

Detailed instructions for CCD dataprocessing in Mathematica to get the *seeing*

Info: across this document, blue background coloured text like this one gives context and detailed explanations before each step. Here, before starting, note that we will read data that had initial calibration made (flat fielding), in a CCD data reducing software like CCDOps (which we use for our CCD SBIG-7XE). This outputs a text file (in our case a 765×510 matrix of pixels with their ADU values). In Mathematica 5.0 we will read such file and calculate the seeing value of the observations.

Action: across this document, green background coloured text like this one lists the actions to make, in order to proceed with data reduction. So, after initial reading of the document, all blue background coloured text may be skipped.

1. Open "RoutineSeeing.nb" with Mathematica 5.0. Save, immediately, with a filename containing the place and date such as "EncumeadaAlta_04jul08". Do not forget to keep saving it frequently as you progress inside Mathematica.

The input textfile in ReadList in Mathematica is a CCD flat-calibrated raw data that must be previously prepared in an external software.

The routines only stop when strictly necessary (for inspection and decisions before resuming).

The initial ADU maximum in the plot is set at 60000, but it can be changed (a must, if you have larger values on the CCD data); only change the viewpoint as a last resource.

2. Change the "SetDirectory" according to the place where you have the CCD text file in your PC; and place its name on "ReadList", including the extension (e.g. Juncal_27jun.txt). Run **Routine 1** (place the cursor in the cell and click "shift-Return").

3. Mathematica requires input before proceeding: the horizontal×vertical dimensions of your CCD chip, in pixels (e.g. 765×510). You do this in this order (horizontal; then, vertical).

The idea of this first routine is to have a good 3D view of the data, where the vertical axis are the ADUs of each pixel. Make a note of its other output: the maximum value (m) of ADUs in these still “one-pixel” data. “m” will help you with the first decision: maximum vertical height in Routine 2 (change, before running it); a good standard is 35000, or 40000, for example, if your maximum is 33456 ADU. Also, before starting, decide on a tighter (x1, x2) - (y1, y2) matrix from the plot of Routine 1: trial and error; in the end, changing the viewpoint might be necessary too. PlotRange and ViewPoint are the two commands to interact with:

PlotRange -> {{200, 350}, {100, 250}, {0, 50000}}, ViewPoint -> {.2, 4, 2.5}

4. The first action is to create a document (e.g. Word) where you will place all relevant information. Right now, make a note of the peak ADU value (output). According to the graph output of Routine 1, run **Routine 2**, after changing the (x,y) and ADU limits in PlotRange (for the latter, use the maximum value of ADUs as a reference – output of Routine 1). For deciding, check the output of Routine 1 (enlarge the graph in Mathematica). Run Routine 2 again (several times), after changing the (x1, x2) - (y1, y2) matrix (and the ViewPoint), until satisfied. Make sure the base is square, i.e., $y_2 - y_1 = x_2 - x_1$. Make a note of the final x1, x2, y1, and y2.

The data will now be organized into 2x2 pixels, preparing “vector c” and “vector f” for analysis. These “vectors” represent the path of Polaris from the peak (vector c) and on the “second peak” side and on the “third peak” side for vector f. Both either stop when reaching a 2x2 pixel with <10000 ADUs or when entering an “infinite” cycle, being stopped at 100. Together, vectors c and f will originate vector h, the full path of Polaris to calculate the seeing in the end. Be patient, this routine might take up to one minute to conclude.

5. Run **Routine 3** (insert the final values of x1, x2, y1, and y2, in this order).

In what follows and, in fact, everytime there is a need to record the data somewhere more efficiently, we recommend that you use a word processor document for all analysis (by copy/pasting from Mathematica images, etc.)

6. You will now look at all output from Routine 3 carefully:

i) vector c – number of cycles; if 100/101 or, if smaller, **in case the vector bends back making a closed curve**, then must inspect its output plot carefully (with the help of peak value information, 2nd maximum and full vector path) to decide where to stop (maximum number of points – make a note; see Fig.1);

ii) vector f – proceed exactly as for vector c (now, however, we are talking about the 3rd maximum).

7. Run **Routine 4 only** if vector c and/or vector f must be revised (input the maximum number of points for each: must be greater than zero). Since vector h will result from “adding” vectors c and f, a final inspection must be made of c+f: insure no “cycling back” (on itself or c+f together) is taking place. It might well happen that further restrictions in the maximum number of points take place to vector c and/or vector f.

8.(If you ran it) Check the output of Routine 4 again, confirming all is well with the final vectors c and f.

9. Now, time has come to define the Polaris path and plot it (several ways): vector h. However, **Routine 5** might have to be edited first: if the “maximum number of points” of #7 is zero for vector c (f) then replace “h=Join[g,c]” with “h=f” (“h=c”).

Before advancing to the seeing calculation, we must ensure that vector h (the Polaris path) is auto-consistent. It might have to be sorted (if too vertical – or parts of it), have “lobe work” done (if more than 1/3 of the raw data does not have the Polaris track on it), or be reversed (c+f instead of f+c) – if the ADU profile is not consistent with the “track-on-top-of-data” plots.

10. Run **Routine 5** and **very carefully** check all (make notes of) its output:

- i) the vector itself;
- ii) number of points in vector h;
- iii) maximum and minimum ADU values in vector h;
- iv) Polaris path as a graph (vector h),
- v) on top of the 1x1 pixel data (greyscale), which is also plotted separately, and
- vi) on top of the 2x2 pixel data (greyscale), which is also plotted separately;
- vii) ADU profile (cf. bright and dark “spots” on the path/greyscale).

11. Following the careful analysis of #10:

i) if the ADU profile is not consistent with the path (as you follow along bright and darker spots) and if it has a “V” shape, then simply replace “h=Join[g,c]” with “h=Join[c,g]” and run **Routine 5** again (go back to #10);

ii) if the ADU profile is not consistent with the path but, this time, the points seem “all over the place”, and the path is almost vertical (say, less than 30° from vertical) – or parts of it, then, **before** the full command line “While [...] e=Append [...] l=1;” put: “e=h[[Ordering[h[[All,2]]]]];h=e;e={};” and run **Routine 5** again (#10); note that a similar correction might be necessary if many parts of the path are horizontal, this time using “h=hr[[Ordering[hr[[All,1]]]]]”;

iii) look at the output of vector h (all of it) and decide on removing points if a >33% rise/drop is present between the first/last point and the one immediately following/preceding (ADU profile); occasionally, two or more points might constitute an initial (or final) “cluster” if their average ADU is within 10% of the ADU value of each individual point (in this case, “cluster” works for “point” in what follows and the ADU value is the calculated average); if points are to be removed:

a) do so “manually”, after copy/pasting vector h in Mathematica and removing the points (deleting) from the vector, producing a final corrected version by giving the command-line “h={{..., ..., ..., ..., { ..., ..., ..., ... }}, { ..., ..., ..., ... } }” which replaces “h=Join[g,c]”;

b) run **Routine 5** again;

iv) if more than 1/3 of the raw data does not have the Polaris track on it (as judged by looking at the 1x1 pixel or 2x2 pixel plots) then “lobe work” must be considered. See #12 and

proceed (only after the final run(s) of Routine 5 have been made, according to i)-iii) above). Otherwise, jump to #16 (but only after all necessary i)-iii) corrections have been made).

When happy, make a note of the maximum and minimum value of vector h , as well as its number of points.

12. For “Lobe work”, the following steps must be made:

- i) jump to the special section “Lobe Fit” (after the Routine 6 section);
- ii) decide the $(x_1-x_2)-(y_1-y_2)$ rectangle where you want your data **removed** [normally corresponding to the part already with the Polaris track marked on it, and a couple 2x2 pixels more in horizontal/vertical distance, as appropriate];
- iii) run **Lobe – Routine 1** and input the x_1 , x_2 , y_1 , and y_2 values (all greater than zero!);
- iv) run **Lobe – Routine 3** and check its output as described in #6 above;
- v) do as #7-#9 above, this time running the corresponding “lobe” routines;
- vi) run **Lobe – Routine 5** and proceed as #11-i) to iii) above.

13. If you are happy with the “lobe work” done but you still need more data to fit (e.g. current data still has more than one-third of the track missing points on it) then you should run the “lobe work” section again, noting that now we proceed to the “lobe 2” routines, repeating #12 for all these.

Note, however, that if you need more than two lobes to fit, you will have to copy/paste the lobe 2 routine changing: “ $h_2=h$ ” to “ $h_4=h$ ”; “ al_2 ” to “ al_4 ” (three places); “ h_4 ” to “ h_5 ” (seven places). However, three lobes plus one main part should be very rare indeed!

14. Proceed to the section “Combining lobes” right at the end of the Mathematica notebook and run the **Routine** there, **after** the following changes:

- i) if more than one lobe, “Join[h_1,h_2]” to “Join[h_1,h_2,h_4]” (two lobes) or “Join[h_1,h_2,h_4,h_5]” (three lobes);
- ii) if the path is too vertical (or parts of it), “ $h = \text{Sort}[hr]$ ” to “ $h=hr[[\text{Ordering}[hr[[\text{All},2]]]]]$ ” (if many parts are horizontal, however, use, instead, “ $h=hr[[\text{Ordering}[hr[[\text{All},1]]]]]$ ”);
- iii) adjust “AspectRatio” to make a realistic plot of the path – iterative with each output.

15. Look at the output of vector h (all of it) and check for any “border effect” (Fig.2). If points are to be removed, Routine 5 must be run again, **after** removing the point(s) from vector h “manually”, after copy/pasting vector h in Mathematica and removing the points (deleting) from the vector, producing a final corrected version by giving the command-line “ $hr=\{\{..., ..., ..., ..., \{..., ..., ..., \dots\}\}$ ” (replacing “ $hr=\text{Join}[\dots]$ ”). When happy, make a note of the maximum and minimum value of vector h , as well as its number of points.

We now start the routines for seeing determination (three independent subroutines, inside “Routine 6”). We will only do parable fit (Routine 6.2) **if** the circle fit (Routine 6.1) fails: the parable fit is usually ridiculous and unphysical (Polaris traces a circle in the sky). There is a third option, even when the fit works (circle or parable) but the seeing values are ridiculous (e.g. 300”).

This usually happens when the path is too vertical. In this case, the “horizontal” fit (Routine 6.3) should be run.

16. [if you executed #11-ii) or #14-ii) then first replace "Sort[e]" by "e[[Ordering[e[[All,2]]]]]" (or "e[[Ordering[e[[All,1]]]]]"); then...] Run **Routine 6.1** and carefully check its output:

- i) fitting quality of the data points (Polaris track);
- ii) overall fitting of the raw path (greyscale): note the curvature. You have to decide on one of two options (circle “up” or circle “down”);
- iii) check if the seeing value makes sense;
- iv) if all is fine, make a note of the seeing value, “peak-to-1st” and “peak-to-last”, and “longitudinal seeing”. And you’re done!
- v) if something did not work, carry on to #17.

17. In case #16 failed, run **Routine 6.2**. This will write the file “parabole.dat” into your working directory. Read this file into, e.g., Excel, organizing it into three columns: you will use the first two (x,y) with the external program zunzun.com (curve fitting¹) doing the following:

- i) from the drop - down menu "2D Functions" pick "polynomial" and then click on "2nd order (quadratic)";
- ii) from the drop - down menu pick "Text data editor", leave the defaults as they are, click on "clear all text" and place in the window a copy of the two Excel columns: make sure they are all into the left (no space there) since the routine ignores any line that does not start with a number - including spaces; click on "submit" above;
- iii) after a few seconds you get the result; the first thing to do is to download the created PDF with all the information needed (click on "a single PDF file");
- iv) go “back” in the browser and, from the drop-down menu "Coefficients and Text Reports", pick the option "Coefficients": copy them into a scratchpad (a, b, and c);
- v) from the drop-down menu "Data Graphs", pick "Y data vs. X data with model" and copy the image to a word processor or save it on disk (to compare it with the fit you will get later in Mathematica).

18. Resume Routine 6.2 where it stopped: inputting the “a, b, and c” coefficient values of #17-iv) – decimal and **not** scientific notation. Check if the fit (cf. #17-v)) and the seeing value make sense. If this is the case, make a note of the seeing value, “peak-to-1st” and “peak-to-last”, and “longitudinal seeing”. And you’re done! Otherwise, go to #19 for a final hypothesis on getting a seeing value out of the data.

19. Only in the case the path is too vertical (and #16 and #17-#18 failed), run **Routine 6.3**. If this “saved the day”, make a note of the seeing value, “peak-to-1st” and “peak-to-last”, and “longitudinal seeing”.

¹ If zunzun fails (offline), try <http://www.xuru.org/rt/PR.asp>. It is enough to open the file “parabole.dat” written into your working directory **from inside** Excel, copy the first two columns and paste them into the "large square" and then click "calculate". Go straight to #18.

Figures

Fig.1:

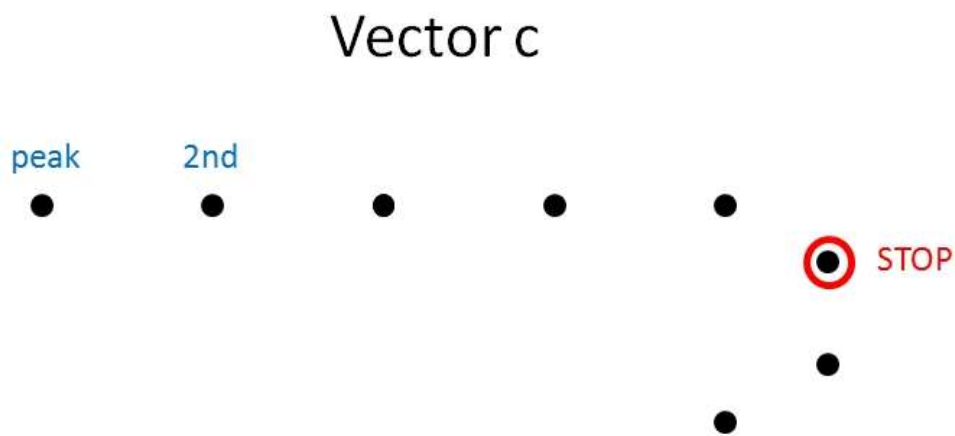


Fig.2:

