Measurement of low-order aberrations in RINGSS

Authors:*A. Tokovinin*
Version: 2
Date: 2021-06-10
File: prj/smass/doc/aberraions.tex

1 Introduction

The theory of the RINGSS turbulence sensor assumes that the defocused ring images are distorted only by the atmoshperic turbulence, while the telescope optics is perfect. In practice, this is not quite true. It has been established by simulation that small static aberrations do not affect RINGSS measurements. However, aberrations with amplitudes larger than ∼1 rad distort the response (namely, the weighting functions, WFs) strongly and therefore bias the resulting turbulence profiles. If the optical aberrations are known, they can be explicitly included in the calculation of the WFs, allowing to get rid of the bias caused bt the imperfect feeding optics.

This document presents a practical method of measuring the aberrations. The aberrations are described by the Zernike polynomials (in the Noll's notation). I use the standard full-aperture polynomials that are not orthogonal for the annular aperture of the instrument, but the orthonormality is not needed for this study. The Zernike terms 1, 2, and 3 (piston and tilts) are irrelevant. The terms 4, 5, 6 characterize the defocus and astigmatism, 7 and 8 are the coma coefficients, 9 and 10 are the trefoil, and 11 is the spherical aberration.

The average image saved in the data processing can be used to iteratively fit the aberrations using the `donut` software. However, this tool is available only in IDL, and its port to python is a substantial work. Meanwhile, the mean radial and angular coefficients computed in the standard processing of the data cubes already contain the information on the low-order aberrations. Therefore, these aberrations can be determined from the regular data using a simple tool, without image analysis.

2 Development in IDL

The program `getimage.pro` generates the undistorted ring image using the instrument parameters. It can optionally take arguments `zn` and `zrad` – numbers and amplitudes of Zernike aberrations (in radians at the wavelength given by the parameters).

Using `getimage`, I studied the effect of aberrations on the cofficients C derived in RINGSS from the data cube. The algorithm of computing these coefficiens, implemented in `cubecoef.pro`, is re-created and applied to the simulated images or to the real average images. As the coefficients are determined by a linear operation, the average coefficients for a data cube equal the coefficients derived from the average image.

The first tests (see `aber1.pro`) have shown that the effects of defocus and spherical aberrations (terms 4 and 11) are almost identical: both affect only the ring's radius. The spherical aberrtion

influences also the ring width, but the width is not captured in the coefficients. Therefore, I restrict the fitting to the 7 Zernike terms from 4 to 10 and do not determine the spherical aberration.

The code `aber1.pro` applies sequentially N_z aberrations and records the resulting coefficients C_i. Their difference from the undistorded values $\Delta C_i = C_i - C_0$, normalized by the amplitude of the applied aberration, represents the *interaction matrix* of the system, I of the size $N_{coef} \times N_z$. For small-amplitude aberrations z (vector of Zernike coefficients in radians), the linearity holds, hence

$$\Delta C = Iz. \tag{1}$$

The left-hand side is measured, I is determined, and we can find the aberrations as the least-squares solution of (1). This is the same idea as in `donut`, except that here we use the coefficients ΔC, while `donut` operates directly on the image pixels.

I found that the least-squares method recovers well all simulated aberrations except coma (terms 7, 8). It is well known that coma displaces the image center. The coefficients C are computed from the centered rings. Although centering is included in the calculation of I, the response of the sector radii to coma is still distorted. Coma mostly affects the sector radii (the first 8 coefficients), but also the $m = 1$ angular signals.

Realilzing that the sector radii are related mostly to the Zernike aberrations 3, 4, and 5, I modified I by nulling the I elements for these terms that contain angular signals, leaving only the radii. On the other hand, for the coma (terms 7, 8), I nulled the I elements corresponding to the radii. With the modified matrix, the least-squares solution of (1) becomes stable and the coma is well recovered. This approach is implemented in `aber1a.pro`.

Finally, the detailed analysis of I shows that each coma and trefoil aberration affects only one angular signal, $m = 1$ for coma and $m = 3$ for trefoil. So, we can further simplify the least-squares problem and get rid of the matrix inversion. This approach is implmented in `aber1b.pro`. The input signals are the 8 sector radii and the 6 angular coefficients with $m = 1, 2, 3$ ($m > 3$ are not needed). These signals C' are selected from the full vector of the mean coefficients. The pre-calculated *selector* matrix S transforms them into quantities proportional to the Zernike coefficients. The piece of code computing S is:

```
smat[0,0:7] = 1.              ; focus 4
smat[1,0:7] = sin(2*phisect)  ; astig 5
smat[2,0:7] = cos(2*phisect)  ; astig 6
smat[3,12] = 1.               ; coma  7
smat[4,8] = 1.                ; coma  8
smat[5,14] = 1.   ; trefoil 9
smat[6,10] = 1.   ; trefoil 10
```

The focus term is related to the sum of the sector radii, the astigmatisms are derived from the radii weighted by $\sin\phi_i$ and $\cos\phi_i$, respectively, where ϕ_i are the angles of the sector centers. The remaining lines of S simply select one angular term related to the appropriate coma or trefoil aberration. The response to the aberration of 1 radian equals $R = SI$ (a 7-element vector). The aberrations z are then estimated from the S-weighted signals as

$$z = (SC')/R. \tag{2}$$

Here C' is the vector of 14 mean coefficients containing the 8 sector radii in pixels and the 6 angular coefficients. The values of the R vector for our prototype instrument are: (8.11, 2.50, 2.47, 0.096, 0.092, -0.156, -0.156). For example, changing the focus by +1 radian increases all sector radii by $\sim$1 pixel, while $z_7 = +1$,rad produces an average $m = 1$ sine coefficiengt of 0.092.

This simple algorithm works well on the simulated images, recovering the input aberrations. I applied it to the two real average ring images. Finally, I ran the algorithm on the coefficients determined by `cubecoef.pro`, instead of the images, and obtained the same results.

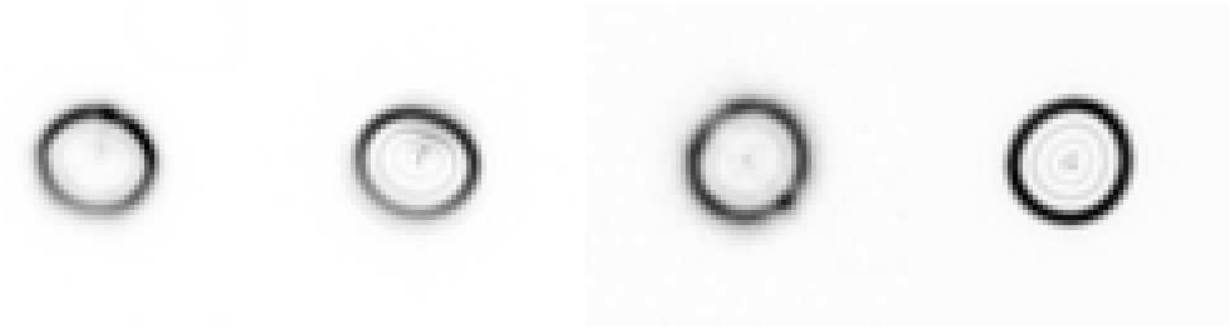

Figure 1: Modeling real average ring images: left: cube-1 taken on May 24, right: cube-2 taken on March 21. Each panel shows the real image on the left and its model on the right, both on the same scale. The real images are slightly blurred by turbulence, while the models are not.

The average images and their models with 7 Zernike terms computed using (2) are illustrated in Fig. 1. The cube-1 data `2021-05-24-233329_cube.fits` is taken on May 24 and suffer from substantial coma aberration. In contrast, on March 21 the telescope was better aligned, and the cube-2 `2021-03-22-000902_cube.fits` has smaller aberrations. The coefficients derived for both cubes are listed below.

Zn	cube-1	cube-2
4	0.10	0.28
5	-0.35	0.34
6	0.59	-0.08
7	1.27	-0.06
8	0.67	0.02
9	-0.03	0.05
10	0.01	0.15

3 Port to python

Calculation of Zernike aberrations is done by the code `zernike.py`, found on the web and slightly tuned. It contains subroutines to compute the radial part, define the n, m indices from the sequential number j, and, finally, compute the Zernike polynomial as `zer = zernikel(j,rho,phi)`, where `rho` is a list or array of radii (normalized by the aperture radius), and `phi` is the corresponding array

of angles. The result is not masked by the pupil, i.e. it returns non-zero values at radii exceeding one. The code was compared with the IDL `ztools.pro` and found to give the same results. The polynomials correspond to the Noll definition and are orthonormal on the pupil of unit radius.

The progam `getimage.py` computes a 2-dimentional monochromatic ring image that optionally includes static Zernike aberrations: `img, rad = getimage(d,eps,pdist,lam,pixel,zn=[],zrad=[])`. The ring radius in pixels is returned as a second parameter. The image is computed on an internal grid of 512x512 'fine' pixels and re-binned on the coarse CCD pixels (the rebin utility, not found in numpy, is included in this module). The image size is a power of 2, e.g. 64x64 pixels. I found that the resulting ring was shifted from the grid center by half-pixel in both coordinates because of the re-binning, and corrected most of this offset by back-shifting the computed image prior to its rebinning. The image is normalized so that its sum equals one.

The code `cube2.py` is slightly modified: now, after computing the radii and angular coefficients for a data cube, it saves the selected mean values needed for the aberration calculus in the `['moments']['mcoef']` section of the resulting dictionary. This is an array of 14 elements: 8 mean sector radii, 3 cosine terms for m=1,2,3, and 3 corresponding sine terms.

The code `aberration.py` determines the 7 response coefficients to the Zernike aberrations and saves them in the `aberration.json` file. The algorithm is ported directly from IDL. Using the instrument parameters dictionary, the function `aberresp(par)` returns the dictionary that contains the 7 response coefficients R, as well as the matrix S and the mean radius of the unaberrated image. This calculus uses `getimage.py` and implements the same algorithm of the coefficients calculation as in `cube2.py`. Once the response is defined, the call `zampl = getzampl(par,mcoef)` computes the 7-element vector of Zernike aberrations (in radians at 600 nm). The main program can be called as `python aberration.py <par-dict> <data-dict>`. The resulting Zernike amplitudes are added to `aberration.json`.

Given the same inputs, the IDL and python codes produce the same response to aberrations. As an example, I processed the real data cube:
`python aberration.py par-mar21.json data/2021-05-24-233329.json`. The table below compares the Zernike coefficients determined for this cube by the IDL and python codes. A small discrepancy in the defocus is caused by the differences in the nominal ring radius: IDL uses the ring radius returned by `getimage`, computed on the fine grid without radial masking, while in python the mean nominal radius is computed from the sector radii of the un-aberrated and binned image, using the radial mask. These radii are similar, but not identical, e.g. 11.56 and 11.41 pixels. The python algorithm is more rigorous.

Zn	IDL	Python
4	0.10	-0.51
5	-0.35	-0.29
6	0.59	0.62
7	1.27	1.61
8	0.67	0.64
9	-0.03	0.01
10	0.01	0.02

Once the aberrations are known, they can be accounted for in the calculation of the weights.

Adding a sum of static aberrations to the conic wavefront in `aweight.py` seemed trivial. However, this modification produced unexpectedly wrong results even for a small-amplitude coma, while it worked for defocus and astigmatism. The reason for this discrepancy was found to be in the subtle differences between the direct and inverse FFTs and their normalization as implemented in IDL and numpy. The problem was corrected.

After correction of `aweight.py`, I compared again the weights calculation by the IDL and python tools. In python, `testweight.py` computes the polychromatic weights for $m = 0...20$ (the $m = 0$ weight refers to the sector motion) and 5 distances $z = 1, 2, ...16$ km and saves the table in `weight3.txt`; it uses `aweight.py`. The instrument parameters are as defined in `par-mar21.json`. Then the IDL code `weightcompare.pro`computes the same weights using `aweight3.pro` and compares them with the results of python.

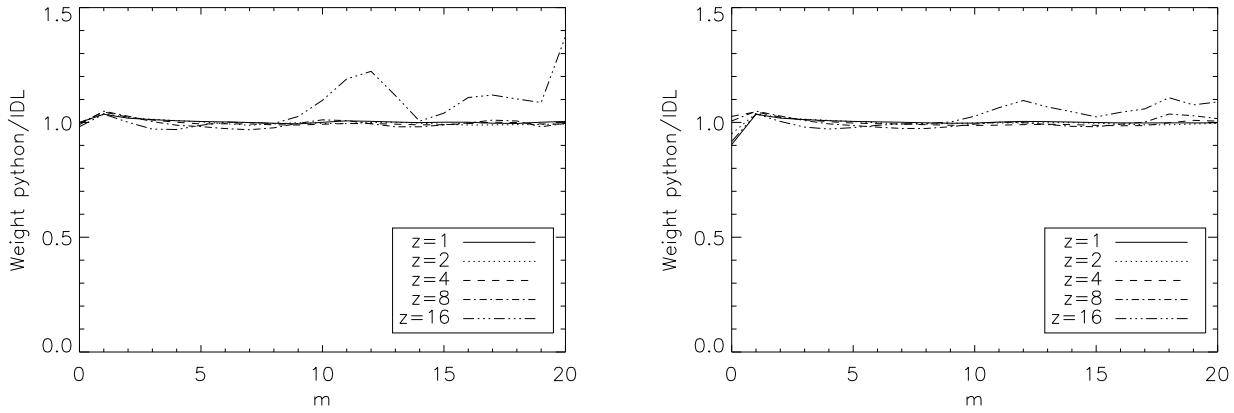

Figure 2: Ratio of weights computed in python and in IDL, for 5 distances from 1 to 16 km shown in the legend. Left: no aberrations, right: coma of 1 radian.

Figure 2 plots the ratio of weights calculated with python and IDL for a perfect optics (left) and for a 1-radian of coma (right). In the latter case, 5 radians of tilt of opposite sign are added to compensate ring displacement produced by the coma and keep the rings approximately centered. The results are very satisfactory. With coma, the sector weight in python is slightly less than in IDL, while for perfect optics their difference does not exceed 2%.

4 Tolerable aberration amplitudes

Using IDL, I explored the impact of low-order aberrations on the WFs. The calculation is programmed in `biases.pro` using instrument parameters `mar21a.par`.

The left panel of Fig. 3 compares the first 10 angular weights without and with coma. The $m = 1, 2$ weights remain unaffected, while at larger m coma increases the weights, and the red curves are elevated above the black ones. The right-hand panel shows the relation between the amplitude A of the $m = 1$ angular coefficient and coma z_7. It is linear up to $\sim$2 rad with a slope of 0.086. The slope is the coma response coefficient, which is found above to be 0.096/0.092 using a different tool.

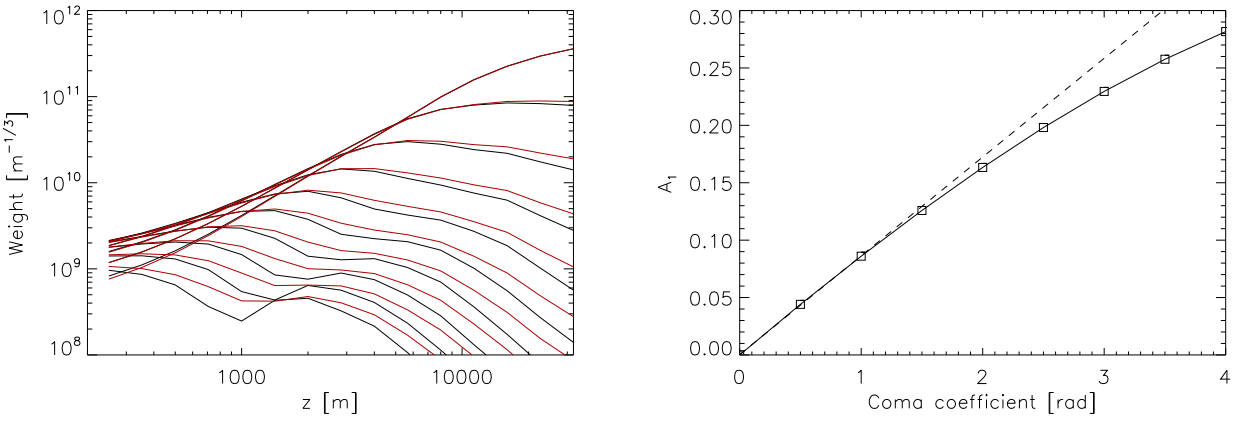

Figure 3: Left: WFs without aberration (black) and with 1 rad of coma (red). Top to bottom: curves for $m = 1, 2, ...10$. Right: relation between the coma amplitude z_7 in radians and the $m = 1$ mean coefficient A. The dashed line has a slope of 0.086.

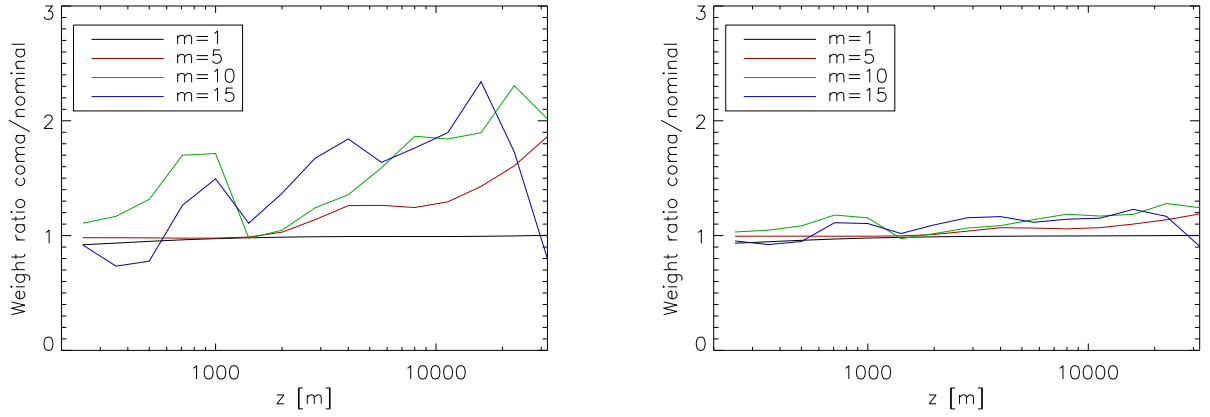

Figure 4: Ratio of the WFs with and without coma. Left: 1 radian, right: 0.5 radian.

The procedure `comaweight` in `biases.pro` evaluates the effect of aberrations on the WFs. In the case of coma, the tilt with a coefficient of -5 is also applied to keep the ring centered. The effect is not proportional to the aberration amplitude but, on the contrary, is highly non-linear. As shown in Fig. 4, 1 rad of coma modifies the WFs by as much as two times (mstly at large m and z), while 0.5 rad of coma causes a much smaller effect, only within 20%. In practice, the impact is even smaller because at large z the WFS are much smaller than at maximum (for a given large m). When the difference of WFs normalized by the WF maximum is plotted, the effect of 0.5 rad coma is under 7%, i.e. quite tolerable.

Figure 5 plots the normalized differences of WFs for astigmatism and trefoil. An astigmatism of 0.5 rad produces an acceptably small change of the WFs, just as coma. On the other hand, a trefoil of

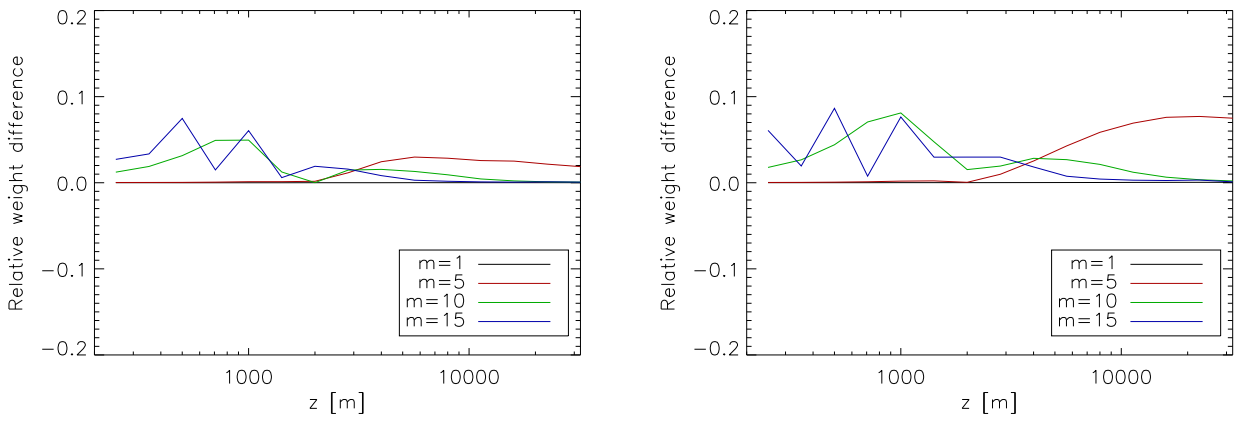

Figure 5: Difference between aberrated and un-aberrated WFs normalized by the WF maximum at each m. Left: astigmatism of 0.5 rad, right: trefoil of 0.3 rad.

the same amplitude gives a much larger effect, and the threshold of acceptable trefoil must be lowered to 0.3 rad, as shown in the right panel. In all cases, the aberrations increase the WFs, leading to an over-estimate of turbulence strength.

5 Summary

Standard processing of the RINGSS data cubes offers a convenient way to evaluate low-order aberrations from the mean angular coefficients and ring radii. The relation between aberrations and coefficients was studied using IDL, and tools for estimating these aberrations are ported to python and tested. This means that the RINGSS instrument can be self-calibrated with respect to aberrations.

In practice, some coma aberration is expected from an imperfect or unstable alignment of the Cassegrain telescope, and some astigmatism arises from a slight de-centering of the re-imaging lenses. The astigmatism also depends on the star position in the field. Processing the real data shows the presence of both aberrations. Their amplitudes in cube-2 above is small and does not cause trouble, while in cube-1 the aberrations arising from poorly aligned optics require re-calculation of the weights.

Aberrations produce a positive bias on the WFs. This effect can be accounted for in the data processing. The bias is a non-linear function of the aberration amplitude. Small (under 0.5 rad) aberrations can be safely neglected. If the data reveal larger aberrations, they can be icluded in the parameters and the weights can be re-computed to cancel this bias. The strategy of dealing with aberrations will be determined during initial test phase of the prototype. This study gives the tools needed to chracterize aberrations and account for them if necessary.