N2 - Photometry of open star clusters

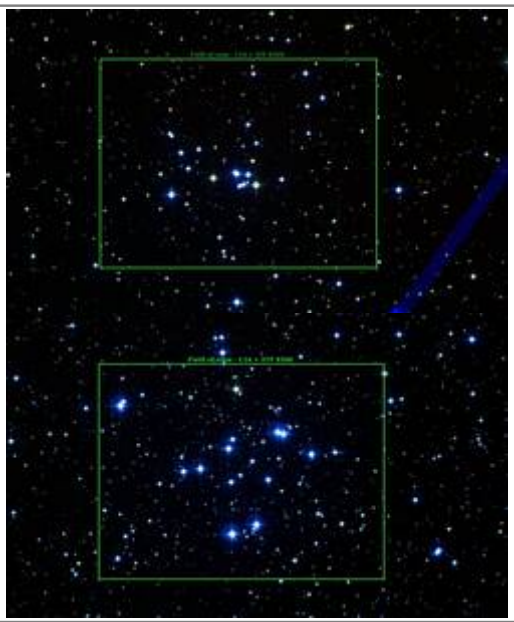
Remark: This article describes the modern data reduction of N2 with help of an half automatic GDL pipeline. In comparison to the classical version (only in German so far), the offsets between the exposures (flatfields, darkframes, and the individual exposures in the different filters) are calculated by the script.

Task

Take photometry of two open star cluster (in two different filters) and create the color-magnitude diagrams for these clusters. The main task is the determination of the cluster ages. Look for suitable clusters in advance of the observation, e.g. at *Simbad* - a help page for the parameter searches can be found here.

Criteria to be fulfilled for the clusters are:

- High number of stars without being too dense
- Field of view with telescope covers most of the cluster
- Size of the cluster is not too small
- No strong brightness contrasts between the individual stars

Good example	**Bad examples**
	

Observation

Nightly observations at the 50cm telescope at the University of Potsdam in Golm are required. Please pay regard to the checklist for night observations. At the OST, the flatfields are taken with a flatfield foil. Bias exposures are not required here as the camera corrects for these errors automatically. Furthermore, darkframes are needed in any case for the data reduction. The exposures of the star clusters are then taken after twilight. To get both a sufficient number of counts and to include dimmer stars, the exposures should last for at least several minutes. If the accuracy of guiding is insufficient,

a series of exposures (e.g. of 1 minute) can be taken and then added up. Moreover, a 2×2 binning can be used with the STF8300 to increase the counts per pixel.

GDL data reduction

The observations can be reduced with *GDL* (or *IDL*). An installation guide for *GDL* can be found here (note that GDL is already installed on the lab computer). A short introduction to *GDL* can be found in this article.

Inspection of the images

Login at the laboratory computer. Copy the observations (FITS files), including flatfield and bias/darkframe exposures from the directory `~/data/<date>` into your own directory `~/data_reduction/`. There are different tools to view the data contained in the FITS files (two dimensional CCD images/data arrays), e.g. *ds9*:

```
ds9 filename.fit
```

opens the image filename.fit with *ds9*. Press and hold the right mouse button and move the cursor to change brightness and contrast. Other options like zoom, color coding, rotation etc. can be performed with *ds9*, too. The pixel position in the current image is displayed in the upper left corner. You can also open multiple images at once (click `Frame` → `new frame`, then open the other file with `File` → `open`). An alternating display mode can be chosen with (i.e. for comparison of different exposures of the same cluster) `Frame` → `blink`.

Alternatively, all FITS files can be opened at once:

```
ds9 *.FIT
```

Scaling the brightness of all images logarithmic can be done with:

```
ds9 -scale log  *.FIT
```

The frames can be viewed in *ds9* as before or as individual frames (buttons: `Frame` → `Singe Frame`). Switch between frames by pressing the TAB button.

Select those images that show nice round stars for the subsequent data reduction and discard those with oval shaped stars.

Reduction pipeline: darkframes, flatfields, and image addition

The analysis of this observation requires the handling of a larger amount of data. Therefore, the reduction is performed with a *GDL* routine which performs the darkframe and flatfield correction and subsequently adds up the individual exposures in each filter. The routine contains no quality control unit, so bad exposures need to be sorted out before (see above).

Preparation

First, copy the *GDL* routine `n2_add_images.pro` from the directory `~/scripts/n2/` to the local working directory and edit it in a text editor of your choice. The paths of the FITS files need to be adjusted, of course. The program „expects“ a certain structure of directories to allow an easy read-in of the data. Each filter's data needs to be stored in a separate directory, also the flatfields and the darkframes need to be sorted accordingly. A possible directory structure could look like:

```
/flats/v/
/flats/b/
/images/v/
/images/b/
/darks/images/
/darks/flat-v/
/darks/flat-b/
```

Remark: The files need to be stored in directories that have no further subdirectories to avoid errors during the read-in (i.e. the darks of the images need to be in `/darks/images/`, not just in `/darks/` following the above example). It is also important that images of different filters may **never** be in the same directory, otherwise the program would mix them (and prematurely add them up). Copy the path names (relative to the script's position or absolute) into the upper part of the *GDL* routine.

Executing the pipeline

The routine is run in the *GDL* environment. Change to the appropriate directory that contains the modified *GDL* routine and start *GDL* by

```
gdl
```

and then compile the routine with

```
.compile n2_add_images.pro
```

In *IDL* the same can be achieved with

```
idl
```

and

```
.rnew n2_add_images.pro
```

In *GDL* and *IDL*, the program can be run by entering the program name:

```
add_images
```

The results will be stored in a new subdirectory `output`.

GDL data analysis

It is more suitable to create a script for the following data reduction than to write each line in the command line of *GDL*. So, open a new file with a text editor of your choice, e.g.

```
kate name.pro &
```

The file needs to contain at least these two lines so GDL will recognize it as valid program:

```
pro programname
end
```

Open *GDL* again

```
gdl
```

compile the script

```
.compile name.pro
```

and run the program

```
programname
```

The commands to read in and to edit the data can now be inserted between the first two commands. To check individual work stages, include `help` commands as needed.

Reading the FITS files

The *GDL* function `readfits` reads a FITS file and saves the image as an array, e.g. the image in the blue filter could be read in as

```
blue = float(readfits('N188B1.fit'))
```

where N188B1.fit is its filename in this example. The variable names should be distinct, such as `blue` for the blue image. This needs to be done for all files. Remark: The data type of the images (by default Integer) should be converted to float to avoid problems when adding high pixel counts.

If the images are stored in different directories, give the relative path to the file, e.g.

```
  blue = float(readfits('data/blue/N188B1.fit'))
```

or, if the script is in yet another subdirectory (the absolute path can be given, too)

```
  blue = float(readfits('../data/blue/N188B1.fit'))
```

To suppress the message „File read: Array xxx x xxxx“, add the parameter `/SILENT`:

```
  blue = float(readfits('data/blue/N188B1.fit',/SILENT))
```

Saving FITS files

Processed FITS files can be saved using the routine `writefits`:

```
writefits, 'filename.fit', <name of the array in GDL>
```

e.g.

```
writefits, 'blue.fit', blue
```

Overlaying the images

The automatic pipeline (`n2_add_images`) from the first step has created one added-up image for each filter. However, also these images are usually shifted against each other and do not necessarily have the same size. This has to be corrected by appropriately cutting of the images.

Measure the offsets

The first step is to determine the relative offsets (with *ds9*, for example). Find one prominent star (not too bright, approx. in the center of the field) and determine its exact pixel position in each frame (the current position of the mouse pointer is displayed in the upper left part of the *ds9* window, use the arrow keys of the keyboard to precisely point to mouse pointer). For quality control, repeat the procedure with a second star.

Furthermore, the FWHM (the full width at half maximum) of a typical star is needed. Measure a few stars' FWHM and take the mean value.

Cutting the images

The left side (x = 0) of the final image is defined by the image with the lowest x coordinate of the „calibration“ star (the same for y).

Example: The blue filter image has the smallest value, i.e. x_b. The visual image will have an x-offset of $\Delta x_v = x_{v} - x_{b}$. Before cutting, the image had $N \times M$ pixel. Consequently, the dimension of the visual image needs to be reduced by $N - \Delta x_{v}$ pixel in the x direction. Remark: *GDL* array-indices start with 0, so the lowest value is x=0 (leftmost column).

Cut the images accordingly with the Δx_i, Δy_i:

```
bluecut = blue[dx_b:N - dx_b, dy_b:M - dy_b]
```

e.g.:

```
bluecut = blue[13:1322, 0:1324]
```

Note: The image with the smallest x value does not necessarily have the smallest y value, too! Check the success of the cutting with a `help` command. The output should then show variables (the image arrays) of equal size.

Now the final visual and blue filter image can be saved with, e.g.

```
writefits, 'blue.fit', bluecut
```

in the FITS format. One of these will be required for the Identification of calibration stars. Check the images with *ds9* (outside *GDL*): The stars should show up as sharp, round objects. Moreover, the stars should stay at the same position, while using the `blink`-option for the images.

Localization of the stars

Finding the stars

The stars are identified, e.g. in the image `badd`, and their brightness is extracted with

```
find, badd, xb, yb, fluxb
```

where `xb`, `yb`, and `fluxb` name variables that will be filled by the `find` routine. There will be a number of queries, asking for the criteria on which basis the stars will be identified (i.e. FWHM, shape, background level). First, the FWHM is asked for (the value determined before), e.g. 3 pixel. The values for the background intensity can be estimated from the added and saved images (using *ds9*). The other parameters (sharpness and roundness) can be left as default (just confirm with Enter).

This procedure returns a table showing the x- and y-position of the stars (`xb`, `yb`), and the flux (`fluxb`). The aim is to identify as many stars in as possible, while excluding „dirt/artefacts“. If too few stars were found, repeat the find procedure and change the criteria, e.g. the background level to include dimmer stars. If the FWHM is too large, multiple stars could be misidentified as one. If it's too small, artefacts could be identified as stars. Change the parameters for a few times and determine the best result. - Apply the method to the final blue and visual image. Change the names of the input array (`badd`), the coordinates (`xb`,`yb`) and the flux (`fluxb`) accordingly.

Once good values for the find routine have been found, it's expedient to put them into the program call and to use the /SILENT option. The full syntax of the `find` call is then:

```
find, image, x, y, flux, sharpness, roundness, hmin, fwhm, roundlim,
sharplim, PRINT=print, SILENT=silent
```

where `sharpness` and `roundness` are both arrays that are filled (like the positions) by the routine; the rest are parameters you have to define, e.g.:

```
find, badd, xb, yb, fluxb, sharpb, roundb, 300, 4.0, [-2.0,2.0], [0.2,1.0],
/silent
```

In this example, the background is 300 and the FWHM is 4.0. The next two variables can be given as an interval, so that the default values can be stated here.

Remark: It has proven useful to make a background correction of the images prior to the call of the `find` routine. The minimum in each image can be determined with

```
bmin = min(badd)
```

A little less than this value should be subtracted, e.g. if `bmin` is 360, the correction could look like this:

```
badd = badd - 300.
```

The value for background can now be set to a lower value in the `find` routine, e.g. 200 instead of 300. Due to the fluctuations it should always be larger than 0.

Converting flux unites to magnitudes

The subsequent parts of the analysis require the fluxes to be converted to magnitudes. Note, that the magnitudes are only certain to an additive constant until a calibration has been performed. Use the routine `flux2mag` of the *AstroLib* for the conversation to magnitudes:

```
magb = flux2mag(fluxb)
magv = flux2mag(fluxv)
```

Cross-correlation of the results

Now determine those stars that were identified in both filters (blue and visual). This cross-correlation is done by the routine `srcor`. It requires as an input the x and y values of the stars as well as a value for the uncertainty, which is set to 2 pixel in the example below. Hence, if the coordinates of a star in the visual and blue image differ by 2 pixel or less, they still will be identified as the same object.

```
srcor, xb, yb, xv, yv, 2, ib, iv
```

`ib` and `iv` are tables containing the index numbers of the common stars in the tables (e.g. `xb`,`xv`,...) that were created by the `find` routine. Using these indices, the positions and magnitudes (fluxes) can be extracted from the original arrays and subsequently be sorted. If, e.g., 226 stars were identified in both filters, an array of type `float` with 226 entries needs to be created that can then be filled with the sorted magnitudes:

```
bmag=fltarr(226)
vmag=fltarr(226)
for i=0, 225 do bmag[i]=magb[ib[i]]
for i=0, 225 do vmag[i]=magv[iv[i]]
```

The dimensioning of these arrays can be automatized, so that it automatically accounts for changes in the number of stars found in both filters (because of changed parameters). The array size can be calculated with the help of the *GDL* command `size`, which itself returns an array with different

information about the original array. Therefore, the command to save the array size in the variable num looks like this:

```
num = (size(ib,/dimensions))(0)
```

Note: It is not important whether ib or iv is used, since both arrays should have the same dimensions. The above loop would then run from i=0 to num-1 instead of 225.

Preliminary color magnitude diagram (CMD)

The next step is to write the dataset that will be used to plot the CMD, i.e. visual magnitude over color (blue minus visual magnitude). The CMD itself can be plotted with different tools, here the version with *gnuplot* is explained (please, don't use *Excel*), also *GDL* can create PostScript plots (see below). First write the data into a file named cmd.dat:

```
openw, out, 'cmd.dat', /get_lun
for i=0,num-1 do printf,out,i,(bmag[i]-vmag[i]),vmag[i]
close, out
free_lun,out
```

Note: This assumes num to be the number of stars that were cross-matched in both fields. Instead of out (and /get_lun) one can use a fixed channel number for the file (here cmd.dat) to be written, too.

Copy the plot script ~/scripts/n2/plot_cmd.py to your current working directory and adjust the plot range (xRangeMin,xRangeMax,yRangeMin,yRangeMax), filename (filename), plot title (nameOfStarcluster), etc. as needed. For this purpose you should open the script in an editor like *kate*.

```
##########################################################################
######
############################## Script Parameters
##################################
##########################################################################
######

# name of cmd data file
CMDFileName = "cmd.dat"
# names of output file
filename    = "?"
# filetype of output, supported filetypes: ps, png, gif, svg, tex
filetype    = "png"
# name of the star cluster
nameOfStarcluster = "?"
## x and y range to plot (change according to your data)
# range of the x coordinate to plot [xRangeMin:xRangeMax]
xRangeMin = "?"
xRangeMax = "?"
```

```
# same for y
yRangeMin = "?"
yRangeMax = "?"
# number of the V column and the B-V column in the 'cmd.dat' file (default:
columnB_V = "2" and columnV = "4")
columnB_V = "2"
columnV   = "4"
```

The script can be executed in the terminal with the following command

```
./plot_cmd.py
```

Afterwards, a figure, showing the CMD, can be found in the current directory. The file type of this figure can be set via the variable `filetype`.

Alternative: *GDL* can also create PostScript files:

```
plot, bmag-vmag, vmag, psym=1, yrange=[-17,-25]
```

Use `yrange` and `xrange` to extract a section of the plot (also to invert the y-axis since higher magnitudes are equivalent to dimmer stars). Also use `psym=1` to get discrete symbols (default is a connected line).

Calibration

The magnitudes are uncalibrated so far (they have no zero point). The calibration is a major problem, in principle there are two possibilities:

1. Alternative

There's a star in the field with known B and V magnitude. If it can be identified in the table, the magnitudes of the respective tables need to be shifted by the difference between the literature magnitude and the determined value.

Identify the stars in the field of view

There are several possibilities to identify the field of view:

The best (and suggested) method is to make use of the Simbad database. In the `basic search` enter the name of the observed cluster and select the `Aladin Previewer`, which one can find in the `Plots and Images` section. Identified the observed field in that image. For comparison, open one of your own observations in *ds9*:

```
ds9 filename.fit
```

The observation might need to be rotated and/or mirrored.

Identify possible calibration stars

Once the field has been identified, find 3-4 stars in the center of the field for which both the B and V magnitudes are known. Note down these values for later use.

- To ease that process, use Simbad again: In the `basic search` use the `query around` (with radius...) and choose `plot this list of objects` to obtain a schematic map of the cluster with clickable stars. Click on the stars to see their magnitudes (B and V magnitudes are of interest).
- The same can be reached with the `Aladin Java Applet` but this tool requires *JAVA* to installed on your computer. Click on the highlighted stars to see their magnitudes. If you work with a computer of the computer pool or at home, this method is the easiest.
- An alternative method (without Simbad) makes use of the Digitized Sky Survey. Compare the best visual exposure with the relevant sections (coordinates) of the sky to identify the region. Then use an overlay with the stars from important catalogs (Bonner Durchmusterung and Henry Draper (HD) Catalogue) to identify stars (note the scrollbar in the menus!). If there are catalog stars in the observed field, use Simbad to find their B and V magnitudes.

In any case check whether the given magnitudes are absolute or apparent ones. If the latter is true, check for the distance (modulus) to convert them.

A *Python* script is available which helps for that task. Copy `~/scripts/n2/plot_stars.py` to your current working directory and adjust the upper part

```
# enter full path to your reduced fits-file
visualPath = 'vadd.fit'

# enter full path to the star position file (format: index, x position, y
position, b magnitude, v magnitude)
starPath = 'stars.dat'
# output file:
outfile = 'starmap.png'
```

in an editor of your choice. State the path to one of your modified images (variable: `visualPath`) and the path to a file that contains the star positions and the magnitudes in the V and B band (variable: `starPath`). You can construct such a file (e.g. `stars.dat`) with your *GDL* program by adding the following commands:

```
openw, comb, 'stars.dat', /get_lun
for i=0,num-1 do printf,comb,i,xv[iv[i]],yv[iv[i]],bmag[i],vmag[i]
close, comb
free_lun,comb
```

Afterwards, recompile and run your *GDL* program once more. Then save and run the *Python* script from a console:

```
./plot_stars.py
```

An image „starmap.png“ will be the result if all went well. It shows the image (e.g. `vadd.fit`) as color-inverted background and all identified stars as overlays **(please include or attach this image to your report)**. This image can be compared with the calibration stars mentioned in the beginning. After one of the star's from the „starmap“ has been matched to a calibration star, the number of the corresponding mark on the „starmap“ can be looked up in the file `stars.dat` to obtain the corresponding magnitude. The stars in the `stars.dat` file are ordered by increasing Y values. So you can search for the position of the star in your image which you used as background image for the star map and then you can search for the coordinates in the `stars.dat` file. The difference between this magnitude and the value given in *Simbad* is the calibration shift. Repeat this procedure for 3-4 stars and calculate the average of the calibration shifts. The variance should not be larger than 0.1 mag.

2. Alternative

A calibration star (with known magnitudes) has been observed (in good local and temporal proximity to the cluster's observation). Reduce these calibration data in the same way as the cluster's data. Important note: The exposure time needs to be the same, otherwise the fluxes of all exposures (cluster and calibration star, in each of the two filters) need to be normalized to, e.g. 1 second. If the total exposure time in the blue filter is 440 seconds, this will e.g. look like:

```
fluxb = fluxb / 440.
```

After the conversion to magnitudes, the difference to the literature values can be calculated and as a correction applied to the observation of the star cluster.

Either way, the B and V magnitudes need to be calibrated accordingly by the calculated shifts:

```
magbcal = magb - bcal
magvcal = magv - vcal
```

With these values the final CMD can be plotted. To ease the output, the color can be calculated already at this point:

```
magbminusv = magbcal - magvcal
```

In comparison with literature diagrams the main sequence is likely to be shifted. This occurs due to the interstellar medium which is spread between the stars of our Galaxy. Like all other baryonic matter, it can be excited by light. It will reemit this energy usually at a longer wavelength. Therefore, this effect is called reddening (do not confuse it with redshift):

$(B-V)_{0} = (B-V) - E_{(B-V)}$

$V_{0} = V - A_{V}$

The reddening is mathematically described by the color excess $E_{(B-V)}$, the difference between the measured, uncorrected color $(B-V)$ (measured here) and the unreddened, „original“ value $(B-V)_{0}$. The redding effects the magnitude, too. The correction term is A_{V}, which relates to $E_{(B-V)}$ by the reddening parameter R_V:

$A_{V} = R_V \cdot E_{(B-V)}$

In the solar neighborhood R_V usually is set to 3.1 (Seaton 1979). Find an appropriate value for $E_{(B-V)}$ for the line of sight to the observed cluster, i.e. in *VizieR* or in *Simbad* by means of the papers that are associated with your object. In any case, refer to the used catalog or paper in your report! Apply this correction to your data and plot the CMD again.

If the calibration was performed on apparent magnitudes, one need to account for dilution due to the distance of the cluster by applying the distance modulus.

To plot the final CMD with the *gnuplot* script, write the corrected data again, e.g. overwrite the preliminary version of `cmd.dat`:

```
openw, cmd, 'cmd.dat',/get_lun
for i=0,num-1 do printf,cmd,i,magbminusv[i],magbcal[i],magvcal[i]
close, cmd
free_lun,cmd
```

Plotting the CMD

Plotting with Python, including isochrones

As previously described, the file `cmd.dat` can be plotted with the script `plot_cmd.py`, which can be run with

```
 ./plot_cmd.py
```

It also offers the possibility to include isochrones, which can be downloaded from the websites of various stellar evolution projects. The variables of the script `plot_cmd.py` need to be adjusted according to the requirements of the downloaded isochrone files. The script expects that the isochrones are given as individual files, which should be located in a single directory (`isoDir`).

```
##########################################################################
######
############################ Script Parameters
#################################
##########################################################################
######

# name of cmd data file
CMDFileName = "cmd.dat"
# names of output file
filename    = "?"
# filetype of output, supported filetypes: ps, png, gif, svg, tex
filetype    = "png"
# name of the star cluster
nameOfStarcluster = "?"
## x and y range to plot (change according to your data)
# range of the x coordinate to plot [xRangeMin:xRangeMax]
```

```
xRangeMin = "?"
xRangeMax = "?"
# same for y
yRangeMin = "?"
yRangeMax = "?"
# number of the V column and the B-V column in the 'cmd.dat' file (default:
columnB_V = "2" and columnV = "4")
columnB_V = "2"
columnV   = "4"
# size of the plotted output figure in cm, default 8cm x 8cm
size_x = "?"
size_y = "?"

#------------------------------------------------------------------------
#####  OPTIONAL: Isochrone fit  #####
# path to the directory with the isochrone data files
isoDir = "?"
# B-V or B column in the isochrone data file?
ISOcolumntype = "B-V"
# number of the B/(B-V) column and the V column in isochrone data file
ISOcolumnB = "?"
ISOcolumnV = "?"
```

Alternative with GDL

As describe above a CMD can be plotted in *GDL* with

```
plot, magbcal-magvcal, magvcal, psym=1, yrange=[18,12]
```

Change x and y range as needed. Add a title and labels to the plot by adding the following parameters to the above call:

```
title="CMD of NGC 123", xtitle="B-V in mag", ytitle="V in mag"
```

Once you are happy with the diagram, it can be exported as a PostScript file:

```
set_plot, 'ps'
device, filename='CMD.ps'
plot, magbcal-magvcal, magvcal, psym=1, yrange=[18,12], title=...
device, /close
```

Then switch back to the X window for graphical output:

```
set_plot, 'x'
```

The PostScript file can then be viewed with, e.g. *Ghostview* via

```
gv CDM.ps
```

Report

Describe the difference between globular clusters and open clusters, with special emphasis on the character of the observed object. Discus the differences of stellar clusters and general accumulations/groups of stars. Shortly describe the evolution of a star and how it manifests in the Hertzsprung-Russell diagram and the color-magnitude diagram, respectively. Describe the age estimation of stellar clusters by means of the analysis of the turn-off point and by means of isochrones. Use these methods to determine the age of the observed clusters. Write a normal laboratory-course report. Discuss your results and compare them to the literature. Address shortcomings in your results and discuss possible causes.

Overview: Laboratory Courses