies only to the equinox/GST case: in the CIO/ERA case, the correct term is "topocentric celestial intermediate place".

3. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

4. Helioprojective coordinates *x,y* lie in a plane normal to the line from the observer to the center of the Sun: the *projection plane*. For a given target, the helioprojective coordinates are where the line of sight intersects the projection plane. The *x,y* origin corresponds to the center of the Sun, the solar north pole is on the *y*-axis, and the edge of the photosphere is a circle of unit radius centered on the origin.

solHc2ap	heliocentric to apparent	solHc2ap

ACTION: Heliocentric vector to topocentric apparent place vector.

GIVEN:

vhc	double[3]	heliocentric vector (ICRS, au)
solpars	Sunp*	solar ephemeris parameters

RETURNED (arguments)**:**

vap	double[3]	topocentric apparent place vector

RETURNED (status)**:**

	int	true = target is hidden by the Sun

NOTES:

1. Heliocentric coordinates are right-handed; their origin is the center of the Sun; their axes are aligned to the ICRS; the units are au.

2. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

3. The apparent place vector **vap** that is returned has an arbitrary magnitude: only its direction is significant.

4. The nomenclature "topocentric apparent place" strictly applies only to the equinox/GST case: in the CIO/ERA case, the correct term is "topocentric celestial intermediate place".

5. The criteria for occultation by the Sun are as follows: (i) if the distance observer-to-target is less than the distance observer-to-limb, the target is not occulted; (ii) otherwise, if the target's apparent direction falls within the Sun's disk, the target is occulted. Test (i) means that sub-photospheric targets are deemed to be unocculted unless they are more distant from the observer than is the limb.

solHc2hg	heliocentric to heliographic	solHc2hg

ACTION: Heliocentric vector to heliographic vector.

GIVEN:

vhc	double[3]	heliocentric vector (ICRS, au)
solpars	Sunp*	solar ephemeris parameters

RETURNED:

vhg	double[3]	heliographic vector (au)

NOTES:

1. Heliocentric coordinates are right-handed; their origin is the center of the Sun; their axes are aligned to the ICRS; the units are au.

2. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

3. Heliographic coordinates are right-handed; their origin is the center of the Sun; the *z*-axis is through the solar north pole and the *x*-axis is through zero Carrington latitude and longitude; the units are au.

solHc2hp	heliocentric to helioprojective	solHc2hp

ACTION: Heliocentric vector to helioprojective *x,y*.

GIVEN:

vhc	`double[3]`	heliocentric vector (ICRS, au)
solpars	`Sunp*`	solar ephemeris parameters

RETURNED (arguments)**:**

hpx,hpy	`double*`	helioprojective coordinates

RETURNED (status)**:**

	`int`	true = target is hidden by the Sun

NOTES:

1. Heliocentric coordinates are right-handed; their origin is the center of the Sun; their axes are aligned to the ICRS; the units are au.

2. The solar ephemeris parameters **`solpars`** are created by a call to one of the functions **`solSolpci`**, **`solSolpeq`** or **`solSolp`** and perhaps subsequently updated by a call to the function **`solSolpt`**.

3. Helioprojective coordinates *x,y* lie in a plane normal to the line from the observer to the center of the Sun: the *projection plane*. For a given target, the helioprojective coordinates are where the line of sight intersects the projection plane. The *x,y* origin corresponds to the center of the Sun, the solar north pole is on the *y*-axis, and the edge of the photosphere is a circle of unit radius centered on the origin.

4. The criteria for occultation by the Sun are as follows: (i) if the distance observer-to-target is less than the distance observer-to-limb, the target is not occulted; (ii) otherwise, if the target's apparent direction falls within the Sun's disk, the target is occulted. Test (i) means that sub-photospheric targets are deemed to be unocculted unless they are more distant from the observer than is the limb.

solHg2ap	heliographic to apparent	solHg2ap

ACTION: Heliographic vector to topocentric apparent place vector.

GIVEN:

vhg	double[3]	heliographic vector (au)
solpars	Sunp*	solar ephemeris parameters

RETURNED (arguments)**:**

vap	double[3]	topocentric apparent place vector

RETURNED (status)**:**

	int	true = target is hidden by the Sun

NOTES:

1. Heliographic coordinates are right-handed; their origin is the center of the Sun; the *z*-axis is through the solar north pole and the *x*-axis is through zero Carrington latitude and longitude; the units are au.

2. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

3. The apparent place vector **vap** that is returned has an arbitrary magnitude: only its direction is significant.

4. The nomenclature "topocentric apparent place" strictly applies only to the equinox/GST case: in the CIO/ERA case, the correct term is "topocentric celestial intermediate place".

5. The criteria for occultation by the Sun are as follows: (i) if the distance observer-to-target is less than the distance observer-to-limb, the target is not occulted; (ii) otherwise, if the target's apparent direction falls within the Sun's disk, the target is occulted. Test (i) means that sub-photospheric targets are deemed to be unocculted unless they are more distant from the observer than is the limb.

sol**Hg2hc**	**heliographic to heliocentric**	sol**Hg2hc**

ACTION: Heliographic vector to heliocentric vector.

GIVEN:

vhg	double[3]	heliographic vector (au)
solpars	Sunp*	solar ephemeris parameters

RETURNED:

vhc	double[3]	heliocentric vector (ICRS, au)

NOTES:

1. Heliographic coordinates are right-handed; their origin is the center of the Sun; the *z*-axis is through the solar north pole and the *x*-axis is through zero Carrington latitude and longitude; the units are au.

2. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

3. Heliocentric coordinates are right-handed; their origin is the center of the Sun; their axes are aligned to the ICRS; the units are au.

solHg2hp	heliographic to helioprojective	solHg2hp

ACTION: Heliographic vector to helioprojective *x,y*.

GIVEN:

vhg	double[3]	heliographic vector (au)
solpars	Sunp*	solar ephemeris parameters

RETURNED (arguments)**:**

hpx,hpy	double*	helioprojective coordinates

RETURNED (status)**:**

	int	true = target is hidden by the Sun

NOTES:

1. Heliographic coordinates are right-handed; their origin is the center of the Sun; the *z*-axis is through the solar north pole and the *x*-axis is through zero Carrington latitude and longitude; the units are au.

2. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

3. Helioprojective coordinates *x,y* lie in a plane normal to the line from the observer to the center of the Sun: the *projection plane*. For a given target, the helioprojective coordinates are where the line of sight intersects the projection plane. The *x,y* origin corresponds to the center of the Sun, the solar north pole is on the *y*-axis, and the edge of the photosphere is a circle of unit radius centered on the origin.

4. The criteria for occultation by the Sun are as follows: (i) if the distance observer-to-target is less than the distance observer-to-limb, the target is not occulted; (ii) otherwise, if the target's apparent direction falls within the Sun's disk, the target is occulted. Test (i) means that sub-photospheric targets are deemed to be unocculted unless they are more distant from the observer than is the limb.

solHp2ap	helioprojective to apparent	solHp2ap

ACTION: Helioprojective *x,y* to topocentric apparent place vector.

GIVEN:

hpx,hpy	double	helioprojective coordinates
solpars	Sunp*	solar ephemeris parameters

RETURNED:

vap	double[3]	topocentric apparent place vector

NOTES:

1. Helioprojective coordinates *x,y* lie in a plane normal to the line from the observer to the center of the Sun: the *projection plane*. For a given target, the helioprojective coordinates are where the line of sight intersects the projection plane. The *x,y* origin corresponds to the center of the Sun, the solar north pole is on the *y*-axis, and the edge of the photosphere is a circle of unit radius centered on the origin.

2. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

3. The apparent place vector **vap** that is returned has an arbitrary magnitude: only its direction is significant.

4. The nomenclature "topocentric apparent place" strictly applies only to the equinox/GST case: in the CIO/ERA case, the correct term is "topocentric celestial intermediate place".

solHp2hg	helioprojective to heliographic	solHp2hg

ACTION: Helioprojective *x,y* to heliographic vector.

GIVEN:

hpx,hpy	double	helioprojective coordinates
solpars	Sunp*	solar ephemeris parameters

RETURNED:

vhg	double[3]	heliographic vector (au)

NOTES:

1. Helioprojective coordinates *x,y* lie in a plane normal to the line from the observer to the center of the Sun: the *projection plane*. For a given target, the helioprojective coordinates are where the line of sight intersects the projection plane. The *x,y* origin corresponds to the center of the Sun, the solar north pole is on the *y*-axis, and the edge of the photosphere is a circle of unit radius centered on the origin.

2. The solar ephemeris parameters **solpars** are created by a call to one of the functions **solSolpci**, **solSolpeq** or **solSolp** and perhaps subsequently updated by a call to the function **solSolpt**.

3. Heliographic coordinates are right-handed; their origin is the center of the Sun; the *z*-axis is through the solar north pole and the *x*-axis is through zero Carrington latitude and longitude; the units are au.

sol**Soleph**	**basic solar ephemeris**	sol**Soleph**

ACTION: Compute basic solar ephemeris values.

GIVEN:

`solpars`	`Sunp*`	solar ephemeris parameters

RETURNED:

`tt`	`double*`	current time (TT MJD)
`apdist`	`double*`	apparent distance to Sun's center (au)
`semid`	`double*`	semi-diameter (rad)
`rastr`	`double*`	ICRS right ascension of Sun's center (rad)
`dastr`	`double*`	ICRS declination of Sun's center (rad)
`rdate`	`double*`	current right ascension of Sun's center (rad)
`ddate`	`double*`	current declination of Sun's center (rad)
`hgl0`	`double*`	heliographic longitude of center of disk
`hgb0`	`double*`	heliographic latitude of center of disk
`pastr`	`double*`	ICRS position angle of SNP (range ±π)
`pdate`	`double*`	current position angle of SNP (range ±π)

NOTES:

1. The solar ephemeris parameters **`solpar`** are created by a call to one of the functions **`solSolpci`**, **`solSolpeq`** or **`solSolp`** and perhaps subsequently updated by a call to the function **`solSolpt`**.

2. No distinction is made here between the ICRS, BCRS and GCRS coordinate systems, all of which are referred to simply as "ICRS".

3. Coordinates described as "ICRS" are topocentric (*i.e.* diurnal aberration and parallax are built in) and are astrometric (but with no light deflection applied). To transform such a topocentric astrometric α,δ into current α,δ would require the application of stellar aberration and precession-nutation.

4. Coordinates described as "current" are topocentric *(i.e.* diurnal aberration and parallax are built in) and are either apparent places or celestial intermediate reference system places depending on the choice of precession-nutation matrix and Earth rotation measure made when the **`solpars`** data structure was set up.

solSolp	use specified solar ephemeris parameters	solSolp

ACTION: Precompute solar ephemeris parameters using specified canonical elements.

GIVEN:

tau	double	light time per au (d)
rnp,dnp	double	ICRS α,δ of Sun's north pole (rad)
w0,wdot	double	heliographic prime meridian coefficients
rau	double	radius of photosphere (au)
tt	double	TT (MJD)
ehp	double[3]	heliocentric Earth position at time tt−tau (au)
ehv	double[3]	heliocentric Earth velocity (au/d)
rpn	double[3][3]	precession-nutation matrix
elong	double	site longitude (rad, east +ve)
phi	double	site latitude (rad)
hm	double	site height above sea level (m)
erm	double	Earth rotation measure (ERA or GST)

RETURNED:

solpars	Sunp*	solar ephemeris parameters:

tau	double	light time for 1 au (d)
rau	double	radius of photosphere (au)
w0	double	J2000 position of prime meridian (rad)
wdot	double	rotation rate (rad/d)
srnp	double	functions of ICRS α,δ of SNP…
crnp	double	...
cdnp	double	...
srsdnp	double	...
crsdnp	double	...
ttref	double	reference time (TT MJD)
rpn	double[3][3]	bias-precession-nutation matrix
ehp	double[3]	Earth position (BCRS, au) at ttref−tau
ehv	double[3]	Earth velocity (BCRS, au/d) at ttref−tau
tt	double	current time (TT MJD)
vobs	double[3]	geocentric vector to observer (au)
vos	double[3]	observer to Sun vector (ICRS, au)
d	double	apparent distance (au)
dl2	double	distance squared, observer to limb
tsd	double	tangent of semi-diameter
rc2hg	double[3][3]	rotation matrix, ICRS to heliographic
rc2hp	double[3][3]	rotation matrix, ICRS to helioprojective

NOTES:

1. No distinction is made here between the ICRS, BCRS and GCRS coordinate systems, all of which are referred to simply as "ICRS".

2. The various transformation functions (**`solAp2hg`** *etc.*) use these parameters to carry out rapid conversions between different forms of solar target coordinates.

3. The parameters become stale at different rates, according to which parameter, how the resulting predictions are used, and the accuracy requirements of the application concerned. It is probably simplest to call the present function say once a minute (though once an hour would be ample), to call the function **`solSolpt`** every few seconds to update the parameters to a new time, and when using celestial coordinates to track the Sun's center and work relative to that.

4. The heliographic prime meridian coefficients **`w0`** and **`wdot`** are the value at J2000 and rate of change of the angle that locates the heliographic prime meridian, in radians and radians/day. The angle is measured along the solar equator from the ascending node with the J2000 equator.

5. The reference time **`tt`** is that corresponding to the Earth rotation measure, **`erm`**.

6. The heliocentric Earth position and velocity vectors **`ehp`** and **`ehv`** are for the instant **`tt-tau`**, approximately when the light left the Sun.

7. The precession-nutation matrix **`rpn`** transforms from ICRS to coordinates of date. It can either be equinox based, in which case the Earth rotation measure must be Greenwich sidereal time, or CIO based, in which case the Earth rotation measure must be Earth rotation angle.

8. The height above sea level, **`hm`**, is included for completeness but has a negligible effect on the results.

sol**Solpci**	**CIO-based ephemeris parameters**	sol**Solpci**

ACTION: Precompute solar ephemeris parameters (CIO based) using default canonical elements.

GIVEN:

tt	double	TT (MJD)
ut	double	UT1 (MJD)
elong	double	site longitude (rad, east +ve)
phi	double	site latitude (rad)
hm	double	site height above sea level (m)

RETURNED:

solpars Sunp* solar ephemeris parameters:

tau	double	light time for 1 au (d)
rau	double	Radius of photosphere (au)
w0	double	J2000 position of prime meridian (rad)
wdot	double	rotation rate (rad/d)
srnp	double	functions of ICRS α,δ of SNP…
crnp	double	...
cdnp	double	...
srsdnp	double	...
crsdnp	double	...
ttref	double	reference time (TT MJD)
rpn	double[3][3]	CIO based bias-precession-nutation matrix
ehp	double[3]	Earth position (BCRS, au) at ttref−tau
ehv	double[3]	Earth velocity (BCRS, au/d) at ttref−tau
tt	double	current time (TT MJD)
vobs	double[3]	geocentric vector to observer (au)
vos	double[3]	observer to Sun vector (ICRS, au)
d	double	apparent distance (au)
dl2	double	distance squared, observer to limb
tsd	double	tangent of semi-diameter
rc2hg	double[3][3]	rotation matrix, ICRS to heliographic
rc2hp	double[3][3]	rotation matrix, ICRS to helioprojective

NOTES:

1. No distinction is made here between the ICRS, BCRS and GCRS coordinate systems, all of which are referred to simply as "ICRS".

2. The various transformation functions (**`solAp2hg`** *etc.*) use these parameters to carry out rapid conversions between different forms of solar target coordinates.

3. The parameters become stale at different rates, according to which parameter, how the resulting predictions are used, and the accuracy requirements of the application concerned. It is probably simplest to

call the present function say once a minute (though once an hour would be ample), to call the function **solSolpt** every few seconds to update the parameters to a new time, and when using celestial coordinates to track the Sun's center and work relative to that.

4. The present function is for the case where the application works with Earth rotation angle and CIO-based CIRS right ascension. To work instead with sidereal time and apparent right ascension use the function **solSolpeq**.

5. For complete control call instead the function **solSolp**.

6. The height above sea level, **hm**, is included for completeness but has a negligible effect on the results.

sol**Solpeq**	**equinox-based ephemeris parameters**	sol**Solpeq**

ACTION: Precompute solar ephemeris parameters (equinox based) using default canonical elements.

GIVEN:

tt	double	TT (MJD)
ut	double	UT1 (MJD)
elong	double	site longitude (rad, east +ve)
phi	double	site latitude (rad)
hm	double	site height above sea level (m)

RETURNED:

solpars Sunp* solar ephemeris parameters:

tau	double	light time for 1 au (d)
rau	double	radius of photosphere (au)
w0	double	J2000 position of prime meridian (rad)
wdot	double	rotation rate (rad/d)
srnp	double	functions of ICRS α,δ of SNP…
crnp	double	...
cdnp	double	...
srsdnp	double	...
crsdnp	double	...
ttref	double	reference time (TT MJD)
rpn	double[3][3]	equinox based bias-precession-nutation matrix
ehp	double[3]	Earth position (BCRS, au) at ttref−tau
ehv	double[3]	Earth velocity (BCRS, au/d) at ttref−tau
tt	double	current time (TT MJD)
vobs	double[3]	geocentric vector to observer (au)
vos	double[3]	observer to Sun vector (ICRS, au)
d	double	apparent distance (au)
dl2	double	distance squared, observer to limb
tsd	double	tangent of semi-diameter
rc2hg	double[3][3]	rotation matrix, ICRS to heliographic
rc2hp	double[3][3]	rotation matrix, ICRS to helioprojective

NOTES:

1. No distinction is made here between the ICRS, BCRS and GCRS coordinate systems, all of which are referred to simply as "ICRS".

2. The various transformation functions (**`solAp2hg`** *etc.*) use these parameters to carry out rapid conversions between different forms of solar target coordinates.

3. The parameters become stale at different rates, according to which parameter, how the resulting predictions are used, and the accuracy requirements of the application concerned. It is probably simplest to call

the present function say once a minute (though once an hour would be ample), to call the function **solSolpt** every few seconds to update the parameters to a new time, and when using celestial coordinates to track the Sun's center and work relative to that.

4. The present function is for the case where the application works with sidereal time and equinox-based apparent right ascension. To work instead with Earth rotation angle and CIRS right ascension use the function **solSolpci**.

5. In computing the sidereal time, the equation of the equinoxes is neglected. The effect on predictions of the solar targets is negligible.

6. For complete control, call instead the function **solSolp**.

7. The height above sea level, **hm**, is included for completeness but has a negligible effect on the results.

solSolpt	update solar ephemeris parameters	solSolpt

ACTION: Update solar ephemeris parameters.

GIVEN:

tt	double	current time (TT MJD)

GIVEN AND RETURNED:

solpars	Sunp*	solar ephemeris parameters:

tau	double	light time for 1 au (d)
rau	double	radius of photosphere (au)
w0	double	J2000 position of prime meridian (rad)
wdot	double	rotation rate (rad/d)
srnp	double	functions of ICRS α,δ of SNP…
crnp	double	...
cdnp	double	...
srsdnp	double	...
crsdnp	double	...
ttref	double	reference time (TT MJD)
rpn	double[3][3]	bias-precession-nutation matrix
ehp	double[3]	Earth position (BCRS, au) at ttref−tau
ehv	double[3]	Earth velocity (BCRS, au/d) at ttref−tau
tt	double	current time (TT MJD)
vobs	double[3]	geocentric vector to observer (au)
vos	double[3]	observer to Sun vector (ICRS, au)
d	double	apparent distance (au)
dl2	double	distance squared, observer to limb
tsd	double	tangent of semi-diameter
rc2hg	double[3][3]	rotation matrix, ICRS to heliographic
rc2hp	double[3][3]	rotation matrix, ICRS to helioprojective

updated

NOTES:

1. No distinction is made here between the ICRS, BCRS and GCRS coordinate systems, all of which are referred to simply as "ICRS".

2. Neither the precession-nutation matrix **rpn** nor the Earth vectors **ehp** and **ehv** are updated by the present function. These can be refreshed by calling one of the **solSolp..** functions occasionally: once a minute is more than adequate, and even once an hour would be acceptable.

3. Only the first two rows of the ICRS to heliographic rotation matrix **rc2hg** are replaced. The bottom row is the ICRS unit vector to the solar north pole, which is constant.

6 Bibliography

Fränz, M. & Harper, D. 2002, *Heliospheric coordinate systems*, Planetary and Space Science, **50**, 2, 217-233

Hapgood, M.A. 1992, *Space physics coordinate transformations: a user guide*, Planetary and Space Science (ISSN 0032-0633), **40**, 5, 711-717

Moisson, X. & Bretagnon, P. 2001, *Analytical Planetary solution VSOP2000*, Celestial Mechanics and Dynamical Astronomy, **80**, 3-4, 205-213

Seidelmann, P.K. (ed) 1992, *Explanatory Supplement to the Astronomical Almanac*, ISBN 0-935702-68-7

Thompson, W.T. 2002, *Standardized coordinate systems for solar image data,* in Advances in Space Research, **29**, 12, 2093-2098.

Wallace, P.T. 2002, *A rigorous algorithm for telescope pointing*, Advanced Telescope and Instrumentation Control Software II. Edited by Lewis, H., Proceedings of the SPIE, 4848, 125

Wallace, P.T., 2016, *Pointing a solar telescope*, Proc SPIE, 9906, Ground-based and Airborne Telescopes VI, 990655 (27 July 2016)

7 Files

The following lists all the SOL files and the contents of each:

`ap2hg.c`	C source for the `sol`**`Ap2hg`** function
`ap2hp.c`	C source for the `sol`**`Ap2hp`** function
`c2hc.c`	C source for the `sol_`**`c2hc`** function
`c2hp.c`	C source for the `sol_`**`c2hp`** function
`demo.c`	C source for the demonstrator program
`hc2ap.c`	C source for the `sol`**`Hc2ap`** function
`hc2c.c`	C source for the `sol_`**`hc2c`** function
`hc2hg.c`	C source for the `sol`**`Hc2hg`** function
`hc2hp.c`	C source for the `sol`**`Hc2hp`** function
`hg2ap.c`	C source for the `sol`**`Hg2ap`** function
`hg2hc.c`	C source for the `sol`**`Hg2hc`** function
`hg2hp.c`	C source for the `sol`**`Hg2hp`** function
`hp2ap.c`	C source for the `sol`**`Hp2ap`** function
`hp2c.c`	C source for the `sol_`**`hp2c`** function
`hp2hg.c`	C source for the `sol`**`Hp2hg`** function
`sol.docx`	documentation, MS Word
`sol.pdf`	documentation, PDF
`soleph.c`	C source for the `sol`**`Soleph`** function
`sollib.h`	C source for the **`sollib.h`** header file
`solp.c`	C source for the `sol`**`Solp`** function
`solpci.c`	C source for the `sol`**`Solpci`** function
`solpeq.c`	C source for the `sol`**`Solpeq`** function
`solpt.c`	C source for the `sol`**`Solpt`** function
`solsys.h`	C source for the `sol`**`sys.h`** header file