Tpoint Software

Concise Pointing Kernel

TPS/PTW/07/002

8 May 2011

Patrick Wallace

Tpoint Software
19, Sutton Wick Lane, Drayton, Abingdon, OX14 4HH
United Kingdom
fax +44–1235–531198
www.tpsoft.demon.co.uk

CHANGE RECORD

Issue	Date	Reason for change
1	12 January 2007	New document
1.1	17 February 2007	cpkEqor added
1.2	7 April 2007	cpkDome added
1.3	5 May 2007	ID,CH allowed for altaz case
1.4	8 May 2007	Turbo refraction
1.5	2 September 2007	Temperature adjustment
1.6	27 January 2008	Revised cpkDome, and DAF term added
1.7	23 May 2009	Minor corrections
1.8	11 May 2011	GCRS added

Contents

1 Introduction

The Tpoint Software *Concise Pointing Kernel* (CPK) is a set of C functions that provides the pointing component of a telescope (or antenna) control system (TCS). Its core capability is to predict how to rotate the axes of an equatorial or altazimuth mount so that a specified celestial target is imaged at the center of the focal plane. In addition, and of particular importance for the altazimuth case, CPK predicts the angle to which the instrument mount (if any) needs to be set in order to stabilize the view of the sky.

CPK consists of a set of 19 C functions callable by a TCS, plus two demonstration applications and a handful of associated files. Some of the functions deal with initializing and maintaining a context that describes the astrometric circumstances and the pointing peculiarities of the telescope; other functions then use this context to make the pointing predictions. The remainder provide low-level support for time/date and vector/spherical transformations, and for string handling.

The end-to-end accuracy objective is better than 2 arcseconds, combined with extremely fast operation, making CPK a good choice for the majority of small telescopes, trackers and antennas. For more demanding applications, such as large professional astronomical telescopes, a much more powerful Tpoint Software package is available, called TCSpk. This not only provides astrometric transformations of milliarcsecond precision, but also allows image location in the focal plane to be controlled and caters for instrument rotators at Nasmyth and coudé focal positions as well as the usual Cassegrain etc.

CPK works in partnership with the interactive pointing analysis package TPOINT. One CPK function logs pointing observations in a form that TPOINT can read, and another accepts the pointing model files that TPOINT writes. A basic repertoire of TPOINT correction terms is provided. Simplified formulations are used, and it is easy to add and test new code to implement any further terms needed by a particular telescope. (The TCSpk package includes the full TPOINT repertoire, and is rigorous.)

The CPK package is supplied simply as source files, ready to be combined with the user's application. There are no platform dependencies, and no assumptions about build and deployment procedures.

2 Basic Principles

2.1 Pointing and tracking

Pointing a telescope means rotating the axes of its mount so that the desired target is seen. The process can be *open loop*, in other words by dead reckoning, or *closed loop*, using feedback from the received image or signal. Often a mixture of the two techniques is used, where first the target is acquired and followed by dead reckoning and then control is passed to an autoguider or other image-tracking device to maintain accurate alignment.

In a computerized telescope, the mount axes are equipped with position encoders, and the motors that drive the axes have the job of following the encoder readings demanded by the controlling device. The demands change with time, to compensate for the rotation of the Earth and, for some targets, to account for the motion of the target across the sky. Typically, the TCS needs to send a stream of such demands, regularly spaced in time, to the servo hardware and/or software that has the job of moving the axes. The principal capability offered by CPK is to generate this open-loop demand stream.

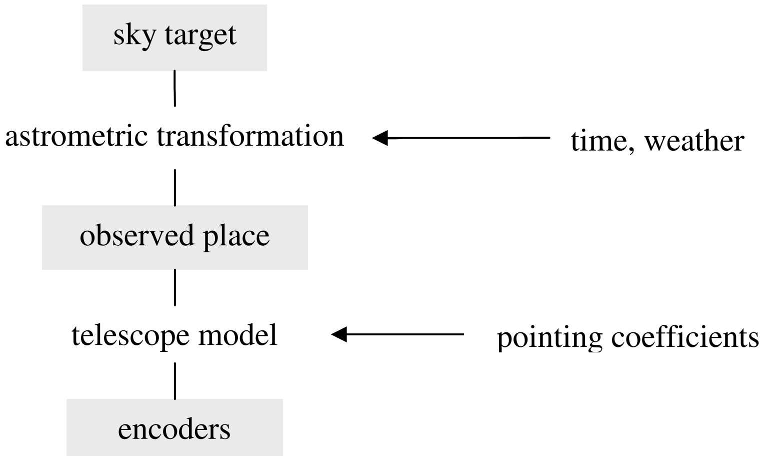
Note that there is no real qualitative distinction between pointing and tracking: the latter is merely the rate of change of the former. However, there may be differences in accuracy requirements for (i) pointing in the sense of target acquisition and (ii) tracking in the sense of maintaining alignment. For a CCD exposure it may well be good enough to begin the exposure with a 10 arcsecond initial misalignment between the center pixel and the nominated (α,δ) coordinates, but during the exposure even a 0.1 arcsec change in pointing may cause unacceptable image trailing.

When designing a TCS it is important to work in terms of position and never to use velocity for high-precision purposes, because of the inevitable and unknown build-up of rounding errors. Velocities can be used to generate changing position demands, and position predictions spaced in time can be differenced in order to deduce current velocity (also acceleration and so on), but the core pointing and tracking calculations must be carried out strictly in terms of absolute position.

CPK is based on the idea that the mount has absolute encoders or, failing that, zero-settable incremental encoders that do not lose their reference during slews to new targets or park positions. It is also assumed that there are no additional inputs into the servo position loops such as guide paddles or offset guiders. Such devices must work through the TCS.

2.2 Relationship between target coordinates and encoder demands

The coordinate transformations that link the desired sky target to the encoder demands are as follows:



The astrometric transformation is independent of the telescope, whereas the telescope model is peculiar to the individual telescope. Each component will be considered separately.

2.3 Astrometric transformation

The chain of coordinate transformations that links the star catalog entry to the observed azimuth and elevation is shown below in some detail. The coordinate systems (uppercase) and corrections (lowercase) that are supported by CPK appear in black:

STAR CATALOG
proper motion, catalog epoch to J2000
INTERNATIONAL CELESTIAL REFERENCE SYSTEM, epoch J2000
proper motion, J2000 to date
BARYCENTRIC CELESTIAL REFERENCE SYSTEM of date
annual parallax
ICRS ASTROMETRIC
light deflection
annual aberration
GEOCENTRIC CELESTIAL REFERENCE SYSTEM
frame bias
precession
nutation
CELESTIAL INTERMEDIATE REFERENCE SYSTEM
Earth rotation
TERRESTRIAL INTERMEDIATE REFERENCE SYSTEM
polar motion
INTERNATIONAL TERRESTRIAL REFERENCE SYSTEM
site longitude
diurnal aberration and parallax
TOPOCENTRIC
refraction
OBSERVED

The gray items are not supported by CPK: with the exception of the catalog handling part they consist of coordinate systems that are not needed and corrections too small to matter. It is the responsibility of the TCS application to deal with the star catalog portion of the chain. Given the CPK accuracy goals, stellar parallax can safely be neglected, but proper motion is significant and should really be taken into account by the user's application.

Note that no distinction is being made at this stage between (i) spherical coordinates such as right ascension and declination and (ii) vector representations. Each coordinate system listed in the chain simply identifies the choice of pole and longitude zero-point. CPK uses exclusively right-handed systems, and in most cases the functions accept directions expressed as unit vectors; the library includes utility functions for transforming between the vector and spherical forms.

The four CPK-supported coordinate systems, their corresponding internal codes, and their possible uses are as follows:

- ICRS (code **`ICRS`**): mainly for stars, though astrometric places are also suitable for planets.

- GCRS (code **`GCRS`**): mainly for spacecraft, but could also be used for planets.[1]

- CIRS right ascension and declination (code **`CIRS`**): this system is referred to the current pole and the celestial intermediate origin (CIO), and may be used for solar-system bodies.

- Topocentric, horizon-based (code **`AZEL_T`**): suitable for Earth satellites.

- Observed, horizon-based (code **`AZEL_O`**): suitable for targets where the entire line-of-sight prediction has been done outside of CPK and only the pointing-model part of the transformation chain is of interest.

CIRS coordinates were introduced by the IAU in 2000 to take over the role of classical geocentric apparent place. The zero point for CIRS right ascension is the celestial intermediate origin (CIO), a kinematically-defined point that is close to $\alpha_{\mathrm{ICRS}} = 0$, whereas apparent right ascension is with respect to the equinox of date. In order to track geocentric apparent place with CPK, use $(\alpha,\delta)_{\mathrm{CIRS}} = (\alpha_{\mathrm{apparent}}+\mathrm{EO}, \delta_{\mathrm{apparent}})$. EO is the equation of the origins (EO) a slowly-changing quantity that can be computed by calling the function `cpk`**`Eqor`** occasionally, say when the system is started.

[1] The distinction between the treatment of stars and solar-system targets is that for stars stellar aberration is applied whereas for planets and spacecraft essentially the same thing is the correction for the body's motion during the light-time (which is why the latter is sometimes called *planetary aberration*).

2.4 Telescope model

The telescope model bridges the gap between the observed direction – what a perfect theodolite would measure – and the corresponding encoder demands. It consists of the nominal mount orientation plus a set of pointing corrections.

CPK supports both equatorial mounts and altazimuth mounts. The CPK names for the final encoder demands are *roll* and *pitch*, meaning the longitude and latitude angles respectively. Conventionally, both equatorial and altazimuth mounts used left-handed coordinate systems whereas CPK's roll/pitch is right-handed. In the equatorial case, roll and pitch correspond to $-h$ and δ, where h (hour angle) is west-positive and δ is declination. In the altazimuth case, roll and pitch correspond to $\pi - A$ and E, where A is azimuth (north through east) and E is elevation (more properly *altitude*).

The pointing corrections comprise a mathematical model of various mechanical deficiencies in the telescope together with a set of small coefficients that define the amplitudes of the various effects. The basic mechanical deficiencies are as follows:

- the index errors (zero points) of the mount encoders (two numbers)
- the non-perpendicularities between the two axes of the mount and between the telescope boresight and the pitch axis (two numbers)
- the misalignment of the roll axis (two numbers)
- vertical droop (one number)

The different pointing terms, each with its own amplitude coefficient, influence the above seven quantities. In some cases, a single quantity receives contributions from multiple terms: for example gear errors will produce cyclic changes in the encoder index errors, adding to the constant zero-point correction itself.

For the telescope concerned, the required selection of pointing terms and the values of the corresponding coefficients are obtained from star observations using the TPOINT pointing analysis application. CPK includes support for logging pointing observations in a form that TPOINT accepts, and can read the model files that TPOINT writes.

2.5 Enclosure

If (i) the telescope's optical axis, (ii) the two axes of the mount and (iii) the center of the enclosure all intersect at a common point, the required settings for the dome and any windscreen aperture are trivially the azimuth and elevation of the star. However, many telescopes are mounted asymmetrically. This is a feature of the German equatorial mount (GEM) in particular, where the telescope is at one end of the declination axis and consequently there are gross differences between the star direction and the dome settings. When using a GEM, even observing the same star may call for quite different dome settings, depending on whether the telescope is east or west of the pier.

To deal with this, CPK includes a support function that, if told the mount orientation, the radius of the dome, the offset of the mount within the dome, the offsets between the roll, pitch and optical axes, and the mount attitude, predicts the dome azimuth and the elevation of any windscreen aperture. This information can then be used either to track the dome continuously or to move it when the beam is about to be vignetted.

3 The CPK library of C functions

3.1 The functions

The CPK library comprises one header file, **`cpk.h`**, and a set of ANSI C functions. Apart from its use of the C run-time library, CPK is self-contained.

Nearly half of the CPK functions are directly concerned with pointing, and work in conjunction with a set of context items (see 3.2):

CONTEXT

- `cpk`**`Model`** read pointing model from a file
- `cpk`**`Upda`** initialize or update the astrometric quantities
- `cpk`**`Updt`** update the Earth rotation

PREDICTIONS

- `cpk`**`Enc2sky`** sky coordinates corresponding to encoder values
- `cpk`**`Guide`** adjust the pointing model
- `cpk`**`Rot`** predict instrument rotator angle
- `cpk`**`Sky2enc`** encoder demands for specified sky target

POINTING TESTS

- `cpk`**`Tpt`** log pointing observation for TPOINT analysis

The remaining CPK functions are freestanding utilities:

ENCLOSURE

- `cpk`**`Dome`** enclosure settings for given encoder values

SUPPORT

- `cpk`**`Eqor`** equation of the origins
- `cpk`**`Mjd`** calendar date to MJD
- `cpk`**`Tday`** h,m,s to days

SPHERICAL/VECTOR

- `cpk`**`C2s`** Cartesian to spherical
- `cpk`**`S2c`** spherical to Cartesian
- `cpk`**`S2v`** spherical to vector
- `cpk`**`V2s`** vector to spherical

STRINGS

- `cpk`**`Length`** string length without trailing spaces
- `cpk`**`Scomp`** compare strings ignoring trailing spaces

Detailed specifications for each of the functions can be found in *Programmer's Reference*, Section 7. Note that radian values are used throughout: degrees, arcseconds etc. are a matter for the TCS application itself, in particular the user interface.

3.2 The data context

The CPK coordinate transformations use a set of precomputed parameters. These reside in a data structure of type **`TRANSP`** that is defined in the **`cpk.h`** header file. The CPK source code uses the name **`pars`** for instances of such structures. The following table lists the data members that comprise the **`TRANSP`** structure:

member	*meaning*	*updated by*
`elon`	site longitude (radians, east +ve)	`cpk`**`Upda`**
`sp` **`cp`**	sine, cosine of site latitude	`cpk`**`Upda`**
`xcip` **`ycip`** **`zcip`**	CIP vector	`cpk`**`Upda`**
`xab` **`yab`** **`zab`**	aberration vector	`cpk`**`Upda`**
`refr`	refraction constant	`cpk`**`Upda`**
`st` **`ct`**	sine, cosine of local ERA	`cpk`**`Updt`** `cpk`**`Upda`**
`tmount`	mount type (**`EQUAT`** or **`ALTAZ`**)	`cpk`**`Model`**
`pco[MAXT]`	pointing coefficients (radians)	`cpk`**`Model`**
`gc` **`gb`**	guiding corrections (radians)	`cpk`**`Guide`** `cpk`**`Model`**

Before any of the CPK pointing functions can be called, the data context must be initialized, by calling the functions `cpk`**`Upda`** and `cpk`**`Model`**: both are required. Subsequently, calls to the function `cpk`**`Updt`** update the context to reflect the rotation of the Earth, and calls to the function `cpk`**`Guide`** update the context to reflect changes to the pointing model.

An application can, if necessary, set up and maintain more than one such data context at once, for example to allow an observation planned for later to be previewed. Another use for this technique is to allow two pointing models to operate at once, perhaps to allow the target image to be switched rapidly between different points in the focal plane, or to account for misalignment and differential flexure between two telescopes on the same mount.

How often to refresh the context is up to the TCS developer. The CPK code is designed to be extremely fast, and it is perfectly reasonable simply to call `cpk`**`Model`** once, to set up the pointing model and reset the guiding corrections, and from then on to call `cpk`**`Upda`** frequently to take account of the changing astrometric transformation. However, because some of the context items are fixed and others change only very slowly, it is less wasteful to make the frequent calls instead to `cpk`**`Updt`**, which recomputes only the Earth-rotation parts. A fresh call to `cpk`**`Upda`** can be made once per new target (or even once per night), to follow air pressure changes and to update the precession-nutation and aberration. The latter never change the pointing by more then 0.5 arcsec/day.

If it will be necessary to use apparent place, the equation of the origins can be precomputed by calling `cpk`**`Eqor.`** This does not need doing more than once a session.

4 Using CPK in a TCS application

4.1 What CPK does not do

CPK provides only a small part of a TCS, namely that associated with pointing predictions. It is up to the TCS developer to provide everything else, including...

- real-time design and interactions with clocks
- provision of UT1[2]
- distributed processing, multi-thread design, mutexes
- the user interface
- sequencing, and coordination with instruments
- servos, including trajectory shaping and management of cable wraps
- above/below pole and side-of-pier choices
- mechanical interference, safety and interlocks
- star proper motion
- predictions for solar system bodies and spacecraft
- world coordinate systems (WCS)
- transformation of guiding demands into sky or mount coordinates
- parking and access positions
- use of brakes and clamps, and slew/track mode switching
- prediction of loss of acquisition
- absolute rotator angle
- management of the rotating focal plane
- Sun avoidance

…and much more.

[2] It is important to understand that CPK is driven by UT1and that this is not the same as UTC. Although the distinction between the two can be glossed over in the equatorial case by allowing a start-of-night rezeroing of the encoders (the so-called “sync”), this doesn’t work for altazimuths.

4.2 Slewing and tracking

The CPK world-view is that the mount drives are able to accept an ever-changing description of the desired locus and do everything needed to follow the demanded path. The CPK demands comprise time-stamped positions, obtained by repeated calls to `cpk`**`Updt`** and `cpk`**`Sky2enc`**. CPK is brutal: it does not generate ramps or make other concessions to the performance limitations of the drives. Those are issues for the motion controllers and/or the TCS application.

A common design is for the servos to accept a revised locus at intervals of say 20-100 ms. It should not be important for successive samples to be exactly spaced in time, but they must be accurately time-stamped – in other words the demand positions must be associated with the exact times used in the calculation. In this connection it should be noted that the time supplied to `cpk`**`Updt`** is UT1, whereas the time-scale in which the servos are working might be different – probably UTC or, ideally, TAI. (See the footnote on p9.)

If velocities are also required, they are obtained by computing two positions a chosen interval apart and taking the difference; similarly for accelerations, which can be obtained by taking three samples (but not too close together, or numerical noise will tend to dominate). These numerical derivatives are suitable for feedforward and extrapolation purposes but must not be used directly for open loop tracking (see also 2.1).

It is up to the servo software/hardware to look after any required signal-shaping, such as limiting of position, speed, acceleration and jerk. Decisions about cable-wrap state are also outside CPK's jurisdiction. In some mounts, management of separate slew/track functions will be required, and perhaps even awkward and old-fashioned devices such as tangent arms; CPK has no direct support for such features.

Ideally, a slew should be defined by the destination alone, with each mount axis simply left to get on with the job of catching up with its demands. However, for most equatorials there is a danger for some slews that the telescope will hit the ground en route. A simple and effective way to avoid this is to limit the available slew speed on the axis that is closest, in such a way that both axes arrive at the same time. Even more elaborate coordination will be needed for German equatorial mounts (GEMs) when crossing the meridian: it may be necessary first to slew in declination alone until the pole is reached, then to complete the hour angle slew, and finally continuing the declination slew.

4.3 Guiding

There are many circumstances in which it is necessary to move the target image in the focal plane, for example:

1. When inaccurate target coordinates have been supplied.
2. To select a particular feature on an extended object.
3. To follow a moving target such as a comet.
4. When the target image does not land in the right place because of pointing errors.
5. When the target image is to be moved from one place on the instrument to another (for example trailing along a slit).
6. When the target image is to be moved from an acquisition device (say an autoguider probe) onto the instrument hot-spot.
7. When the instrument rotator is turned and the instrument hot-spot isn't on the rotator axis.

CPK provides basic support for all of these, but it is crucial for the TCS (and the operator) to select the right method. If this is not done correctly, bad things can happen, in particular:

- Loss of acquisition as the instrument mount is turned.
- Incorrect logging of target coordinates and/or WCS information.

In the numbered list, above, cases 1, 2 and 3 are handled by changing the sky coordinates of the target. As a result (say) of pressing buttons on a handset, or clicking on a feature in a TV image, the TCS application changes the sky coordinates used in the calls to `cpk`**`Sky2enc`**, thus influencing the encoder demands and moving the telescope.

The remaining cases, 4-7, are handled by making adjustments to two terms in the pointing model, namely the boresight/pitch non-perpendicularity and the pitch zero-point. This is done by calling the function `cpk`**`Guide`**, which stores in the specified **`TRANSP`** structure the members **`gb`** and **`gc`**. These are pointing coefficients that supplement the TPOINT-derived model that was input with the `cpk`**`Model`** function when the **`TRANSP`** structure was initialized (or last updated). Because the call to `cpk`**`Guide`** has changed the model, calls to `cpk`**`Sky2enc`** generate revised encoder demands and consequently move the telescope – as intended.

Because both techniques – changing the target coordinates and changing the pointing model – support multiple applications, it is a good plan for the

TCS to maintain a list of offsets that are summed before being used in CPK calls. In the case of target coordinates, such a table might contain:

- direct offsets, supplied through the user interface
- offsets resulting from differential tracking
- offsets derived from point-and-click image-based interfaces

In the case of pointing-model adjustments, the table might contain:

- adjustments obtained during start-of-night calibration
- “trim” adjustments that will be discarded before the next acquisition
- rotator-angle-dependent adjustments

Rotator-angle-dependent adjustments are considered in more detail in 4.5.

4.4 Rotator

An instrument rotator is a normal part of an altazimuth telescope, where it is needed to compensate for field rotation, a gross effect. In an equatorial telescope, the field does not rotate, at least to first order, and so the main purpose of any instrument mount is to provide access to different picture orientations. CPK's rotator predictions, available through the function `cpk`**`Rot`**, work for both mount designs but will most often be used for the altazimuth case.

Just as to first order the rotator on an equatorial does not need to move, the rotator on an altazimuth to first order tracks the parallactic angle (the angle between north and vertical). However, it turns out that pointing errors can disturb these simple results. For fields near the pole of the mounting, small corrections on the sky may produce large changes to the "roll" coordinate (hour angle or azimuth), which will rotate the field by similar amounts. This effect is particularly severe on altazimuths, because the pitch coordinate (i.e. elevation) is changing and hence the pointing corrections are also changing. CPK sidesteps this difficulty by using encoder to sky transformations to sample the mapping between mount coordinates and the chosen sky coordinates (for example ICRS α,δ).

The call to `cpk`**`Rot`** requires the latest encoder demand, a **`TRANSP`** transformation parameters structure and a nominated sky coordinate system. The angle that is returned is the rotator position angle required to stabilize the northwards (or upwards) direction in the sky with respect to a fixed line in the rotating focal plane. The absolute value of the returned angle should be regarded as arbitrary: the rotator orientation should be set up directly by the TCS, the `cpk`**`Rot`** predictions being used to apply the rotation starting from that initial setting.

4.5 Focal-plane coordinates

The nominal *pointing origin* is the rotator axis: pointing tests normally involve placing a series of known stars on that point and logging the encoder demands, and the resulting pointing model will aim to place the target image at the same spot. However, if it is necessary to use a different pointing origin – say the center of a CCD that is slightly offset from the rotator axis – then the pointing will be wrong and will change as the rotator angle changes.

The solution is for the TCS application to provide for one or more pointing origins at nominated (x,y) coordinates in the rotating focal plane. As the telescope tracks, these (x,y) coordinates should be rotated through the instrument mount position angle to generate coordinates (ξ,η) in the non-rotating focal plane. These (ξ,η) coordinates can then be included in the table of pointing-model adjustments suggested in 4.3. One effect is that as the telescope tracks, the target image remains on the off-axis pointing origin. Another effect is that if the instrument mount is turned, the field rotates about the off-axis pointing origin.

4.6 Encoder to sky transformations

The cpk**Enc2sky** function provides a good approximation to the inverse of the cpk**Sky2enc** sky-to-encoders function that delivers CPK's basic tracking capability. As we saw in 4.4, encoder-to-sky transformations allow CPK to determine field orientation; what other uses are there?

A naive TCS designer may be tempted to use cpk**Enc2sky** to translate encoder readings into sky coordinates in order to find out "where the telescope is actually pointing". However, this is wasteful and potentially inaccurate. Because the tracking function cpk**Sky2enc** generates encoder *demands*, the pointing model must relate true pointing direction to the demands necessary to centre the target image, and *the best estimate of where the telescope is pointing is simply where it has been asked to point*. If, for servo reasons, there happens to be an offset between the encoder demands and the encoder readouts, this will cancel; if, instead, the encoder readings are used, the offset will appear as an error in the sky coordinates. Moreover, if the encoder readings are to be visible to the TCS, they must be properly time-stamped and will in general not match the times used to generate the transformation parameters.

It is sometimes thought that the encoders-to-sky transformation is needed to show the progress of a slew. This is not really the case, as the instantaneous servo errors – i.e. how far left to go in each axis – are at least as useful as an α,δ readout.

However, there is one case where use of encoder readings is justified, and this is when the servo cannot keep up. If the mount provides very high velocities and accelerations in order to track rapidly-moving nearby objects, it is possible that the encoder readings are indeed a better guide than the demands. Under these circumstances, resorting to cpk**Enc2sky** may be best after all.

4.7 Beyond the pole

All gimbal mounts can in principle reach the same point in the sky in two ways, one with pitch angles in the range ±90º, the other in the region out to ±180º with an accompanying change of 180º change in the roll angle. This is particularly easy to see in the case of a fork equatorial, where the telescope can (apart from mechanical collisions) reach the region of sky beneath the pole. Such a reversal is of course an essential feature of GEMs, where the entire declination range is used in order to place the telescope on either the east or the west of the pier. Even some altazimuth mounts can do it, though it is generally a bad idea: no extra sky can be observed, and the gravity reversal may lead to optics shifts and other forms of hysteresis.

One of the arguments to the `cpk`**`Sky2enc`** function is **`jbp`**, a flag that selects which of the two mount settings to use. For an altazimuth mount, this should normally be set to zero, so that pitches in the convenional 0-90º range are generated. For a fork or horseshoe equatorial there may – depending on mechanical constraints – be occasional uses for the **`jbp`** = 1 case. For a GEM, the values 0 and 1 will each appear half the time, reflecting which side of the pier the telescope is on; it is up to the TCS to decide which option is appropriate.

4.8 Pointing tests

It is a TCS responsibility to provide pointing test data from which the TPOINT model can be derived, needed so that CPK can generate appropriate corrections. The pointing observations comprise records of (i) where the telescope was actually pointing and (ii) what encoder demands brought about the stated alignment. This information can be logged directly by CPK: calling the `cpk`**`Tpt`** function when the telescope is pointing at a known place produces a TPOINT record in the form of a character string. The procedure is demonstrated in the T_TPT example application (see section 6.2)[3].

Notes:

1. The actual pointing direction may be (and usually is) a star, but it may be a determination of the field center (α,δ) made by an automatic astrometry system.

2. Choose stars, or sky positions, that are more or less equally spaced in the sky – TPOINT comes with suitable star lists. Do not space your stars equally in hour angle or azimuth as this will produce concentrations around the pole or zenith and introduce unwanted weighting in the TPOINT fits.

3. The pointing origin should be fixed in the non-rotating field. This can be achieved either by not rotating the measurement device (camera or graticule-eyepiece) or by using the rotator axis as the reference point.

4. An exception to Rule 3 can be made where an instrument rotator is being used to compensate for field rotation in an altazimuth telescope. Including the equatorial pointing terms ID and CH in the otherwise altazimuth model will have the effect of correcting for any offset between the pointing origin and the rotator axis. The terms IE and CA will be for the rotator axis, while the terms ID and CH represent the offset. However, for the resulting pointing model to work, the rotator must always be used in the same way as during the pointing test, for example to keep the CCD axis running north. If another orientation is chosen the offset will no longer be correctly compensated.

Do not be tempted to...

[3] In addition to the observation records themselves, the TPOINT file contains certain other information, including a caption and the site latitude. For further details see the T_TPT source code (**`t_tpt.c`**) and the TPOINT documentation.

- ... log encoder readings instead of demands. This introduces unnecessary extra needs for precise timing, and servo peculiarities may mean that the demands and readouts do not agree perfectly.

- ...log the guiding *corrections*, relative to the original blind acquisition. This will generate a TPOINT model that is *relative* rather than absolute, and require meticulous records to be kept of what model was in service.

5 The TPOINT pointing model

CPK uses the same pointing-model conventions as the analysis package TPOINT, supporting a selection of the most important terms and using the same names. TPOINT models written with the `OUTDAT` command are read directly by the `cpk`**`Model`** function and set up the appropriate parts of the **`TRANSP`** data structure; unrecognized terms are simply ignored, and terms not present in the model file are set to zero amplitude.

5.1 Supported terms

The terms supported by CPK (as delivered) comprise the basic geometrical set plus some common flexures and cyclic effects. With a few exceptions (ID, CH and TF), terms are either equatorial or altazimuth and must not be mixed. A model file containing a mixture of the two will be read, but the discordant terms will simply be ignored when the pointing corrections are evaluated.

term	***meaning***	**`EQUAT`**	**`ALTAZ`**
IH	hour angle index error	✓	✗
ID	declination index error	✓	✓
NP	h/δ non-perpendicularity	✓	✗
CH	boresight/δ non-perpendicularity	✓	✓
ME	polar axis vertical misalignment	✓	✗
MA	polar axis left-right misalignment	✓	✗
FO	fork flexure	✓	✗
DAF	δ axis flop	✓	✗
IA	azimuth index error	✗	✓
IE	elevation index error	✗	✓
NPAE	A/E non-perpendicularity	✗	✓
CA	boresight/E non-perpendicularity	✗	✓
AN	azimuth axis tilt north	✗	✓
AW	azimuth axis tilt west	✗	✓
HESE	elevation runout, sine component	✗	✓
TF	Hooke's Law vertical flexure	✓	✓

The terms ID and CH, which are always present in the equatorial case, are useful also in the altazimuth case to compensate for an off-axis pointing origin when a rotator is being used to stabilize the field orientation. The term TF nominally represents flexure in the telescope tube and is likewise supported in both the equatorial and altazimuth cases.

The formulations of the individual terms are not rigorous and assume fairly small coefficients – a few hundred arcseconds or less in most cases – but are good enough to satisfy the CPK accuracy goals.

5.2 Identification of mount type

All TPOINT models include index errors: IH and ID for an equatorial mount, IA and IE for an altazimuth. The CPK function cpk**Model** exploits this fact to identify the mount type: if the IH term is present the mount type is **EQUAT**, and if the IA term is present the mount type is **ALTAZ** (shaded items in above table).

5.3 Adding new terms

TPOINT supports hundreds of different pointing terms, and only the most common are implemented in CPK as delivered. If a telescope is found to require additional terms, changes in several places are needed:

- In the header file **cpk.h**, the names of the additional terms must be added to the **enum** definition.

- In the cpk**Model** function, add extra lines to recognize the term name and deposit the coefficient value in the appropriate element of the **pars->pco** array.

- In the cpk**Sky2enc** function, add appropriate extra code. This may be tricky. The new term should be identified as to its natural place in the 7-term basic model – most usually it will contribute to the **ta** variable (direct changes to the roll coordinate), the **tb** variable (direct changes to the pitch coordinate) or the **tc** variable (changes to the collimation correction).

- In the cpk**Enc2sky** function, add appropriate extra code. This will often be similar to the changes made to the cpk**Sky2enc** function.

- Add the new term(s) to one or both of the example model files **equat.mod** and **altaz.mod**.

- Add to the table in Section 5.1

To verify that the changes have worked, build and run the example applications T_STAR and T_TPT (see Section 6 below).

- Run T_STAR and check that the ICRS upstream coordinates match the original catalog coordinates. This proves that the changes to the `cpk`**`Sky2enc`** function match those in the `cpk`**`Enc2sky`** function.

- Make a copy **`model.mod`** of the appropriate example model file and run T_TPT. Run TPOINT, input (using the INDAT command) the data file written by T_TPT, input (using the INMOD command) the model file, and perform a fit. If all the terms in **`model.mod`** are reproduced, with coefficients of the right values and correctly signed, and if the residuals are small, this proves that the implementation of the new term(s) in `cpk`**`Sky2enc`** and `cpk`**`Enc2sky`** matches TPOINT's.

When implementing harmonic (or polynomial) terms in azimuth, note that the TPOINT internal convention is used, where south is zero and azimuth increases anticlockwise: $A_{TPOINT} = (180° - A_{CONVENTIONAL})$. Appropriate care must be taken with signs. The T_TPT test will verify that all is well.

6 Example applications

Two example applications are supplied with CPK. Apart from demonstrating some of the CPK function calls, they both have important roles in verifying the implementation of new pointing terms – see 5.3.

6.1 T_STAR

The T_STAR example application demonstrates how to point a telescope at a given star. It also demonstrates the reverse transformation and the rotator predictions.

The telescope is at N 35º 12' 36" W 111º 37' 12", elevation 2300m (near Flagstaff, Arizona); the time is 2006 December 28, $04^h\ 05^m\ 12^s$ UT and the star is β Orionis (Rigel), $(\alpha,\delta)_{ICRS} = 5^h\ 14^m\ 32.27^s$ –8º 12' 05.9". The local air pressure (QFE) is 766 hPa. Using a **`model.mod`** file that is a copy of the supplied example altazimuth pointing model **`altaz.mod`**, the following output is produced:

```
Catalog:

  RA =    78.63446, Dec =  -8.20164  ICRS

Downstream:

  RA =    78.63445, Dec =  -8.19196  GCRS
  RA =    78.63993, Dec =  -8.20213  date
  Az =   138.28760, El  =  36.83526  topo
  Az =   138.28760, El  =  36.85149  obs

  A  =  +138.33516, B   = +36.81436  enc

Same with guiding:

  dC =    +0.00100, dB  =  +0.00200

  A  =  +138.33641, B   = +36.81236  enc

Upstream:

  A  =  +138.33641, B   = +36.81236  enc

  Az =   138.28760, El  =  36.85149  obs
  Az =   138.28760, El  =  36.83526  topo
  RA =    78.63445, Dec =  -8.19196  date
  RA =    78.63992, Dec =  -8.20213  GCRS

  RA =    78.63446, Dec =  -8.20164  ICRS

Rotator:

  PA =   +33.32489Catalog:
```

The "downstream" (i.e. star to encoders) topocentric (A,E) result is (138.28758º, 36.83528º). This can be compared with USNO predictions:

```
                              Rigel

                   Apparent Topocentric Positions
                     Local Zenith and True North

                         FLAGSTAFF, ARIZONA
           Location:  W111°37'12.0", N35°12'36.0",  2300m
             (Longitude referred to Greenwich meridian)

    Date        Time                 Zenith              Azimuth
         (UT1)                      Distance             (E of N)
              h  m   s               °  '   "             °  '   "
 2006 Dec 28 04:05:12.0             53 09 53.8          138 17 15.0
```

The USNO figures are thus (138.28750º, 36.83506º), and the two sets of predictions differ by (0.00008º, 0.00022º), equivalent to an on-sky difference of 0.7 arcseconds.

6.2 T_TPT

The T_TPT example application writes a file of pointing observations into a form that can be read by the TPOINT analysis package. When fitted to the model that was present in the **`model.mod`** file when T_TPT was run, the observations enable the original coefficient values to be recovered. This is an invaluable check on the model term formulations implemented in the `cpk`**`Sky2enc`** function.

The format used for the TPOINT observation records depends on whether the model was for an equatorial or an altazimuth. In the equatorial case, INDAT Format 1 is used, with the star and telescope coordinates expressed as (h,δ) but written as if they are apparent (α, δ) positions for local sidereal time zero. In the altazimuth case, INDAT Format 4 is used, so that the star and telescope coordinates are both azimuth and elevation in degrees.

Here are the two supplied example model files, **`equat.mod`**...

```
Example equatorial model file
T    0   0.0000    0.000   0.0000
  IH       +80.0000     0.00000
  ID       +70.0000     0.00000
  FO       +60.0000     0.00000
  TF       +50.0000     0.00000
  NP       +40.0000     0.00000
  CH       +30.0000     0.00000
  ME       +20.0000     0.00000
  MA       +10.0000     0.00000
END
```

...and **`altaz.mod`**:

```
Example altazimuth model file
T    0   0.0000    0.000   0.0000
  IA       +80.0000     0.00000
  IE       +70.0000     0.00000
  HESE     +60.0000     0.00000
  NPAE     +50.0000     0.00000
  CA       +40.0000     0.00000
  AN       +30.0000     0.00000
  AW       +20.0000     0.00000
  TF       +10.0000     0.00000
END
```

Here is the resulting TPOINT file generated by T_TPT for the equatorial case[4] (abridged):

```
Simulated observations
: ALLSKY
: EQUAT
: NODA
+35 12 36.0
23 12 44.0560   -33 42 35.626   23 12 50.9382   -33 44 19.806   00 00
08 06 54.8962   +65 57 46.270   08 07 03.6774   +65 57 58.236   00 00
21 42 35.4478   -25 39 11.399   21 42 43.3271   -25 40 52.506   00 00
06 29 58.5343   +49 00 16.864   06 30 04.4479   +48 59 50.147   00 00
     :
     :
23 11 47.9844   +42 49 31.043   23 11 58.9441   +42 46 54.706   00 00
00 10 01.7730   +27 25 03.511   00 10 10.2325   +27 22 40.826   00 00
00 24 26.0848   +41 44 29.364   00 24 35.3707   +41 41 55.358   00 00
23 38 45.9999   +35 03 53.486   23 38 55.5764   +35 01 23.134   00 00
END
```

Here is the corresponding file for the altazimuth case (abridged):

```
Simulated observations
: ALLSKY
: ALTAZ
+35 12 36.0
190.457571  20.198268 190.494174  20.168491
 21.720256  20.582352  21.762662  20.565333
213.015350  21.004280 213.051274  20.977495
 44.320619  21.424502  44.364074  21.403636
      :
      :
312.810534  77.952853 312.940209  77.927452
163.962106  81.919499 164.139634  81.874438
 34.347491  81.909835  34.611845  81.878059
269.614696  85.657075 269.856607  85.626785
END
```

[4] As already pointed out, what are being presented to TPOINT as apparent (α,δ) positions are in fact observed ($-h$,δ) positions. Setting the LST to zero eliminates any need to distinguish between equinox-based and CIO-based right ascensions, and omitting weather data from the site parameters record, which consists of the site latitude alone, removes the distinction between topocentric and observed coordinates.

The TPOINT fits for the two cases are as follows:

```
       coeff       change     value    sigma

  1     IH         -0.000    +79.99    0.006
  2     ID         +0.000    +70.02    0.002
  3     FO         +0.000    +59.99    0.004
  4     TF         +0.000    +50.01    0.002
  5     NP         +0.000    +40.00    0.004
  6     CH         +0.000    +30.01    0.006
  7     ME         -0.000    +20.00    0.003
  8     MA         +0.000    +10.00    0.001

Sky RMS =   0.01
Popn SD =   0.01
```

```
       coeff       change     value    sigma

  1     IA         +0.000    +80.00    0.003
  2     IE         -0.000    +69.97    0.004
  3     HESE       +0.000    +60.03    0.003
  4     NPAE       +0.000    +50.00    0.003
  5     CA         -0.000    +40.00    0.004
  6     AN         +0.000    +30.00    0.000
  7     AW         -0.000    +20.00    0.000
  8     TF         -0.000     +9.96    0.003

Sky RMS =   0.00
Popn SD =   0.00
```

The small disagreements between the actual and fitted models are caused by CPK's slight lack of rigor in the formulation of the pointing corrections. This also manifests itself as rapidly worsening residuals as the pole of the mounting is approached. The effect can be reduced by restricting pointing tests to stars where $|\,\delta\,| < 85°$ or E < 85°, say, for equatorials and altazimuths respectively.

7 Programmer's reference

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

- `cpk`**`C2s`** Cartesian to spherical
- `cpk`**`Dome`** enclosure settings
- `cpk`**`Enc2sky`** sky coordinates corresponding to encoder values
- `cpk`**`Eqor`** equation of the origins
- `cpk`**`Guide`** adjust the pointing model
- `cpk`**`Length`** string length without trailing spaces
- `cpk`**`Mjd`** calendar date to MJD
- `cpk`**`Model`** read pointing model from a file
- `cpk`**`Rot`** predict instrument rotator angle
- `cpk`**`S2c`** spherical to Cartesian
- `cpk`**`S2v`** spherical to vector
- `cpk`**`Scomp`** compare strings ignoring trailing spaces
- `cpk`**`Sky2enc`** encoder demands for specified sky target
- `cpk`**`Tday`** h,m,s to days
- `cpk`**`Tpt`** log pointing observation for TPOINT analysis
- `cpk`**`Upda`** initialize or update the astrometric quantities
- `cpk`**`Updt`** update the Earth rotation
- `cpk`**`V2s`** vector to spherical

cpk**C2s**	**Cartesian to spherical**	cpk**C2s**

ACTION: Transform Cartesian coordinates into spherical coordinates.

GIVEN:

x	`double`	x-component
y	`double`	y-component
z	`double`	z-component

RETURNED:

a	`double*`	longitude angle (e.g. α) in radians
b	`double*`	latitude angle (e.g. δ) in radians

NOTES:

1. No validation of the arguments is performed.

2. The Cartesian coordinates need not be for a unit vector. The original magnitude is lost and the results contain only the direction.

3. Right-handed coordinate systems are assumed: α = **a**, h = −**a**, A = π−**a**. For the (h,δ) case, the alternative is to flip y first. For the (A,E) case, the alternative is to flip x first.

cpk**Dome**	**enclosure settings**	cpk**Dome**

ACTION: Predict the azimuth and elevation of the dome aperture for an asymmetrically mounted telescope.

GIVEN:

`phi`	`double`	elevation of roll axis in radians
`rdome`	`double`	radius of dome
`xm,ym,zm`	`double`	offset of mount in dome
`p`	`double`	separation of roll and pitch axes
`q`	`double`	distance along pitch axis
`r`	`double`	separation of pitch axis and telescope
`ta,tb`	`double`	telescope roll/pitch in radians

RETURNED:

`az,el`	`double`	dome-aperture az/el in radians

RETURNED (function value)**:**

`int`	status: 0 = OK −1 = illegal **`rdome`** −2 = no solution

NOTES:

1. The purpose of this function is to predict the point on the dome (azimuth and elevation) through which the telescope's optical axis passes. This depends on where the telescope is pointed, where within the dome the mount is located, the mount geometry, and where on the mount the telescope is fixed.

2. The function supports all types of mount based on two axes at right angles. The longitude or *roll* axis is the polar axis in the case of an equatorial mount and the azimuth axis in the case of an altazimuth mount. The latitude or *pitch* axis is the declination axis in the case of an equatorial mount and the elevation axis in the case of an altazimuth mount.

3. The telescope roll/pitch coordinate system matches $-h,\delta$ in the equatorial case. This means that it is right-handed, longitude increasing anticlockwise as seen from the positive pole. It means also that zero roll occurs when the telescope is pointing south. Thus in the altazimuth case the telescope roll/pitch system matches ($\pi-A$, E).

4. For the roll/pitch values **`ta`** and **`tb`**, relatively low precision will suffice. For example, the pointing model (corrections for the effects of minor flexures and misalignments etc.) can safely be ignored.

5. The type of mount is specified through the **`phi`** argument, which is the elevation above the horizon of the north, or positive, direction of the roll axis. For an equatorial mount, **`phi`** is equal to the latitude; for an altazimuth mount it is equal to $\pi/2$. Except in the altazimuth case, the tilt of the roll axis defines "north" for the dome. If the roll axis is not aligned north-south then an appropriate adjustment needs to be made to the returned azimuth.

6. The enclosure is presumed to be hemispherical (or some other portion of a sphere). The radius of the sphere is specified through the **`rdome`** argument. Any desired units can be used as long as the other "length" arguments are in the same units.

7. The arguments **`xm`**,**`ym`**,**`zm`** specify the offset of the mount from the center of the dome. The “mount" in this context is that point along the roll axis (and hence fixed in space) that lies nearest to the pitch axis. The x,y,z coordinate system is oriented east,north,up.

8. The arguments **`p`**, **`q`** and **`r`** form a sequence of offsets linking the roll axis, the pitch axis and the telescope's optical axis:

   - **`p`** is the separation between the roll and pitch axis at their closest approach. For most GEMs (German equatorial mounts) and altazimuth mounts, the two axes intersect and hence **`p`** is zero. Occasionally (some horseshoe mounts for example) the two axes do not intersect and **`p`** is non-zero.

   - **`q`** is the distance along the pitch axis to where the telescope assembly turns, starting from the point on the pitch axis closest to the roll axis. For most altazimuths and for equatorial forks etc., **`q`** is zero, but for GEMs **`q`** is a substantial distance.

   - **`r`** is the separation between the pitch and optical axes. It is usually zero. Very occasionally, for example where a second telescope is mounted on the"side" of the main telescope, the pitch axis and optical axis do not intersect and consequently **r** is non-zero.

9. The sign conventions for **p**, **q** and **r** are most easily described by considering the altazimuth and equatorial cases separately, though the two cases are in fact equivalent.

   - Altazimuth: with the telescope horizontal and pointing south, **p** is positive to the north, **q** is positive to the east and **r** is positive upwards.

   - Equatorial: with the telescope pointing at $h = \delta = 0$, **p** is positive towards $h = \pi$, $\delta = 0$, **q** is positive towards $h = -\pi/2$, $\delta = 0$ (i.e. the east) and **r** is positive towards $\delta = +\pi/2$.

10. The offsets **p** and **q** are at right angles, and the offsets **q** and **r** are at right angles, but the angle between the offsets **p** and **r** varies with the pitch angle.

11. The units of **rdome**, **xm**, **ym**, **zm**, **p**, q and r must all be the same.

12. The dome azimuth/elevation coordinate system follows the normal convention. Azimuth increases clockwise from zero in the north, through 90º (π /2 radians) in the east. The value returned is in the range zero to 2π. At the zenith, zero is returned.

13. Example:

   - The mount is a GEM and is at latitude +36.18°.
   - The dome is 3.8 meters in diameter.
   - The optical axis and the declination axis intersect.
   - The declination axis and the polar axis intersect.
   - When the mount is set to zero *h,δ*, the telescope is east of the pier and the counterweight is west of the pier.
   - The distance from the polar axis along the declination axis to the optical axis is 505 mm.
   - The point at which the polar axis and declination axis intersect is 35 mm west of, 370 mm north of, and 1250 mm above the center of the dome.
   - The telescope is pointing at a star 10^{m} west of the meridian, at $\delta = +37.9°$, and is east of the pier.

   The **cpkDome** arguments are as follows:

phi	0.6315	elevation of roll axis

`rdome`	1900.0	radius of dome
`xm`	–35.0	offset of…
`ym`	370.0	…mount from…
`zm`	1250.0	…dome
`p`	505.0	separation of mount axes
`q`	505.0	distance of telescope along δ axis
`r`	0.0	separation of δ and optical axes
`ta`	–0.0436	telescope roll (i.e. $-h$)
`tb`	0.6615	telescope pitch (i.e. δ)

14. The telescope roll/pitch are mechanical rather than celestial, so that above/below pole and east/west of the pier cases are distinguished. When the mount is in the "below the pole" configuration, the pitch value **`tb`** will lie outside the range $\pm\pi/2$. Though rare for most mount types, in the GEM case this will happen half the time because of the pier reversal necessary as targets cross the meridian. For the telescope in Note 13, to acquire the same star but with the telescope west of the pier would require **`ta`** = 3.098 and **`tb`** = 2.480 (radians).

cpk**Enc2sky**	**encoders to sky**	cpk**Enc2sky**

ACTION: Evaluate a specified part of the chain of transformations involved in determining where a telescope is pointing:

ENC encoder demands: where the mount was aimed
↓
pointing model
↓
AZEL_O observed (refracted) A,E
↓
refraction
↓
AZEL_T topocentric (unrefracted) A,E
↓
Earth rotation
↓
CIRS α,δ of date (RA is with respect to CIO)
↓
precession-nutation
↓
GCRS α,δ with respect to ICRS axes
↓
aberration
↓
ICRS astrometric α,δ

GIVEN:

pars	TRANSP*	transformation parameters
a	double	roll angle demand (radians)
b	double	pitch angle demand (radians)
to	COSYS	final system (**AZEL_O**, **AZEL_T**, **CIRS**, **GCRS** or **ICRS**)

RETURNED:

vobs	double[3]	**AZEL_O** vector (i.e. topocentric A,E)
vtopo	double[3]	**AZEL_T** vector (i.e. topocentric A,E)
vdate	double[3]	**CIRS** vector (i.e. CIO-based α,δ of date)
vgcrs	double[3]	**GCRS** vector (~aberrated J2000 mean α,δ)
vicrs	double[3]	**ICRS** vector (~ J2000 mean α,δ)

RETURNED (function value)**:**

int	true = illegal final system **to**

NOTES:

1. The starting point is always the demanded encoder readings **ENC**. The sky directions **AZEL_O**, **AZEL_T**, **CIRS**, **GCRS** and **ICRS** are returned, but optionally stopping short at the specified place.

2. The transformation parameters **pars** must be set up before the call. See the functions cpk**Model**, cpk**Upda** and cpk**Updt**.

3. The accuracy of the first part of the transformation, namely the pointing corrections, depends on the size of the model coefficients. For typical values, the error compared with a rigorous implementation is unlikely to be as large as 1 arcsecond.

4. Using the functions cpk**Upda** etc., the accuracy of the remaining **AZEL_O** to **ICRS** portion is about 1 arcsecond for dates between 1950 and 2100, elevations above 25° and V color.

5. The inverse transformations are performed by the function cpk**Sky2enc**. The two functions are not precisely complementary, but for typical pointing coefficients should deliver consistency at the 1 arcsecond level except close to the pole of the mounting.

6. All results are right-handed 3-vectors with axes as follows:

	CIRS, GCRS, ICRS	**AZEL_O, AZEL_T**
x	$\alpha = 0^h$, $\delta = 0°$	south
y	$\alpha = 6^h$, $\delta = 0°$	east
z	$\delta = 90°$	zenith

7. The returned vectors are not precisely unit vectors in all cases.

8. The encoder demand angles are h,δ for the case of an equatorial mount and A,E for an altazimuth mount. Hour angle is zero on the meridian and west-positive. Azimuth is zero in the north and increases clockwise as seen from above. The angles may exceed normal ranges, and represent actual mechanical attitude.

cpk**Eqor**	**equation of the origins**	cpk**Eqor**

ACTION: Equation of the origins.

GIVEN:

`date`	`double`	date as MJD (JD–2400000.5)

RETURNED (function value)**:**

`double`	equation of the origins (radians)

NOTES:

1. The date is TDB in principle, but it is perfectly acceptable to use UTC or for that matter UT1.

2. The equation of the origins (EO) is the accumulated precession-nutation since J2000. It is the difference between Greenwich sidereal time and Earth rotation angle and, equivalently, between the equinox based apparent right ascension and the CIO-based CIRS right ascension:

$$\begin{aligned} EO &= ERA - GST \\ &= \alpha_{CIRS} - \alpha_{apparent} \end{aligned}$$

3. This function need be called only occasionally, say once per slew or even once per night, and only if support for geocentric apparent place is required. The value returned can then be used to transform apparent (α,δ) target coordinates into the CIRS coordinates used by CPK: simply add EO to apparent α and leave δ as it is.

4. The series used is accurate to about 0.25 arcsec during the 21st century.

cpk**Guide**	**adjust pointing model**	cpk**Guide**

ACTION: Make adjustments to the pointing model in order to move the target image to the right place in the focal plane.

GIVEN:

`dc`	`double`	guiding correction in collimation (radians)
`db`	`double`	guiding correction in pitch (radians)

RETURNED:

`pars`	`TRANSP*`	updated

NOTES:

1. No validation of the arguments is performed. For example, omitting a scale factor from arcseconds to radians would simply produce extremely large guiding corrections.

2. The guiding correction in collimation is added to the pointing coefficient CH for an equatorial or CA for an altazimuth. The guiding correction in pitch is added to the pointing coefficient ID for an equatorial or IE for an altazimuth.

3. The guiding correction is an offset in position that persists until explicitly canceled, by a call to the present function with both arguments zero. Except for the initial transient, it does not produce a *tracking rate* adjustment.

4. This form of guiding moves the telescope without changing the celestial coordinates of the target. The TCS application will normally also support a mode where the guiding comes instead from adjusting the celestial coordinates of the target, which does not involve the present function. The third option, namely controlling which place in the focal plane is to receive the target's image, does involve the present function, but using guiding corrections that depend on the instrument mount position angle. It is the responsibility of the telescope user to decide which of these modes is appropriate to the task in hand, and it is the responsibility of the TCS application developer to implement suitable code.

5. The character of the guiding corrections (i.e. as collimation adjustments) makes them suitable for start-of-night "trimming" corrections, matching

such effects as optical alignment and temperature differentials. If the guiding corrections required to center the first star (or their average over a few stars) are saved by the TCS application and used as a fixed basis for subsequent operations, the all-sky pointing will probably be improved. However, the character of the corrections does not match loss of encoder zero points, at least in the "roll" coordinate (i.e. h or A). A start-of-night alignment procedure for a mount lacking absolute encoders would require stars at different pitch angles to be observed and the results analyzed to distinguish between (i) index errors in h or A and (ii) non-perpendicularity between the boresight and the δ or E axis.

cpk**Length**	**length of string**	cpk**Length**

ACTION: Length of a string excluding any trailing whitespace.

GIVEN:

`string`	`char*`	the string

RETURNED (function value)**:**

	`int`	length excluding trailing blanks

NOTES:

1. The minimum length returned is zero.

cpk**Mjd**	**Gregorian date to MJD**	cpk**MJd**

ACTION: Gregorian calendar date to Modified Julian Date.

GIVEN:

`iy`	`int`	year
`im`	`int`	month
`id`	`int`	day

RETURNED (function value)**:**

	`double`	Modified Julian Date: JD–2400000.5

NOTES:

1. The algorithm used is valid from –4800 March 1.

2. No validation of the arguments is performed.

3. In early eras the conversion is from the "Proleptic Gregorian Calendar"; no account is taken of the date(s) of adoption of the Gregorian Calendar, nor is the AD/BC numbering convention observed (hence the minus sign in Note 1).

REFERENCES:

1. Explanatory Supplement to the Astronomical Almanac, P. Kenneth Seidelmann (ed), University Science Books (1992), Section 12.92 (p604).

cpk**Model**	**read pointing model file**	cpk**Model**

ACTION: Read the telescope pointing model from a file. The file must conform strictly to the format produced by the OUTMOD command in the TPOINT package, except that certain records and fields are ignored (Notes 1,2).

GIVEN:

`file`	`char[]`	the name of the file to be read

RETURNED:

`pars`	`TRANSP*`	pco[MAXT] array and tmount set

RETURNED (function value)**:**

	`int`	status: 0 = OK −1 = unable to open input file −2 = I/O error or premature EOF −3 = unrecognized record

NOTES:

1. The file containing the pointing model must, as far as the basic layout is concerned, be exactly as written by TPOINT's OUTMOD command (which indeed is the ideal way of generating the file). However, unlike TPOINT's INMOD command, the present function does not support format changes and the insertion of comment records. The first two records, which are ignored, may be used to record comments; however, both must be present, as must the END marker. In the term records, the term name must start in column 3, and the coefficient value must not start before column 11.

2. The order of the term records is ignored. Only selected TPOINT terms are supported; if absent they are set to zero; unknown terms cause an error status.

3. The supported terms are as follows. Each appears as a radian value in the specified **`pars->pco[]`** element.

IH	hour angle index error
ID	declination index error
TF	tube flexure
FO	fork flexure

NP	h/δ nonperp
CH	boresight/δ nonperp
ME	polar axis misalignment up-down
MA	polar axis misalignment left-right on sky
IA	azimuth index error
IE	elevation index error
HESE	elevation centering, sine
NPAE	A/E nonperp
CA	boresight/E nonperp
AN	azimuth axis tilt north
AW	azimuth axis tilt west

4. Here is an example file:

```
"Comment 1"
"Comment 2"
" IA       -256.0000"
" IE       +380.2140"
" HESE      -24.5223"
" TF        +27.2402"
" NPAE      -13.7719"
" CA        +18.1204"
" AN        -32.0555"
" AW        +17.1147"
"END"
```

↑
col 1

Note the two initial records, which must be present but which are not interpreted, and the mandatory END record. Each of the remaining records defines a term and the corresponding coefficient value (in arcseconds). For example, the record

```
" IA       -256.0000"
```

defines a term called IA with a value of −256 arcsec. The additional flags and numbers found in records output by the TPOINT command OUTDAT are ignored.

5. If any invalid record is detected, the pointing model is reset and the rest of the input file is ignored.

6. The guiding corrections are reset to zero whenever this function is called.

cpk**Rot**	**predict rotator angle**	cpk**Rot**

ACTION: Rotator angle required to stabilize view of sky.

GIVEN:

`pars`	`TRANSP*`	transformation parameters
`a`	`double`	roll angle demand (radians)
`b`	`double`	pitch angle demand (radians)
`sky`	`COSYS`	coordinate system (**`ICRS`**, **`GCRS`**, **`CIRS`**, **`AZEL_T`** or **`AZEL_O`**)

RETURNED:

`pa`	`double*`	rotator angle (radians)

RETURNED (function value)**:**

	`int`	true = illegal coordinate system **`sky`**

NOTES:

1. The transformation parameters **`pars`** must be set up before the call. See the functions `cpk`**`Model`**, `cpk`**`Upda`** and `cpk`**`Updt`**.

2. The sign convention for the rotator angle **`pa`** is such that when **`pa`** increases it drives the projection of the rotator on the sky counter-clockwise.

cpk**S2c**	**spherical to Cartesian**	cpk**S2c**

ACTION: Transform spherical coordinates into Cartesian coordinates.

GIVEN:

a	`double`	longitude angle (e.g. α) in radians
b	`double`	latitude angle (e.g. δ) in radians

RETURNED:

x	`double*`	x coordinate
y	`double*`	y coordinate
z	`double*`	z coordinate

NOTES:

1. No validation of the arguments is performed.

2. The Cartesian coordinates returned are for a unit vector.

3. Right-handed coordinate systems are assumed: call using (α, δ), ($-h$, δ) or ($\pi-A$,E). For the h, δ case, the alternative is to call using (h, δ) and then flip y. For the (A,E) case, the alternative is to call using (A,E) and then flip x.

cpk**S2v**	**spherical to vector**	cpk**S2v**

ACTION: Transform spherical coordinates into unit vector.

GIVEN:

a	`double`	longitude angle (e.g. α) in radians
b	`double`	latitude angle (e.g. δ) in radians

RETURNED:

v	`double[3]`	unit vector

NOTES:

1. No validation of the arguments is performed.

2. Right-handed coordinate systems are assumed: call using (α, δ), ($-h$, δ) or ($\pi - A$, E).

cpk**Scomp**	**compare strings**	cpk**Scomp**

ACTION: Compare two strings ignoring trailing whitespace.

GIVEN:

`string1`	`char*`	first string
`string2`	`char*`	second string

RETURNED (function value)**:**

`int`	`0` =	strings are the same

cpk**Sky2enc**	**sky to encoders**	cpk**Sky2enc**

ACTION: Evaluate a specified part of the chain of transformations involved in pointing a telescope:

`ICRS`	astrometric α,δ
↓	
aberration	
↓	
`GCRS`	α,δ with respect to ICRS axes
↓	
precession-nutation	
↓	
`CIRS`	α,δ of date (RA is with respect to CIO)
↓	
Earth rotation	
↓	
`AZEL_T`	topocentric (unrefracted) A,E
↓	
refraction	
↓	
`AZEL_O`	observed (refracted) A,E
↓	
pointing model	
↓	
`ENC`	encoder demands: where to aim the mount

GIVEN:

`jbp`	`int`	0 = normal attitude, else = beyond pole
`pars`	`TRANSP*`	transformation parameters
`v`	`double[3]`	starting vector
`from`	`COSYS`	starting system (**`ICRS`**, **`ICRS`**, **`CIRS`**, **`AZEL_T`** or **`AZEL_O`**)

RETURNED conditionally:

`vgcrs`	`double[3]`	**`GCRS`** vector (~aberrated J2000 mean α,δ)
`vdate`	`double[3]`	**`CIRS`** vector (i.e. CIO-based α,δ of date)
`vtopo`	`double[3]`	**`AZEL_T`** vector (i.e. topocentric A,E)
`vobs`	`double[3]`	**`AZEL_O`** vector (i.e. topocentric A,E)

RETURNED always:

`a`	`double`	roll angle demand (radians)

b `double` pitch angle demand (radians)

RETURNED (function value)**:**
`int` true = illegal starting system **`from`**

NOTES:

1. The starting point may be any of the points in the chain marked as **`ICRS`**, **`GCRS`**, **`CIRS`**, **`AZEL_T`** and **`AZEL_O`**, and the returned directions comprise all those below that point. A common use would be to start with star catalog position (marked as **`ICRS`**) and to use the final **`ENC`** predicted encoder readings to command the mount, discarding all the intervening results.

2. The **`jbp`** argument is almost always false for an altaz, may occasionally be true for a fork or horseshoe equatorial and will be true half the time for mounts that have to be reversed at the meridian such as a GEM. When true, the "pitch" coordinate (i.e. E or δ) will be outside the range $\pm\pi/2$ and the "roll" coordinate (i.e. $180^\circ - A$ or $-h$) will be π different from the nominal value.

3. The transformation parameters **`pars`** must be set up before the call. See the functions `cpk`**`Model`**, `cpk`**`Upda`** and `cpk`**`Updt`**.

4. The transformation neglects proper motion, parallax and light deflection. Using the functions `cpk`**`Upda`** etc., the accuracy of the **`AZEL_O`** to **`ICRS`** portion is about 1 arcsecond for dates between 1950 and 2100, elevations above 25° and V color.

5. The accuracy of the remainder of the transformation, namely the pointing corrections, depends on the size of the model coefficients. For typical values, the additional error is unlikely to be as large as 1 arcsecond, except close to the pole of the mounting.

6. The inverse transformations are performed by the function `cpk`**`Enc2sky`**. The two functions are not precisely complementary, but for typical pointing coefficients should deliver consistency at the 1 arcsecond level.

7. All except the **`ENC`** results are right-handed 3-vectors with axes as follows:

	`CIRS, GCRS, ICRS`	**`AZEL_O, AZEL_T`**
x	$\alpha = 0^h$, $\delta = 0^\circ$	south
y	$\alpha = 6^h$, $\delta = 0^\circ$	east

z	$\delta = 90^\circ$	zenith

8. The starting vector v must be a unit vector except in the **`AZEL_O`** case. The returned vectors **`vdate`**, **`vtopo`** and **`vobs`** are not in all cases guaranteed to be unit vectors. However, the vectors remain close to unit length, so that the successive returned vectors can be passed back into the function without renormalizing.

9. The encoder demand angles are for the case of an equatorial mount and for an altazimuth mount. Hour angle is zero on the meridian and west-positive. Azimuth is zero in the north and increases clockwise as seen from above. The angles may exceed normal ranges, and represent actual mechanical attitude.

10. Which of the conditional vectors are returned depends on the choice of coordinate system for the starting vector:

	GCRS	**CIRS**	**AZEL_T**	**AZEL_O**	**v**
ICRS	✓	✓	✓	✓	
GCRS	×	✓	✓	✓	
CIRS	×	×	✓	✓	
AZEL_T	×	×	×	✓	
AZEL_O	×	×	×	×	

from

11. The returned demand angles are h,δ for the case of an equatorial mount and A,E for an altazimuth mount. Hour angle is zero on the meridian and west-positive. Azimuth is zero in the north and increases clockwise as seen from above. The angles may exceed normal ranges, and represent actual mechanical attitude.

cpk**Tday**	**time of day to days**	cpk**Tday**

ACTION: Time of day from hours, minutes, seconds to fraction of a day.

GIVEN:

`ih`	`int`	hour
`im`	`int`	minute
`s`	`double`	seconds

RETURNED (function value)**:**

`double`	fraction of 1 day

NOTES:

1. No validation of the arguments is performed. The three numbers are simply combined arithmetically, and can be negative or lie outside the conventional ranges.

cpk**Tpt**	**create TPOINT record**	cpk**Tpt**

ACTION: Format a pointing observation for TPOINT analysis.

GIVEN:

`pars`	`TRANSP*`	transformation parameters
`vobs`	`double[3]`	**`AZEL_O`** vector (i.e observed *A*,*E*)
`a`	`double`	roll angle demand (radians)
`b`	`double`	pitch angle demand (radians)

RETURNED:

`string`	`char[]`	formatted record (null terminated)

RETURNED (function value)**:**

	`int`	length of record (not counting the null)

NOTES:

1. No validation of the arguments is performed.

2. The transformation parameters **`pars`** must be set up before the call. See the functions **`cpkModel`** and **`cpkUpda`**. (The parts used are the mount type and the functions of latitude.)

3. It is the caller's responsibility to provide a character array big enough to receive the record. The recommended length is 81 characters.

cpk**Upda**	**update astrometry quantities**	cpk**Upda**

ACTION: Update the astrometric portion of the transformation parameters.

GIVEN:

`ut1`	`double`	UT1 as Modified Julian Date (JD–2400000.5)
`press`	`double`	air pressure (hPa = mb)
`elong`	`double`	longitude (radians, east-positive)
`phi`	`double`	latitude (radians)
`pars`	`TRANSP*`	transformation parameters

RETURNED:

`pars`	`TRANSP*`	updated

NOTES:

1. No validation of the arguments is performed.

2. The difference between UT1 and TT is neglected.

3. The main purpose of this function is to update the aberration, precession and nutation quantities, which change very slowly. In addition, the refraction is renewed to take account of any change in air pressure and, for completeness, the rapidly changing Earth rotation component is also updated.

4. This function need be called only occasionally, say once per slew or even once per night. However, during tracking, continuous updates to the transformation parameters are required, using the `cpk`**`Updt`** function, in order to account for Earth rotation.

5. The pressure **`press`** is the measured value at the telescope, not a sea-level version: at high-altitude sites it will be a smaller number than at low-altitude sites.

6. CPK uses a simple (and extremely fast) refraction algorithm that gives good results for most purposes, especially above 20º elevation and at temperatures not too far from 10ºC. Improved results can be obtained by applying to the supplied pressure **`press`** a correction factor 1.0419 – 0.00380*t, where t is the ambient temperature in degrees C. For

the most demanding applications, a supplementary function called `cpk`**`Refacc`** (licensed separately from CPK) is available that takes full account of color, pressure, humidity, temperature, site location and tropospheric lapse rate and that offers much improved performance at very low elevations. Supplied with the current pressure reading and topocentric (i.e. unrefracted) elevation, `cpk`**`Refacc`** produces an adjusted pressure that can be used in the `cpk`**`Upda`** call in place of the actual reading. Using this adjusted pressure, CPK's fast refraction algorithm then delivers extremely accurate predictions.

cpk**Updt**	**update Earth rotation**	cpk**Updt**

ACTION: Update the rapidly-changing Earth rotation portion of the transformation parameters.

GIVEN:

`ut1`	`double`	UT1 as Modified Julian Date (JD−2400000.5)
`pars`	`TRANSP*`	transformation parameters

RETURNED:

`pars`	`TRANSP*`	updated

NOTES:

1. No validation of the arguments is performed.

2. **It is the responsibility of the application to distinguish between UT1 and UTC.**

cpk**V2s**	**vector to spherical**	cpk**V2s**

ACTION: Transform vector to spherical coordinates.

GIVEN:

v	`double[3]`	vector

RETURNED:

a	`double*`	longitude angle (e.g. α) in radians
b	`double*`	latitude angle (e.g. δ) in radians

NOTES:

1. The vector **v** does not have to be a unit vector: only its direction is significant.

2. Right-handed coordinate systems are assumed: α = **a**, h = −**a**, A = π−**a**.

8 Bibliography

McCarthy, D.D. & Petit, G. (eds) 2004, *IERS Conventions (2003)*, IERS Technical Note 32, Frankfurt am Main: Verlag des Bundesamts für Kartographie und Geodäsie

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

Wallace, P.T. & Capitaine, N. 2006, *Precession-nutation procedures consistent with IAU 2006 resolutions*, Astronomy & Astrophysics, **459**, 981-985