

Detailed instructions for CCD data processing in Mathematica to get the *seeing* (FWHM METHOD)

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 electronic ADU values). In Mathematica 5.0 we will read such file and calculate the seeing value of the observations (in this case, using the **FWHM Method**).

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 "RoutineSeeingFWHM.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. 040708_B2.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 data. "m" is critical to fix the FWHM (at half its value); it will also be equal to the vertical height in Routine 2. Before starting this routine, you must 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 (first two parameters) and ViewPoint are the two commands to interact with:

PlotRange -> {{200, 350}, {100, 250}, {0,m}}, 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) limits in PlotRange. 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., $y_2 - y_1 = x_2 - x_1$ (and that x1 and y1 are different from zero). Make a note of the final x1, x2, y1, and y2.

In Routine 3, the data will now be chopped at the FWHM and presented (view from the top). The user must indicate the rough angle of the data with the xx and yy axis (by saying if he judges the direction to be left-right or up-down) and, then, pick the two points that are extremes on the path (a guess is enough at this stage – suggestion: pick them a bit out of the actual extremes of the dataset). Always have in mind the “border-path” orthogonality (Appendix). The “midpoint” calculations will be performed in columns or in rows, depending on the initial guess of direction. Six graphs are, then, presented: a 2D plot of the FWHM values; the path points (only); the plot of the two points that define the extremes of the path (in red); a 2D superposition of these in the 2D FWHM plot; again, the full data chopped at FWHM seen from top; a 2D plot presenting the resulting path calculated via the “midpoint” analysis, column by column (or row by row) – to be analysed later – superposed on the 2D full FWHM data. This will allow deciding if the extremes of the path were correctly chosen (or if they were too short, too long or wrongly placed vertically/horizontally). There is a chance to do several iterations until satisfied. Then, the vector e created (with the path) is final. This should now be analysed, before proceeding. If there is a “jump” in the dataset (different blocks of data) the space in between will be filled with points with $x=0$ (or $y=0$). These ought to be removed first. In case of “jumps” the longest continuous section should be chosen for seeing analysis (by running Routine 4).

5. Run **Routine 3** (insert the final values of x1, x2, y1, and y2, in this order). Then, indicate the approximate direction of the path (left-right or up-down); and pick the two points that are extremes on the path (a guess is enough at this stage – suggestion: pick them a bit out of the actual extremes of the dataset). Carefully inspect the output of the 2D graph with the two extreme points of the path superposed (in red); to help, the top-viewed 2D data are also presented (and four other graphs): iterate as much as necessary until correctly “designing” the path. Then, it is time to look at the actual Polaris path from where Vector e is derived: does it make sense? Are there any discontinuities? If yes: i) do they result from a “jump” in the data? In this case, remove all points with $x=0$ (or $y=0$), and run **Routine 4** a first time (to get the correct number of total points in the original vector e); beware of long diagonal paths that suggest failure of the fit (check against the 2D plot); one way to reduce its length is to switch the orientation chosen (e.g. from top-down to left-right); ii) if the total points in “discontinuous” (and/or diagonal) sections amount to less than 1/3 of the total number of points, simply remove them from the data (editing and redefining vector e – place it on **Routine 4** in Mathematica and run it – to check the final path – must be continuous), making a note on the number of points removed and their distribution (e.g. if in three parts write 4, 3, 5). If more than 1/3 have to be discarded, consider this dataset out of the global data analysis.

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.)

We now start the routines for seeing determination (three independent subroutines, inside "Routine 5"). We will only do parable fit (Routine 5.2) **if** the circle fit (Routine 5.1) 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 5.3) should be run.

6. Run **Routine 5.1** and carefully check its output:

- i) fitting quality of the data points (Polaris track);
- ii) overall fitting of the raw path: 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. And you're done!
- v) if something did not work, carry on to #7.

7. In case #6 failed, run **Routine 5.2**. This will fit a parabola to the data. Again, carefully check its output:

- i) fitting quality of the data points (Polaris track);
- ii) overall fitting of the raw path;
- iii) check if the seeing value makes sense;
- iv) if all is fine, make a note of the seeing value. And you're done!
- v) if something did not work, carry on to #8.

8. Only in the case the path is too vertical (and #6 and #7 failed), run **Routine 5.3**. If this "saved the day", make a note of the seeing value.

Appendix

(ortogonality of "borders" and the "path" to correctly choose seed points)

