

Observation Planning Software

Observation planning software helps astronomers answer their questions before their nights on the telescope. Here are some examples:

- During which period of the night will my science target be observable?
- During that time interval, when is that galaxy, star, or solar system object *best* observed?
- Is there a bright star near my faint science target that I can use as a reference point?

To make efficient use of valuable telescope time (one dollar per second in some cases) observers must arrive well prepared for their nights. Answers to questions like those given above must be worked out days, or even weeks, ahead of time. This chapter provides an introduction on how to write the software tools required for this preparation.

Airmass Plotting Tools

Following is a typical scenario for an astronomer preparing for their night:

My favorite target is in Orion. Is it observable tonight? Seven weeks from now? Next Tuesday a few hours before sunrise?

These are typical astronomer questions for which simple software tools can provide quick, easy answers. The most popular output format for tools that answer this type of question is an hour-by-hour air-mass table (see Table 1) or plot (see Fig. 1).

Let's quickly review the quantity "air mass." The term originates from the quantity of atmosphere ("air") that the line of sight takes from our telescope to your target (star, galaxy, or solar system body). When your target is "at zenith" (directly over head) the air mass is defined to be 1.0. When it is much lower, perhaps a few hours after rising or a few hours before setting, the distance through the air doubles; the air-mass is 2.0. At what viewing angle does this occur? The answer is easy thanks to the simple

The erratum of this chapter can be found under DOI: [10.1007/978-1-4614-7058-8_7](https://doi.org/10.1007/978-1-4614-7058-8_7)

Table 1 Hour-by-hour airmass table. Airmass values greater than 2.2 are italicized. Hourly airmass for Vega on 2012 Aug 10 (UT date)

Local	UT	LMST	HA	airmass
20 00	1 00	16 50	-1 48	1.086
21 00	2 00	17 50	-0 48	1.022
22 00	3 00	18 50	-0 12	1.008
23 00	4 00	19 50	1 13	1.042
0 00	5 00	20 50	2 13	1.130
1 00	6 00	21 50	3 13	1.295
2 00	7 00	22 51	4 13	1.590
3 00	8 00	23 51	5 13	2.149
4 00	9 00	0 51	6 13	3.416
5 00	10 00	1 51	7 14	8.148

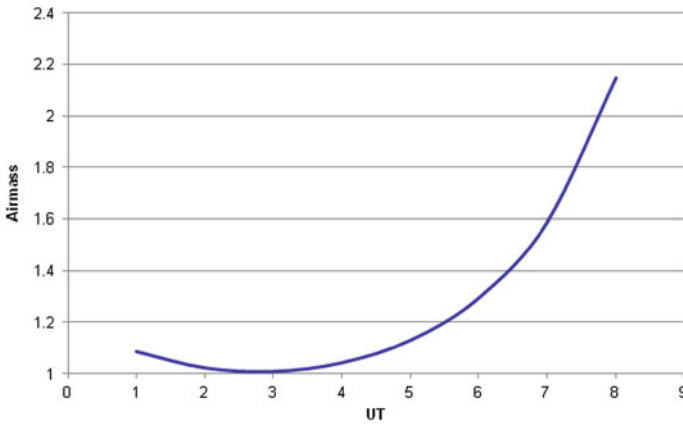


Fig. 1 Hour-by-hour airmass plot

properties of a 30-60-90 triangle (see Fig. 2). This 30° “elevation above horizon” is often as far as an astronomer wants to go. (Several factors distort telescope images of objects when they are low in the sky and this can reach a point, at roughly air-mass 2.2 or so, where it becomes problematic). So it is useful if an air-mass plotting tool produces output that designates this limit. For example, as shown in the last two lines of Table 1, an hourly table of airmass should highlight epochs corresponding to excessive airmass by changing the style or color of the font.

We are now ready for our first block of exercises. But, first a few notes about the exercises you will find in this textbook.

- 1.1 Exercises are given at the end of each section. These require the student to apply the software development concepts and techniques given in that section. For many of these exercises, pseudo-code¹ is required to complete the answer. Several textbooks that cover software development contain sections on how to

¹ Also commonly referred to as ‘Structured English’.

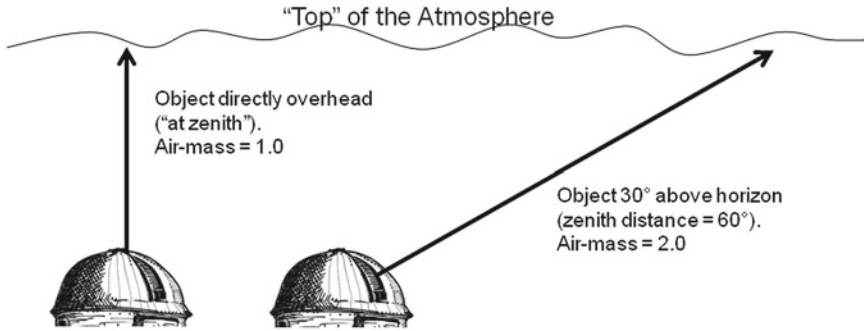


Fig. 2 Airmass concept

express algorithms using this technique (see references [1] and [2]). Also, we provide here, in the next section, a brief review of pseudo-code basics.

- 1.2 Pseudo-code. Our basic criteria for pseudo-code is that it should be possible to transcribe the pseudo-code to a given language (e.g., C++, Python, or IDL) with each line of pseudo-code corresponding to not more than 10 lines of code in the target language.

Suppose we wish to express the Eratosthenes sieve algorithm for computing all prime numbers between 1 and N . Here is an example for pseudo-code for that algorithm.

```

markers ← enumerated type { "unmarked", "prime",
                             "not prime" }
A ← new array of integers = [2, 3, 4, ..., N]
M ← new array of markers = {unmarked}
p ← 2
all_primes_found ← False
while ( ! all_primes_found )
  for i = 1 to N/p
    M[ p x i ] ← not prime
  q ← p
  while ( q < N )
    q++
    if q > N
      all_primes_found ← True
      break
    if M[q] is unmarked
      M[q] ← prime
      p ← q
      break
  end of while loop
end of while loop

```

When pseudo-code is requested in the exercises, provide a result which is at roughly the same level of detail as this example.

Exercises:

1. As above, the air-mass is 1.0 when the elevation is 90 and 2.0 when the elevation is 30. Give the general equation in terms of sin and or cos for air-mass as a function of elevation.
2. Use the result of exercise #1 to write pseudo-code to generate Table 1.
3. Use the language of your choice to implement the pseudo-code for exercise #2 above.

Target Planning

Catalog Search Tools

To determine a target's observability, first we have to learn its coordinates on the sky: its right ascension and declination. By virtue of having taken the prerequisites for this course, you are familiar with right ascension (RA) and declination (Dec), but let's review a few basics.

Copernicus taught us that the Earth is not the center of the universe, but the concept of the 'celestial sphere,' which stands as the basis of the RA/Dec coordinate system, takes us back to this overly-exalted position.

The earth rotates on its axis, every 24 h. So, it is very natural to give every observable object an 'X' coordinate which is a value between 0 and 24. The 'zero' is a well defined point in the sky referred to as the 'first point of Aries.' For astronomers, the east-west location of an object is given *not* in degrees between 0 and 360, but in *hours* between 0 and 24. The 'Y' coordinate, that designates how far north or south an object sits with respect to the Earth's equator, does not require this special treatment and is therefore expressed in good old-fashioned degrees. RA is a value between 0 and 24 h; declination is a value between $+90^\circ$ and -90° .

For small distances between objects, for example the distance between two of Jupiter's moons at a given instant, astronomers switch to a system in which both coordinates are given in arcseconds. An arcsecond is a small fraction of a degree; a sixtieth of a sixtieth (i.e., 3,600 arcseconds per degree). No spherical trigonometry is required in this case; the two objects are so close to one another that the small portion of the celestial sphere that they share can be treated as a Cartesian coordinate system. If Callisto is 3 arcseconds north of Europa and 4 arcseconds west, the total distance can be computed using basic Euclidian geometry; $\sqrt{(3^2 + 4^2)} = 5$ in this case. We will return to this example in more detail in the section titled "Offset Guide Star Planning" below.

Exercises:

1. Perform a simple catalog search as follows:
 - a. Use the IDL routine QUERYVIZIER² to find all stars brighter than seventeenth R-mag and within 180 arcseconds of RA = $11^h 33^m 47^s$ and Dec = $5^\circ 26' 12''$.
 - b. Repeat for RA = $11^h 33^m 47^s$ and Dec = $72^\circ 26' 12''$.
 - c. Was the difference in the two results expected? Why?
 - d. Write pseudo-code for wrapping this routine in a simple GUI.

Ephemeris Tools

In the previous section we reviewed the RA/Dec coordinate systems used for stars and galaxies. Every star and galaxy has a unique RA/Dec coordinate which, for the most part, it is stuck with for all of its very long life. (There are small drifts over time due to effects like proper motion, but this drift is relatively small.)

Now consider the *Wanderers* (the Greek origin for our English word “planet”). Every object in the Solar System has an RA/Dec which changes significantly from day to day (for outer planets like Saturn and Jupiter), from hour to hour (for nearby planets like Mars and Venus), and minute to minute (for very near objects like our moon and nearby asteroids).

Why? The closer an object is, the less we can account for all of its motion in the sky as being dominated by the Earth’s rotation. The RA distance between a star overhead, and the star that was overhead 1h ago, is just that: one “hour” of distance. An asteroid overhead, and a star a bit north of overhead,³ are at the same RA, but 1h previous the asteroid may have been one minute west of that same star. In this case we say that this asteroid has a “differential” or “non-sidereal” motion of one minute per hour.

There exist several “tips and tricks” for writing software tools to assist with planning for observations of Solar System objects (often referred to as non-sidereal objects):

² QUERYVIZIER comes with the IDL ASTRO LIB package which can be downloaded from <http://idlastro.gsfc.nasa.gov/>.

³ Thus far we have been using the word “overhead” to broadly refer to any object (star, galaxy, or planet) that you would see if you looked straight up. Astronomers, of course, use words that more precisely define the situation. A star which is really, dead nuts, straight up from your position on Earth (AKA, your “ground position”) is said to be “at zenith.” An object which is at its highest point in the sky (i.e., that instant at which it is no longer rising, yet has not yet started setting) is “transiting” or “on the meridian” (short for “on the local meridian”). Any two objects, both in this condition, will be one north of the other. Here we use the astronomer’s version of “north” which means: closer to that fixed point in the sky, directly above the Earth’s axis, and quite close to the North Star. So, in our example here, the asteroid and the star both lie “on the meridian,” but only one could be actually “at zenith.”

1. Do not use *Apparent* coordinates:

There is not only one system for RA/Dec coordinates. In fact there are at least 3 and these are called: FK4, FK5, and Apparent.⁴ The “apparent” system is in many ways a more intuitive choice, for solar system objects, but in fact, 95 % of astronomers use the same system as is used for stars and galaxies: FK5/2000. As a software developer working in the field of astronomy, you only need to remember this one “take away” message: If you are pressured by a purist to use apparent coordinates for your solar system software tools, resist.

2. Use JPL Horizons for sign and unit conventions:

In our example above, we noted that the asteroid’s “differential motion” was “one minute per hour;” and that the asteroid went from being one minute west of our star to being at the same RA as that star, over the space of 1h. There exists a web site, hosted by Jet Propulsion Laboratory (JPL) that, at the time of this writing, stands as the “Bible” for this type of information. That site would report the differential motion of the asteroid as “dra = +60 arc-seconds/hour.” As you might imagine, there are a dozen or more ways to represent this same information (degrees-per-minute, arcseconds-per-second, sign reversal for east vs. west, etc). The “take away” message here is: Just use JPL-Horizons conventions for all your software that deals with the differential motion of Solar System objects (period).

Exercises:

- (1) Generate an airmass plot for Mars as it could be observed from Mauna Kea on January 1, 2010, UT date, by following these steps:
 - (a) Use JPL Horizons to generate the airmass at 1h intervals between sunset and sunrise.
 - (i) Save the web page output as a file.
 - (ii) Edit the web page output file to generate a comma separated values (CSV) data file.
 - (iii) Open the CSV file in Excel© and generate the chart.
 - (b) Repeat the exercise using skybot.
 - (c) Write a one-page report describing how one might write software to automate this process. Include discussion of the pros and cons of using skybot versus JPL-Horizons with respect to automation.

⁴ The list could increase to 5 since a factor called the “equinox” (the zero point for an effect called “precession”) can be either the year 1950 or the year 2000 for FK4 and FK5; however, the only system used by serious astronomers (at the time of this writing) is FK5/2000. So we suggest never using any of the other 3 (FK5/1950, FK4/1950, or FK4/2000). So there are really just these two to chose from: FK5/2000 and Apparent; and, as in the text, really really only one: FK5/2000; even for solar system objects.

Table 2 Sample science targets and their offset guide stars

Science object	Ra	Dec	GuideStar	RA	Dec
Sagittarius A ^{*a}	17 45 40.04	−29 0 27.9	SO−2	17 45 40.04	−29 0 27.7
Ry Scuti	18 25 31.48	−12 41 24.2	TYC 5698	18 25 42.38	−12 44 44.5

^aThe black hole at the center of our galaxy. This RA/Dec for SO-2 given here is actually for a specific date; when it is furthest from the black hole during its 16 year orbit around it

Offset Guide Star Planning

To observe interesting objects, astronomers must frequently make use of otherwise boring, but very nearby, bright stars. Table 2 gives a few examples.

The software required for this planning requires the same basic distance calculation, so we will cover that operation first. Let's return to the example given in the previous section. As before, suppose Callisto is 3 arc-seconds north of Europa and 4 arc-seconds west of Europa. But now let's look at the absolute coordinates of each:

```
Europa    12 34 56 +30 06 07.1
Callisto  12 34 56 +30 06 10.1
```

For the second coordinate, declination, the differential is clear as written. The difference is only in the arc-seconds field (degrees and minutes are identical) and that difference is 3 arc-seconds; Callisto being north of Europa since the former's declination value is greater. But what happened to the RA value? It did not change value at all between the two, yet Callisto is 4 arc-seconds west of Europa.

Let's rewrite the absolute coordinates of the two bodies, but this time with more digits of precision.⁵

```
Europa    12 34 56.23 +30 06 07.100
Callisto  12 34 56.00 +30 06 10.100
```

OK, things are becoming a bit clearer. The Europa coordinates now indicate that it is west of Callisto, but by less than a half of one second so that, with the lower precision format used previously, the two values of RA for Callisto and Europa appeared to be identical. But how did 4 *arc-seconds* become 0.23 s.

Arc-seconds (seconds of *arc* as the name implies) differ from *seconds* (or what are sometimes, for clarity, called “seconds of time”). Let's re-write the coordinates with the superscript format⁶:

⁵ Notice that we always give declination one more digit of precision than right ascension. This is a real stickling point with some purists and used as an immediate indication of whether or not you know what you are doing. Why do we do this? See exercise 1 below for a hint to the answer.

⁶ The six number coordinate format used by astronomers for locating points in the heavens, sometimes called “HMS/DMS” comes in three popular varieties: superscript (usually the clearest, but hardest to display in a GUI; only for the true software ‘artiste’); colon-separated; or white-space-separated (the vehemence with which the relative benefits of these latter two is debated is only

Europa $12^h 34^m 56.23^s + 30^\circ 06' 07.100''$

Callisto $12^h 34^m 56.00^s + 30^\circ 06' 10.100''$

Arc-seconds differ from seconds by two factors:

- (a) There are fewer seconds in a complete circle than there are arc-seconds ($24 \times 60 \times 60$ vs. $360 \times 60 \times 60$; a factor of 15) and
- (b) It's a short walk around the world when you live on the north pole.

In this second cryptic comment we refer to the difference between lines of constant right ascension and lines of constant declination on the celestial sphere. These are directly analogous to lines of constant longitude and lines of constant latitude on the Earth's surface.

What's the difference? These full circles on the earth that pass through the north and south poles, the lines of longitude, are all the same size; great circles of approximately 40,000 km circumference. Same for the celestial sphere: The full circles that include both the north celestial pole (NCP) and the south celestial pole (SCP) are all the same size. For this reason, the declination difference we see in the respective declination fields in our Europa versus Callisto comparison is clear.

But the lines of constant latitude (on the earth), analogous to the lines of constant declination (on the celestial sphere), are not so nice. Up near the north pole, say at declination (for the celestial sphere) or latitude (for the earth's surface) 89.9° , the circle is tiny (the "short walk around the north pole" that we referred to earlier). How much smaller in circumference is this line of constant declination at say 60° declination than it is at the line of constant declination at say 0° declination (i.e., the celestial equator)? The answer in this case is: exactly one half. The answer in general is:

$$\text{circle circumference at declination } \theta = \text{circle circumference at equator} \times \cos(\theta)$$

We chose a simple case to arrive at a conversion factor of "exactly one half" for declination 60° .⁷

We have now given the two halves of the puzzle that lead to the following formula (and the explanation of how the Callisto/Europa east-west distance of 4 arc-seconds became 0.23 s of time):

$$\text{seconds of arc} = \text{seconds of time} \times 15 / \cos(\theta).$$

where θ is the declination. At the celestial equator, one second of time is 15 s of arc. At declination 60° , one second of time is 7.5 s of arc.

(Footnote 6 continued)

exceeded by debates over which is better: vi or emacs). For any of the formats, always put the sign on the first (degrees) field of declination (the fourth of the 6 values); even when it is positive (i.e., northerly) be sure to include the '+' sign.

⁷ Recall that $\cos(60)$ is 0.5 by virtue of the simple properties of a 30-60-90 triangle.

Let's dig a little deeper into the software techniques required to produce this result: "at declination 60° , one second of time is 7.5 s of arc." For our new example, we consider two stars, one is a single second of time west of the other.⁸ Here are their respective coordinates:

```
Hokusterne 12 34 55.0 +60 00 00.00
Sternehoku 12 34 55.1 +60 00 00.00
```

Hokusterne is west of Sternehoku by one arcsecond; i.e., Sternehoku will occupy the same points in the sky as Hokusterne, but constantly lagging behind by one second.⁹ To

Δ RA (the west-to-east distance) in arcseconds we must apply: *seconds of arc* = *seconds of time* $\times 15/\cos(\theta)$.

So we will be needing to compute the cosine of declination. Suppose we are doing this in C/C++ and therefore using the *cos* routine in *libm.so*. Clearly it is not possible to invoke the time-honored trig functions in that library with something like:

```
char *s = "60 00 00.0";
double c_dec = cos( s );
```

or even:

```
double d = 60.0;
double c_dec = cos( d );
```

since, as with most hard core math libraries, all angles are passed in as radians, not degrees.

There are two ways to go with this issue of converting "human readable" formats (i.e., HMS/DMS¹⁰) to "machine readable" formats. The first is the traditional method: scalar variables in pairs (SI units¹¹ for internal and HMS/DMS for external). For the second method we take the opportunity to employ an object oriented approach. Pros

⁸ We could not possibly use Europa and Callisto for our example at 60° declination. Why not?

⁹ Being able to determine this assertion directly from a reading of the coordinates is why astronomers measure distances using units of time.

¹⁰ Note that even for this shorthand notation for the 6-number coordinates of stars, the confusion between seconds of time is reinforced. Strictly speaking, the mnemonic should be "HMS/DMA" to indicate that the value in third position differs from the one in sixth position ('S' for seconds of time and 'A' for seconds of arc, respectively). But DMS is used since in a non-astronomy context, the 'S' stands for the shortened version of arcseconds, which can be called just "seconds." When working in astronomy, do not fall into this lazy habit. Always use arcseconds for "seconds of arc." A last comment on this HMS/DMS nomenclature: Sometimes the word "sexagesimal" is mistakenly used to refer to the HMS/DMS 6-number format. Avoid this. Sexagesimal has nothing to do with the number 6, but refers to base 60 and is therefore only applicable to the DMS part of HMS/DMS. Just use "HMS/DMS" and ignore the fact that "HMS/DMA" would be more correct in this context.

¹¹ This abbreviation actually comes, originally, from the French: *Système international d'unités*. For linear measures the suggested measure is clear: meters (even wavelengths for which there are always many leading zeros). As a programmer working in the field of astronomy, the key is to stick with the standard for angles. Always use radians (avoid degrees) for internal variables.

and cons exist for both approaches; the time-honored tradeoff between cutting edge versus tried-and-true. In the next few paragraphs, we delve into these two possible approaches a little deeper, and take the opportunity to generalize with respect to overall design considerations regarding software infrastructure.

The approach taken at many observatories for human-readable versus machine-readable coordinate representations is built up from SLALIB routines. In particular the SLALIB routines [3, 4] `SLA_DAF2R` and `SLA_CTF2R` convert DMS and HMS to radians, respectively. Though traditionally linked into C/C++ applications as a library, other bindings exist. For example, there exists a Python interface (`pyslalib`). Also, there are competing packages; for example `TPM` (Telescope Pointing Machine) which also has a Python interface (`PyTPM`). However, `SLALIB` is recommended. Why? It is mature and reliable. Although the conversions we are discussing appear simple, there are pitfalls (see exercise 5 below) and subtleties. These are addressed in `SLALIB` with care and rigor. The package benefits from nearly 50 years of refinement and debugging.¹²

So, the conversion method is straightforward (i.e., Use `SLALIB`), but what conventions and/or structures work best for representation. Is it better to simply carry around both representations, radians and HMS/DMS throughout your code in pairs of scalar variables, a character string for HMS/DMS and a floating point (e.g., a C double) scalar for radians; with some clunky naming convention like “_str” and “_rad” as the end of each name? Or does this situation call out for an object oriented approach with a class called “TargetCoordinates” with the appropriate methods (wrappers around `SLA_DAF2R` and `SLA_CTF2R`, for example) and class variables for both representations. Why stop there? Include Az/El representation, second order effects (e.g., proper motion), etc. But can we keep going? Introduce or adopt an existing framework like `LABVIEW`, `EPICS`, or `CORBA` (see commandment #4 in the subsection at the end of the control system chapter).

Pragmatism should rule the day, both here and for similar design decisions that will come up in your assignments for programming software systems for astronomy. For a 200 line Python utility required for a specific observing run, a clunky naming convention like that described above is sufficient. For a 3000 line Java GUI, create the object. For a 100,000 line instrument control system (ICS), adopt a framework.

Exercises:

1. Compute an offset as follows:
 - a. Repeat the search described in exercise 1a on page 7.
 - b. Calculate the offset from the search center ($RA = 11^h 33^m 47^s$ and $Dec = 5^d 26^m 12^s$) to the brightest found star.
 - c. Now compute the same offset in “arcseconds on the sky.”
 - d. Use one of these results to determine the distance as a single, radial value (in arcseconds on the sky).

¹² Consider the first sentence of the `SLALIB` manual: “`SLALIB` is descended from a package of routines written for the AAO 16-bit minicomputers in the mid-1970s.”

- e. To compute 1d above, which offset format (the result from 1b or the result from 1c) did you use? Why?

2. Pseudo-code

- a. Write the pseudo-code to perform the steps you determined for exercise 1 above.
 - b. Extend the pseudo-code for step 2a to operate on a list of stars.
3. Sketch a graphical user interface (GUI) to control the functionality which you designed in pseudo-code in step 2b above.
 4. Design a display that presents the following information, laid out in a way that is intuitive for the astronomer:
 - a. Base RA/Dec in HMS/DMS
 - b. Current RA/Dec in HMS/DMS
 - c. Offset delta RA/Dec; in arc-seconds
 - d. LST
 - e. HA

Hint: How are RA, LST, and HA related?

5. Write C-code to convert the declination in the following coordinates to radians:

- a. 13 02 44 +05 25 53.1
- b. 10 22 55 -00 01 10.3
- c. 10 20 55 +00 01 10.3

- i Give the difference in sky angles, north to south, between (b) and (c).
 - ii What extra conditional was required to properly handle 5b. Is there a way to convert DMS to radians without this extra conditional?
 - iii Generalize your code to work with all three formats (superscript, colon-separated, and white-space-separated).
6. Design and code C++ for the *TargetCoordinates* class for coordinate conversion discussed in the text above and re-factor your solution given in exercise 5 above to use this class.
 7. The term “dithering” is, at the time of this writing, the general term used by astronomers for taking images (or sometimes spectra) with the science target placed at different points on the detector. Figure 3 shows a typical pattern used.
 - a. Write pseudo-code for a dither script that moves the telescope in this pattern.
 - b. Re-write the script so that it is general purpose in that it is data driven (i.e., it reads a config file).
 - c. For (b) above, did you use absolute RA/Dec or relative motion in arcseconds for the contents of your config file? Which is better? Explain why.

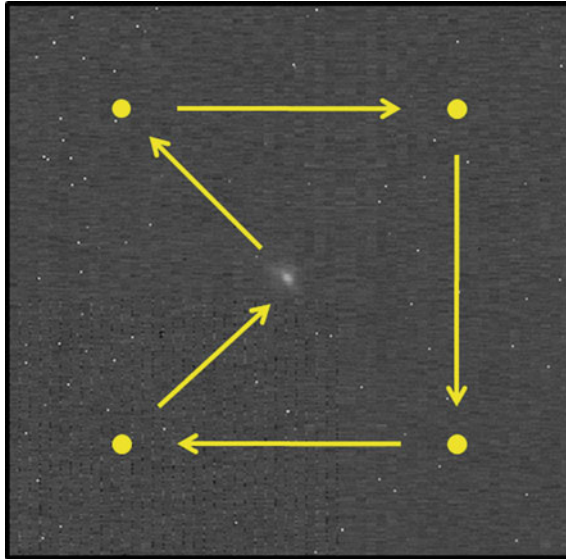


Fig. 3 The box4 dither pattern

References

1. The Design and Analysis of Computer Algorithms, Alfred Aho, J. Hopcroft, and Jeffrey Ullman, Addison-Wesley, 1974.
2. Introduction to Algorithms, Thomas Cormen, Charles Leiserson, Ronald Rivest, MIT Press and McGraw-Hill, 1990.
3. A rigorous algorithm for telescope pointing, Patrick Wallace, SPIE 4848, 2002.
4. SLALIB/C Users Manual, Patrick Wallace, 2006.

Software Systems for Astronomy

Conrad, A.R.

2014, XI, 95 p. 38 illus., 18 illus. in color., Softcover

ISBN: 978-1-4614-7057-1