LSST Image Trailing

LSST/PTW/003

P.T.Wallace

CHANGE RECORD

Issue	Date	Reason for change
1	January 12, 2005	New document
2	January 31, 2005	Additional material
3	February 2, 2005	Additional material
4	February 4, 2005	New logo
5	February 7, 2005	Table 3 added
6	August 12, 2005	Caption amendment
7	November 6, 2006	Blank page removed
8	January 22, 2007	More on simulator software

Contents

1 Introduction

This report considers unwanted image motions in the LSST focal plane. Even in the absence of wind-shake and other random tracking disturbances, such motions can occur systematically for several reasons:

i. Variations in the apparent directions of the stars during exposures.
ii. Variations in the apparent directions of stars over months and years.
iii. Failure of the instrument rotator to follow the rotating field.
iv. A pointing model that fails to describe the actual behavior of the telescope

Type (i) motions are dominated by differential refraction effects. They are a feature of the view of the star-field as seen from any ground-based observatory and there is no practical way (rubber-sheet detectors?) to eliminate them. They are one of the limits on exposure duration and on minimum usable elevation.

Type (ii) motions, which happen principally because of precession, may not matter, depending on the way the LSST observations are reduced and archived. They do mean, however, that exposures of the same star-field taken under identical conditions but years apart, though untrailed as individual exposures, may display a relative rotation that is undesirable.

Type (iii) motions are a crucial aspect of LSST performance. The impeccable star images at the center that the pointing and tracking software will aim to deliver must be matched by equally sharp images at the edge of the 3.5° field.

Type (iv) motions are simply the differential manifestation of telescope pointing errors. They can be minimized by spending enough time modeling the telescope and performing frequent recalibrations (or continuous recalibration—see the notes at the end of Section 4), but all telescopes have some limit to their blind pointing accuracy. This is another crucial aspect of LSST performance, if there is to be no guiding during exposures.

The report begins by describing the tools that have been used to explore some of these issues (Section 2), then goes on to look at rotator control (Section 3) and finally describes the effects of pointing calibration errors in a realistic LSST model (Section 4).

2 LSST Exposure Simulator

The tool used to study the aspects of LSST performance considered in this report is a C program that simulates the tracking of star fields. The telescope-control algorithm is based on the proprietary TCSpk and SLALIB/C software libraries and is the one used on the SOAR and LBT telescopes. However, any other control algorithm could be used instead and would give identical results, as long as it (i) correctly performs the positional-astronomy transformations to transform catalog star position to telescope-mount encoder demands and (ii) correctly (and rigorously) applies the set of pointing corrections peculiar to the telescope and mount in question. But although most of the key results could have been obtained with a rather simpler program than the one actually used, the advantage of the latter is that it exploited software that already existed and is likely eventually to be used in the LSST telescope control system.

2.1 The simulator program

The LSST Exposure Simulator has the following structure:

- Create a telescope control system for a given site and telescope.
- Set up *two* pointing models:
    - The *nominal* model, used to generate the pointing/tracking demands.
    - The *true*, but in practice unknown, underlying model.
- Track a (fictitious) star, using the nominal pointing model and a choice of several rotator-control algorithms
- At the beginning of the track, create a fixed pattern of (fictitious) stars that correspond to sample *x,y* positions in the focal plane.
- During the track, knowing where the telescope is currently being asked to point, predict the *x,y* positions of the images of the star pattern.
- Record the changing *x,y* positions for subsequent analysis.

The assumed site for all the simulations in the present report is Cerro Pachón. The pointing model is one from Gemini South and is quite typical: see Table 1.

Table 1: Nominal pointing model

term	*coeff*	*meaning*
IA	**-157.5340**	**azimuth index error**
IE	**-520.8787**	**elevation index error**
PEE	230.1083	elevation scale error
TF	**-7.3000**	**Hooke's-law tube flexure**

HASA2	4.1144	miscellaneous
HACA2	9.5761	miscellaneous
HACA4	0.6496	miscellaneous
HASA10	0.5662	miscellaneous
HESA	-3.6480	miscellaneous
HESA3	2.1142	miscellaneous
HECA3	1.6600	miscellaneous
HESA5	0.5423	miscellaneous
HECA5	0.4097	miscellaneous
NPAE	**-6.3266**	**az/el nonperpendicularity**
CA	**51.6312**	**OTA/el nonperpendicularity**
AW	**19.4365**	**azimuth axis tilt west**
AN	**-24.2447**	**azimuth axis tilt north**
	arcsec	

The terms in bold type are the generic seven that are always expected. These are the terms that are varied in the experiments reported in Section 4.

The test pattern consists of a star on the rotator axis and (for the LSST simulations) eight distributed around the edge of a 3.5° circular field.

To introduce the computational methods used, a listing of the simulator source code (not necessarily identical to that supplied to the LSST project, but representative of it) is included as Appendix A.

2.2 Configuring the simulator program

The simulator program is a highly experimental tool that was designed to get reliable and precise results quickly and with the utmost flexibility. There is a noticeable absence of convenience features, each type of simulation involving the compilation of a new version of the program. The program lacks even a configuration-file interface (let alone a GUI). On the other hand, most simulations involve very small changes, often merely the editing of the pointing model files. The output is an alphanumeric listing suitable for subsequent analysis and plotting: see 2.4.

The bulk of the code is in a file **`trails.ccc`**. This begins with a set of macro definitions that control the behaviour of the program but which are in fact merely illustrative dummies, rendered inactive by means of an **`#if 0 ... #endif`** construct. Individual versions of the simulator are generated by preparing a file that first contains live definitions of these macros and then incorporates the common code with an **`include`** statement. Here is an example, configured to test different rotator control algorithms and currently set to use the “field-oriented ray-tracing” option:

```
#include <stdio.h>
#include <string.h>
#include <slalib.h>
#include "tcs.h"
#include "tcsmac.h"

int main ( )

#define YES 1
#define NO 0

#define VERBOSE YES           /* Full reporting */
#define REP_HEAD YES          /* Head the report */

#define MOUNT ALTAZ           /* Mount type */
#define RLOCN OTA             /* Instrument rotator location */
#define SITE_LAT -30.24075    /* Site latitude (deg) */
#define SITE_HM 2738.0        /* Site elevation (m) */
#define PMB 730.0             /* pressure (hPa) */

#define FIELD 3.5             /* Field width (deg) */
#define F_CIRC YES            /* Circular (rather than square) field */
#define FOCAL_L 10499.8       /* Focal length (mm) */

#define TPS 5L                /* Ticks per second */
#define T_SLOW (60L*TPS)      /* Ticks per slow update */
#define T_MED (5L*TPS)        /* Ticks per medium update */

#define PM_NOMINAL "lsst.nom" /* Nominal pointing model */
#define PM_TRUE "lsst.tru"    /* True pointing model */
#define DXI 0.0               /* Rotator axis error in xi (arcsec) */
#define DETA 0.0              /* Rotator axis error in eta (arcsec) */
#define PM_ENABLED YES        /* Pointing models enabled */

#define RMA_PAR NO            /* Control rotator using parallactic angle */
#define RMA_KECK NO           /* Control rotator using Keck formula */
#define RMA_SLIT NO           /* Control rotator for slit */
#define RMA_FIXED NO          /* Rotator motion disabled */

#define LEN_TEST (60L*TPS)    /* Duration of test (ticks) */
#define LOG_INT (1L*TPS)      /* Ticks per line of report */

#define REP_DIST NO           /* Whether to report distortions and shifts */
#define REP_TRAIL NO          /* Whether to report trailed image positions */
#define MAG 500               /* Trail magnification */
#define N_TOP YES             /* Whether north at the top */
#define E_LEFT YES            /* Whether east at the left */
#define ONE_STAR NO           /* One star or multiple stars */

#include "trails.c"
```

The various items are as follows:

- **VERBOSE** controls the brevity of the report. For human-readable reports use `YES`, otherwise `NO`.
- **REP_HEAD** controls whether the run parameters are to be listed at the start of the report. For human-readable reports use `YES`, otherwise `NO`. `REP_HEAD YES` is overridden by `VERBOSE NO`.

- **MOUNT** selects the mount type. EQUAT, ALTAZ and GIMBAL are all accepted. For LSST use, only ALTAZ is ever required.
- **RLOCN** defines where the instrument rotator is located. OTA, NASMYTH_L, NASMYTH_R, COUDE_L and COUDE_R are all accepted. For LSST the setting is always OTA.
- **SITE_LAT**, **SITE_HM** and **PMB** are doubles that describe the observing site, being the latitude (degrees), elevation (meters) and local air pressure (hPa or mb).
- **FIELD** defines the field width in degrees (double), and **F_CIRC** selects between a square (NO) and circular (YES) field. **FOCAL_L** is the focal length (mm, double).
- **TPS**, **T_SLOW** and **T_MED** control the TCS simulation, respectively the number of ticks per second, the number of ticks per "slow" update and the number of ticks per "medium" update. They can be set to values that reduce the amount of computation without affecting the results significantly. They are of type long int; example values are 1L, (60L*TPS) and (5L*TPS), setting the values to 1s, 1m and 5s.
- **PM_NOMINAL** and **PM_TRUE** are the file names for the "nominal" and "true" pointing models.
- **DXI** and **DETA** specify the inaccuracy in the knowledge of the rotator axis position. They are double values in arcseconds.
- **PM_ENABLED** is a quick way of turning on (YES) and off (NO) the pointing corrections.
- **RMA_PAR**, **RMA_KECK**, **RMA_SLIT** and **RMA_FIXED** specify the method of calculating the instrument rotator position angle. To use parallactic angle, set values of YES, NO, NO and NO. To use the Keck formula, set values of NO, YES, NO and NO. To use the TCSpk raytracing method but slit-optimized, set values of NO, NO, YES, NO. To use the best available method, namely the field-optimized TCSpk raytracing method, set values of NO, NO, NO and NO. To fix the rotator, set values of NO, NO, NO and YES.
- **LEN_TEST** and **LOG_INT** specify the duration of the test (in ticks – see above) and the number of ticks per line of report. They are long int values. With the 1s ticks suggested earlier, a 2-hour simulation with a report every minute requires values of (2L*60L*60L*TPS) and (60L*TPS).
- **REP_DIST** selects whether to report distortions and shifts (YES) or not (NO).
- **REP_TRAIL** selects whether to report trailed image positions (YES) or not (NO).

- **MAG** is an `int` (for example `1000`) that specifies the desired trail magnification. It should be set by trial and error so that the trail lengths are a pleasing match with respect to the star spacings.
- **N_TOP** and **E_LEFT** allow the picture to be flipped, in order to facilitate comparisons with published work. YES and YES place north and east at the top and left, NO and NO place south and west at the top and left, and so on.
- **ONE_STAR** selects whether the simulation expects only one star position (YES) or a list of stars (NO). With a setting NO, to invoke a version of the simulator for a sky position of $(h = 45^\circ, \delta = -10^\circ)$ might require a command like "`test > report.txt`" and the response "`45 -10`" once the program is running. With a setting YES, to invoke a version of the simulator for a list of stars might require a command like "`test < stars.dat > report.txt`".

2.3 Controlling the simulator output

This section contains samples of the possible outputs and how they are turned on and off.

For human-readable reports, headings can be enabled by setting both **VERBOSE** and **REP_HEAD** to YES:

```
Test circumstances:

  mount type is equatorial
  site longitude and height = +30 deg, 0 m
  pressure = 1013.25 hPa
  field is 0.500000 deg (radius)
  focal length = 12000 mm
  ticks per second = 1
  slow update every 60.000000 sec
  medium update every 5.000000 sec
  pointing models (nominal, true) are in files xptl.mod, xptl.mod
  rotator fixed
  total duration of test in seconds = 7200
  seconds per report = 60
  trails will be logged as shifts and distortions
  trails will be logged as dX,dY magnified x1000, with
  north at the top and west at the right
  results for multiple stars
```

The star positions (see 2.4.1) are logged if **VERBOSE** was set to YES:

```
<  4.675055 -10.854400>
```

The distortions and fits report (see 2.4.2) can be included by setting **REP_DIST** to YES:

```
   0.00  0.00000  -0.00000   1.00000   1.00000   0.00000  -0.00000   -10.83
```

The trail report (see 2.4.3) can be included by setting **`REP_TRAIL`** to `YES`:

```
  -52.359855   -52.359997  +0.000000  +0.000000 -1 -1 00 00 00.00  164.7  47.9
```

The trail statistics report is included only if **`VERBOSE`** has been set to `YES`:

```
Max trail = 0.592 arcsec, RMS = 0.236 arcsec, mean el = 48.0.
```

2.4 Interpreting the simulator output

Several of the possible output lines are intended to be human-readable rather than for machine interpretation. The three that are most likely to provide numerical inputs into further stages of processing and display are the star position record (2.4.1), the distortions and fits record (2.4.2) and the trail record (2.4.3).

Turning the numerical tabulations into graphical displays is left to the user. Suitable tools include MS-Excel (which was used for Figure 12 for example), MATLAB and so on. Most of the (relatively crude) plots in the present report were made using specially-written Fortran programs calling a GKS-based drawing package.

2.4.1 Star position record

```
<  4.675055 -10.854400>
```

The two numbers are simply the hour angle and declination of the field center, both in degrees. The position is *observed*, meaning as affected by refraction.

2.4.2 Distortions and fits record

```
   0.00  0.00000  -0.00000   1.00000   1.00000   0.00000  -0.00000   -10.83
```

This record (rarely used) expresses the instantaneous image positions as an affine transformation of the starting positions. The eight numbers are as follows:

- Field 1: time in seconds relative to start of exposure.
- Fields 2-3: overall x,y offsets.

- Fields 4-5: overall x and y scale factors.
- Field 6: nonperpendicularity between x and y axes (degrees).
- Field 7: field rotation (degrees).
- Field 8: pitch coordinate (degrees).

2.4.3 Trail record

```
 -52.359855   -52.359997  +0.000000  +0.000000 -1 -1 00 00 00.00  164.7  47.9
```

This record shows the changing position of one of the stars in the 9-point test pattern. The 11 numbers have the following meanings:

- Fields 1-2: the x,y coordinates of the image. The position is the sum of the x,y position of the image at the start of the exposure and the magnified Δx,Δy offset of the current position.
- Fields 3-4: the Δx,Δy offset in mm (unmagnified).
- Fields 5-6: coordinates identifying which star of the nine is being described. The values are x,y relative to the center, each value being −1, 0 or +1.
- Fields 7-9: the time since start of exposure in hours, minutes and seconds.
- Fields 10-11: the (observed) azimuth and elevation in degrees.

2.5 Example simulation

In 1967, Ira S. Bowen published[1] plots of star trails for the Palomar 1.2m Schmidt camera, to demonstrate the effects of differential refraction for exposures centered on the meridian but at different declinations. The plot for the $\delta = 0$ case is shown in Figure 1:

[1] Bowen, I.S., *Future Tools of the Astronomer*, Quarterly Journal of the Royal Astronomical Society, Vol. 8, Pages 9-22, Figure 1 (1967)

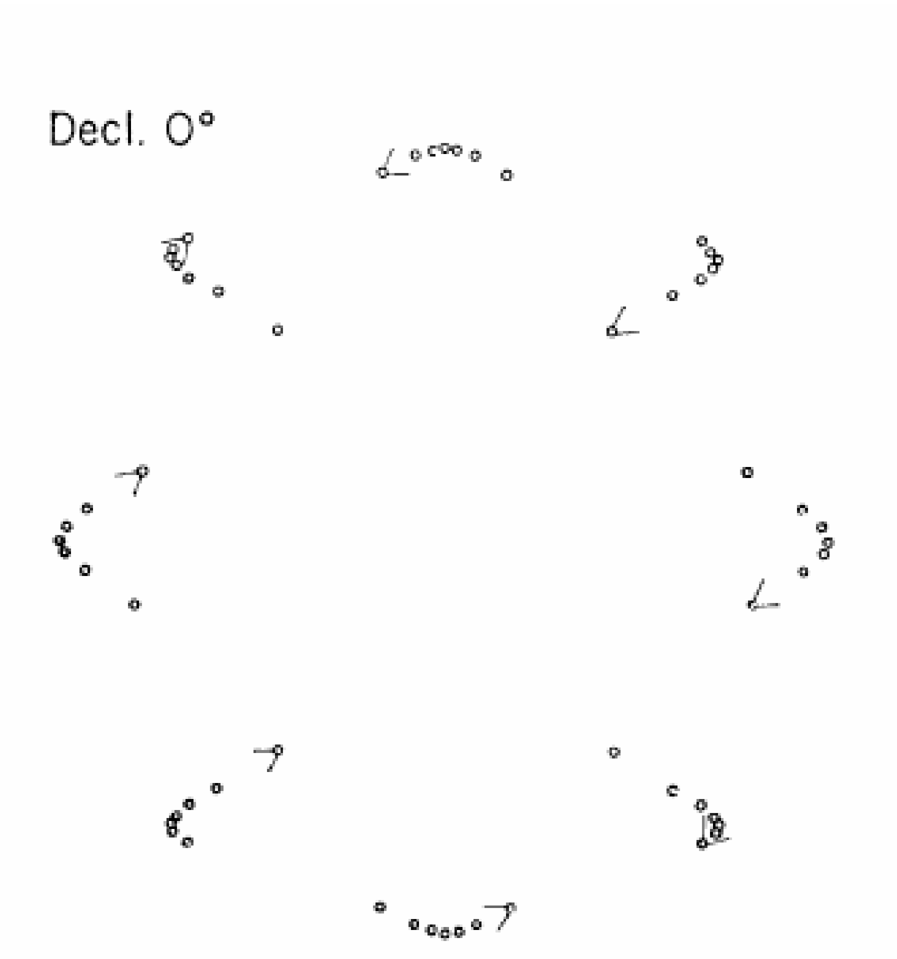


Figure 1: I.S.Bowen's 1966 demonstration of a Palomar Schmidt exposure at $\delta = 0$.

To produce a similar plot using the trail simulator, we start with the following parameter settings, taken from the file **`bowen.c`**:

```
#include <stdio.h>
#include <string.h>
#include <slalib.h>
#include "tcs.h"
#include "tcsmac.h"

int main ( )

#define YES 1
#define NO 0

#define VERBOSE NO           /* Full reporting */
#define REP_HEAD YES         /* Head the report */

#define MOUNT EQUAT          /* Mount type */
#define RLOCN OTA            /* Instrument rotator location */
#define SITE_LAT 30.0        /* Site latitude (deg) */
#define SITE_HM 1892.0       /* Site elevation (m) */
#define PMB 819.0            /* pressure (hPa) */

#define FIELD 2.0            /* Field width (deg) */
#define F_CIRC YES           /* Circular (rather than square) field */
#define FOCAL_L 3048.0       /* Focal length (mm) */

#define TPS 1L               /* Ticks per second */
#define T_SLOW (60L*TPS)     /* Ticks per slow update */
#define T_MED (5L*TPS)       /* Ticks per medium update */

#define PM_NOMINAL "bowen.mod" /* Nominal pointing model */
#define PM_TRUE PM_NOMINAL     /* True pointing model */
#define DXI 0.0              /* Rotator axis error in xi (arcsec). */
#define DETA 0.0             /* Rotator axis error in eta (arcsec). */
#define PM_ENABLED YES       /* Pointing models enabled */

#define RMA_PAR NO           /* Control rotator using parallactic angle */
#define RMA_KECK NO          /* Control rotator using Keck formula */
#define RMA_SLIT NO          /* Control rotator for slit case */
#define RMA_FIXED YES        /* Rotator motion enabled */

#define LEN_TEST (8L*60L*60L*TPS) /* Duration of test (ticks) */
#define LOG_INT (60L*60L*TPS)     /* Ticks per line of report */

#define REP_DIST NO          /* Whether to report distortions and shifts */
#define REP_TRAIL YES        /* Whether to report trailed image positions */
#define MAG 500              /* Trail magnification */
#define N_TOP YES            /* Whether north at the top */
#define E_LEFT NO            /* Whether east at the left */
#define ONE_STAR YES         /* One star or multiple stars */

#include "trails.ccc"
```

The specified pointing model in this case, **`bowen.mod`**, is null, meaning in particular that the polar axis is set to the true (rather than refracted) pole.

When the program is run, the file produced consists entirely of trail records (see 2.4.3). If this is read into MS-Excel using “import foreign data” and the first two columns selected, the “XY (Scatter)” plot can be used to produce a graph like Figure 2:

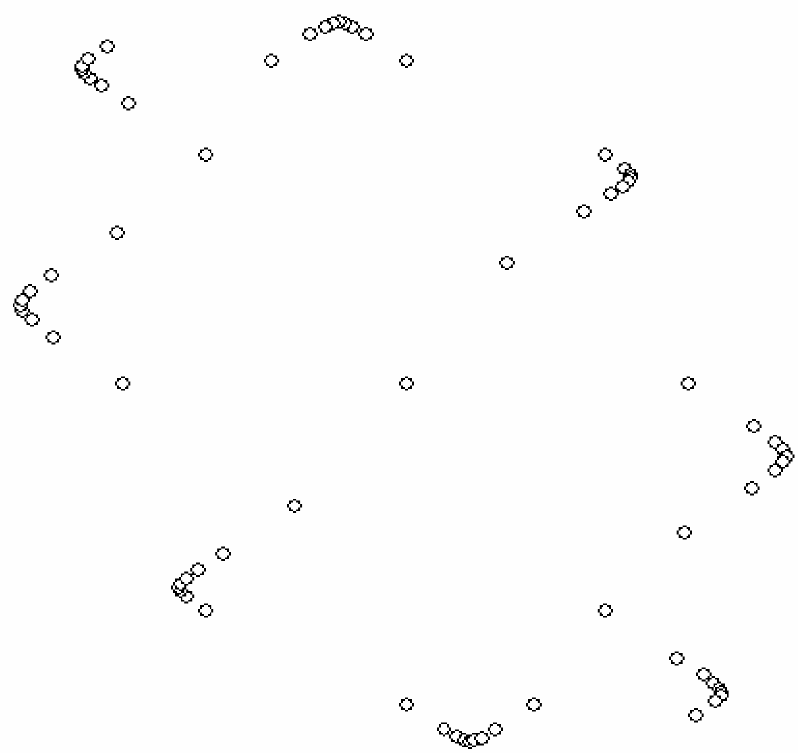


Figure 2: Equivalent Palomar Schmidt predictions made with the trail simulation program.

3 Choice of Rotator Control Algorithm

In an altazimuth-mounted telescope with a focal plane fixed to the optical telescope assembly (OTA)—i.e. prime focus, Newtonian, Cassegrain etc.—the field rotates as the telescope tracks across the sky. The rotation can be compensated by mounting the instrument on a rotator driven continuously to follow the field. On most such telescopes the command signal is derived from the parallactic angle.[2] This method has two disadvantages:

- The north reference is towards the celestial pole of date, which slowly moves with respect to the stars as a result of precession, whereas the ICRS pole would be preferable as it would match star catalogs and not move.
- Various deficiencies in the mounting, especially non-perpendicularities between the two axes and between the elevation axis and the OTA, change the field rotation slightly with respect to the parallactic angle. This is particularly apparent near the zenith, where small pointing corrections require large azimuth changes that in turn cause changes to the field orientation.

With its exceptionally large field, LSST will be particularly sensitive to field rotation effects: to keep imaging trailing below 0.1 arcsec, the rotator must track to about 3 arcsec, only 1–2 orders of magnitude less than the accuracy of the tracking of the telescope itself. To achieve consistent field orientations from year to year and to minimize trailed images at the edge of the field, rotator control using the parallactic angle is unlikely to be acceptable.[3] Two improved methods are available:

1. Some recent control systems, such as the Keck telescopes, refer the field angle to north in the chosen tracking frame (usually ICRS) and apply field rotation corrections using formulas that take into account the pointing coefficients as well as refraction (Wallace 1988[4]).

[2] Parallactic angle is the angle at the star between the direction to the celestial pole and the direction to the zenith. Standard spherical-trig formulas give parallactic angle as a function of star hour angle and declination, and site latitude.
[3] ESO discovered this the hard way, finding that on VLT exposures a few degrees from the zenith stars at the center of the field were trailed into "sausages", short arcs of circles centered on the guide star—see Spyromilio, J. et al., *Commissioning of the Unit Telescopes of the VLT,* ESO Messenger, 98, pp21-24 (1999).
[4] Wallace, P.T., *Pointing and Tracking Algorithms for the Keck 10-Meter Telescope,* Proc. 9th Santa Cruz Summer Workshop in Astronomy and Astrophysics, July 13- 24, 1987, Lick Observatory. Ed. Robinson, L.B., Publisher, Springer-Verlag, New York, NY, pp691-706 (1988)

2. A bolder technique (Wallace 2002[5]), used by Gemini, SOAR and LBT, is to take two sample [*x,y*]s in the focal plane and to perform a rigorous and exhaustive raytrace back through the entire pointing model up to the sky in order to predict to what [α,δ] each of the focal-plane points corresponds. The position angle between them is compared with that desired for the line through the two [*x,y*]s (called the *instrument principal direction*, IPD), and appropriate corrections sent to the instrument rotator. For example, the two sample points might on some telescopes be at the ends of a spectrograph slit, and the desired sky position angle 0°, while tracking in ICRS. The corrections sent to the rotator will then keep the slit aligned accurately north-south in ICRS coordinates.

The effect of pointing-model corrections is to shift, and as a side-effect rotate, the field. The effect of differential refraction is to distort the field geometry slightly, in a way that changes during the exposure and that may have a rotation component. If the instrument were a slit spectrograph, Option 2 (above) clearly delivers the optimum result: as long as the instrument principal direction is chosen to be the slit, then any image motion due to differential refraction can only lie along the slit. However, in the case of a 2D imager, such as LSST's, it turns out that with Option 2 the exposure in general retains some residual overall rotation even though along the chosen line the image motion due to differential refraction is entirely longitudinal. The solution is to make predictions for two directions at right angles—the camera's *x*- and *y*-axes respectively would be an obvious choice—and then to take the mean of the resulting two slightly different rotator angles. (The principle extends to averaging over more than two directions, but the results are not significantly changed: any two sample directions at right angles are enough.)

The LSST exposure simulator described in Section 2 was used to demonstrate the different rotator-control algorithms. All of the plots assume the usual Gemini South pointing model and Cerro Pachón conditions. The trails are magnified by different amounts depending on the plot.

[5] Wallace, P.T., *A Rigorous Algorithm for Telescope Pointing,* Advanced Telescope and Instrumentation Control Software II, Ed. Lewis, H., Proc. SPIE, 4848, pp125-136 (2002).

Figure 3 shows what can happen when the parallactic angle is used to control the instrument rotator. The exposure is an hour long, to illustrate the changing image positions over such a period rather than the trailing that would occur on an individual short exposure. The field transits 1.5° from the zenith. Most of the trailing comes from the pointing corrections that are being applied in order to track the central star accurately and the consequent field rotation**.**

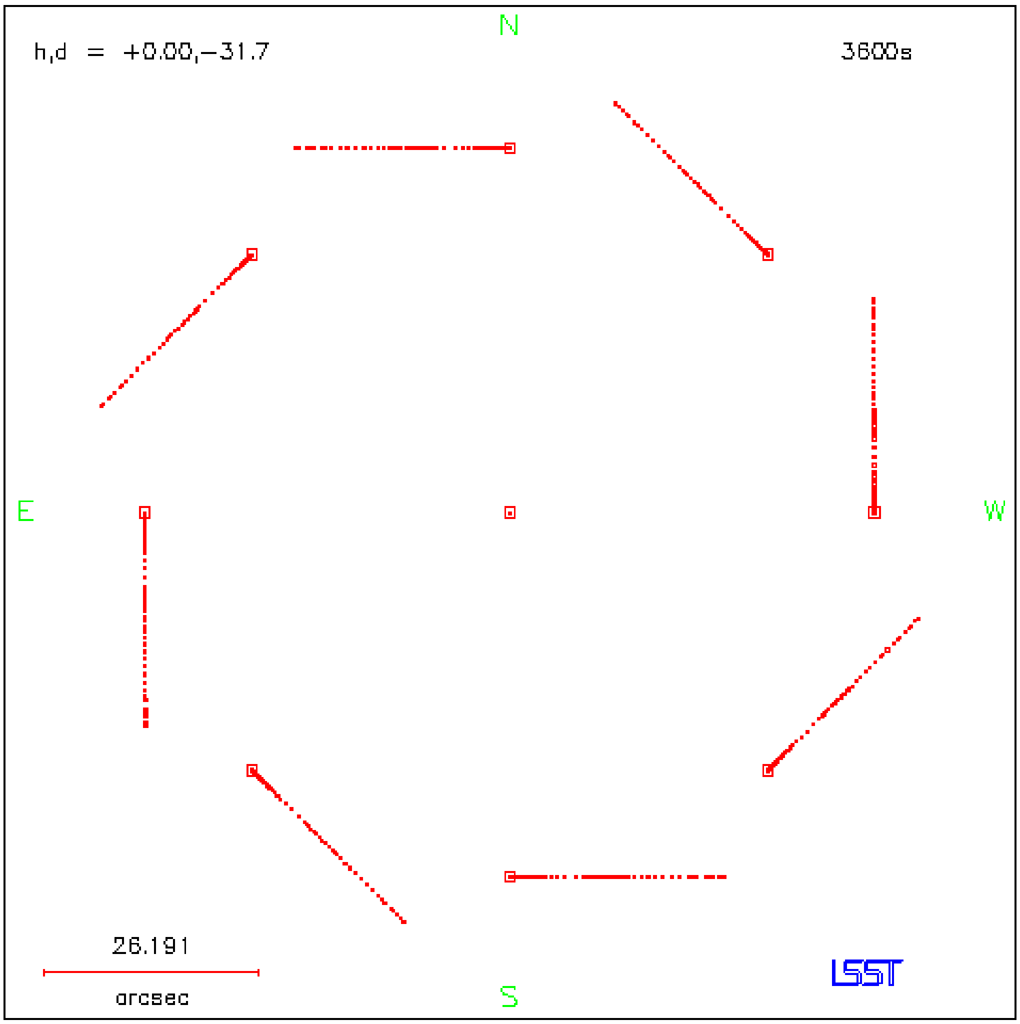


Figure 3. A 1-hour exposure transiting 1.5° from zenith, using parallactic angle to control the instrument rotator.

Figure 4 shows the dramatic improvement from using one of the improved methods, in this case the Keck formula approach. The trails have all but been eliminated. (The slit-optimized raytrace option gives an equally good result for this case, while the field-optimized raytrace option is just detectably better, reducing the maximum trail by 1 mas.)

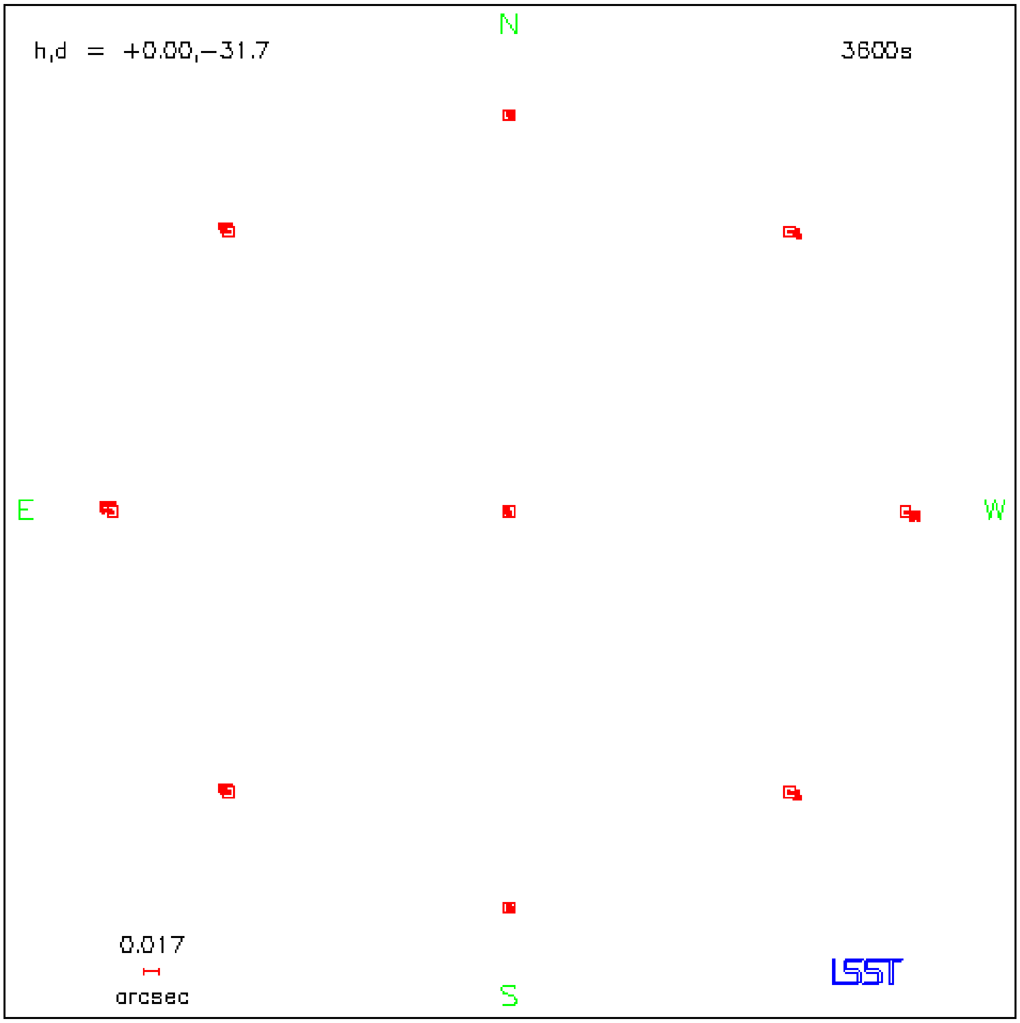


Figure 4. The same as the previous Figure but using the Keck formulas to provide rotator control.

Away from the zenith, the pointing-model effects are far less pronounced. Figure 5 is for a field low in the west, centered on 4^h hour angle and 0° declination. The 1-hour exposure covers an elevation range of 32°–19°, and rotator control is via parallactic angle.

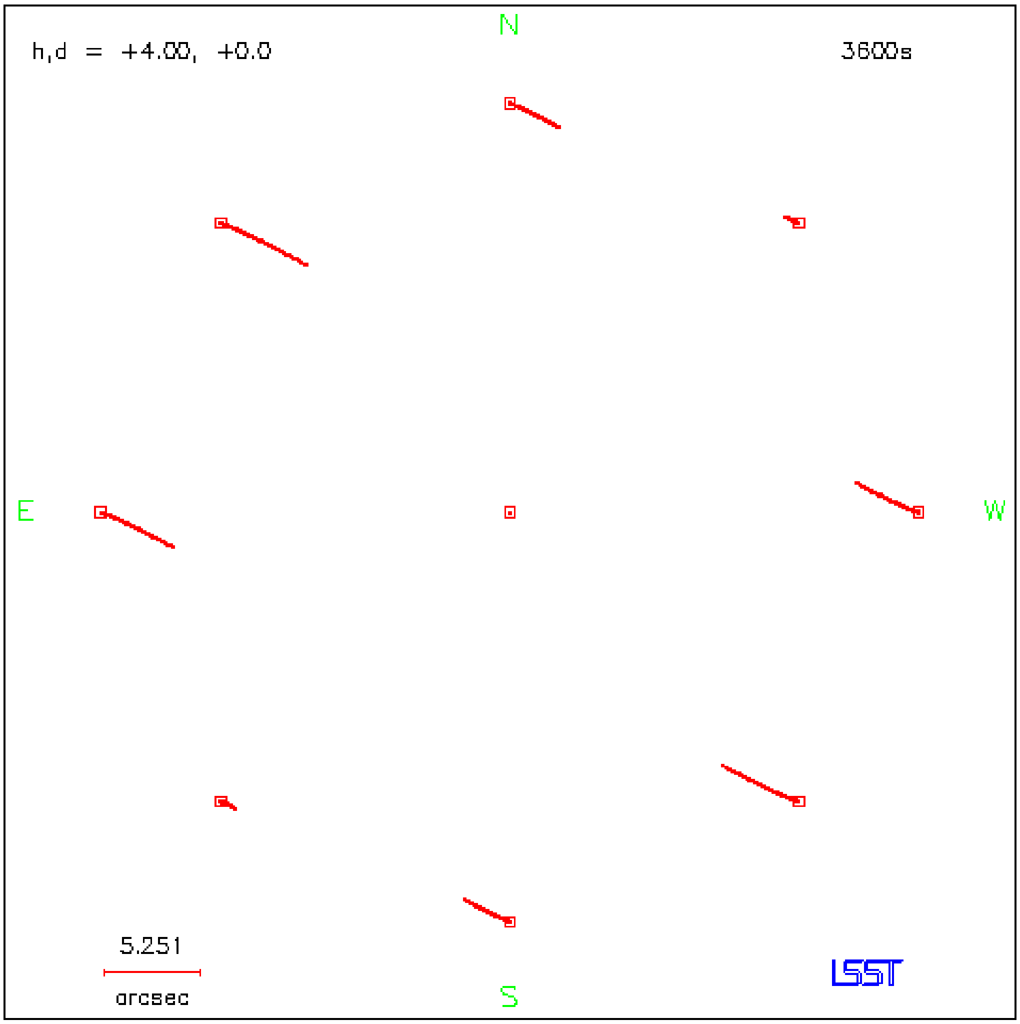


Figure 5: A field low in the west, with rotator control via parallactic angle.

For this field, the trailing is mainly the result of refraction, with the pointing model doing little to rotate the field. Consequently, the different methods for controlling the instrument rotator offer comparable performance, changing the character of the trailing somewhat without much affecting the maximum trailing. Figure 6 demonstrates the Keck algorithm, at 6.002 arcsec slightly worse than the much simpler parallactic-angle approach. The slit-optimized ray-tracing method, seen in Figure 7, is only slightly better, achieving 5.979 arcsec; note the way the rotation has been selectively eliminated for the north and south stars. However, the field-optimized ray-tracing method, Figure 8, does finally manage to do better than the parallactic-angle method, but only just, achieving a 5.125 arcsec maximum trail.

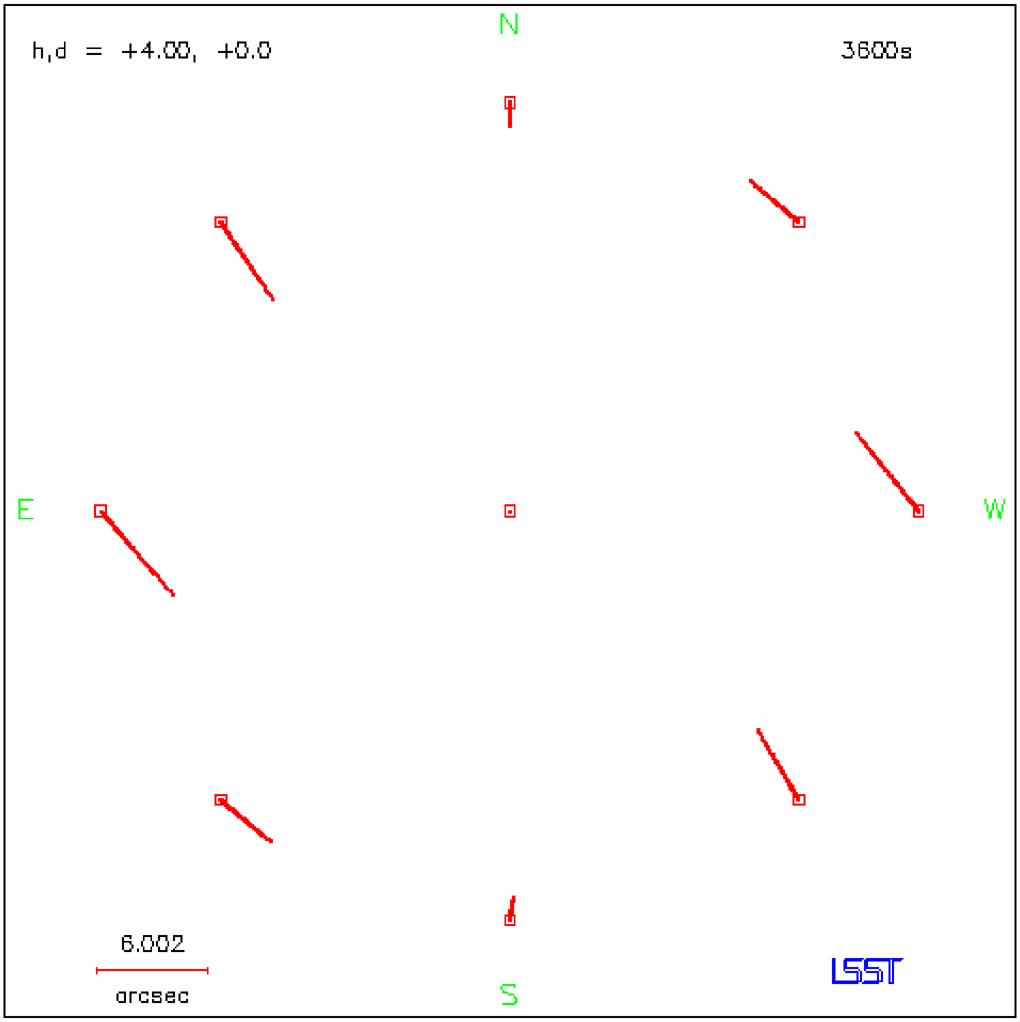


Figure 6: The same, but using the Keck algorithm for controlling the rotator.

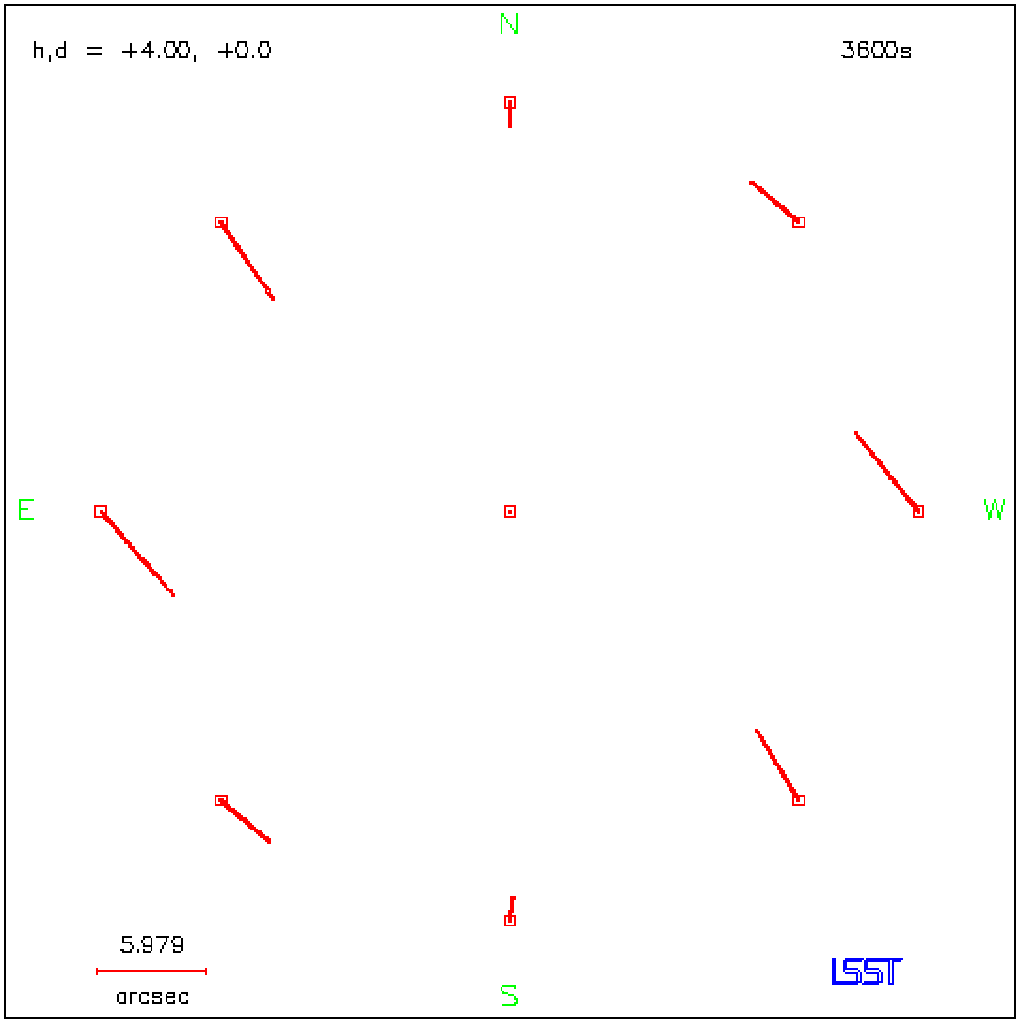


Figure 7: The same but using the ray-tracing method, slit-optimized.

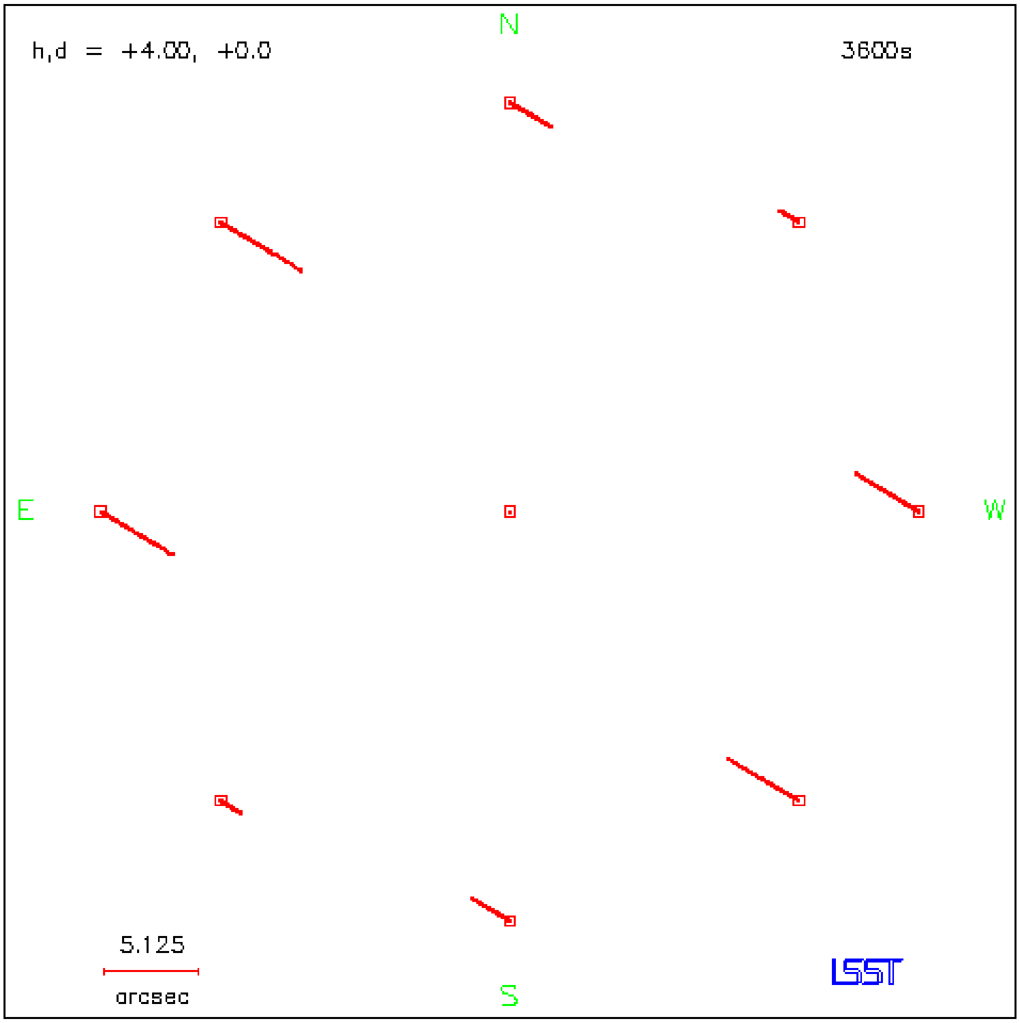


Figure 8: As before, but field-optimized.

For the lowest elevations there is significant image trailing even in a short exposure. Figure 9 shows an 1-minute exposure at about 16° elevation. Despite using field-optimized rotator control, trailing of up to 0.309 arcsec occurs.

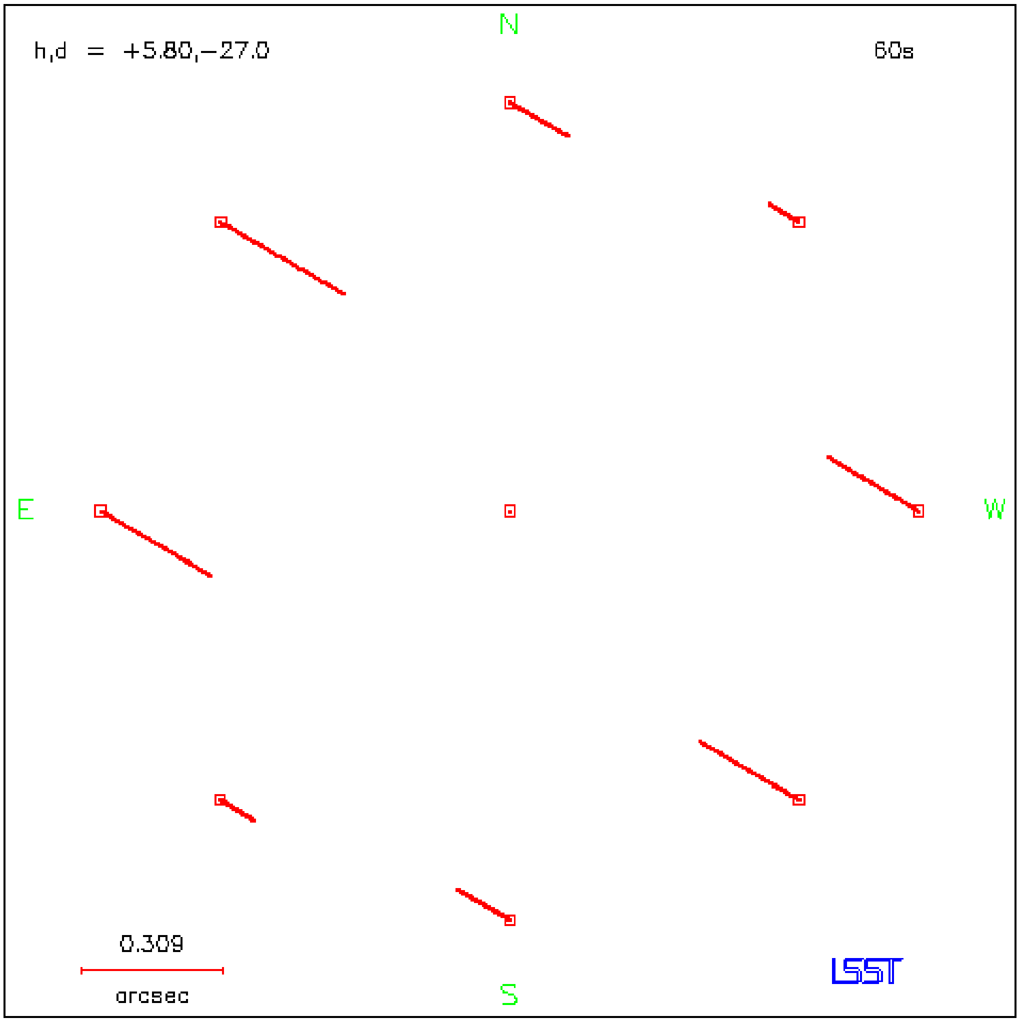


Figure 9: Low-elevation 1-minute exposure.

Low elevation *per se* does not mean there will be image trailing. Figure 10 shows a field at about $\delta = +45°$ in transit at 15° elevation. The trailing in a 1-minute exposure is only 53 mas.[6] If, instead of being in transit, the field is setting and therefore changing rapidly in elevation, the effects are much larger—see Figure 11.

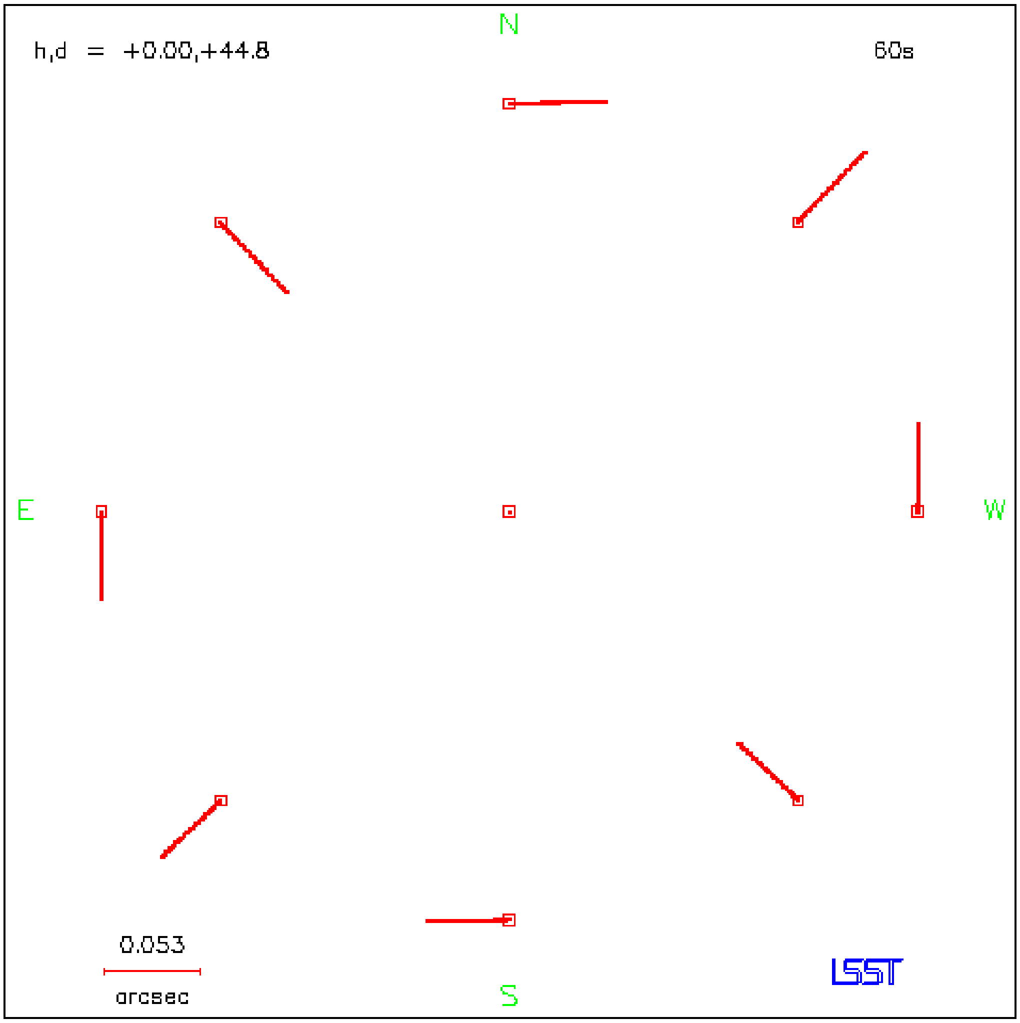


Figure 10: A field north of the zenith in transit at the 15° elevation limit.

[6] The amount of trailing is not to be confused with the amount of differential refraction, which is several arcseconds at this low elevation: the trailing is a consequence of the *change* in differential refraction during the exposure.

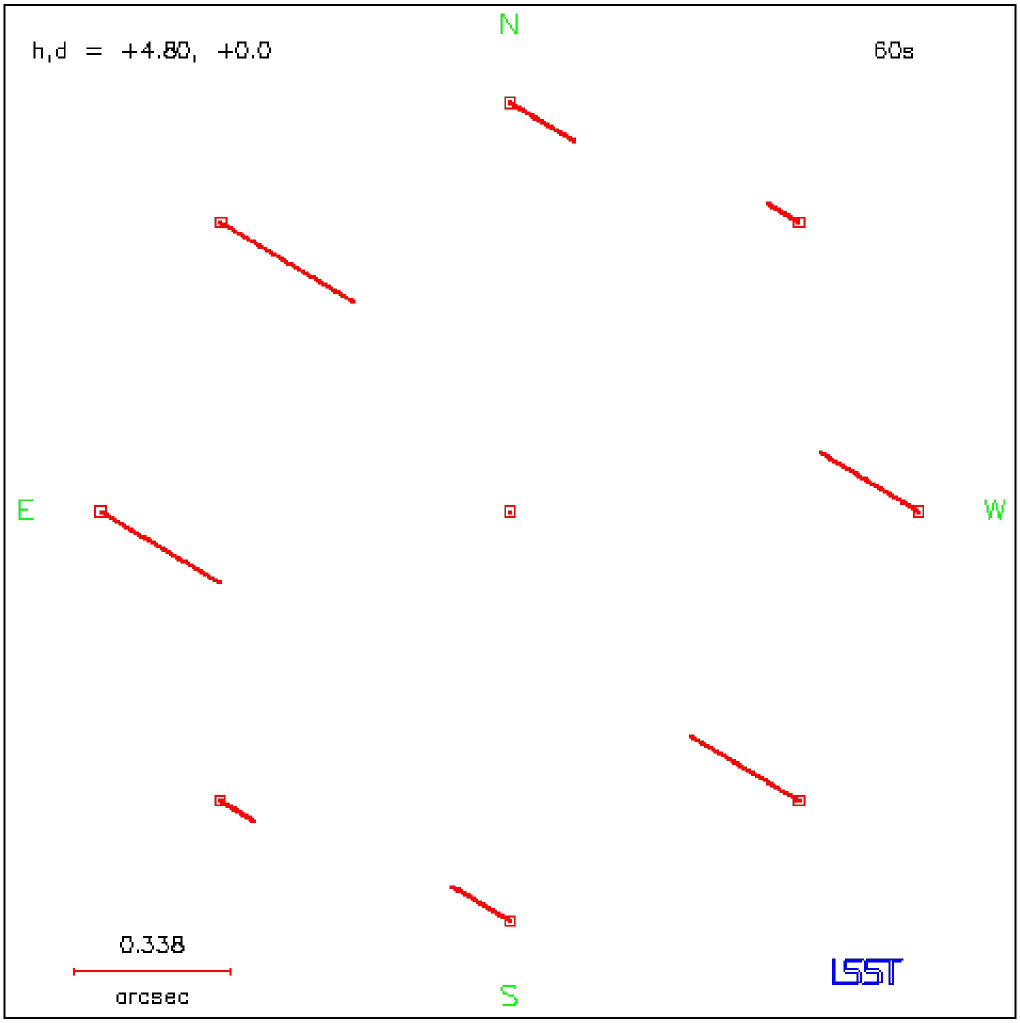
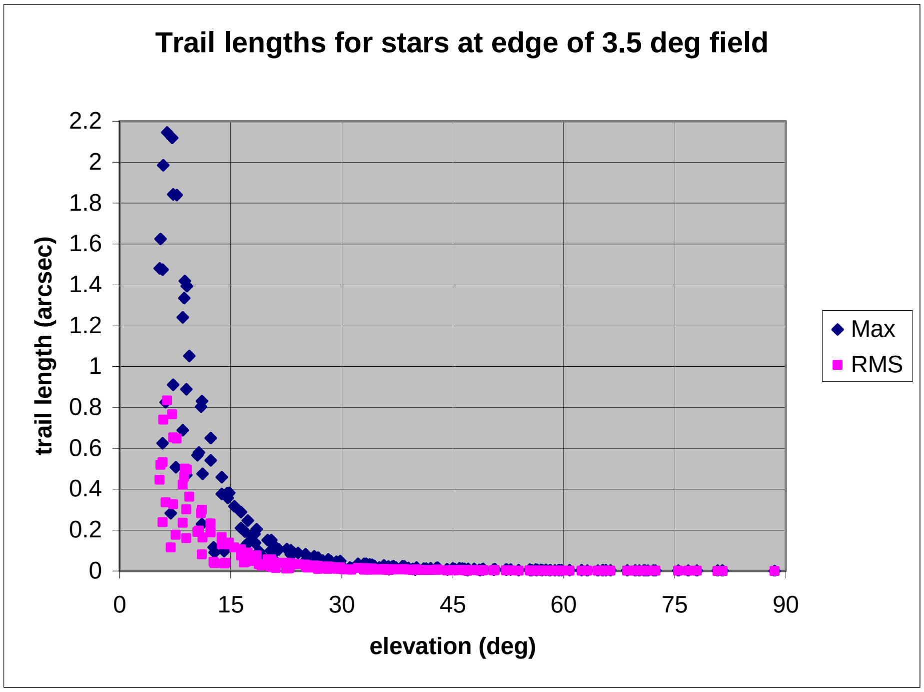


Figure 11: Like the previous Figure, a 1-minute exposure at the 15° elevation limit, but this time for a field that is setting.

However, the worst trailing due to differential refraction alone does indeed occur at low elevations, as can be seen in Figure 12, where the maximum and RMS trail lengths for the fields listed in Appendix B are plotted. At the nominal 15° elevation limit, trails of up to 0.4 arcsec can occur in a 1-minute exposure.



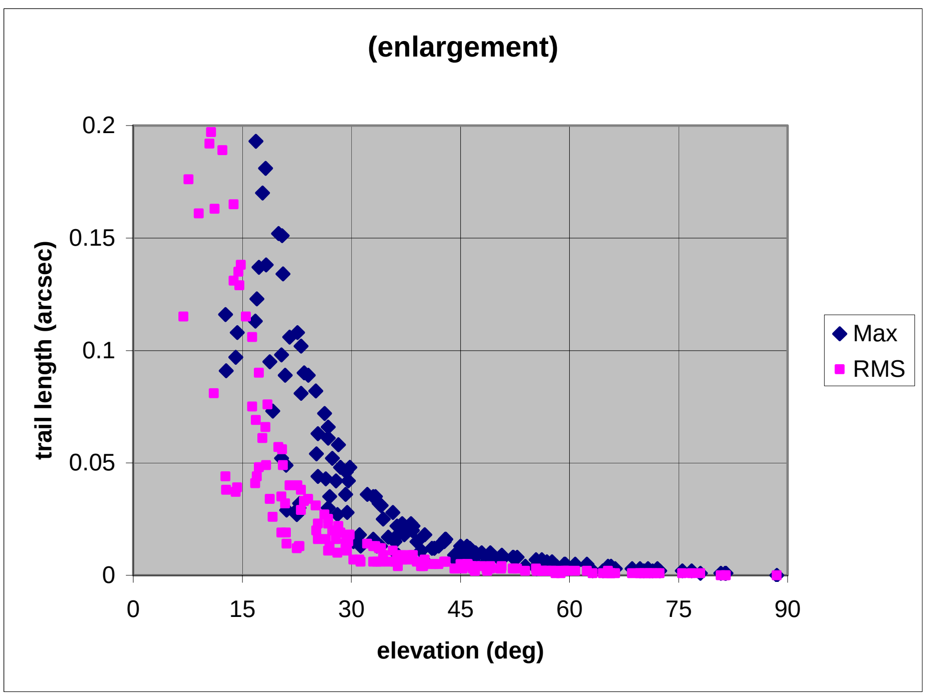


Figure 12: Trail length in 1 minute as a function of elevation.

Conclusions:

- Text-book rotator control via the parallactic angle is not good enough for LSST. Although it is adequate for low-elevation exposures, where the limits on image trailing are set inescapably by differential refraction, at higher elevations it worsens rapidly. In the zenith regions, where pointing corrections to the mount inject unwanted rotations into the field, the trailing becomes gross.
- LSST can instead adopt special rotator formulas that take into account the pointing coefficients, as in the Keck control system. This approach delivers excellent results, but the algorithm is somewhat opaque.
- The ray-tracing approach used on SOAR and Gemini is extremely straightforward in principle and guarantees zero rotation for the user-specified “instrument principal direction”. However, it relies on complete rigor in the pointing transformation algorithms, and in practice this means resorting to proprietary software (TCSpk).
- From a performance point of view, the best algorithm for LSST to adopt would be the ray-tracing method in its field-optimized variant.
- Even then, some fields lower than 25° will, in an exposure 1 minute long, exhibit trailing that exceeds 0.1 arcsec at the edge. The lower elevations, down to the 15° limit, are usable, but only for fields near culmination.

4 Effect of Errors in the Pointing Model

So far, the simulations have made two assumptions:

1. There is a perfect match between the nominal and true pointing models, so that the tracking is perfect and the central star in the test pattern is always free of trailing.
2. The observations are made through air, so that in most cases the trailing is due to differential refraction.

These assumptions will now be reversed, so that (i) the whole field will be trailed because of systematically imperfect tracking, and (ii) all trailing comes from the pointing errors and is not mixed up with refraction. Moreover, only one pointing term will be disturbed at one time, so that the effects due to that one term are exposed. Bear in mind that in the real LSST case, not only will *all* pointing terms be in error to some unknown extent but also the refraction *will* be present: consequently, trailing will occur which is the *sum* of several of the plots in this report.

Of the 17 terms in the Gemini South model used for the LSST simulations, only seven are included in the tests. The reasons for this are (i) the other terms are peculiar to Gemini South—there will be a similar set of empirical terms for LSST, but probably not the same ones—and (ii) the basic set of 7 terms explores all of the chief modes in which pointing terms can operate—incorrect encoder zero points, non-perpendicularities, tilts and vertical deflections.[7]

In each case, the coefficient of the pointing term under test is changed by exactly +10 arcsec, so that the nominal model, that is being used to control the telescope, is 10 arcsec more than that describing the real telescope. This is at the upper end of the expected range for LSST. A modern large altazimuth telescope should point blind to better than 5 arcsec, worst case, suggesting that individual terms are known to perhaps 2 arcsec or better.

Simulated exposures of 60 seconds were used throughout (of course unguided), initially all over the sky (at the places listed in Appendix B) and then finally in a place where the effect was at a maximum. The trail plots are with respect to the

[7] It will be noted that nothing is said about the accuracy with which the pixel coordinates of the rotator axis are known. This is because the simulations are by definition with respect to the rotator axis, and the uncertainty manifests itself as errors in the pointing coefficients CA (left-right component) and IE (up-down component). However, this does not mean that the pixel coordinates of the rotator axis are unimportant, because they represent where the calibration-star images must fall during pointing tests. In fact this turns out to be such a critical part of the LSST performance that it would be wise to locate the pixel coordinates of the rotator axis in the instrument test lab, not on the telescope. On-sky determination of rotator axis location is always problematical because it depends on perfect open-loop tracking.

image positions at the start of the exposure: the initial setting error is thus ignored. Field-optimized rotator control was used.

The first term to be tested was the azimuth index error, IA. The coefficient for the true telescope was –157.5340 arcsec. The corresponding value used in the nominal model was –147.5340 arcsec. The resulting trails were never greater than 38 mas, slightly better than this at low elevations. Figure 13 is for a field transiting about 10° north of the zenith.

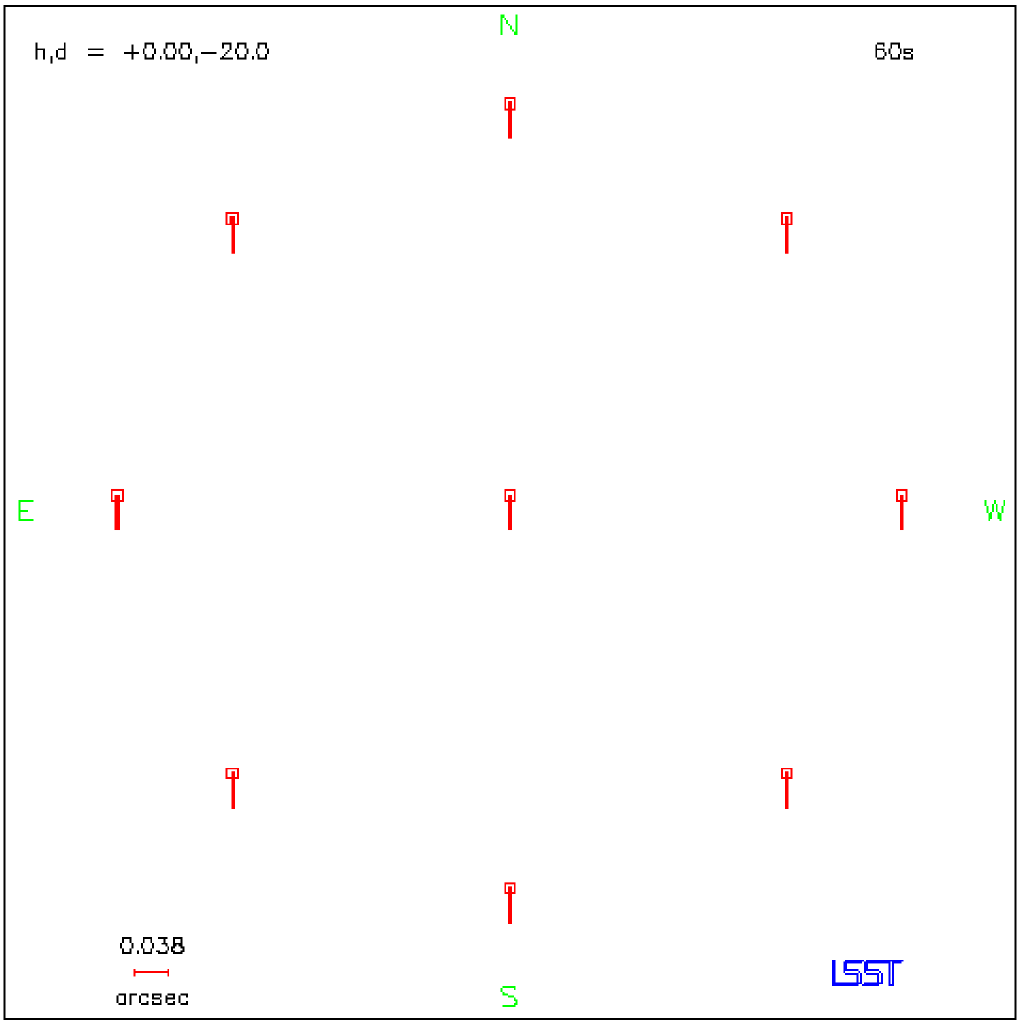


Figure 13: Trailing caused by a 10 arcsec error in the azimuth zero point IA.

The elevation index error, IE, is –520.8787 arcsec for the true telescope. When this was changed to –510.8787 arcsec, fields close to the zenith showed significant trailing. Figure 14 is for a field in transit 1.5° south of the zenith, where the trails during the 1-minute exposure reach 1.444 arcsec.

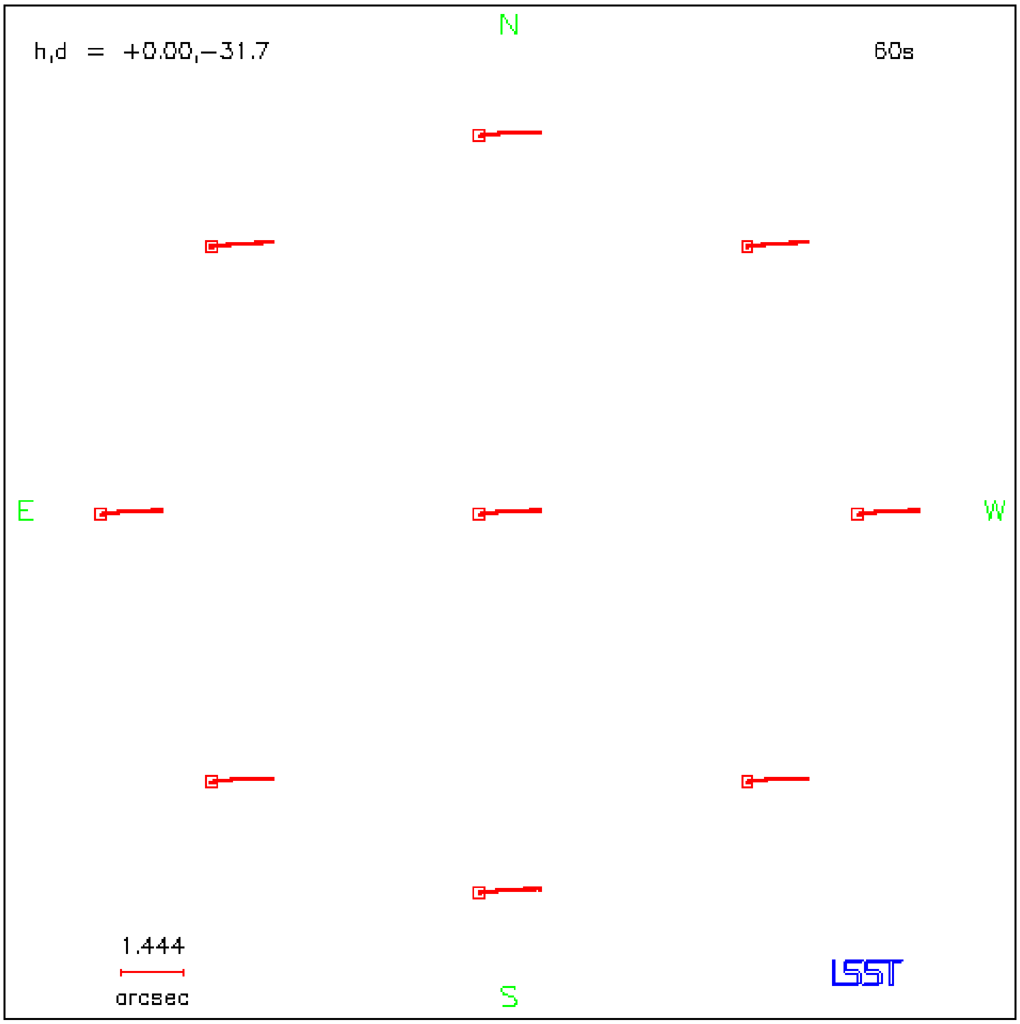


Figure 14: Trailing caused by a 10 arcsec error in the elevation index error IE.

This result may qualitatively seem surprising, but is merely a reflection of the rapid change of rotator angle when in transit near the zenith. On the meridian, a displacement of the rotator axis in zenith distance corresponds to a displacement in declination alone. 30s later, the same displacement in zenith distance causes a change in right ascension (together with a smaller change in declination).

The TF term represents Hooke's-Law vertical flexure, proportional to sin ζ. For the real telescope, a coefficient of –7.3000 was used, while the nominal pointing model used +2.7 arcsec. The resulting trails were small, and consistent over wide areas of sky. Figure 14 shows a field in transit about 30° north of the zenith, producing trails of 38 mas.

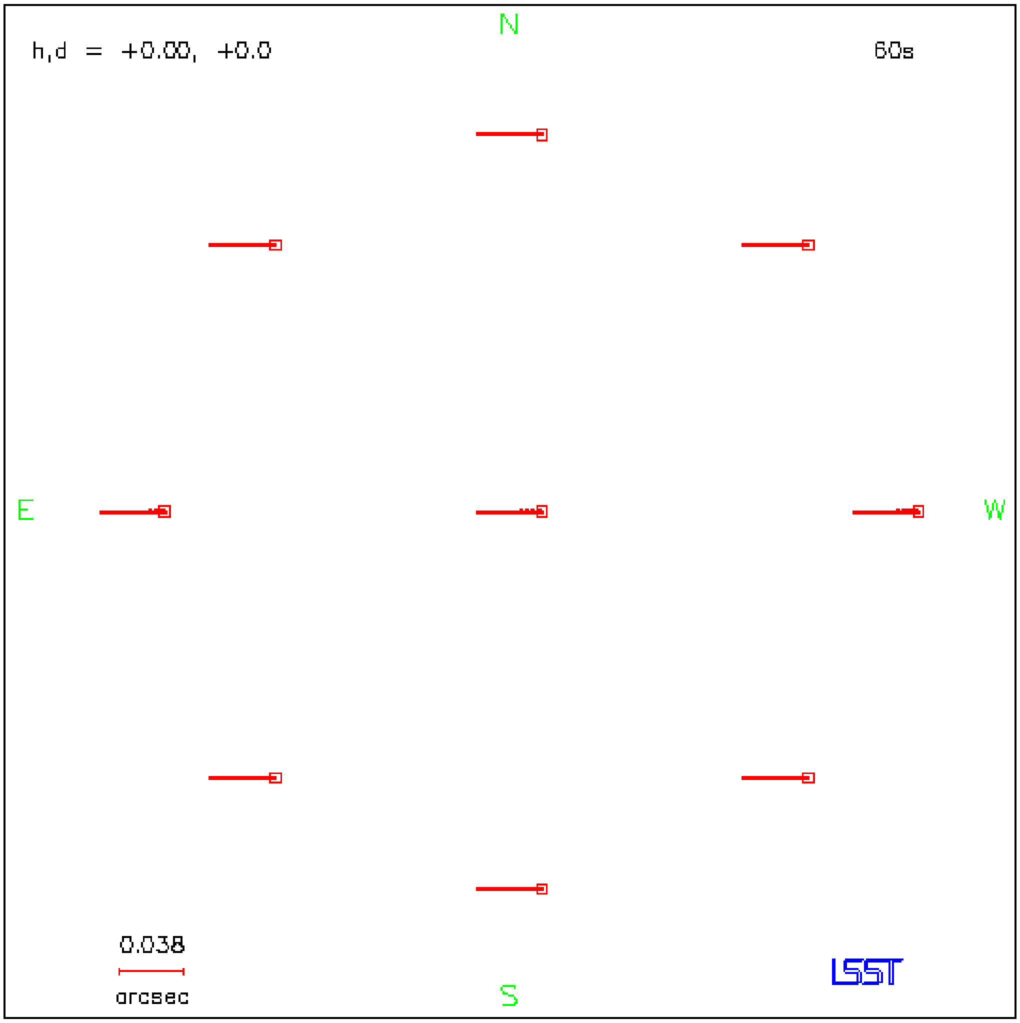


Figure 15: Trailing caused by a 10 arcsec error in the tube flexure term TF.

The reason the effects are so much smaller than those for the elevation index error, IE, even though both affect elevation, is that near the zenith, where IE errors matter most, the TF correction is small.

A 10 arcsec error in the az/el nonperpendicularity term NPAE produced large trails for fields in transit near the zenith. The coefficient values used in Figure 16 were –6.3266 arcsec for the real telescope and +3.6734 arcsec for the nominal telescope. For a field in transit 1.5° south of the zenith, trails of 1.439 arcsec are predicted.

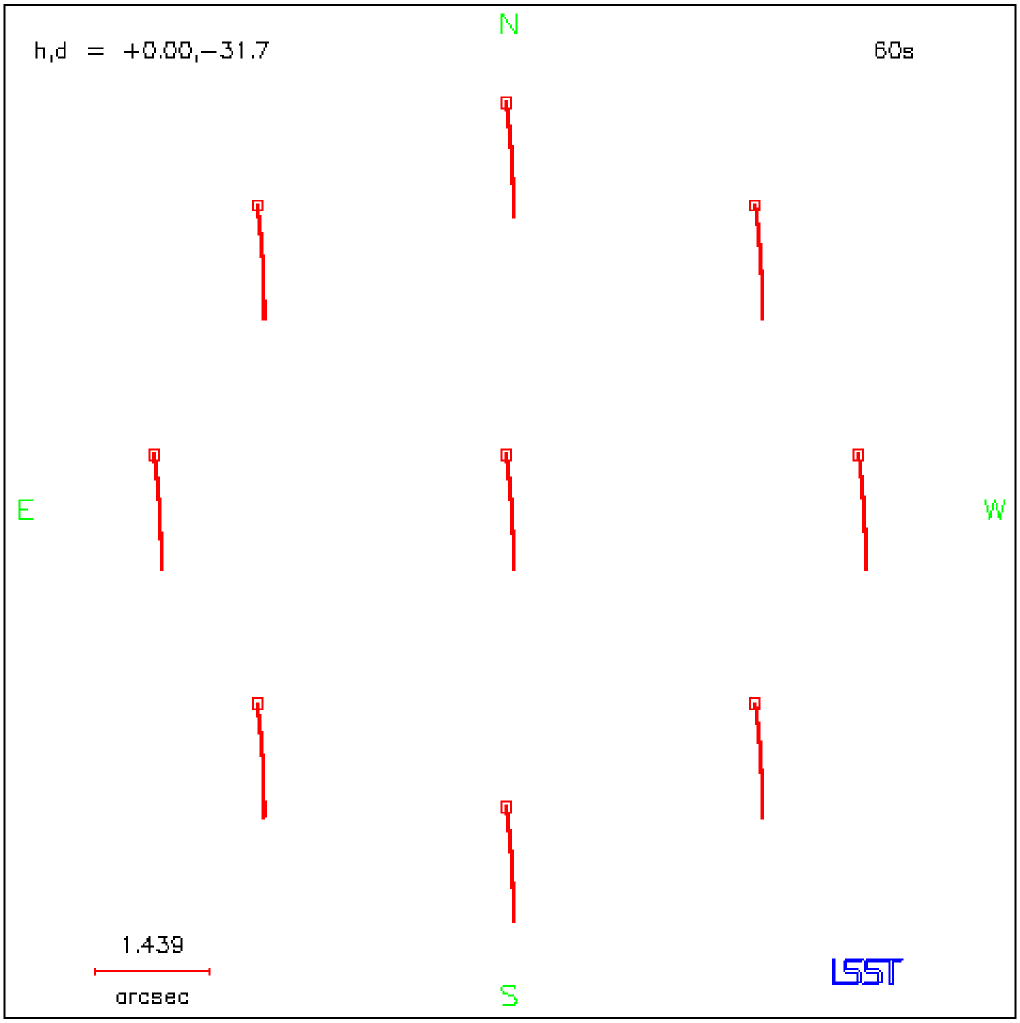


Figure 16: Trailing caused by a 10 arcsec error in the az/el nonperpendicularity NPAE.

On the face of it, this places stringent limits on the accuracy with which the LSST non-perpendicularity will need to be known. Fortunately, this is not the case, as is explained below.

The collimation error CA is the nonperpendicularity between the telescope and the elevation axis. For the true telescope, a coefficient value of + 51.6312 arcsec was used, while in the nominal model the value was +61.6312 arcsec. The resulting trails are shown in Figure 17, for the same field that was used in the NPAE case, above. The two plots—NPAE and CA respectively—are almost identical, with trails of up to 1.439 arcsec developing during the 1-minute exposure.

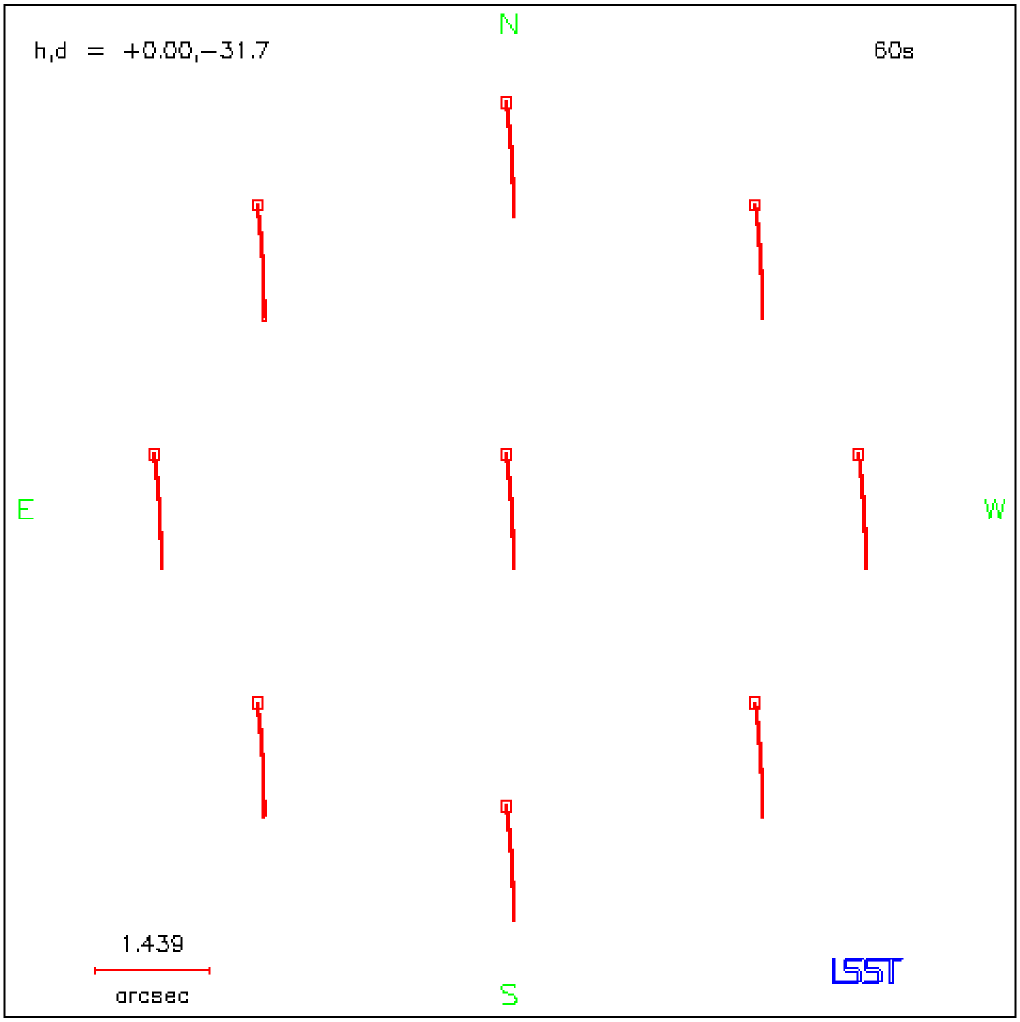


Figure 17: Trailing caused by a 10 arcsec error in the OTA/el nonperpendicularity CA.

The reason the NPAE and CA plots are so similar is that close to the zenith the two terms produce the same pointing corrections; the difference comes at low elevations, when NPAE produces a much smaller horizontal shift (and a small field rotation) where CA still produces the same horizontal shift as it did near the zenith. This relationship also has the effect of relaxing the accuracy requirements for each of the two terms. When the pointing model is fitted, they trade off with one another: for the zenith region, where the worst trailing effects are encountered, it is the blend of NPAE and CA which matters rather than the individual values.

As for IE, the NPAE/CA trails are a consequence of the rapid change of rotator angle when in transit near the zenith, this time for a horizontal displacement of the rotator axis. On the meridian, such a displacement corresponds to a displacement in right ascension alone. 30s later, the same horizontal displacement causes a change in declination (together with a smaller change in right ascension).

Figure 18 shows the trailing caused by a 10 arcsec error in the azimuth axis westwards tilt (true AW = +19.4365 arcsec, nominal value +29.4365 arcsec). The test exposure, 1.5° south of the zenith, suffers only 25 mas of trailing, and away from the zenith the trailing becomes even less.

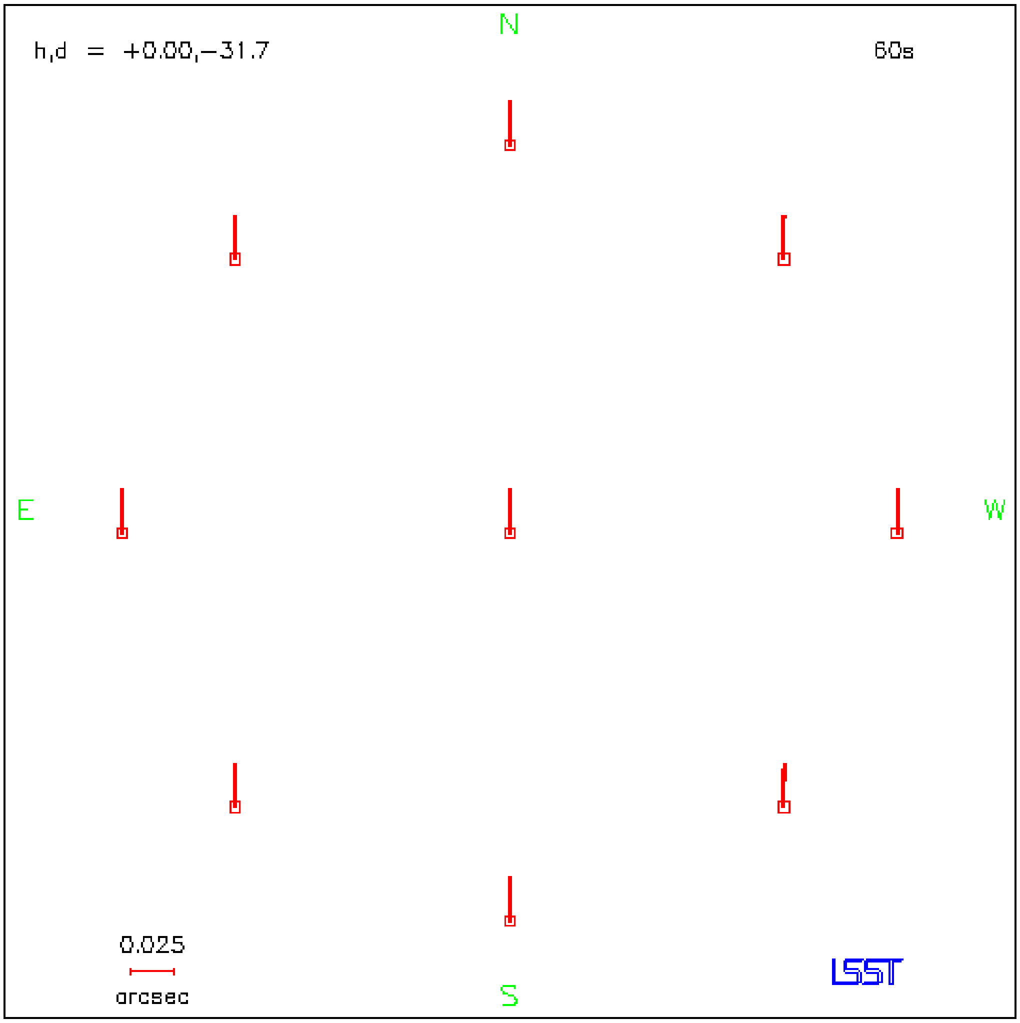


Figure 18: Trailing caused by a 10 arcsec error in the azimuth axis westwards tilt AW.

Finally, Figure 19 shows the trailing that comes from a 10 arcsec error in the azimuth axis northwards tilt (true –24.2447 arcsec, nominal –14.2447 arcsec). As for the westwards tilt, the error is very small—only 44 mas for the test exposure, transiting 5° above the south celestial pole—but unchanged over large parts of the sky.

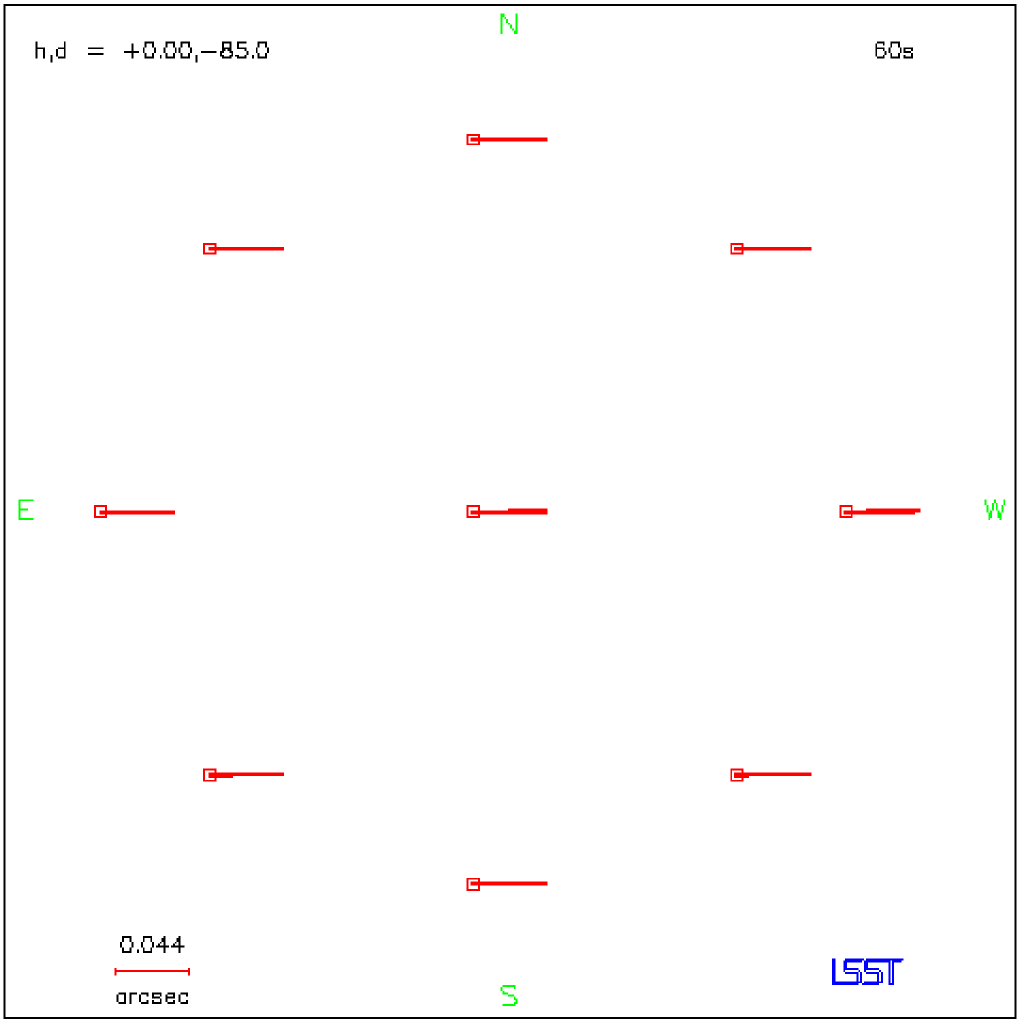


Figure 19: Trailing caused by a 10 arcsec error in the azimuth axis northwards tilt AN.

Conclusions:

1. In the absence of any autoguiding, LSST will be highly dependent on having an accurate pointing model. Table 2 shows the error in each of the seven basic pointing terms that can cause a 0.1 arcsec trail in a 1-minute exposure. Most of the tolerances are easily achievable—IA, TF, AW and AN need to be known to 20-40 arcsec, which is very lax. The remaining three terms—IE, NPAE and CA—are a different matter: an error of 1 arcsec in any one of them will cause more than 0.1 arcsec per minute trailing for fields transiting near the zenith. The position for NPAE and CA is not quite as serious as it seems, because near the zenith they have similar effects and so it is only the combination of the two that has to be known to rather better than 1 arcsec.

2. It might be thought that keeping the local pointing in a high state of tune by frequent adjustments of IE and CA would suppress the IE/CA problems, but this is only true to the extent that the natural pointing variations really do correspond to those terms. Thus, while differential thermal expansion in the OTA would be amenable to correction by IE and CA changes, instabilities in the azimuth axis position would not. If *ad hoc* IE and CA changes are introduced merely in order to force the local pointing to match the rotator axis, and if the underlying cause is not of an IE/CA character, the adjustments will do more harm than good as far as tracking is concerned.

3. The high accuracy with which IE and CA must be known means in turn that precise calibration of the pixel coordinates of the rotator axis is needed. To avoid relying on the open-loop tracking of LSST (though this has in any case to be excellent if the telescope is to be operated without autoguiding) it may be wise to carry out the calibration in the workshop, using an artificial star.

4. As a dedicated wide-field imager, LSST is in an excellent position to use its science observations as the principal source of pointing data. It could do this continuously—recognizing astrometric calibration stars such as those in the UCAC2 star catalog and deriving for each exposure an accurate ICRS α,δ for the rotator axis—and run fitting software to keep its pointing model up to date automatically. This capability should be built into the observing software from the outset. (The approach is normally not practical because it is vulnerable to inaccurate user-supplied science target coordinates and the limited sky coverage that an individual science program tends to provide.)

5. The worst effects of pointing-model inaccuracy occur near the zenith, and it may turn out to be necessary to accept a larger zenith blind spot than the present 1.5°; only experience will tell. Table 3 shows how the accuracy requirements for the IE and CA coefficients ease as the distance from the zenith at transit is increased to 5°. The 0.4 arcsec trails correspond to 10 arcsec errors in IE and CA respectively. A 1.6 arcsec error in each of IE and CA would produce an overall 0.1 arcsec trail in 60s.

6. At the lowest elevations (i.e. 15° elevation) there is excessive trailing in a 1 minute exposure from differential refraction alone.

term	*tolerance*
IA	26
IE	0.7
TF	26
NPAE	0.7
CA	0.7
AW	40
AN	23
	arcsec

Table 2: Required accuracy of pointing-model terms. Each of the above errors may cause 0.1 arcsec of trailing in a 1-minute exposure.

Δ	*ζ*	*ΔIE=10"*	*ΔCA=10"*
−25.24	5	0.433	0.433
−28.74	1.5	1.427	1.424
−30.24	0		
−31.74	1.5	1.443	1.439
−35.24	5	0.435	0.434
deg	*deg*	*arcsec*	*arcsec*

Table 3: Trailing from 10 arcsec IE and CA errors at different transit zenith distances.

APPENDIX A – Exposure Simulator Source Code

As mentioned in Section 2, this Appendix lists representative source code for the exposure simulator program in order to document the way in which the trailing predictions were made. Note that:

- The actual code delivered to the LSST project is not identical to the listing. The latter is in a more concise form, dedicated to the particular case of demonstrating different rotator control algorithms.

- Compiling and linking the program requires the proprietary SLALIB/C and TCSpk libraries.

More details about the program, including how it is configured for the different tests and how the output can be interpreted, are given in Section 2.

```
#include <slalib.h>
#include "tcs.h"
#if 0
/*
** - - - - - - - -
**  T R A I L S
** - - - - - - - -
**
** Common code for LSST trail tests.
**
** This revision:   19 January 2007
**
** Copyright P.T.Wallace.  All rights reserved.
*/

/*
** -------------------------------------
** Example of parameters defining a test
** -------------------------------------
*/

#define YES 1
#define NO 0

#define VERBOSE YES           /* Full reporting */
#define REP_HEAD YES          /* Head the report */

#define MOUNT EQUAT           /* Mount type */
#define RLOCN OTA             /* Instrument rotator location */
#define SITE_LAT -31.0        /* Site latitude (deg) */
#define SITE_HM 1300.0        /* Site elevation (m) */
#define PMB 893.0             /* pressure (hPa) */

#define FIELD (6.0*sqrt(2.0)) /* Field radius (deg) */
#define F_CIRC NO             /* Circular (rather than square) field */
#define FOCAL_L 3048.0        /* Focal length (mm) */

#define TPS 1L                /* Ticks per second */
#define T_SLOW (60L*TPS)      /* Ticks per slow update */
#define T_MED (5L*TPS)        /* Ticks per medium update */

#define PM_NOMINAL "schmidt.mod" /* Nominal pointing model */
#define PM_TRUE PM_NOMINAL       /* True pointing model */
#define DXI 0.0               /* Rotator axis error in xi (arcsec) */
#define DETA 0.0              /* Rotator axis error in eta (arcsec) */
#define PM_ENABLED YES        /* Pointing models enabled */

#define RMA_PAR NO            /* Control rotator using parallactic angle */
```

```
#define RMA_KECK NO           /* Control rotator using Keck formula */
#define RMA_SLIT NO           /* Control rotator for slit case */
#define RMA_FIXED YES         /* Rotator motion enabled */

#define LEN_TEST (60L*60L*TPS) /* Duration of test (ticks) */
#define LOG_INT (60L*TPS)      /* Ticks per line of report */

#define REP_DIST NO          /* Whether to report distortions and shifts */
#define REP_TRAIL YES        /* Whether to report trailed image positions */
#define MAG 2000             /* Trail magnification */
#define N_TOP YES            /* Whether north at the top */
#define E_LEFT YES           /* Whether east at the left */
#define LAT_INV NO           /* Whether to laterally invert trails */
#define ONE_STAR YES         /* One star or multiple stars */

#endif

/* ---------------------------------------- */
/* The live common code segment starts here */
/* ---------------------------------------- */

/* Euler angles wrt terrestrial [Az,El] for generalized gimbal case */
#define GIM1Z 0.0        /* 1st rotation, about z-axis */
#define GIM2Y 0.0        /* 2nd rotation, about y-axis */
#define GIM3X 0.0        /* 3rd rotation, about x-axis */

{

/* ------------------------------------------ */
/* Context for imperfectly-modelled telescope */
/* ------------------------------------------ */

#include "tcsctx.h"

/* -------------------------- */
/* Context for real telescope */
/* -------------------------- */

   static int model_true[MAXTRM];
   static double coeffv_true[MAXTRM];
   static int nterml_true;
   static int ntermx_true;
   static int nterms_true;
   static char coeffn_true[NTROOM][9];
   static char coform_true[NTROOM][9];
   static double ia_true;
   static double ib_true;
   static double np_true;
   static double xt_true;
   static double yt_true;
   static double zt_true;
   static double ae2mt_true[3][3];
   static double m_spm1_true[3][3][3][3], m_spm1_i_true[3][3][3][3],
                 m_spm2_true[3][3][3][3], m_spm2_i_true[3][3][3][3];
   static double r_spm1_true[3][3], r_spm1_i_true[3][3],
                 r_spm2_true[3][3], r_spm2_i_true[3][3];

/* ----------------------------------------- */
/* Context for "topocentric az,el" telescope */
/* ----------------------------------------- */

   static int model_topo[MAXTRM];
   static double coeffv_topo[MAXTRM];
   static int nterml_topo;
   static int ntermx_topo;
   static int nterms_topo;
   static char coeffn_topo[NTROOM][9];
   static char coform_topo[NTROOM][9];
   static double ia_topo;
   static double ib_topo;
   static double np_topo;
   static double xt_topo;
   static double yt_topo;
   static double zt_topo;
   static double ae2mt_topo[3][3];
```

```
   static double m_spm1_topo[3][3], m_spm1_i_topo[3][3],
                 m_spm2_topo[3][3], m_spm2_i_topo[3][3];
   static double r_spm1_topo[3][3], r_spm1_i_topo[3][3],
                 r_spm2_topo[3][3], r_spm2_i_topo[3][3];

/* ------------------------- */
/* End of telescope contexts */
/* ------------------------- */


/* Field halfwidth (radians). */
   double hw = FIELD * D2R / sqrt(8.0);

/* Minimum elevation (degrees). */
   double elmin = 5.0*D2R;

/* Array of test RA,Decs. */
   double tp[3][3][2];

/* Arrays of corresponding x,y positions. */
   double xye[3][3][2], xym[3][3][2];

/* Statistics. */
   long n;
   double sdr2, dr2max, sel;

/* Affine transformation coefficients. */
   double coeffs[6];

   int istar, low, i, ix, iy, j, ih, im;
   long itick, l;
   char s, line[200];
   double tai, vca, vnpae, ha, dec, t, rap, dap, hob, dob,
          aia, st, az, el, sst, cst, x, y, z, theta, rota_p,
          rtar, dtar, cd, xtar, ytar, ztar,
          ru, sb, xu, yu, zu, xut, yut, zut, xst, yst, zst, rst,
          xr, yr, zr, sq, cq, q, xi, rxy2, qc, rota_k, xiim, etaim,
          xim, yim, dx, dy, xtrail, ytrail, dr2, xz, yz, xs, ys,
          perp, orient, sec;


/*
** --------------------------
** Report the test conditions
** --------------------------
*/

#if VERBOSE && REP_HEAD
   printf ( "Test circumstances:\n\n" );
   printf ( "  mount type is %s\n", MOUNT == ALTAZ ? "altazimuth" :
                                                    "equatorial" );
   printf ( "  site longitude and height = %+g deg, %g m\n",
                                                   SITE_LAT, SITE_HM );
   printf ( "  pressure = %7.2f hPa\n", PMB );
#if F_CIRC
   printf ( "  field is %f deg (radius)\n", FIELD/2.0 );
#else
   printf ( "  field is %f x %f deg (%f radius)\n",
                                            hw/D2R, hw/D2R, FIELD/2.0 );
#endif
   printf ( "  focal length = %g mm\n", FOCAL_L );
   printf ( "  ticks per second = %d\n", (int) TPS );
   printf ( "  slow update every %f sec\n", ( (double) T_SLOW ) /
                                            ( (double) TPS ) );
   printf ( "  medium update every %f sec\n", ( (double) T_MED ) /
                                              ( (double) TPS ) );
#if PM_ENABLED
   printf ( "  pointing models (nominal, true) are in files %s, %s\n",
            PM_NOMINAL, PM_TRUE );
#else
   printf ( "  pointing models disabled\n" );
#endif
#if RMA_PAR
   printf ( "  rotator controlled by parallactic angle\n" );
#elif RMA_KECK
```

```
   printf ( "  rotator controlled by Keck formula\n" );
#elif RMA_SLIT
   printf ( "  rotator controlled for slit case\n" );
#elif RMA_FIXED
   printf ( "  rotator fixed\n" );
#else
   printf ( "  rotator controlled by TCSpk (2-axis)\n" );
#endif
   printf ( "  total duration of test in seconds = %g\n",
                          ( (double) LEN_TEST ) / ( (double) TPS ) );
   printf ( "  seconds per report = %g\n",
                          ( (double) LOG_INT ) / ( (double) TPS ) );
#if REP_DIST
   printf ( "  trails will be logged as shifts and distortions\n" );
#endif
#if REP_TRAIL
   printf ( "  trails will be logged as dX,dY magnified x%d, with \n", MAG );
#if N_TOP
   printf ( "  north" );
#else
   printf ( "  south" );
#endif
   printf ( " at the top and " );
#if E_LEFT
   printf ( "west" );
#else
   printf ( "east" );
#endif
   printf ( " at the right\n" );
#endif
#if ONE_STAR
   printf ( "  results for one star only\n" );
#else
   printf ( "  results for multiple stars\n" );
#endif
   printf ( "\n" );
#endif

/*
** --------------
** Initialization
** --------------
**
** n.b. C's default initialization to zero is assumed.
*/

/* No special refraction handling required. */
   rfun = NULL;

/* Site and telescope. */
   tlongm = 0.0*D2R;
   tlatm = SITE_LAT*D2R;
   hm = SITE_HM;
   fl = FOCAL_L;
   rnogo = 0.05*D2R;
   mount = MOUNT;
   rotl = RLOCN;

/* Time. */
   delut = 0.746/86400.0;
   delat = 29.0/86400.0;
   ttmtai = 32.184/86400.0;
   xpmr = 0.0*AS2R;
   ypmr = 0.0*AS2R;

/* Met. */
   temp = 275.0;
   press = PMB;
   humid = 0.3;
   tlr = 0.0065;
   wavelr = 0.55;

/* Mount above/below pole state. */
   jbp = 0;
```

```
/* Initial mount [roll,pitch] (arbitrary). */
   roll = 0.0;
   pitch = 1.0;

/* Frames. */
   m_cosys = FK5;
   m_eqx = 2000.0;
   m_wavel = 1.0;
   r_cosys = m_cosys;
   r_eqx = m_eqx;
   r_wavel = m_wavel;

/* Pointing models, nominal (used to control telescope), true and null. */
#if PM_ENABLED
   if ( tcsIntpm ( PM_NOMINAL, MAXTRM, NTROOM, model,
                   &nterml, &ntermx, &nterms,
                   coeffn, coform, coeffv ) ) return -1;
   if ( tcsIntpm ( PM_TRUE, MAXTRM, NTROOM, model_true,
                   &nterml_true, &ntermx_true, &nterms_true,
                   coeffn_true, coform_true, coeffv_true ) ) return -1;
#else
   if ( tptMinit ( MAXTRM, NTROOM, model,
                   &nterml, &ntermx, &nterms,
                   coeffn ) ) return -1;
   if ( tptMinit ( MAXTRM, NTROOM, model_true,
                   &nterml_true, &ntermx_true, &nterms_true,
                   coeffn_true ) ) return -1;
#endif
   if ( tptMinit ( MAXTRM, NTROOM, model_topo,
                   &nterml_topo, &ntermx_topo, &nterms_topo,
                   coeffn_topo ) ) return -1;

/* Extract selected pointing coefficients. */
   vca = 0.0;
   vnpae = 0.0;
   for ( i=0; model[i] > 0; i++ ) {
      if ( ! strcmp ( "CA",   coeffn[model[i]-1] ) ) vca = coeffv[i];
      if ( ! strcmp ( "NPAE", coeffn[model[i]-1] ) ) vnpae = coeffv[i];
   }


/* Guiding adjustments. */
   ga = 0.0*AS2R;
   gb = 0.0*AS2R;

/* Secondary initialization. */
   if ( tcsInit2 ( tlongm, tlatm, xpmr, ypmr, hm,
                   mount, GIM1Z, GIM2Y, GIM3X, ae2nm,
                   &tlong, &tlat, &uau, &vau, &ukm, &vkm, &diurab ) ) {
      printf ( "\nSecondary initialization has failed.\n" );
      return -1;
   }

/* Simulate hardware that by now would be running. */
   rma = 0.0*D2R;    /* Rotator angle (n.b. achieved, NOT demanded) */
   rmav = 0.0;       /* Velocity (achieved). */
   rmat = 0.0;       /* Timestamp. */

/* Target defaults. */
   m_cosys = FK5;
   m_eqx = 2000.0;
   m_wavel = 0.55;
   r_cosys = m_cosys;
   r_eqx = m_eqx;
   m_tar_dt [ 0 ] = 0.0;
   m_tar_dt [ 1 ] = 0.0;
   m_tar_t0 = 0.0;
   m_tar_ob [ 0 ] [ 0 ] = 0.0;
   m_tar_ob [ 0 ] [ 1 ] = 0.0;

/*
** ------------
** Star by star
** ------------
*/
```

```
   for ( ; ; ) {

   /* The target. */
      if ( gets ( line ) == NULL ) break;
#if VERBOSE
      printf ( "<%s>\n", line );
#endif
      sscanf ( line, "%lf %lf", &ha, &dec );
      ha *= D2R;
      dec *= D2R;

   /*
   ** -----------------------------------------------
   ** Run the SLOW routine for the first of many times
   ** -----------------------------------------------
   */

   /* Get the (approximate) current time. */
      (void) tcsTime ( &tai );

   /* Perform the slow update for that time. */
      tcsSlow ( tai, delut, delat, ttmtai,
                temp, press, humid, wavelr, tlong,
                &t0, &st0, &tt0, &ttj, amprms, &refa, &refb );

   /*
   ** -----------------------------------------------
   ** Obtain a star near the nominated HA,Dec mid-test
   ** -----------------------------------------------
   */

      t = tai + ( (double) LEN_TEST ) / ( (double) TPS ) / 86400.0 / 2.0;
      slaOap ( "H", ha, dec, t-delat, delut, tlong, tlat, hm, 0.0, 0.0,
               temp, press, humid, wavelr, tlr, &rap, &dap );
      slaAmp ( rap, dap, t, 2000.0, m_tar_p0, m_tar_p0+1 );

   /*
   ** -------------------------------------------------
   ** Run the MEDIUM routine for the first of many times
   ** -------------------------------------------------
   */

   /* Get the (approximate) current time. */
      (void) tcsTime ( &tai );

   /* Perform the medium update for that time. */
      (void) tcsMedium ( tai, MAXTRM, model, coeffv, nterml, ntermx, nterms,
                         coeffn, coform, mount, ae2nm, roll, pitch, jbp, aux,
                         m_cosys, m_eqx, m_wavel, r_cosys, r_eqx, r_wavel,
                         m_tar_p,
                         t0, st0, ttj, temp, press, humid, tlr, wavelr,
                         refa, refb, rfun, hm, tlat, diurab, amprms,
                         &ia, &ib, &np, &xt, &yt, &zt, ae2mt,
                         m_spm1, m_spm1_i, m_spm2, m_spm2_i,
                         r_spm1, r_spm1_i, r_spm2, r_spm2_i );

   /*
   ** ----------------------------------------------
   ** Set the field-orientation to something harmless
   ** ----------------------------------------------
   */

      aia = 0.0;             /* IAA */

      pai = 0.0;             /* IPA */
      sia = sin ( aia );     /* sin(IAA) */
      cia = cos ( aia );     /* cos(IAA) */

#if RMA_SLIT

   /* Rotator predictions slit-optimized. */
      jf = 0;

#else
```

```
   /* Rotator predictions field-optimized. */
      jf = 1;

#endif

   /* Tell the FAST routine. */
      tcsFast ( PA, jbp, tai,
                rotl, rmat, rma, rmav,
                t0, st0,
                ia, ib, np, xt, yt, zt, ga, gb, rnogo,
                m_tar_t0, m_tar_op0, m_tar_dt,
                fl, m_por_p,
                sia, cia, pai, jf,
                m_cosys, m_spm1, m_spm2, r_cosys, r_spm1_i, r_spm2_i,
                &roll, &pitch, &rota );

   /*
   ** ---------------------------------------------
   ** Set the pointing-origins to something harmless
   ** ---------------------------------------------
   */

      npo = 1;
      m_por_p0[npo][0] = 0.0;
      m_por_p0[npo][1] = 0.0;
      m_por_ob[npo][0][0] = 0.0;
      m_por_ob[npo][0][1] = 0.0;

   /* Incorporate the offsets from base. */
      tcsPorup ( m_por_p0[npo], m_por_ob[npo], m_por_p );

   /* Tell the FAST routine. */
      tcsFast ( PO, jbp, tai,
                rotl, rmat, rma, rmav,
                t0, st0,
                ia, ib, np, xt, yt, zt, ga, gb, rnogo,
                m_tar_t0, m_tar_op0, m_tar_dt,
                fl, m_por_p,
                sia, cia, pai, jf,
                m_cosys, m_spm1, m_spm2, r_cosys, r_spm1_i, r_spm2_i,
                &roll, &pitch, &rota );

   /* Tell the FAST routine about everything initialized so far. */
      tcsFast ( ALL, jbp, tai,
                rotl, rmat, rma, rmav,
                t0, st0,
                ia, ib, np, xt, yt, zt, ga, gb, rnogo,
                m_tar_t0, m_tar_op0, m_tar_dt,
                fl, m_por_p,
                sia, cia, pai, jf,
                m_cosys, m_spm1, m_spm2, r_cosys, r_spm1_i, r_spm2_i,
                &roll, &pitch, &rota );

   /* Get the (approximate) current time. */
      (void) tcsTime ( &tai );

   /* Calculate target position. */
      tcsTargup ( tai, m_tar_t0, m_tar_p0, m_tar_dt, m_tar_ob,
                  m_tar_op0, m_tar_p );

   /* Now run the MEDIUM routine to generate the new pointing model */
   /* and SPMs for mount tracking.                                  */
      (void) tcsMedium ( tai, MAXTRM, model, coeffv, nterml, ntermx, nterms,
                         coeffn, coform, mount, ae2nm, roll, pitch, jbp, aux,
                         m_cosys, m_eqx, m_wavel, r_cosys, r_eqx, r_wavel,
                         m_tar_p,
                         t0, st0, ttj, temp, press, humid, tlr, wavelr,
                         refa, refb, rfun, hm, tlat, diurab, amprms,
                         &ia, &ib, &np, &xt, &yt, &zt, ae2mt,
                         m_spm1, m_spm1_i, m_spm2, m_spm2_i,
                         r_spm1, r_spm1_i, r_spm2, r_spm2_i );

   /* Tell the FAST routine about the new target and mount SPMs. */
      tcsFast ( TARGET + TRANSFORM + MODEL, jbp, tai,
                rotl, rmat, rma, rmav,
```

```
             t0, st0,
             ia, ib, np, xt, yt, zt, ga, gb, rnogo,
             m_tar_t0, m_tar_op0, m_tar_dt,
             fl, m_por_p,
             sia, cia, pai, jf,
             m_cosys, m_spm1, m_spm2, r_cosys, r_spm1_i, r_spm2_i,
             &roll, &pitch, &rota );

/*
** -----------------------------------------------------
** Run the FAST routine every tick for the test duration
** -----------------------------------------------------
*/

/* Reset "went below elevation limit" flag. */
   low = 0;

/* Initialize the image trailing statistics. */
   n = 0;
   sdr2 = 0.0;
   dr2max = 0.0;
   sel = 0.0;

/* Initialize the clock. */
   (void) tcsTime ( &tai );

/* Simulate successive calls to FAST. */
   for ( itick = -T_SLOW; itick <= LEN_TEST; itick++ ) {

   /* TAI (MJD). */
      t = tai + ( (double) itick ) / ( (double) TPS ) / 86400.0;

   /* Perform the SLOW update if due. */
      if ( ! ( itick % T_SLOW ) ) {
         tcsSlow ( t, delut, delat, ttmtai,
                   temp, press, humid, wavelr, tlong,
                   &t0, &st0, &tt0, &ttj, amprms, &refa, &refb );
      }

   /* Perform the nominal and null pointing-model medium updates if due. */
      if ( ! ( itick % T_MED ) ) {

      /* SPMs for nominal pointing model. */
         (void) tcsMedium (
                   t, MAXTRM,
                   model,
                   coeffv,
                   nterml,
                   ntermx,
                   nterms,
                   coeffn,
                   coform,
                   mount, ae2nm, roll, pitch, jbp, aux,
                   m_cosys, m_eqx, m_wavel, r_cosys, r_eqx, r_wavel,
                   m_tar_p,
                   t0, st0, ttj, temp, press, humid, tlr, wavelr,
                   refa, refb, rfun, hm, tlat, diurab, amprms,
                   &ia,
                   &ib,
                   &np,
                   &xt,
                   &yt,
                   &zt,
                   ae2mt,
                   m_spm1,
                   m_spm1_i,
                   m_spm2,
                   m_spm2_i,
                   r_spm1,
                   r_spm1_i,
                   r_spm2,
                   r_spm2_i );

      /* SPMs for null pointing model and no refraction. */
         (void) tcsMedium (
```

```
                     t, MAXTRM,
                     model_topo,
                     coeffv_topo,
                     nterml_topo,
                     ntermx_topo,
                     nterms_topo,
                     coeffn_topo,
                     coform_topo,
                     mount, ae2nm, roll, pitch, jbp, aux,
                     m_cosys, m_eqx, m_wavel, r_cosys, r_eqx, r_wavel,
                     m_tar_p,
                     t0, st0, ttj, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, rfun, hm, tlat, diurab, amprms,
                     &ia_topo,
                     &ib_topo,
                     &np_topo,
                     &xt_topo,
                     &yt_topo,
                     &zt_topo,
                     ae2mt_topo,
                     m_spm1_topo,
                     m_spm1_i_topo,
                     m_spm2_topo,
                     m_spm2_i_topo,
                     r_spm1_topo,
                     r_spm1_i_topo,
                     r_spm2_topo,
                     r_spm2_i_topo );
      }

   /* Tell the FAST routine about everything that has changed. */
      tcsFast ( 127, jbp, t,
                  rotl, rmat, rma, rmav,
                  t0, st0,
                  ia, ib, np, xt, yt, zt, ga, gb, rnogo,
                  m_tar_t0, m_tar_op0, m_tar_dt,
                  fl, m_por_p,
                  sia, cia, pai, jf,
                  m_cosys, m_spm1, m_spm2, r_cosys, r_spm1_i, r_spm2_i,
                  &roll, &pitch, &rota );

   /* Ask the FAST routine to calculate new demands. */
      tcsFast ( 0, jbp, t,
                  rotl, rmat, rma, rmav,
                  t0, st0,
                  ia, ib, np, xt, yt, zt, ga, gb, rnogo,
                  m_tar_t0, m_tar_op0, m_tar_dt,
                  fl, m_por_p,
                  sia, cia, pai, jf,
                  m_cosys, m_spm1, m_spm2, r_cosys, r_spm1_i, r_spm2_i,
                  &roll, &pitch, &rota );

   /* Prepare for other rotator predictions. */

      rtar = m_tar_op0[0];
      dtar = m_tar_op0[1];
      cd = cos(dtar);
      xtar = cos(rtar)*cd;
      ytar = sin(rtar)*cd;
      ztar = sin(dtar);
      st = st0 + ( t - t0 ) * STRPD;
      sst = sin ( st );
      cst = cos ( st );

   /* Parallactic angle, calculated from topocentric az,el. */

      tcsSky2a_c ( xtar, ytar, ztar,
                     m_spm1_topo, m_cosys, sst, cst, m_spm2_topo,
                     &x, &y, &z );
      az = x != 0.0 || y != 0.0 ? atan2 ( y, -x ) : 0.0;
      el = atan2 ( z, sqrt ( x*x + y*y ) );
      slaDh2e ( az, el, tlat, &hob, &dob );
      theta = slaPa ( hob, dob, tlat );
      rota_p = PI - theta;
```

```
      /* Keck algorithm. */

         ru = sqrt ( xtar*xtar + ytar*ytar );
         sb = ztar / ( ru > 1e-10 ? ru : 1e-10 );
         xu = -xtar*sb;
         yu = -ytar*sb;
         zu = ru;
         tcsSky2a_c ( xu, yu, zu,
                      m_spm1, m_cosys, sst, cst, m_spm2,
                      &xut, &yut, &zut );
         tcsSky2a_c ( xtar, ytar, ztar,
                      m_spm1, m_cosys, sst, cst, m_spm2,
                      &xst, &yst, &zst );
         rxy2 = xst*xst + yst*yst;
         rst = sqrt ( rxy2 );
         sb = zst / ( rst > 1e-10 ? rst : 1e-10 );
         xr = -xst*sb;
         yr = -yst*sb;
         zr = rst;
         sq = xst*yr*zut + yst*zr*xut + zst*xr*yut
            - zst*yr*xut - xst*zr*yut - yst*xr*zut;
         cq = xr*xut + yr*yut + zr*zut;
         q = atan2 ( sq, cq );
         xi = - ( vca + vnpae*zst );
         qc = q - z*atan2(xi,sqrt(rxy2-xi*xi)) + vnpae*sqrt(rxy2);
         rota_k = PI - qc;

/* Select which rotator prediction override to use. */

#if RMA_PAR

      /* Parallactic angle based. */
         rota = rota_p;

#elif RMA_KECK

      /* Keck algorithm. */
         rota = rota_k;

#elif RMA_FIXED

      /* Rotator not tracking. */
         rota = rma;

#endif

      /* Put rotator demand into the normal range (0-2pi). */
         rota = slaDranrm ( rota );

      /* Make the rotator follow the demands. */
         rmav = slaDrange ( rota - rma ) * 86400.0 * ( (double) TPS );
         rma = rota;
         rmat = t;

      /* Tell the FAST routine about it. */
         tcsFast ( 1, jbp, t,
                   rotl, rmat, rma, rmav,
                   t0, st0,
                   ia, ib, np, xt, yt, zt, ga, gb, rnogo,
                   m_tar_t0, m_tar_op0, m_tar_dt,
                   fl, m_por_p,
                   sia, cia, pai, jf,
                   m_cosys, m_spm1, m_spm2, r_cosys, r_spm1_i, r_spm2_i,
                   &roll, &pitch, &rota );

      /* Is it time to log? */
         if ( ! ( itick % LOG_INT ) ) {

         /* Yes: obtain LAST. */
            st = st0 + ( t - t0 ) * STRPD;

         /* Obtain (approximate topocentric) az,el. */
            slaMapqkz ( m_tar_p[0], m_tar_p[1], amprms, &rap, &dap );
            slaDe2h ( st - rap, dap, tlatm, &az, &el );
```

```
         /* Abandon this star if too low or too near pole of mounting. */
            if ( el < elmin || pitch > D90-rnogo ) {
               low = 1;
               break;
            }

         /* Is the run in progress yet? */
            if ( itick >= 0L ) {

            /* Yes.  Functions of LAST. */
               sst = sin ( st );
               cst = cos ( st );

            /* Is this the very start of the run? */
               if ( ! itick ) {

               /* Yes.  Create the pattern of test points. */
                  for ( iy = 0; iy < 3; iy++ ) {
                     for ( ix = 0; ix < 3; ix++ ) {

                     /* Image x,y (radians). */
                        xye[ix][iy][0] = ( (double) ( ix - 1 ) ) * hw;
                        xye[ix][iy][1] = ( (double) ( iy - 1 ) ) * hw;
#if F_CIRC
                        if ( !(ix%2) && !(iy%2) ) {
                           xye[ix][iy][0] /= sqrt(2.0);
                           xye[ix][iy][1] /= sqrt(2.0);
                        }
#endif

                     /* Exact x,y to approximate sky RA,Dec. */
                        tcsVTsky ( roll, pitch, rotl, rma,
                                   xye[ix][iy][0], xye[ix][iy][1],
                                   m_spm1_i, m_cosys, sst, cst, m_spm2_i,
                                   ia, ib, np, xt, yt, zt, ga, gb,
                                   &tp[ix][iy][0], &tp[ix][iy][1] );

                     /* SPMs for this star and true pointing model. */
                        (void) tcsMedium (
                           t, MAXTRM,
                           model_true,
                           coeffv_true,
                           nterml_true,
                           ntermx_true,
                           nterms_true,
                           coeffn_true,
                           coform_true,
                           mount, ae2nm, roll, pitch, jbp, aux,
                           m_cosys, m_eqx, m_wavel,
                           r_cosys, r_eqx, r_wavel,
                           tp[ix][iy],
                           t0, st0, ttj, temp, press, humid, tlr, wavelr,
                           refa, refb, rfun, hm, tlat, diurab, amprms,
                           &ia_true,
                           &ib_true,
                           &np_true,
                           &xt_true,
                           &yt_true,
                           &zt_true,
                           ae2mt_true,
                           m_spm1_true[ix][iy],
                           m_spm1_i_true[ix][iy],
                           m_spm2_true[ix][iy],
                           m_spm2_i_true[ix][iy],
                           r_spm1_true,
                           r_spm1_i_true,
                           r_spm2_true,
                           r_spm2_i_true
                                         );

                     /* Use the SPMs to get slightly revised x,y.*/
                        tcsVTxy ( tp[ix][iy][0], tp[ix][iy][1],
                                  m_spm1_true[ix][iy],
                                  m_cosys, sst, cst,
                                  m_spm2_true[ix][iy],
```

```
                        rotl, rma, roll, pitch,
                        ia_true,
                        ib_true,
                        np_true,
                        xt_true,
                        yt_true,
                        zt_true, ga, gb,
                        &xye[ix][iy][0], &xye[ix][iy][1], &j );

               /* Scale to mm. */
                  for ( i = 0; i < 2; i++ ) {
                     xye[ix][iy][i] *= fl;
                  }
               }
            }
         /* End of "start of run" code. */
         }

      /* Time to log, and run in progress. */

      /* Time for MEDIUM update? */
         if ( ! ( itick % T_MED ) ) {

         /* Yes: SPMs for each star and true pointing model. */
            for ( iy = 0; iy < 3; iy++ ) {
               for ( ix = 0; ix < 3; ix++ ) {
                  (void) tcsMedium (
                     t, MAXTRM,
                     model_true,
                     coeffv_true,
                     nterml_true,
                     ntermx_true,
                     nterms_true,
                     coeffn_true,
                     coform_true,
                     mount, ae2nm, roll, pitch, jbp, aux,
                     m_cosys, m_eqx, m_wavel,
                     r_cosys, r_eqx, r_wavel,
                     tp[ix][iy],
                     t0, st0, ttj, temp, press, humid, tlr, wavelr,
                     refa, refb, rfun, hm, tlat, diurab, amprms,
                     &ia_true,
                     &ib_true,
                     &np_true,
                     &xt_true,
                     &yt_true,
                     &zt_true,
                     ae2mt_true,
                     m_spm1_true[ix][iy],
                     m_spm1_i_true[ix][iy],
                     m_spm2_true[ix][iy],
                     m_spm2_i_true[ix][iy],
                     r_spm1_true,
                     r_spm1_i_true,
                     r_spm2_true,
                     r_spm2_i_true
                                   );
               }
            }

         /* MEDIUM update complete. */
         }

      /* Time to log, and run in progress. */

      /* Compute true image x,y positions. */
         for ( iy = 0; iy < 3; iy++ ) {
            for ( ix = 0; ix < 3; ix++ ) {

            /* Start by getting (non-rotating) xi,eta. */
               tcsVTxe ( tp[ix][iy][0], tp[ix][iy][1],
                         m_spm1_true[ix][iy],
                         m_cosys, sst, cst,
                         m_spm2_true[ix][iy],
                         roll, pitch,
```

```
                              ia_true,
                              ib_true,
                              np_true,
                              xt_true,
                              yt_true,
                              zt_true, ga, gb,
                              &xiim, &etaim, &j );

#if PM_ENABLED
                  /* Introduce rotator axis error. */
                     xiim += DXI * AS2R;
                     etaim += DETA * AS2R;
#endif

                  /* Transform into (rotating) x,y. */
                     tcsXe2xy ( xiim, etaim, rotl, rma, roll, pitch,
                                &xym[ix][iy][0], &xym[ix][iy][1] );


                     for ( i = 0; i < 2; i++ ) {
                        xym[ix][iy][i] *= fl;

                     /* If start of run, note. */
                        if ( ! itick ) {
                           xye[ix][iy][i] = xym[ix][iy][i];
                        }
                     }

                  /* Update the trail statistics. */
                     dx = xym[ix][iy][0]-xye[ix][iy][0];
                     dy = xym[ix][iy][1]-xye[ix][iy][1];
                     dr2 = dx*dx + dy*dy;
                     sdr2 += dr2;
                     dr2max = dr2max > dr2 ? dr2max : dr2;
                     sel += el;
                     n++;
                  }
               }

#if REP_DIST
            /* Report distortions and shifts. */
               slaFitxy ( 6, 9, (double(*)[2]) xye,
                                (double(*)[2]) xym, coeffs, &j );
               printf("%7.2f",(t-tai)*86400.0);
               slaDcmpf ( coeffs, &xz, &yz, &xs, &ys, &perp, &orient );
               printf ( "%9.5f %9.5f %9.5f %9.5f %9.5f %9.5f %8.2f\n",
                        xz, yz, xs, ys, perp/D2R, orient/D2R, pitch/D2R );
#endif

#if REP_TRAIL
            /* Report trailed image positions. */
               for ( iy = 0; iy < 3; iy++ ) {
                  for ( ix = 0; ix < 3; ix++ ) {
                     dx = xym[ix][iy][0]-xye[ix][iy][0];
                     dy = xym[ix][iy][1]-xye[ix][iy][1];

                     l = itick;
                     ih = (int) ( l / ( 3600L * TPS ) );
                     l = l % ( 3600L * TPS );
                     im = (int) ( l / ( 60L * TPS ) );
                     l = l % ( 60L * TPS );
                     sec = ( (double) l ) / ( (double) TPS );

                     xtrail = xye[ix][iy][0] + dx * ( (double) MAG );
                     ytrail = xye[ix][iy][1] + dy * ( (double) MAG );
#if N_TOP == NO
                     ytrail = -ytrail;
#endif
#if E_RIGHT
                     xtrail = -xtrail;
#endif

                     printf (
   "%+12.6f %+12.6f %+10.6f %+10.6f %2d %2d %2.2d %2.2d %05.2f %6.1f %5.1f\n",
                             xtrail, ytrail,
```

```
                              dx, dy,
                              ix-1, iy-1, ih, im, sec,
                              az/D2R, el/D2R );
                  }
               }
#endif

            /* End of "run in progress" code. */
            }

         /* End of "time to log" code. */
         }

      /* End of "running" code. */
      }

#if VERBOSE
   /* Report statistics. */
      if ( ! low ) {
         printf (
    "\nMax trail = %5.3f arcsec, RMS = %5.3f arcsec, mean el = %4.1f.\n",
                     sqrt ( dr2max ) / ( AS2R * fl ),
                     sqrt ( sdr2 / ( (double) n ) ) / ( AS2R * fl ),
                     sel / ( (double) n ) / D2R
                );
      }
#endif

   /* Next simulation, using a new star (if any). */
      if ( ONE_STAR ) break;
   }

/* Finished. */
   return 0;

    }
```

APPENDIX B – Test Points Used to Sample the Celestial Sphere

In one of its modes of operation, the exposure simulator carries out multiple tests, each in a different place in the sky. Afterwards, the maximum trail length as a function of sky position can be studied. Here is the list of test hour angles and declinations (both in degrees). The pattern covers the whole celestial sphere, from pole to pole; for any given series of tests only the visible subset will be used.

```
 11.595749 -83.667263
272.386633 -85.488041
194.637223 -80.564199
 66.514868 -79.210667
277.811224 -76.740584
137.709769 -80.281524
 26.516307 -73.865900
239.648990 -73.563275
 97.686882 -73.735881
308.295970 -72.959418
148.326710 -71.682098
348.936060 -72.771322
204.990247 -72.434655
 67.658071 -68.900155
274.939648 -66.715471
120.313564 -66.585021
325.338904 -65.807507
170.028707 -66.478778
 10.223086 -65.778485
210.460449 -64.512325
 45.663414 -65.135020
248.494057 -65.017739
 94.309879 -64.860207
302.612105 -63.665177
146.779590 -61.804073
347.661656 -61.041838
187.469131 -60.127617
 26.530485 -58.682643
225.758284 -58.217930
 63.814567 -58.056690
261.717331 -57.612448
 98.916944 -56.714968
293.157566 -56.726319
130.125704 -57.544474
329.191404 -57.002071
167.609400 -56.979200
  6.384161 -56.622996
205.844047 -55.977160
 45.861449 -55.568397
243.876494 -55.132577
 81.147101 -54.610337
277.405811 -53.196075
113.672721 -52.217158
309.140454 -51.480796
144.655839 -50.510769
340.357752 -49.458298
175.409203 -48.534457
  9.846326 -47.667880
203.634563 -47.286247
 37.603518 -47.986437
232.314131 -48.443175
 67.217790 -48.441337
261.829592 -48.134460
 96.750436 -47.753278
```

```
292.340347 -47.647465
128.117000 -48.107850
323.585676 -48.271150
159.198691 -48.503875
354.671260 -48.388677
189.506508 -47.884807
 24.005224 -46.861054
218.497069 -46.054391
 52.700370 -45.226603
246.342739 -44.401899
 79.661343 -43.487661
272.765599 -42.182046
106.060345 -40.752245
299.203622 -39.420437
131.785932 -38.705794
324.206568 -38.356922
156.683419 -38.350095
349.360816 -38.302547
182.148427 -38.147409
 14.946179 -37.921688
207.668951 -37.932907
 40.132542 -38.229173
232.430381 -38.404043
 64.797656 -38.428379
257.362129 -38.446998
 90.345808 -38.722617
283.829112 -39.434910
117.556066 -40.362091
310.884994 -40.740391
143.841298 -40.521221
336.726987 -39.985799
169.693851 -39.326272
  2.681354 -38.453726
195.567946 -37.502093
 28.291241 -36.536550
220.635123 -35.698161
 52.674153 -35.021658
244.555770 -34.484375
 76.409019 -33.994350
268.143965 -33.426556
 99.767958 -32.889068
291.221918 -32.422153
122.570572 -32.151541
313.976911 -31.892814
145.350264 -31.542835
336.512492 -31.078003
167.474927 -30.622904
358.424121 -30.143606
189.487998 -29.663252
 20.666423 -29.141846
211.810819 -28.677803
 42.921312 -28.336063
234.036077 -28.025509
 65.151310 -27.727180
256.235940 -27.471332
 87.357826 -27.310781
278.558146 -27.165930
109.835787 -26.926557
301.096267 -26.559332
132.319151 -26.265552
323.547646 -26.039750
154.796074 -25.925327
346.040442 -25.868616
177.254296 -25.868822
  8.474422 -25.842499
199.710838 -25.749405
 30.954919 -25.527188
```

```
222.119705 -25.224718
 53.212990 -24.911728
244.261853 -24.623144
 75.306251 -24.358901
266.331342 -24.084241
 97.364128 -23.847580
288.439654 -23.646007
119.663849 -23.524099
310.988184 -23.361359
142.341402 -23.107273
333.708918 -22.703766
165.070755 -22.213367
356.366732 -21.652658
187.523923 -21.084823
 18.539609 -20.506904
209.447013 -19.958533
 40.299079 -19.449918
231.065600 -18.967727
 61.750784 -18.506871
252.344754 -18.060887
 82.865624 -17.625909
273.287179 -17.163791
103.596034 -16.671203
293.774824 -16.142626
123.836452 -15.631040
313.797660 -15.129846
143.683165 -14.682263
333.514176 -14.284898
163.320451 -14.003332
353.150041 -13.837654
183.041393 -13.777601
 13.007332 -13.758357
203.045664 -13.755918
 33.161470 -13.734770
223.321340 -13.669766
 53.508931 -13.564614
243.696368 -13.428031
 73.882354 -13.275007
264.041617 -13.103564
 94.177660 -12.934428
284.280396 -12.764112
114.367064 -12.603022
304.424837 -12.422210
134.454132 -12.230607
324.457755 -12.016468
154.458075 -11.781100
344.482613 -11.508476
174.552123 -11.200015
  4.675055 -10.854400
194.844970 -10.487720
 25.057140 -10.109298
215.280910  -9.726966
 45.510725  -9.341839
235.727696  -8.953785
 65.936433  -8.567222
256.121111  -8.180957
 86.295315  -7.801321
276.453109  -7.421485
106.608707  -7.050218
296.748688  -6.681974
126.875777  -6.336655
316.988915  -6.007286
147.086785  -5.710442
337.171032  -5.444135
167.232373  -5.223482
357.276158  -5.036726
187.306436  -4.873262
```

```
 17.341999  -4.711524
207.379529  -4.536972
 37.429539  -4.337832
227.482824  -4.106596
 57.545812  -3.848428
247.606515  -3.566850
 77.673642  -3.272899
267.736804  -2.967163
 97.804846  -2.659620
287.866573  -2.349040
117.931672  -2.040837
307.994731  -1.732137
138.059584  -1.427631
328.125976  -1.128558
158.194585  -0.837816
348.268422  -0.556914
178.342400  -0.285921
  8.421305  -0.021325
198.500310  +0.242650
 28.586896  +0.510310
218.670870  +0.789778
 48.759464  +1.079494
238.842508  +1.376910
 68.927387  +1.672715
259.003997  +1.964852
 89.080085  +2.247321
279.145323  +2.523440
109.208830  +2.790710
299.261214  +3.049010
129.307270  +3.294517
319.343664  +3.524578
149.370828  +3.736509
339.388603  +3.923991
169.392953  +4.087620
359.392451  +4.230985
189.390641  +4.367263
 19.408689  +4.509200
209.462053  +4.680910
 39.573532  +4.895389
229.732238  +5.165259
 59.932902  +5.486289
250.151444  +5.853606
 80.384394  +6.254475
270.615565  +6.684237
100.850564  +7.132898
291.073921  +7.590618
121.298258  +8.058043
311.522087  +8.525593
141.754689  +8.999960
331.996945  +9.470568
162.246144  +9.946441
352.500763 +10.417013
182.736534 +10.875966
 12.940598 +11.297731
203.092929 +11.664249
 33.196413 +11.946264
223.245758 +12.145200
 53.271825 +12.277467
243.286038 +12.372439
 73.311132 +12.448798
263.342725 +12.523644
 93.391670 +12.602124
283.446949 +12.692437
113.526179 +12.794669
303.624339 +12.885469
133.744202 +12.960797
323.879511 +12.990787
```

```
154.019932 +12.975244
344.157414 +12.887258
174.261130 +12.753532
  4.322488 +12.602791
194.320589 +12.492681
 24.250628 +12.475508
214.104328 +12.634089
 43.928587 +12.975184
233.756613 +13.471505
 63.635807 +14.066151
253.581264 +14.712899
 83.621787 +15.369852
273.767151 +16.013822
104.051956 +16.620301
294.462323 +17.160478
124.998445 +17.663928
315.645581 +18.121921
146.405624 +18.570297
337.278675 +18.981695
168.239635 +19.396740
359.295080 +19.801593
190.457571 +20.198268
 21.720256 +20.582352
213.015350 +21.004280
 44.320619 +21.424502
235.596529 +21.809694
 66.819283 +22.140828
257.952339 +22.436697
 88.994319 +22.728472
279.967596 +23.054315
110.960016 +23.422411
301.975176 +23.778998
133.017688 +24.156163
324.082586 +24.520540
155.198113 +24.912464
346.393262 +25.236692
177.618933 +25.469048
  8.831581 +25.602288
200.031529 +25.700057
 31.267974 +25.755573
222.527099 +25.848639
 53.808302 +26.037101
245.090738 +26.331573
 76.397763 +26.671624
267.721859 +26.974714
 99.039608 +27.130850
290.244490 +27.198075
121.374401 +27.338042
312.480717 +27.508496
143.601544 +27.717428
334.711481 +27.873879
165.819330 +28.103741
356.993661 +28.402514
188.148352 +28.690797
 19.163412 +28.947636
210.114869 +29.327265
 41.189176 +29.790242
232.433578 +30.262509
 63.819662 +30.696953
255.243022 +31.119520
 86.725979 +31.627580
278.267726 +32.244476
109.896706 +32.931439
301.653747 +33.541536
133.538190 +34.133286
325.528428 +34.719252
157.750855 +35.466758
```

```
350.365618 +36.233119
183.286255 +36.952006
 16.299560 +37.638563
209.192725 +38.371156
 41.955352 +39.033032
234.652105 +39.518105
 67.521672 +39.663226
260.822248 +39.258083
 94.600729 +38.539530
288.051699 +38.084749
120.932789 +38.010792
313.448856 +37.980548
145.834470 +37.918058
338.243481 +37.510624
170.813442 +37.183102
  3.666922 +37.178738
196.584906 +37.387472
 29.368355 +37.407280
221.852016 +37.370987
 54.142322 +37.532921
246.445356 +38.071290
 79.019913 +39.000022
272.065395 +40.476145
105.419639 +42.188760
298.766381 +43.472006
132.400895 +44.419387
326.389414 +45.147680
160.869096 +46.000166
355.534323 +46.806357
190.081842 +47.473121
 25.063862 +47.560964
220.303845 +47.463739
 55.696788 +47.277910
251.149799 +47.047559
 86.574055 +46.596551
281.760985 +46.697549
116.909561 +47.417977
311.912036 +47.795543
147.080447 +47.816467
341.886184 +46.831001
175.847824 +46.083177
  9.646210 +46.480923
203.976742 +47.722105
 39.014532 +49.113367
234.325322 +50.269304
 69.646865 +51.439975
265.517757 +52.441503
102.175894 +53.698705
298.973308 +54.758364
136.702616 +55.134615
335.327700 +54.893489
174.769624 +54.845439
 13.559242 +55.287086
211.958201 +56.000359
 50.771371 +56.431313
249.554649 +56.463323
 86.414351 +55.390413
281.544769 +55.420503
118.402668 +56.746222
316.357123 +57.334541
154.097879 +58.361760
353.107530 +58.965535
192.629380 +60.372124
 32.933458 +61.111622
234.237076 +61.420121
 78.277192 +62.651836
284.479455 +63.925858
```

```
129.962435 +64.583120
333.737933 +64.381035
171.489398 +63.512215
  9.926947 +64.365920
210.756087 +65.615647
 53.829315 +65.974983
259.030471 +66.409622
105.044573 +67.706948
312.050115 +68.077117
168.367824 +71.844854
 25.645325 +71.979696
234.156397 +71.102597
 74.770805 +71.597554
279.731845 +73.143546
134.935537 +73.716237
350.672704 +74.335886
211.329987 +77.419819
 94.437508 +79.384604
312.810534 +77.952853
163.962106 +81.919499
 34.347491 +81.909835
269.614696 +85.657075
```

The file is produced by the program **`stars.c`**. Inside that program the coordinates appear as constants. The original values were generated as follows:

1. Points were placed at roughly equally-spaced locations on a sphere, arranged so as to give smooth coverage of both axes (as opposed to being at discrete elevations for example).

2. In an iterative process, the points were allowed to move under the influence of an inverse square law repulsive force.

3. The simulation was stopped after a certain number of iterations, once a sufficiently "low energy" state had been reached. (Convergence to some final state never occurs: the points continue to move.)