Slalib Replacement

Below is the list of SLALIB functionality we will need to replace in the catalogs framework. It is all located in /lsst/sims/catalogs/measures/astrometry /Astrometry.py

I will keep track of how the functions are called, what the arguments are, and whether or not they appear to be declared in palpy/cextern/pal/pal.h. I will save the question of whether or not they are actually equivalent until I can test that identical inputs result in identical outputs for the two functions (the SLALIB version and the PALPY version)

- slaDsep – accepts 4 doubles; returns a double; in the function angularSeparation – exists in pal.h
- slaEpj – accepts a double; returns a double; in functions applyPrecession, applyProperMotion – exists in pal.h
- slaPrenut – accepts 2 doubles and a pointer to a 3 by 3 ndarray of doubles; in function applyPrecession – exists in pal.h; **however, it does not appear to be wrapped into python**
- slaPm – accepts 8 doubles and 2 pointers to doubles; in function applyProperMotion – exists in pal.h
- slaEqgal – accepts 2 doubles and 2 pointers to doubles; in functions equatorialToGalactic, galacticToEquatorial – exists in pal.h
- slaMappa – accepts 2 doubles and a pointer to a 1-D ndarray of doubles (21 elements long); in function applyMeanApparentPlace – exists in pal.h
- slaMapqk – accepts 6 doubles, a pointer to a 1-D ndarray of doubles (21 elements long) and 2 pointers to doubles; in function applyMeanApparentPlace – exists in pal.h
- slaAoppa – accepts 12 doubles and a pointer to a 1-D ndarray of doubles (14 elements long); in function applyMeanObservedPlace – exists in pal. h
- slaAoppa_nr – accepts 12 doubles and a pointer to a 1-D ndarray of doubles (14 elements long); in functions applyMeanObservedPlace, applyApparentToTrim – does not exist in pal.h; maybe this is something we defined; it also does not exist in the slalib.h that comes with pyslalib
- slaAopqk – accepts 2 doubles, a pointer to a 1-D ndarray of doubles (14 elements long) and 5 pointers to doubles; in function applyMeanObservedPlace – exists in pal.h; **does not appare to be wrapped into python**
- slaAopqk_nr – accepts 2 doubles, a pointer to a 1-D ndarray of doubles (14 elements long) and 5 pointers to doubles; in functions applyMeanObservedPlace, applyApparentToTrim – does not exist in pal.h; maybe this is something we defined; it also does not exist in the slalib. h that comes with pyslalib
- slaDe2h – accepts 3 doubles and 2 pointers to doubles; in functions applyMeanObservedPlace, applyApparentToTrim, equatorialToHorizontal – exists in pal.h
- slaRefco – accepts 9 doubles and 2 pointers to doubles; in function refractionCoefficients – palRefco accepts 8 doubles and 2 pointers; palRefro accepts 9 doubles and 2 pointers; when I look at slalib.h in pyslalib, I see the same dependency as in PALPY; **does not appear to be wrapped into python**
- slaRefz – accepts 3 doubles and a pointer to a double; in function applyRefraction – exists in pal.h; **does not appear to be wrapped into python**
- slaGmsta – accepts 2 doubles; returns a double; in function calcLast – exists in pal.h

Though some of the functions we require are not wrapped into Python, it is, in fact, very easy to make them so. Simple edits to the pal.pyx file in PALPY declaring the missing functions and calling them from C makes the functions available from Python.

I have compiled SLALIB both from the shared object file slalsst.so provided me by Andy Connolly and from the publicly available software package pyslalib. I have compared the outputs of PALPY with the outputs of SLALIB given identical inputs. I used unit test subroutines in which each PALPY /SLALIB function was given many random inputs. My results are as follows.

- Aoppa – according to documentation, this function "Precomputes apparent to observed place parameters required by slaAopqk and slaOapqk". Comparison between PALPY and pyslalib shows that the two routines give consistent outputs to within one part in 10^5. Comparison between PALPY and slalsst.so gives a discrepancy of a few 0.1 arcseconds in the output longitude and latitude parameters. I do not understand the origin of this discrepancy.
- Aopqk – "Quick apparent to observed place." As long as you are considering a location on the sky whose zenith distance is less than 0.4 pi radians, PALPY is consistent with both pyslalib and slalsst.so to within 1 part in 10^10. If the zenith distance is greater than that, the agreement between PALPY and SLALIB quickly becomes unacceptably high. This could be because of nonsense inputs.
- De2h – "Convert equatorial to horizon coordinates." PALPY agrees with both pyslalib and slalsst.so to one part in 10^10
- Dsep – gives the angular separation of two points on the sky. PALPY agrees with both pyslalib and slalsst.so to 3 parts in 10^10
- Epj – "Conversion of modified Julian Date to Julian Epoch." PALPY agrees with pyslalib and slalsst.so exactly.
- Eqgal – "Transformation from J2000.0 coordinates to IAU1985 equatorial coordinates." PALPY agrees with both pyslalib and slalsst.so to 2 parts in 10^10
- Gmsta – Converts from universal to sidereal time. PALPY disagrees with pyslalib and slalsst.so by 0.3 arcseconds. However, this is because PALPY uses the IAU2006 conventions. SLALIB uses the IAU1985 conventions.
- Mappa – "Compute star-independent parameters in preparation for conversions between mean place and geocentric apparent place." PALPY agrees with pyslalib and slalsst.so to within a few 10^-5 arcseconds. Note: PALPY and pyslalib disagree about whether the precession-nutation matrix should be output in row-major or column-major order. PALPY and slalsst.so agree on this ordering.
- Mapqk – "Quick mean to apparent place: transform a star RA,Dec from mean place to geocentric apparent place, given the star-independent parameters." PALPY agrees with pyslalib and slalsst.so to within a few 10^-5 arcseconds.
- Pm – "Apply corrections for proper motion to a star RA,Dec." PALPY agrees with pyslalib and slalsst.so to within a few 10^-4 arcseconds. However, this agreement breaks down if the parallax to the star being considered is very low. The threshold value appears to be a parallax of 0.00045 arcseconds. Below this value, PALPY disagrees with SLALIB on the order of a few hundred arcseconds. Above this value, PALPY agrees with SLALIB at the sub-milliarcsecond level.
- Prenut – calculate the precession-nutation matrix. PALPY agrees with pyslalib and slalsst.so to within a few parts in 10^6.
- Refco – calculate refraction coefficients. PALPY agrees exactly with pyslalib. PALPY agrees with slalsst.so to with a few 10^-3 arcseconds.
- Refz – "Adjust an unrefracted zenith distance to include the effect of atmospheric refraction, using the simple A tan z + B tan3 z model." PALPY agrees exactly with pyslalib. PALPY agrees with slalsst.so to with a few parts in 10^10.