IDL software for turbulence measurements with RINGSS, version 5

Authors:*A. Tokovinin*
Version: 3
Date: 2023-03-13
File: prj/atm/smass/doc/ringss-idl.tex

1 Introduction

This document contains a short description of the software package to measure optical turbulence profiles from series of ring-like images produced by the RINGSS instrument. The concept of RINGSS and processing algorithms are covered in the MNRAS paper [1]. Additional information can be found at the web site [2].

Briefly, the input data are series of consecutive ring-like images of a bright taken with a 1-ms exposure time (data cubes). A typical cube has 64x64x1000 format. Each image is slightly distorted by turbulence. First, several parameters (coefficients) are determined for each frame. Then statistics of these coefficients (variances and covariances) are used to estimate turbulence profile and seeing. Relation between statistical moments and turbulence parameters is defined by the weighting functions which depend on the instrument parameters, star color, etc. The instrument parameters are assembled in the accompanying parameter file.

This version of the IDL code is called 'version 5'. It works under IDL 7.1 and higher or under GDL. The IDL `astro` library is used for reading and writing the fits files. Ideally, the package could work by placing all programs, auxiliary data (star catalog, zmatrix), and the parameter file in one directory, launching the IDL, and typing two commands:

```
IDL> par = readpar('par-file')
IDL> @pipe
```

Figure 1 illustrates the processing flow. The data cube is taken by the image acquisition software (not part of this package) and saved, e.g. in the FITS format. It is processed by `cubecoef` to determine the signals, and their statistical moments (variances and covariances) are computed by `statmom`. A python version of the code does a similar job, and the statistical moments saved in a text file (one line per cube) can be used for profile measurement by both codes. The profile restoration uses weighting functions which are computed by a dedicated module and saved as IDL variable. Finally, the simulation package is used for checking the code and exploration of various issues.

2 Parameter and data files

This is an ASCII file that defines the IDL structure with all relevant parameters, including directories where input files live, telescope characteristics, etc. It is used by several programs. An example of parameter file `sep12a.par` with comments is given below. The parameters are read by `readpar.pro`. The number and name of parameters can be flexibly changed, but it will then require re-starting the IDL.

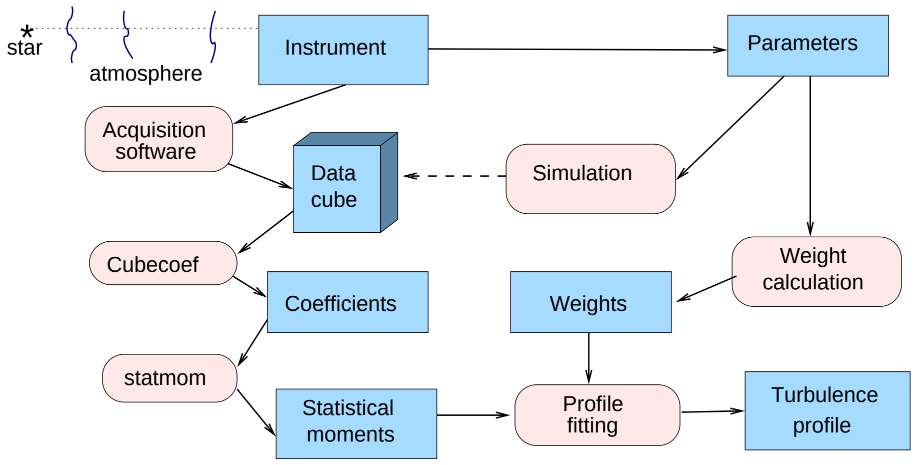

Figure 1: Scheme of the data processing. Blue boxes are data objects, pink ovals are program modules.

```
site: 'LaSerena',        ; site
lon: -71.2425,           ; longitude [deg]
lat: -29.91744,          ; latitude [deg]
D:   0.127,              ; aperture diam [m]
eps: 0.5,                ; central obstruction
pixel: 1.78,   ; pixel scale [arcsec]
lambda: 0.6e-6,          ; central wavelength [m]
pdist: 460.,             ; defocus exressed as conjug. height
ringradpix: 11.2,        ; default ring radius [pixels]
drad:  1.5,              ; ring width in lambba/(0.5*D*(1-eps)) units
abercoef: [6.562, 2.00, 2.00, 0.073, 0.074, -0.127, -0.127], ; for aber. calc.
zrad:  [-0.24,-0.15, 0.42,-0.58,-0.13, 0.09,-0.02],  ; Zernike 4-10 [rad]
mmax: 20,                ; max order of angular signals
nsect: 8,                ; number of sectors for radius calculation
interpol: 1,             ; use sub-pixel shifts
ron: 1.1,                ; readout noise, el
dirdat: '/uwa1/atokovin/RINGSS/Tololo20220501/',      ; fits data cubes
stmfile: '/uwa1/atokovin/RINGSS/stm2/2022-08-25.stm', ; .stm file
datfile: 'data/STM-20220825.idl',    ; results file, average
weightfile:  'wt-aug2022.idl',  ; weighting functions
qedata: 'qedata3.txt',    ; spectral response
aber: '',        ; no aberrations
zmat: 1}        ; apply saturation correction
```

The results of the data processing are stored in the structure array `dat`, one element per data cube. This structure, created by `allcubes5.pro`, is saved as an IDL variable. The elements of the data structure are:

```
el={fname:'',jd:0D0,hr:0,bv:0.,zen:0.,texp:1e-3,gain:0.,nx:0,nz:0,impar:impar, $
flux:0.,rad:0.,rrms:0.,pow:fltarr(mmax+1),   cov:fltarr(mmax+1), anoise:0.,rnoise:0.,  $
prof:fltarr(8),see:0.,fsee:0.,noise:0.,scint:0.,rmserr:0.,see2:0.,rnoise1:0.,wind:0.,tau0:0.}
```

The embedded sub-structure of the image parameters is

```
impar = {backgr:0.,flux:0.,fluxvar:0.,rad:0.,rwidth:0., $
       xc:0.,yc:0.,xcvar:0.,ycvar:0.,coma:0.,angle:0.,contrast:0.,noisepar:fltarr(4)}
```

The `dat` contains the star HR number, its B-V color, and the zenith distance. The turbulence profile is an 8-element array of $C_n^2 dh$ integrals (in $\mathrm{m}^{1/3}$) in eight pre-defined layers. The parameter `dat.flux` is expressed in electrons, using the gain value from the header and the hard-coded conversion from ADU to electrons appropriate for the camera model used.

3 Main program modules

Each .pro file contains one or several programs.

3.1 allcubes5.pro

allcubes5 creates the data structure and does most of the processing by calling other modules. Inputs: par. The optional keyword /flat indicates that a flat-field correction from a file flat.fits (must be in the input directory) should be applied to equalize pixel sensitivity of the detector. Optional keyword /list=list gives the list of file names (string array) to be included in the data. Used modules: cubecoef, statmom. This code should be adapted to the file naming conventions and the header content of the data acquisition software. The acquisition time is decoded from the file names, rather than from the headers. The average images derived from the cubes are saved on the disk.

getzen computes the zenith distance. It uses the site coordinates from the parameter file and needs valid HR numbers of the stars, saved in the `dat` structure. The star coordinates and B-V colors are found in the subset of the bright stars catalog saved as IDL structure in `bscb.idl` (see `getbsc.pro`). Used modules: none. The result is saved in the data structure.

noisecubes evaluates the noise. Input: parameters. Used modules: none. The result is saved in the data structure.

listcubes lists some parameters of the data structure in the text file `listcubes.txt`. Input: par. Used modules: none.

3.2 cubecoef.pro

This code processes one data cube. Inputs: image cube (3D array) and parameters. Optional inputs: flat=flatfield.fits for flat-field correction, /display to display the processing and stop at appropriate moments Used modules: none. Outputs: coef – matrix of coefficients extracted from the cube, impar – various parameters associated with this data cube. Optional outputs: imav – average re-centered ring image.

3.3 statmom.pro

Statistical processing of coefficients. Inputs: coef – coefficients determined by `cubecoef` (2D array), and the parameters. Modules used: none.

Outputs: pow – array representing the angular power spectrum size (1+mmax), where mmax is the maximum angular frequency specified by the parameters, rrms – dispersion of the differential sector variance in pixels, rnoise – estimate of the rms noise of differential sectors in pixels. Optional outputs: seeing – crude estimate of seeing from differential sector variance without correction for turbulence profile (normally not used), cov – covariance of the coefficients with a time lag of 1 frame.

3.4 profrest5.pro

getweight5 calculates the weighting functions (WFs) used for the turbulence profile restoration. Modules used: aweight3. Inputs: parameters. The WFs are computed for the distance grid of 16 log-spaced points from 0.25 to 32 km, including zero distance. The instrument spectral response is used, the weights are computed for black-body spectra of 10213 and 3938 K that approximate stars with $B - V$ of 0 and 1, to be linearly interpolated for any color. The U-functions are also computed for the two star temperatures, and the sets of the six U-coefficients needed for the wind measurement are found. The WFs for $m = 0$ expresses the differential sector variance in λ/D units.

The outputs are saved as IDL variables in the weights file with the name specified in parameters:

```
save, fi=par.weightfile, z,wt0,wtslope,ucoef0,ucoefslope,mm,lameff,zmat,hslope
```

The outputs are: z – grid of 16 distances [m] from 0 to 32 km, log-spaced, wt0 – matrix of weights [$\mathrm{m}^{-1/3}$], size 21x16, wtslope – weight slope vs. B-V star color, ucoef0 – U-coefficients for wind calculation and B-V=0, usoefslope – their slope vs. B-V, mm=[1,3,6,7,8,9] – angular frequencies corresponding to the U-coefficients, lameff – effective wavelengths for B-V of 0 and 1. The `zmat`, saved together with weights, is used for saturation correction which can be disabled in the parameter file; `zmat.idl` contains this matrix. The parameter `hslope` links the conjugation distance H to the ring radius in pixels as $r_{\mathrm{ring}}H = hslope$, so specifying both parameers is redundant. The code uses only the ring radius and estimates the conjugation altitude H, which is used then by `aweight3` for the WF calculation.

conjugation is an auxiliary routine to test the relation $r_{\mathrm{ring}}H = hslope$. Modules used: aweight3.

profrest5 fits the turbulence profile to the measured angulr power spectrum (APS). Inputs: pow – the APS vector (only the first 15-17 elements are used), cov – vector of covariances with time lag 1. The weights and other parameters computed by `getweight5` are given at input as well, interpolated for the B-V color of the specific star and its zenith distance.

Outputs: prof – turbulence integrals in $\mathrm{m}^{1/3}$ at 8 altitudes (not distances!) of 0, 0.25,0.5,... 16km, corrected for the zenith angle; erms – relative rms discrepancy between APS and its model, wind – effective wind speed (no zenith correction). Modules used: getprof, nnls.

testprof restores turbulence profile and other parameters for one data record, given as input. Used mostly for debugging.

allprof5 restores profiles for all elements of the data structure by calling `profrest5`. It also computes the seeing from the sector motion and estimates the wind speed. Inputs: parameters. Modules used: profrest5. The results are saved in the data structure.

3.5 getprof.pro

Restoration of turbulence profile. Inputs: z – distance grid [m], wt – weighting functions, scint – APS corrected for noise, exposure time, and saturation, escint – estimated errors of the APS. Modules used: nnls. Outputs: prof – turbulence integrals [$\mathrm{m}^{1/3}$] at 8 distances defined by the vector z, CHI2 – χ^2 metric of the discrepancy between APS and its model. The nnls code must be compiled before compiling `getprof` (see `pipe.pro`)

3.6 aweight3.pro

Function to compute the WFs used in the profile restoration. Inputs: z – vector of distances [m], mmax – maximum angular frequency, d – telescope diameter [m], eps – central obscuration ratio, pdist – conjugation distance below pupil [m], wav – array of wavelengths [m], sp – corresponding array of spectral response. Optional inputs: drho – with of the ring mask in FWHM units (default 1.5), pixel – angular pixel size [arcsec], zn – vector of static aberration numbers, zrad – vector of their amplitudes [radians at mean wavelength], blur – exposure-time blur [m]. Modules used: ztools (only if aberrations are specified).

Outputs: returns the matrix of weights of the size $N_m \times N_z$, in $\mathrm{m}^{-1/3}$, expressing the scintillation power for $m = 1...m_{max}$ and the sector motion for $m = 0$ for a turbulence integral of $J = 1\,\mathrm{m}^{1/3}$.

Optional output: ufunc – matrix of U-functions used for the wind estimation, same size as WF, in $\mathrm{m}^{7/3}$. If the keyword /ringrad is set, the function only returns the ring radius in CCD pixels.

4 Auxiliary code

This section describes programs not used directly for the data processing.

4.1 xd5.pro

This is a simple data browser GUI adapted to the current structure of the data files. Inputs: parameters. Modules used: various.

For each element of data, the GUI shows the main parameters (file name, ring radius, seeing, profile). Buttons allow to edit the data element, plot the APS and covariance, display the average image, repeat the profile restoration, and repeat `cubecoef` in display mode. Additional commands: seeing – plot two estimates of the seeing (from profile and differential sector motion) and the free-atmosphere seeing; corel – plot correlation coefficient, tempspec – plot temporal spectrum for a given frequency m, fluxvar – print flux uniformity among 8 sectors, conjugation – plot conjugation distance computed from the ring radius.

4.2 readprof.pro

This code reads the statistical-moments text files (.stm) produced by the python code and created the corresponding data structure compatible with the IDL package (e.g. with `xd5`). This is useful for testing alternative profile restoration (e.g. with different WFs) and for detailed data analysis with IDL. The python software normally does not save the data cubes, only the .stm files (statistical moments) and the restored profiles (.prof files). The module contains routines `readstm` and `readprof`, `readlist` for ingesting several nights of data, and `filterdat` to remove records that do not pass the quality control. An older version of the code `allstm.pro` does a similar job without ingesting the python profiles.

4.3 getimage.pro

Generate a nominal ring image without turbulence. Input: parameters. Optional input: static aberrations. Outputs: impix – the simulated image in CCD pixels, optional output: imh1 - high-resolution image. Calculation is performed for monochromatic light with the wavelenhth given by the prameters.

4.4 simatm.pro

Simulate complex amplitude of light at the ground after propagation through one or two turbulent layers. Used in simulations. The complex amplitude at the ground (a 2D complex array) is saved in `atm.idl`.

4.5 ringsim.pro

Simulate a data cube using the light amplitude saved in `atm.idl`. Inputs: parameters. Optional inputs: zn – vector of static aberration numbers, zrad – vector of their amplitudes in radians, wind – wind speed (default 10 m/s), blur – simulate exposure-time blur (default none), /debug – debug mode with intermediate stops. Optional inputs: starmag and ron. Ring motion on the detector on a circle can be optionally simulated by specifying the jitter parameter (circle radius in arcseconds). Modules used: ztools (only if aberrations are specified).

Outputs: data cube of 64x64xN size, where N is defined by the exposure time and the accumulation time (hard-coded). Every 20th ring image is displayed on the screen during simulation.

4.6 testsimul1.pro

Script to simulate a monochromatic data cube and process it. Modify as required to test the code or explore different situations. As the simulation is achromatic, the code uses the special parameter file sim1.par and a one-line spectral response qesimul.txt

4.7 aber.pro

This module estimates static aberrations of low order (Zernike numbers 4-10) from the average ring image or from the saved image parameters. Within the module, `abercoef` establishes the relation between Zernike amplitudes and measured coefficients for a particular instrument (they are included in the parameter file, see the example above), while `getaber` evaluates these aberrations from the information in `dat` (either average for all records or for one specific record). If the aberrations are substantial (e.g. exceed 1 radian), their effect can be included in the calculation of the WFs, and thus accounted for in the profile restoration (self-calibration). An optional file with static aberrations for weight calculation has two lines that specify the numbers of Zernike terms and the amplitudes in radians (at the parameter-specifiec wavelength):

```
znum:  4  5  6  7  8  9 10
zrad: -0.00 -0.15  0.42 -0.58 -0.13  0.09 -0.02
```

Note, however, that including arbitrary aberrations in the WF calculation is not yet implemented in `profrest5.pro`, although the underlying engine `aweight3.pro` does contain this option.

4.8 rdser.pro

If the data cubes are saved in the SER format (e.g. by the ASICap software for operating a ZWO camera), this code converts them into fits cubes. The details depend on particular acquisition software. Call `IDL> allser, 'dir'` to convert all cubes in the directory dir from .ser to .fits.

References

[1] Tokovinin A. Measurement of turbulence profile from defocused ring images. 2021, MNRAS, 502(1), 794-808.

[2] RINGSS description, documents, and code: *http://www.ctio.noirlab.edu/˜atokovin/ringss/*