POLARIMETRIC REDUCTION TUTORIAL

DANIEL BEDNARSKI

August 1, 2015

Abstract

This tutorial presents a brief description of polarimetric reduction of OPD data (Observatório Pico dos Dias) to be performed through IRAF software. The process applies for observations in which have been employed CCD Ikon/Ixon and IAGPOL polarimeter with a calcite and a half wavelength retarder plate for measurements of linear polarization.

Contact: daniel.bednarski.ramos@gmail.com. This tutorial is not finished and probably has many gramatical errors yet.

CONTENTS

Part I

ON REDUCTION

The reduction process inside IRAF environment is the first step to be done at the some raw night. The first requirement is to install Beacon package, whose tasks will be used in this step (instalation and IRAF configuration are explaned in section 1).

The reduction procedure can be done through two ways: automatically, by running *calib* and *reduce* routines (described in section 2); or manually, by performing directly the same procedures called by these tasks (described in section 3). Although the automatic mode is much more simple and fast, it is a quite limited for particular cases. Therefore, sometimes it is needed to use the manual mode over one atypical data serie.

In addition, some general failures that you can experience at both manual and automatic modes are described in section 4.

1

THE BEACON PACKAGE

INSTALL BEACON PACKAGE

Assuming you are inside your IRAF installation directory, go to './extern/' subdirectory to install the Beacon package and type:

```
$  git clone https://github.com/dbednarski/beacon.git
```

Edit './unix/hlib/extern.pkg' inside IRAF directory, adding the following lines before the line containing the *keep* command:

```
reset   beacon      = iraf$extern/beacon
task    beacon.pkg  = beacon$beacon.cl
```

SET IRAF

Download the script to open IRAF:

```
$  sudo apt-get install wmctrl
$  sudo wget http://www.astro.iag.usp.br/~bednarski/obs/irafw -P /usr/local/bin
$  sudo chmod a+x /usr/local/bin/irafw
```

Open a new terminal and create a "working directory" in '~/iraf' and the login.cl file, by typing:[1]

```
$  mkdir ~/iraf
$  cd ~/iraf
$  mkiraf   # Select xgterm
```

Change the following lines inside '~/iraf/login.cl':

- Line ∼ 24: `set  stdimage  = imt1024`
- Line ∼ 31: `set  imtype  = "fits"`

Just type *irafw*[2] command to open IRAF (ecl+DS9) or *irafw -cl* to open only ecl. At this first time that you call IRAF, set instrument, display parameters and compile .f codes:

```
$  irafw     # Opening IRAF
ecl>  beacon
ecl>  setinst   # Select 'camera' and type ':q' and ':q' to exit from the
                                                    following screens

ecl>  epar display   # change xcenter=0 and ycenter=0 (':q' to save and exit)

ecl>  cd PATH_TO_BEACON_PACKAGE/pccd   # Change PATH_TO_BEACON_PACKAGE to
                                         the path for your Beacon package
ecl>  !sudo chmod -R 777 .
ecl>  !rm pccd2000gen05.mac.e ccdrap_e.e
ecl>  fc pccd2000gen05.mac.f -o pccd2000gen05.mac.e
ecl>  fc ccdrap_e.f -o ccdrap_e.e
```

Done! Beacon package and IRAF are already installed and configured!

1 Don't forget to type `ur_setup` before `mkiraf` command if you have installed Ureka's IRAF (`http://ssb.stsci.edu/ureka/`).

2 IRAF's login.cl file is supposed inside '~/iraf'. Edit this value inside script if is not it. May you need to change 'ecl' to 'cl' in the two lines starting with 'xgterm' also.

2

THE AUTOMATIC REDUCTION

The automatic reduction of one night has just two steps: combination of calibration images and reduction of the objects one by one. In the first step is used *calib* task, which is described in section 2.1, whereas in the second is used *reduce* (described in section 2.2).

2.1 CALIB.CL: GETTING READY THE CALIBRATIONS

PRESENTATION The first step is to run *calib* task to generate a "mean" bias image and to combine the flat images for each filter. However, you need to run *calib* for every CCD configuration. For example, if there are calibrations with both 1.0 and 5.1 pre-amplifier gain, you must run twice, one for each configuration.

PARAMETERS The only *calib* parameter to be setted is `suf`, whose value must be equals to the suffix of calibration images – the complement between the root name and fits number. In the above example, suffix for 5.1 gain would be "_g5" if their names were "bias_g5_0001.fits"/"flat_g5_**F**_0001.fits", where **F** is the filters[3].

```
  PACKAGE = beacon
     TASK = calib

     (suf =     _f) Suffix to be applied to ALL the calib images
  (verify =    yes) * Stop script if no bias images are found?
```

RUNNING

- Load the Beacon package typing *beacon*;
- Go to calib directory inside the night to be reduced;
- Set *suf* parameter through the command *epar calib* and run (*:g* to go!).

2.2 REDUCE.CL: REDUCING AN OBJECT

PRESENTATION Task *reduce* is now used to reduce an entire object. It musts be called inside the object subdirectory. As like *calib* task, *reduce* musts be runned for each CCD configuration used for such object.

PARAMETERS The parameters of current version and their default values are:

```
   PACKAGE = beacon
      TASK = reduce

      pref =     tpyx  Prefix of filenames (WITHOUT '_')
       suf =       _f  Suffix of filenames (same of calib. images)
    (calib = ../calib) Path to calib directory (WITHOUT last '/')
     (dov1 =      yes) Do the reduction without calibrations?
     (dov2 =      yes) Do the reduction using the calib. images?
     (head =      yes) Use gain and readnoise values from headers?
 (ver1stwp =      yes) Verify if first image is the WP position L0?
(usecoords =      yes) Use previous daofind coordinates to next filters?
  (dograph =       no) Generate all .png graphs?
```

3 *Calib* also works on the reversal names "flat_**F**_g5_0001.fits".

```
   (ganho =    INDEF) CCD gain to be used if head=no (e/adu)
  (readno =    INDEF) ReadNoise (ADUs) to be used if head=no (adu)
  (reject =        0) Reject images with counts larger than this value
                                        (0 to use value from ccdrap)
(pccdpath = /iraf/iraf-2.16.1/extern/beacon/pccd/) Path to .e pccd files (blank to
                                                   use values from ccdrap/pccdgen)
(graphpol = /iraf/iraf-2.16.1/extern/beacon/grafpol.py) Path to the grafpol.py code
```

Let's make clear each one:

- `pref`: the main parameter, concerning to the root of filenames (e.g. for dsco_*.fits images, pref="dsco");
- `suf`: suffix as in *calib* task, very important to take the correct calibration images (those with the same suffix);
- `calib`: path to the calib directory concerning to the current object subdirectory. Useful if being used calibrations of another day;
- `dov1/dov2`: literally "do version 1"/"do version 2". These reduction versions are, respectively, the reduction without and with the calibration images;
- `head`, `ganho` and `readno`: most observations keep stored the gain (sensitivity) and readout noise values in header of .fits files. However, there are not such fields in some nights. Therefore, `head` parameter queries if you want to use the values from headers – if not, `ganho` and `readno` must be specified[4];
- `ver1stwp`: queries if you want to abort the procedures if the first WP position is not the position L0. It is suggested to use ver1stwp=yes, but some nights have wrong headers and all positions are registered as the same erroneously (L4 for example). In this case, unable this parameter.
- `usecoords`: if enabled, *reduce* tries to find automatically the star position to a certain filter from the position of previous filter;
- `dograph` and `graphpol`: if `dograph` is enabled, the polarimetric modulations are printed into images at the end of *reduce* procedure. The location to *grafpol.py* should be specified through the parameter `graphpol`;
- `reject`: saturation level for CCD configuration. The algorithm won't use the images whose max counts be larger than `reject`. If reject=0, the value setted in *ccdrap* task will be used. Case the configuration readen from image headers indicates it is CCD iXon, bin 2, EM or CCD iXon/iKon, bin 2, CON, `reject` (or `ccdrap.reject`) will be compared with the expected reject value[5];
- `pccdpath`: The location to 'pccd' Beacon's subdirectory. If it is blank, the paths setted in *ccdrap* and *pccdgen* tasks will be used.

4 These values are available in `http://www.lna.br/opd/instrum/ccd/detccd.html` (columns "CCD Sensitivity" and "Single Pixel Noise"). Be careful to use the correct units (e/ADU and ADU): the table of Ixon features readnoise in e unit, which needs to be divided by the gain to the suitable value in ADU.

5 For CCD iXon/iKon, bin 2, if `reject` parameter should be greater than 30,000 for observations using Electron Multiplier (EM) amplifier mode and 63,000 for Conventional (CON) amplification (even though the safe values are 22,000 and 62,000, respectively). Otherwise, an error message will be printed.
The CCD configurations can be accessed through the image headers by calling *imhead IMAGE.FITS l+* for some image named 'IMAGE.FITS'.

- Go to the object directory;
- Case `pccdpath=""`, type *epar ccdrap* and *epar pccdgen* to verify `fileexe` parameters for ccdrap_e.e and pccd2000geno5.mac.e files location. Once you has changed *ccdrap* and *pccdgen* parameters, you can skip to the next item;
- Set the parameters through the command *epar reduce* (*:q* to save and exit). Don't worry about set `pref` and `suf` parameters here, because they must be passed directly when calling *reduce*;
- Run *reduce* just typing *reduce "pref" "suf"*, where "pref" and "suf" are those parameters passed directly).
- When running, **is very important** you be consistent and to select the beams in CCD at the same order for each object! Otherwise, the polarization angle can be wrong!

3 THE MANUAL REDUCTION

The manual reduction of one night has the same two steps of those for automatic reduction: combination of calibration images and reduction of the objects. In the first step we use directly *read3Dfits*, *ccdproc*, *zerocombine* and *flatcombine* tasks (described in section 3.1). The second step has two parts: do the photometry through *ccdrap* task and combine its results in polarimetric modulations through *polrap* task (described in sections 3.2 and 3.3, respectively).

3.1 GETTING READY THE CALIBRATIONS

Further.

3.2 CCDRAP: DOING THE PHOTOMETRY

PRESENTATION Task *ccdrap* is the first step to reduce one data serie for some object. In contrast to *reduce*, it is reduced only one filter by run. This is why this manual mode is not practical, but can be essential to solve some specific problems.

Ccdrap calculates the photometry of ordinary/extraordinary beams for several values of aperture radius. As a larger radius can sum not only more signal of the star, but more noise also and a smaller radius can get less noise, but throw away signal, the objective is to perform photometries for many apertures values and to get the best one.

PARAMETERS The parameters of current version and their default values are:

```
   PACKAGE = beacon
      TASK = ccdrap

    rootin =           Filename root (DO NOT PUT THE '_' AT THE END)
   (intera =       no) Use interactive selection for objects?
     (modo =        2) 1-Register images; 2-Run PHOT
  (version =       .1) Version of the output files
     (trim =       no) Trim the image?
 (overscan =       no) Apply overscan strip correction?
  (zerocor =       no) Apply zero level correction?
  (darkcor =       no) Apply dark count correction?
  (flatcor =       no) Apply flat field correction?
 (coordref =       no) Use a reference coordinate file?
 (ver1stwp =      yes) Verify if first image is the WP position L0?
  (biassec =         ) Overscan strip image section
  (trimsec =         ) Trim data section
     (zero =         ) Zero level calibration image
     (dark =         ) Dark Count calibration image
     (flat =         ) Flat field images
    (coord =         ) Reference coordinate file
(readnoise =     2.19) CCD readnoise (adu)
    (ganho =      3.7) CCD gain (e/adu)
      (nap =       10) phot: number of apertures (maximum 10)
(apertures =  5:14:1) phot: List of aperture radii in pixels
  (annulus =     30.) phot: Inner radius of sky annulus in scale units
 (dannulus =     10.) phot: Width of sky annulus in scale units
  (boxsize =        7) Imalign: Size of the small centering box
   (bigbox =       11) Imalign: Size of the big centering box
   (reject =    62000) Reject images with pixel values larger than this value
 (stack1st =       no) Attempt to automatically stack 1st WP position?
  (fileexe = /iraf/iraf-2.16.1/extern/beacon/pccd/ccdrap_e.e) CCDRAP executable file
```

Let's make clear those that you may want to change:

- `rootin`: prefix of fits image to be processed. In contrast to the *reduce* task, here prefix is the whole name (prefix, in the sense of *reduce*, + filter + suffix) until the last numbers before '.fits' extension;
- `intera`: run in interactive mode? (i.e., asks for confirmation of coordinates in every WP position?);
- `version`: version of reduction (the string appended before '.out'/'.log'/ etc files). Remember that '.1' and '.2' must be concerning to the reduction without and with calibration images. However, you can generate a third or fourth version '.3', '.4' if you need;
- `zerocor` and `zero`: the first one queries if you will use a bias image. Case 'yes', you need to specify the bias file on `bias` parameter (you can indicate the path to the file considering the current path as the object directory.);
- `flatcor` and `flat`: idem to the bias, but now it is about the flat;
- `coordref` and `coord`: variables used by *reduce*. `coordref` asks if you want to use a coordinate file to find the positions of beams automatically. Case 'yes', specify the coord file '.ord' in `coord` variable;
- `ver1stwp`: queries if you want to abort the procedures if the first WP position is not the position L0. It is suggested to use ver1stwp=yes, but some nights have wrong headers and all positions are registered as the same erroneously (L4 for example). In this case, unable this parameter.
- `gain`: like *reduce*'s `ganho` parameter;
- `readnoise`: like *reduce*'s `readno` parameter;
- `reject`: like *reduce*'s `reject` parameter;
- `nap` and `apertures`: the first one is the number of distinct radius values to be used as aperture to do the photometry. The second one specify the radius values, a string with format 'min:max:step', in pixels units;
- `annulus` and `dannulus`: variable concerning to the inner and outer radius of the "ring" around the star (in pixels), inside which the sky counts will be computed;
- `boxsize` and `bigbox`: variables concerning to the size of boxes, in pixels, used for images alignment;
- `fileexe`: The location of 'ccdrap_e.e' program. It is inside pccd subdirectory into Beacon package;
- `icom`: The location of 'icom.sh' script. It is inside pccd subdirectory into Beacon package also.

RUNNING FOR VERSION ".1"

- Go to the object directory;
- Type *epar ccdrap* and set the parameters for version ".1": `version`=".1", `zerocor`=no, `flatcor`=no, `gain` and `readnoise`[6]. Thereby `zero` and `flat` parameters will not be used;

6 See the footnote 4 (on page 7).

- Be sure that the `reject`[7], `fileexe` and `icom` values are correct;
- Set some other *ccdrap* parameter if needed (*:q* to save and exit);
- Run *ccdrap* just typing *ccdrap "rootin"*, where "rootin" is the root of the object+filter to be runned.
- When running, **is very important** you be consistent and to select the beams in CCD at the same order for each object! Otherwise, the polarization angle can be wrong!

RUNNING FOR VERSION ".2"

- Similar to the version ".1", you need change the parameters `version`=".2", `zerocor`=yes and `flatcor`=yes (or "no" in case of missing bias or flat image);
- Edit `zero` and `flat` parameters also, by putting the path to the bias/flat fits images. Remember you must use the calibrations done on the same CCD configuration (like pre-amplifier gain, CCD section, output amplifier CON/EM, binning).

RUNNING FOR ANOTHER VERSION

- You can run more than these two versions also, editing the parameters right as you need. It can be done, for instance, if you have two distinct sets of calibration images and want to have the results for both.

3.3 POLRAP: GROUPING THE WP POSITIONS

PRESENTATION Task *polrap* is the last step to reduce the data serie. *Polrap* calculates the polarization level for each WP position and groups them, generating a modulation pattern, on which adjusts the model for polarization as function of the WP angle (position).

Once runned *ccdrap*, you can run *polrap* at any time over such data serie. This is very useful, because sometimes is needed to combine another group than those generated automatically by *reduce* routine.

PARAMETERS The parameters of current version and their default values are:

```
  PACKAGE = beacon
     TASK = polrap

        (n =     8) Number of waveplate positions in each group
     (type =   dat) Input type: .mag or .dat files
(version1 =    .1) Version of the input fits files (output of ccdrap routine)
(version2 =    .1) Version of the input mag files
 (rootout =     w) Root for output filenames
  (rootin =      ) Root for input filenames
 (minimum =  full) MACROL: minimum? (first,full)
    (pout =    00) WP position(s) to be ignored. Sintax: # of WP to ignore
                   (2 digit number) followed by the list of WP pos. (2 digit
                   numbers) separated by 1 space
```

Let's present each one:

- `n`: amount of waveplate positions to group in each group;
- `type`: extension of the files containing the photometric data (".dat" or ".mag"), i.e., extension of *ccdrap* output files;

7 See the footnote 5 (on page 7).

- `version1` and `version2`: the first one is the reduction version to be combined (see *ccdrap*'s `version` parameter). The last one is used only when `type`=".mag" (is not our case);
- `rootout`: the root for .out/.log output files. We use the same value used for *ccdrap*'s `rootin` parameter, just appending the "w" letter before it. HINT: *ccdrap* sets this parameter automatically right after running;
- `rootin`: the root for .dat (.mag) input files (the output of *ccdrap* task). Such root is the same that *ccdrap*'s `rootin` parameter, just appending "sum_" at the begining. HINT: *ccdrap* also sets this parameter automatically right after running;
- `pout`: this is a powerful parameter. You can use it to exclude one or more WP positions from the groups. The syntax is a string begining with the amount of WP to be disregarded (in "C-like format" %2d), followed by such position numbers (in format %2d also), using the space (" ") as delimiter. Example: to ignore the positions #3 and #11, `pout`="02 03 11". To use all WP, `pout`="00".

RUNNING

- *polrap* uses *pccdgen* task, and then, be sure that *pccdgen* parameters are correct for polarimeter configuration: `wavetype`="half", `retar`=180, `calc`="c" and `fileexe` is pointing to the correct 'pccd2000gen05.mac.e' file;
- Now there is no secret: set the parameters as you need;
- Run *polrap* as many times as needed, grouping how many positions you want for each reduction version.

4 GENERAL FAILURES OF REDUCTION TASKS

In this section, we summarize some problems that you may experience when running *calib*, *reduce*, *ccdrap* or *polrap* routines.

a) case reduce/ccdrap task has missed the coordinates for some lamina position.

This is partially prevented to happening, but not entirely. If you have experienced it, there are two distinct ways to fix:

1. Improving the `bigbox` and `boxsize` *ccdrap* parameters. They are concerning to the bounds inside of which the object should stands at (see section 3.2).
2. Setting 'yes' for the *ccdrap* parameter `intera` (see section 3.2). It will activate the interactive mode and you'll be queried about the correct coordinates at each lamina position.

b) case reduce task has reported that reject parameter is wrong.

It can happen because the saturation/linearity level changes according to the CCD model/configurations. Check the value and change manually `reject` parameter.

c) case reduce/ccdrap task has reported:

```
INTERNAL ERROR: dictionary full
(...)
# Grouping the 0 WP positions in -7 groups of 8 positions
(...)
# Grouping the 0 WP positions in -15 groups of 16 positions}
```

It can happen due to a memory error. Are you running the tasks for more than 64 WP positions? Case yes, split the files for each 64-positions into a new directory and run for each one.

d) case reduce/ccdrap task has reported right at beginning:

```
column 3 not found in coordtmp.ord
ERROR: segmentation violation
```

It can happen due to a spurious coordinate file. Verify the .ord file for the filter in which the error have occurred. Is it void or incomplete? Delete it and run again.

a) case reduce/ccdrap task has reported:

```
ERROR: the first WP position for "//root//"_*.fits is not the position L0!
   Verify through imhead task and try again.
   If the first position was performed at L0 but the headers are wrong, you can set ver
```

Part II

ON VALIDATION OF MODULATIONS

In the sections 2 and 3 we have described the reduction procedures. Once concluded, you have to generate the logfiles that links the informations regarding the observations and points the selected outfiles to use for each object.

Task *polrap* called directly or internally by *reduce* has generated many polarimetric results by grouping the set of WP positions in several ways. For instance, an object observed in 16 positions results in the default combination with all 16 positions, as well as grouping 8-by-8 positions sequentially (i.e. 1-8, 2-9, 3-10, etc). The aperture radius featuring the best polarimetric result for each one of these combinations is placed in a '.out' file.

The point is: which lamina combination to use? The usual is the one which has the best error. But in some cases it's necessary some caution, by comparing the modulations through the png images.

This process is done python xThis is introduced in section 6 and the simplified statements to get manually right these results are presented in section 7.

5 THE PYHDUST PACKAGE

PyHdust is a miscellany of python modules (documentation on `http://www.astro.iag.usp.br/~moser/doc/`)

INSTALLING PYHDUST

First, install the requeriments:

```
$  sudo apt-get install python-pip python-matplotlib ipython python-pyfits
$  sudo pip install numpy jdcal scipy
```

Assuming you are inside whre you want to install PyHdust, type:

```
$  git clone https://github.com/danmoser/pyhdust.git
$  cd pyhdust
$  python setup.py install --user
```

UPDATING PYHDUST

Case pyhdust is installed already, just update and run setup.py. At some times, you need to update when there is a new version.

```
$  git pull
$  python setup.py install --user
```

Further.

The appendix **??** presents many examples of bad modulations and a brief discution about them and their choices.

7 EXTRACTING THE STOKES PARAMETERS FROM OUT FILES

In the sections 2 and 3 we have described the reduction procedures, whose process generates a lot of out files. Let's explain how to get the final Stokes parameters for each object/filter.

Task *ccdrap* called by *reduce* has performed photometries for each lamina position using a set of values for aperture radius; thereafter, task *polrap* has generated many polarimetric results by grouping the various lamina positions in several ways. For illustrating, an object observed in 16 positions results in the default combination on all 16 positions, as well as grouping 8-by-8 positions sequentially (i.e. 1-8, 2-9, 3-10, etc). The aperture radius featuring the best polarimetric result for each one of these combinations is placed in a '.out' file.

The point is: which lamina combination to use? The usual is the one which has the best error. But in some cases it's necessary some caution, by comparing the modulations through the png images.

Suppose it was selected a good '.out' file. First, if such observation was performed before 2015, the real angle is not θ_{out} value inside that file, but the applied transformation $-\theta_{out}$ (because the lamina motor was rotating in opposite direction). Second, because the calcite is not aligned with the celestial equator, it's needed to add a correction factor obtained through the standard star: that factor is nothing else than the difference between the published angle at the equatorial system and the observed value[8].

If P_{out} and θ_{out} are the polarization percent/angle of reduced object, changed to $theta_{out}$ if it applies; and Θ_{out} and Θ_{publ} are, respectively, the polarization angle of standard star (from such standard's '.out' file or changed to $-\Theta_{publ}$ if it applies) and the published angle in correct reference system:

$$\begin{array}{ll} \textit{calibration factor:} & \Delta\Theta = \Theta_{publ} - \Theta_{out} \\ \textit{object's angle calibrated:} & \theta_{out} + \Delta\Theta \end{array}$$

So, the polarization of object is:

$$\boxed{\begin{array}{l} P = P_{out} \\ \theta = \Theta_{publ} + \Theta_{out} - \theta_{out} \end{array}}$$

and the new Stokes Q-U parameters must be calculated from these values. ***Q-U* parameters inside '.out' file should be discarded**.

Attempt to the fact that the angle can be changed $\theta \rightarrow \theta \pm n\,.\,180^{\circ}$ ($n = 1, 2, 3, ...$), in order to shift it to the interval $[0, 180]$ (or, likewise, $[-90, 90]$, depending the intervals with which you are working). It can be done because for polarimetry these are the same angle.

8 Polarization angles of standard stars are available at `http://astroweb.iag.usp.br/~polarimetria/padroes/`.

Part III

ON TO GET ALL POLARIMETRIC DATA FOR SOME OBJECT

This is an after-reduction step and consists in sweep ‘obj.dat’ and ‘std.dat’ files for every night and generate a polarimetric data table with all observations for some target.

8 GENTARGET.PY

Further.