PUAKO notes #02:
Telemetry processing facility

O. Beltramo-Martin*

January 6, 2021

Contents

1 Rationale

PUAKO calls several Matlab class objects to perform the PSF reconstruction. One of them is the so-called *telemetry* class that aims to (i) grab calibrated data of the system environment (pupil model, NCPA,field static,..) and get the seeing and C_n^2 profile from the MASS-DIMM, (ii) read the AO telemetry from the Keck *Telemetry Recording System*, model the loop temporal transfer function and estimate the noise variance and the seeing, (iii) recover the NIRC2 raw image and process it. This object is designed to process either on-sky Keck data (TRS and NIRC2 so far) or simulated data using the OOMAO framework [3, 2] and more specifically the *aoSystem* object implemented within OOMAO (see KASP notes). I detail in this document how to handle the *telemetry* object as well as the methodology currently implemented to estimate the WFS noise and the seeing, with application to Keck II AO data.

2 The Telemetry class

The *telemetry* class must be called from five required inputs, on top of which the user can define and eleven optional parameters as follows

*olivier.beltramo-martin@lam.fr

```
trs = telemetry(objname,pathTRS,pathIMAG,data_folder_path,fitsHeader,'flagNoisemethod',
'autocorrelation','badModesList',[],'jMin',4,'jMax',120,'fitL0',true,'flagBest',false,
'flagMedian', false, 'wvl',0.5e-6,'aoMode','NGS','D1',9,'D2',2.65)
```

where

- `objname` is a cell containing the object name as it appears in the .fits and .sav files, like 'n0004' for instance.
- `pathTRS` is a vector of character, like ´/home/path´that points the telemetry folder location.
- `pathIMAG` is a vector of character, like ´/home/path´that points the NIRC2 images folder location.
- `data_folder_path` is a structure with fields *calibration* and *massdimm* that refers respectively to the calibration data and MASSDIMM folders.
- `fitsHeader` is a cell obtained by reading the header fits file.
- `flagNoisemethod` denotes the method used to estimate the WFS measurements noise among 'autocorrelation' (default), 'theoretical' and 'rtf'. See Sect. 3.2 for a complete description.
- `badModesList` is an array of integers that define the Zernike modes that are excluded from the model-fitting of Zernike modes variance. None are excluded by default and the fit is performed over the range of Nolls's index from `jMin` (4 by default, i.e. focus) to `jMax` (120 by default).
- `fitL0` permits to fit the outer scale if set to true (default). On top of that, the user can choose different fitting strategies by setting `flagBest` and `flagMedian`. See Sect. 3.2 for a complete description. The final r_0 estimation will be given at the wavelength `wvl`, which is 500 nm by default.
- The parameter `aoMode` is only useful when dealing with simulated data to identify whether the system is working with a LGS or not.
- Parameters `D1` and `D2` allow to define an annular pupil of outer diameter D1 and inner diameter D2 on which the phase is reconstructed and the Zernike modes variance estimated. This is particularly useful for mitigating pupil-edges effects due to salved actuators.

Practically, it can be annoying to define to call the *telemetry* class by defining all these inputs. I advice so to instantiate PUAKO (see PUAKO note #01) use the *grabData* facility as follow

```
p = puako('path_imag',path_imag,'path_trs',path_trs,'path_calibration',path_calib);
p.grabData({'n0004'},'flagNoisemethod','autocorrelation','badModesList',[],'jMin',4,'jMax',
120,'fitL0',true,'flagBest',false,'flagMedian',false,'wvl',0.5e-6,'D1',9,'D2',2.65)
p.trs
```

PUAKO will automatically pass the appropriate fields to instantiate the telemetry class that will be accessible from the *trs* properties of the PUAKO object. If you want to use the *telemetry* class on simulated data, we must use the *aoSystem* class implemented within the OOMAO framework (see KASP notes) as follows

```
aoSys = aoSystem(parm,'runSimulation',true);
trs = telemetry(aoSys,'none','none',struct(),{''});
```

The *aoSystem* will automatically run a simulation from the structure *parm* containing the simulation parameters and update the *aoSys.loopData* with the simulated telemetry and PSF. The *telemetry* class will recognized the *aoSystem* class if provided in input and calls the function *fromAoSystemClassToTelemetry* to fill all the required fields. It will then model the loop transfer functions and estimate the noise variance and the seeing using the exact same functions than if it was on-sky telemetry. I present this processing in the next section.
If the user wants to deal with on-sky data, the data reading scheme is the following:

1. Restore calibrated data using the *restoreCalibratedData* function. Form the calibrate folder, PUAKO will (i) restore the static aberrations maps according to the observing data, (ii) define the pupil model from the fits file header, (iii) model the DM influence functions and (iv) get all the calibration of dark, flat, bad pixels and background maps if none is found in the images folder.

2. Restore telemetry data, which are slopes, tip-tilt, DM commands, pixels intensity, reconstructed wavefront in the actuators space using the function *restoreKeckTelemetry*. This function also gets the AO and telescope configuration, such as the servo-lag transfer loop, the zenith angle, ... and fills in the *telemetry* object properties.

3. Restore MASS-DIMM data using the *profiler* object class (another one !). This latter has a *fetchData* function that does read the MASS-DIMM folder inside the CALIBRATION folder so as to look for a .dat file containing all the MASS-DIMM measurements for a single night. If the file corresponding to the observing night you want to the process the data, the *fetchData* facility will directly read the url http://mkwc.ifa.hawaii.edu/current/seeing/ to download the missing .dat file if it exists. If not, the atmospheric parameters are set to median conditions at Mauna Kea, i.e.

```
r0 = 0.16*airmass^(-3/5);
weights  = [0.517 0.119 0.063 0.061 0.105 0.081 0.054];
% Fixed parameters independent from the MASS-DIMM
L0 = 25
windSpeed = [6.8 6.9 7.1 7.5 10.0 26.9 18.5];
windDirection = [0 pi/2 pi/4 3*pi/2 -pi/4 pi/6 -pi/4];
heights = [0 .5 1 2 4 8 16]*1e3*airmass;
```

   From the MASS-DIMM data, the routine will look for the closest measurements to the data given in the fits header. Eventually, we get (i) *seeingDIMM* as the seeing provided by the DIMM at wvl=500 nm, (ii) *seeingAlt* as the free atmosphere seeing obtained from the MASS data and (iii) *cn2MASS* as the C_n^2 profile in the free atmosphere measured by the MASS. Then, PUAKO updates the atmosphere characteristics as follows

```
r0 = 0.976*3600*180/pi*wvl/seeingDIMM*airmass^(-3/5);
seeing0 = (seeingDIMM^(5/3) - seeingAlt^(5/3))^(3/5);
mu0    = (0.976*constants.radian2arcsec)^(5/3)*0.423*4*pi^2/wvl^(1/3);
Cn2h = [seeing0^(5/3)/mu0 cn2MASS];
weights = Cn2h/sum(Cn2h(:));
```

4. Read and process the fits file using the *processDetectorImage* function. The processing is quite standard and simple and does (i) compensate for dark, flat and remove hot pixels, (ii) subtract precisely the residual background by centering the pixel histogram around zero, (iii) subtract a quadratic background map model fitted over the whole CCD and (iv) interpolate bad pixels by taking a weighted averaged of closest neighbors. Starting from this point, the image should be quite clean and the algorithm detects the PSF

using the max method after applying a median-filter and crops the PSF. The image is then passed to the *psfStats* object that calculates the PSF figures of merits, such as the Strehl-ratio, the FWHM and so on.

The *telemetry* class gets through all these steps successively and automatically and for each object whose name has been provided by the user from the *obj_name* field. If the user enters a multi-dimensional cell when calling PUAKO, like { 'n0004','n0005' }, PUAKO creates a *telemetry* for each object that can be accessed from the command `p.trs(k)`. If you want to do so, PUAKO does check the memory for you and will either warn you if loading all the telemetry requires a lot of memory or discard the loading if it requires more than 95% of your memory. Note that this facility is only available on UNIX platform.

3 Telemetry processing

3.1 AO Transfer functions modeling

The AO temporal transfer functions modeling is straightforward and necessitates only few parameters from the telemetry, which are:

- The high-order and tip-tilt gains, sampling frequency and latency. Thus latter is calibrated from the detector clock and a look-up table is given from the function estimateLoopDelay.
- The servo-lag transfer function coefficients for the high-order ('DM_SERVO') and tip-tilt ('DT_SERVO') loops. This transfer function is described using four parameters (a_0 to a_3) for the numerator and three parameters for the denominator (b_1 to b_3) as follows

$$\kappa_{\text{servo}}(f) = \frac{a_0 + a_1 z + a_2 z^2 + a_3 z^3}{1 + b_1 z + b_2 z^2 + b_3 z^3}, \tag{1}$$

where f is the temporal frequency defined from 0 to $F_e/2$ where F_e is the temporal sampling frequency and $z = \exp(-2i\pi f/F_e)$ the Laplace's variable. PUAKO also models the WFS and the loop delay transfer functions as follows

$$\begin{aligned} \kappa_{\text{wfs}}(f) &= \text{sinc}(f/F_e)\exp(-if/F_e) \\ \kappa_{\text{lag}}(f) &= \exp(-2if\Delta t/F_e), \end{aligned} \tag{2}$$

where Δt is the latency in seconds. Then, PUAKO calculates the open-loop, closed-loop, rejection and noise transfer function using

$$\begin{aligned} \kappa_{\text{ol}}(f) &= \kappa_{\text{wfs}}(f)\kappa_{\text{lag}}(f)\kappa_{\text{servo}}(f) \\ \kappa_{\text{ctf}}(f) &= \kappa_{\text{ol}}(f)/(1 + \kappa_{\text{ol}}(f)) \\ \kappa_{\text{rtf}}(f) &= \kappa - \kappa_{\text{ctf}}(f) \\ \kappa_{\text{ntf}}(f) &= \kappa_{\text{servo}}(f)/(1 - \kappa_{\text{ol}}(f)). \end{aligned} \tag{3}$$

Such transfer functions are useful to derive the noise propagation factor for AO error breakdown assessment, data analysis and WFS measurement noise estimation. The user can have a quick look on transfer function by calling the property *displayAoTransferFunction* of the *telemetry* object.

3.2 Noise covariance estimation

Several techniques exist to get an estimate of the noise covariance function which is required for the PSF reconstruction stage (see PUAKO notes #03). The point is to describe the impact of the noise contaminating the slopes calculation on the PSF shape. The variance, more generally the covariance matrix, of this noise

depends on the centroiding method the slopes calculation relies on, as well as the hardware characteristics of the whole optical chain and the guide stars magnitude. The theoretical background is well established [14, 13]; however estimating the noise from theoretical formulas would require to have specific calibration of the total optical throughput, detector quantum efficiency and jnow the exact shape of the spot in the WFS detector focal plane. Moreover, one may understand how this measurement noise propagates through the AO loop to really capture its impact on the PSF shape.
A more pragmatic approach consists in estimating the noise covariance matrix from the AO telemetry directly. We must distinguish two noise contribution then; at each loop iteration, the WFS sees the noise that has been measured at previous iterations and propagated through the loop, plus a new realization of the detector and shot noise coming from the guide star and the sky. However, only the first one impacts the PSF shape eventually, meaning we must be able to determine the unpropagated noise so as to subtract it to get the PSF [15]. Either we perform this estimation using the WFS measurements and then propagate in the DM actuators space [12, 15], or determine the propagated noise using the DM commands [7, 4, 5] directly.
Within the PUAKO framework, the user can choose three different methods to estimate the noise contribution, which are

- **Theoretical calculation**: Assuming the noise variance is the same for each The noise covariance matrix is estimated by

$$\widehat{C_\eta} = \frac{\lambda}{2\pi d \Delta\theta}\sigma_\phi^2 \boldsymbol{R}\boldsymbol{R}^t, \tag{4}$$

  where λ is the WFS central wavelength, d the sub-aperture size, $\Delta\theta$ the detector pixel scale, σ_ϕ^2 the WFS noise variance in rad^2 for a single sub-aperture and $\boldsymbol{R}$ is the command matrix. The value of σ_ϕ^2 for a quad-cell is given by [14]

$$\sigma_\phi^2 = 4\pi^2\left(\frac{\sigma_{e^-}}{n_{\rm ph}}\right)^2 + \frac{\pi^2}{n_{\rm ph}}, \tag{5}$$

  assuming a diffraction-limited spot and where σ_{e^-} is the read-out noise and $n_{\rm ph}$ the number of photons collected during a frame and for a sub-aperture.

- **The auto-covariance method**: the auto-covariance peak of any actuator commands can be split into a turbulence and a noise part. However, if the turbulence coherence time is much larger than the WFS frame rate, the one frame delay cross-correlation, which does not include anymore the noise contribution as it obeys to a white distribution, contains the same turbulence contribution. Therefore, by subtracting the cross-covariance to the auto-covariance, one may estimate the noise variance. Concretely, the noise cross-variance σ_η^2 between DM actuators u and v is estimated as follows

$$\widehat{\sigma}_\eta^2(u,v) = 0.5 \times (\langle(\boldsymbol{u}_0(t) - \boldsymbol{u}_{-1}(t))\,\boldsymbol{v}_0(t)\rangle + \langle(\boldsymbol{u}_0(t) - \boldsymbol{u}_1(t))\,\boldsymbol{v}_0(t)\rangle), \tag{6}$$

  where $\boldsymbol{u}_k$ is the k-frames shifted command of the actuator u. Note that contrary to what proposed by [4], I average the noise estimations obtained by including a positive and negative one frame delay, which makes the estimation more robust in practice.

- **The rejection transfer function method**: This technique has been proposed by [7] and consists in looking for a plateau in the DM commands spectrum. This necessitates to compensate DM commands for the loop transfer function, which must be modeled regarding the RTC configuration. The noise cross-variance σ_η^2 between DM actuators u and v is estimated as follows

$$\widehat{\sigma}_\eta^2(u,v) = \mathrm{med}_n\left(\mathcal{R}\left[\frac{\tilde{\boldsymbol{u}}(f)\tilde{\boldsymbol{v}}^*(f)\Pi_n(f)}{(\kappa_{\rm cl}(f)/\kappa_{\rm wfs}(f))^2}\right]\right) \tag{7}$$

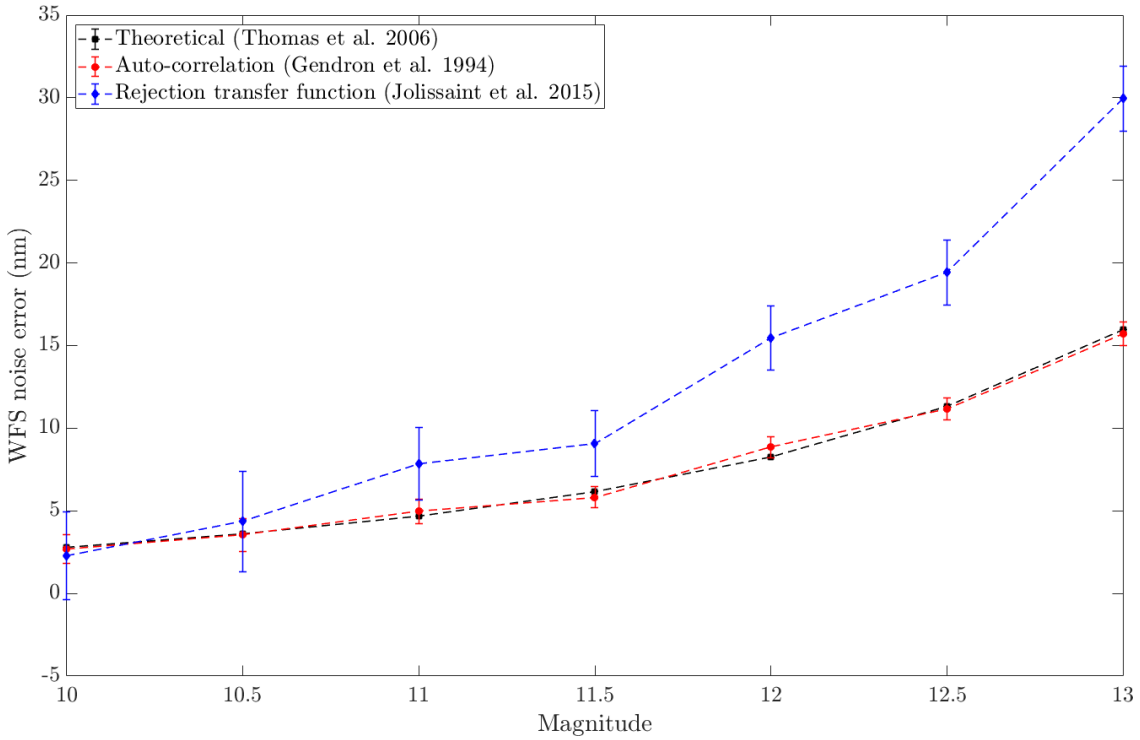

Figure 1: WFS noise error estimates in nm obtained from the three different techniques detailed above as function of the NGS magnitude. Results are obtained on NGS Keck simulated data using the OOMAO simulator [3] with $\sigma_{e-} = 1e-$, no photon noise, a monochromatic beam at 640 nm at a sampling frequency of 1kHz.

where $\text{med}_n(x)$ is the median value of $x((f > n \times f_s/2)\ \&\ (f < f_s/2)$, $\mathcal{R}(x)$ is the real part of x, $\tilde{\boldsymbol{u}}(f)$ is the 1D Fourier transform of the actuator u command and x^* the conjugate of x. We select a sub sample of high-spatial frequencies by truncating the Fourier transforms thanks to the multiplication with the gate function $\Pi_n(f)$ defined by

$$\Pi_n(f) = \begin{cases} 1 \text{ if } (f > n \times f_s/2)\ \&\ (f < f_s/2) \\ 0 \text{ otherwise} \end{cases}, \tag{8}$$

with f_s the WFS sampling frequency and n an arbitrary number that allows to truncate low temporal frequencies, which has been practically set to $n = 0.9$.

Both these techniques should be comparable as both rely on a similar assumption, e.g. the turbulence coherence time is negligible toward the WFS frame rate. If not, the noise estimation will be biased by a residual atmospheric turbulence whatever the technique employed: the temporal cross-variance will contain a different turbulence signature than the auto-covariance term, while the plateau in DM commands PSD will be masked by the servo-lag error contribution that contains high temporal frequencies owing to high velocity conditions. My preferred alternative remains the auto-covariance method as it is computationally faster, with a more robust methodology and compares very well to theoretical expressions as presented in Fig. 1

The results of this process can be found in the structure *trs.res.noise* that contains the noise covariance matrices (high-order and tip-tilt modes) as well as the noise standard-deviation (obtained from the matrix trace) and the method identification. It is possible to switch between a method to another by specifying the field 'noisemethod' when calling the *grabData* PUAKO facility or instantiating the *telemetry* object directly. The possible option are 'theoretical', 'autocorrelation' (the initial method described in [4]), 'autocorrelationSym' (default value) and 'rtf'.

3.3 Seeing estimation

Estimating the seeing is a critical step for PSF reconstruction so as to model the PSF wings that contain spatial frequency outside the AO DM cut-off frequency. This functionality is provided by the function *estimateSeeingFromTelemetry* that takes the full *telemetry* object as an input. It uses the standard technique [6] that consists in estimating the Zernike expansion of the atmospheric phase reconstructed from the DM commands and retrieve r_0, L_0 by adjusting a model of the Zernike coefficients variance.
In practice, actuators at the pupil edges are usually slaves which includes phase non-stationary that biases the Zernike variance estimation. In order to mitigate this effect, we reconstruct the phase over a sub-pupil of outer and inner diameter of 9 m and 2.65 m respectively. Moreover, we must limit the number of reconstructed Zernike modes to the maximal Noll's index $j_{\max}$ so as to mitigate the noise propagation in the estimated coefficients variance, as well as exclude the tip-tilt modes.
From these assumptions, a rough r_0 estimates can be deduced by the model of the Zernike coefficients variance [11] that says that the tip-tilt excluded atmospheric phase variance is given by $0.134(D/r_0)^{5/3}$. When subtracting the contribution of non-reconstructed high order Zernike modes, we get

$$\begin{aligned} r_0^{\text{init}} &= D \times \left(\frac{0.134 - 0.2944 \times j_{\max}^{-\sqrt{3}/2}}{\widehat{\sigma^2}_\phi} \right)^{3/5} \\ \widehat{\sigma^2}_\phi &= \frac{4\pi^2}{\lambda^2 \times n_{\text{act}} \times n_{\text{frame}}} \sum_{i=1}^{n_{\text{act}}} \sum_{k=1}^{n_{\text{frame}}} (\boldsymbol{u}_i(k) - \bar{\boldsymbol{u}}_i)^2 - \widehat{\sigma^2}_\eta, \end{aligned} \tag{9}$$

where $\widehat{\sigma^2}_\phi$ is estimated from the DM actuators commands covariance subtracted from the noise covariance estimated previously. This value is considered as an initial guess of the model-fitting step that will estimate jointly the r_0 and L_0 by minimizing the following criterion

$$\mathcal{J}(r_0, L_0) = \left\| \sum_j^{n_j} \boldsymbol{\sigma^2}_z(\text{idx}(j)) - v_z(\text{idx}(j), D, r_0, L_0) \right\|^2, \tag{10}$$

where $\boldsymbol{\sigma^2}_z$ is the empirical Zernike coefficient variance, $v_z(j, D, r_0, L_0)$ the r_0, L_0-dependent model assuming a Von-Kármánn turbulence and idx the Noll's index of Zernike coefficients considered in the model-fitting, that is performed thanks to a Levenberg-Marquardt algorithm. By default, the initial L_0 is set to 25 m and be keep as fixed by specifying the field 'fitL0' as false. Then, the algorithm runs the following steps

1. Call of the function *dmCommandsToZernike* to estimate $\boldsymbol{\sigma^2}_z$. The Zerkike coefficients are estimated from

$$\begin{aligned} \boldsymbol{z}(k) &= \boldsymbol{R}_z \boldsymbol{u}(k), \\ \boldsymbol{R}_z &= \left[(\boldsymbol{Z}_{\text{trunc}}^t \boldsymbol{Z}_{\text{trunc}})^{-1} \boldsymbol{Z}_{\text{trunc}}^t \right] \boldsymbol{F} \end{aligned} \tag{11}$$

   where $\boldsymbol{F}$ is the DM influence functions matrix and $\boldsymbol{Z}_{\text{trunc}}$ is the masked Zernike polynomials within the circular tore defined using the outer and inner diameter on which the phase is reconstructed. The user can specify these diameters using the fields 'D1' and 'D2' respectively. This operation done in Eq. 11 allows to perform the projection of the DM commands over the truncated Zernike modes. The variance is then obtained from the variance empirical estimator applied to the retrieved Zernike coefficients, which is then subtracted from the DM noise projected onto the Zernike modes using the reconstructor $\boldsymbol{R}_z$ introduced in Eq. 11.

2. Correction of aliasing the empirical variance by using a 5th order polynomial model of the variance excess due to the aliasing [6], that remains valid up to $j_{\max} = 120$.

3. Remove of some Zernike coefficients in the variance estimates and the model if the users believes that they can not be trusted. If so, one must specify the field 'badModesList' as a vector containing the Noll's index of modes to be excluded from the fit.

4. Perform the model-fitting (function *fitZernikeVariance*). Three different options are accessible: (i) do the fit once, (ii) do successive fits by increasing $j_{\min}$ progressively up to j_4 (arbitrary value) and keep the one that provides the best overall precision on r_0 and L_0 (put the field 'flagBest' as true) and (iii) do successive fits by removing systematically the empirical modes variance that are above 3-σ from the adjusted model until all samples remains less far from this threshold to the model (put the field 'flagMedian' as true).

5. Check if the 3-σ precision on the r_0 is lower than 1 m; if not, the algorithm redo the full model-fitting process with a fixed L_0 value set to 25 m by default in order to mitigate any possible degeneracy.

Results are accessible from the fields 'trs.res.zernike' (empirical modes variance plus adjusted model, propagated noise and Noll's index) and 'trs.res.seeing'. This latter contains the estimated r_0, L_0 as well as (i)

$$\begin{aligned} w_0^K &= \frac{k_1}{\widehat{r_0}} \\ k_1 &= \frac{0.976 \times \lambda \times 180 * 3600}{\pi} \end{aligned} \tag{12}$$

the Kolmogorov PSF FWHM (or the standard seeing definition) (ii) the Von-K'arm'ann PSF FWHM accounting for the outer scale [10]

$$\begin{aligned} w_0^{VK} &= w_0^K \times \delta \\ \delta &= \sqrt{1 - k_2 \times (\widehat{r_0}/\widehat{L_0})^{k_3}}; \quad k_2 = 2.813 \text{ and } k_3 = 0.356 \end{aligned} \tag{13}$$

On top of that, the 95% confidence intervals Δr_0 and ΔL_0 can be found in the 'res.seeing' structures. Both of them are derived from the fit residual and the Jacobian matrix thanks to the `nlparci` Matlab native function. Similar confidence intervals on the seeing values are deduced by

$$\begin{aligned} \Delta w_0^K &= \frac{k_1 \Delta r_0}{r_0^2} \\ \Delta w_0^{VK} &= \sqrt{\left(\Delta w_0^K \delta\right)^2 + \left(w_0^K \Delta\delta\right)^2} \\ \Delta\delta &= \sqrt{\left(\frac{\Delta r_0 \partial\delta}{\partial r_0}\right)^2 + \left(\frac{\Delta L_0 \partial\delta}{\partial L_0}\right)^2} \\ \frac{\partial\delta}{\partial r_0} &= \frac{k_2 k_3 L_0^{-k3} r_0^{k_3-1}}{2\delta} \\ \frac{\partial\delta}{\partial L_0} &= \frac{k_2 k_3 L_0^{-k3-1} r_0^{k_3}}{2\delta} \end{aligned} \tag{14}$$

Both seeing values will be determined regardless whether the outer scale is fitted or not.

4 Application to Keck data

I have processed 304 files obtained during four nights (February 3rd 2013, August 1st 2013, September 14th 2013 and March 14th 2017) whose 226 were obtained in NGS mode and 78 in LGS mode (in 2017).

Table 1: Statistics on r_0 and L_0 estimates obtained from the AO telemetry acquired during four different nights. The total sample was composed of 304 data. In 2017, the data set gathers 78 samples in LGS mode and 30 in NGS mode.

	r_0 (cm)		L_0 (m)	
	Median	1-σ std	Median	1-σ std
2013 02 03	26.0	5.9	10.2	1.4
2013 08 01	23.4	6.2	11.7	3.1
2013 09 14	16.7	3.8	10.4	4.8
2017 03 14	14.6	2.9	10.6	2.3
All	17.1	6.4	10.7	3.3

In Fig. 2, I present the WFS noise error calculated using the auto-covariance method as function of the Guide star flux expressed in ADU/m^2/s. We observe trends of the standard-deviation error in $1/$ flux, which is expected for read-out noise limited measurements [14, 13].

We also observe different trends regarding the WFS clock: faster frame-rate are set up for brightest stars as it should be; besides the noise error can be maintained to a low level by decreasing the WFS frame rate at a cost of a higher servo-lag error (see PUAKO note #05). I do not exclude that the auto-covariance method becomes biased for low WFS frequency. In this situation, WFS clock rate may be significantly larger than the atmospheric turbulence coherence time. Therefore, the temporal cross-correlation of DM actuators will differ from the auto-covariance due to the WFS noise plus a delta on the turbulence contribution regarding its temporal spectrum. Thus, the noise level should be overestimated as it would include an additional term, which does not seem to be the case when looking at Fig. 2. There are two groups of samples that give a similar error of 30 nm for a clockrate of 149 Hz and 438 Hz and a flux of 2.10^6 and 7.10^6 respectively. From the fastest to slowest clockrate, the number of photon is divided by 3.5 so the wavefront error in nm should be increased by a factor 1.9. By decreasing the sampling frequency from 438 Hz to 149 Hz, a factor 2.9 thus, we should lessen the WFS noise standard-deviation by a factor $\sqrt{(2.9)} = 1.7$, which compensate the loss in wavefront error due to a poorer SNR. Eventually, we should not have a big difference between those two configurations as we see it on Fig. 2.

I present in Fig. 3 histograms of r_0 and L_0 estimates during these nights, as well as median and 1-σ standard-deviation in Tab. 1. In average, we measure a median r_0 of 17 cm, which looks to comply with Mauna Kea statistics although the size of the sample we have is not really large enough to draw sharp conclusions. About the outer scale, I retrieve 10-ish outer scale, which looks underestimated what we think the free atmosphere outer scale should be [16]. My understandings is that the dome seeing dominates the atmospheric contribution to the residual phase and the outer scale we measure is ultimately limited by the dome size. This assumption is probably difficult to verify as we do not have external profilers sensitive to the outer scale (I've got an idea to do so, though). However, similar experiments on CANARY at WHT [9] and AOF at VLT [8] obtain similar values for L_0, which suggests two possibilities: either the method is biased or there is a real physical reason to explain why we measure small L_0. Answering this question is above the scope of this work, but with PUAKO, Keck has a pipeline to measure and analyze the data automatically. In the future, I'd like to implement the method proposed by [1] which is supposed to be more robust and accurate than the current method implemented.

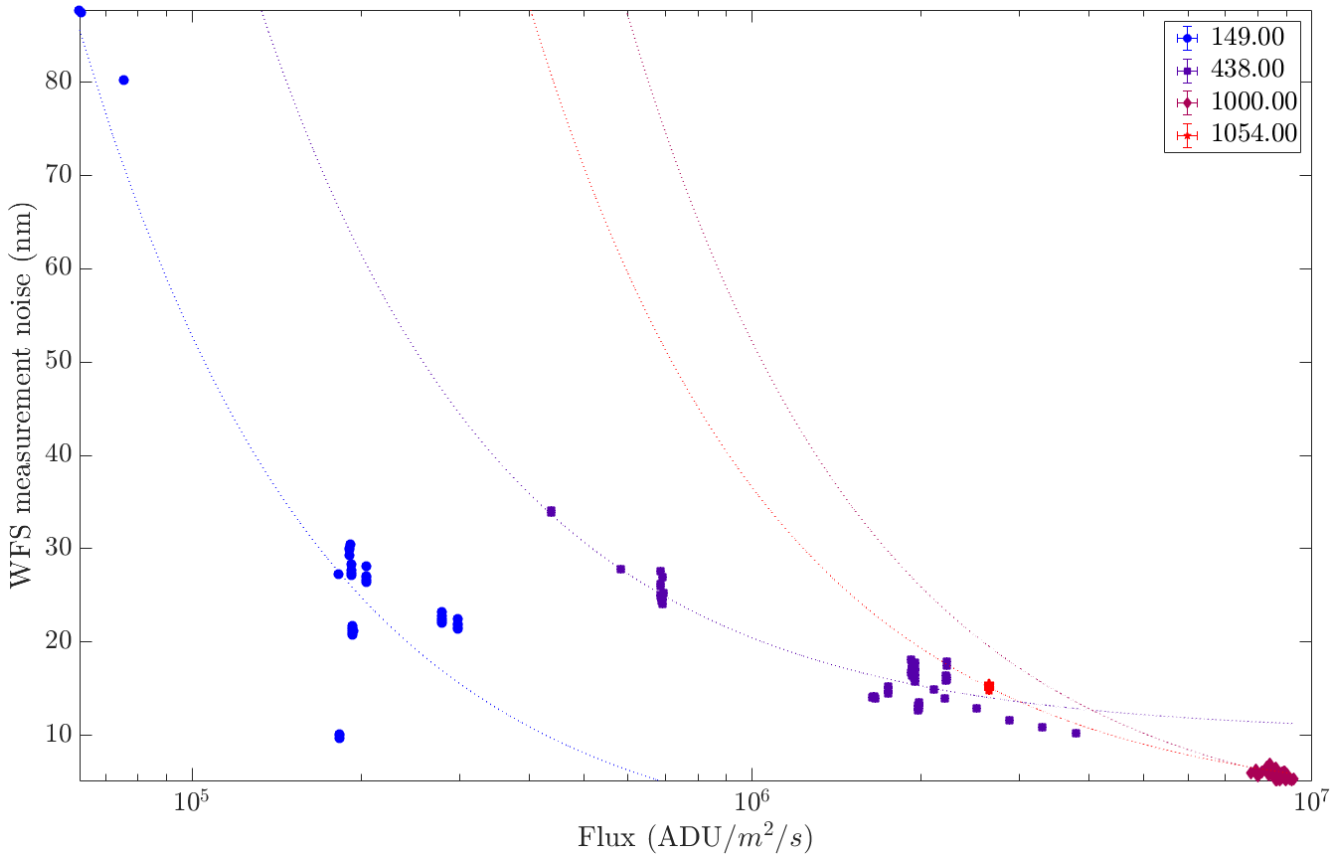

Figure 2: WFS measurements noise in nm estimated on the DM commands using the auto-covariance method as function of the flux for different WFS clocks. Dotted lines result from a fit in 1/flux, where the flux is estimated from the WFS pixels intensity maps.

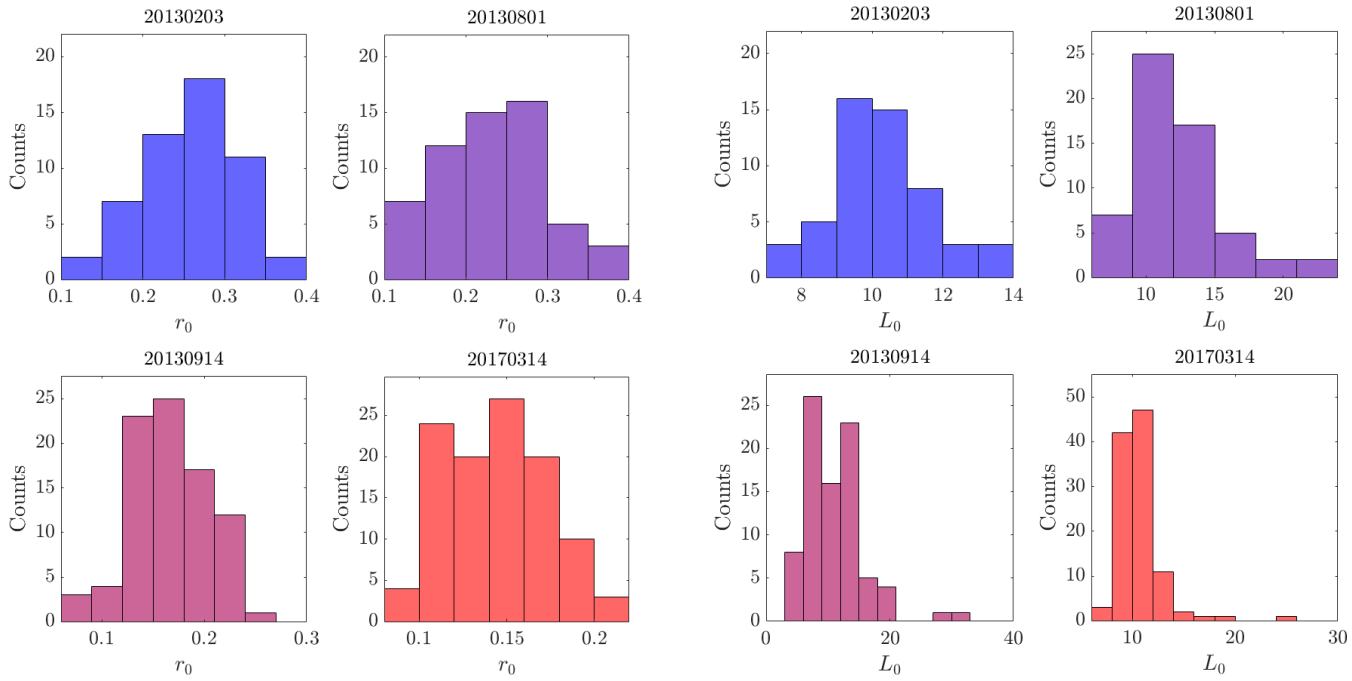

Figure 3: Histograms of r_0 (left) and L_0 (right) estimated from the AO telemetry and sorted by observing dates.

References

[1] Paulo P. Andrade, Paulo J. V. Garcia, Carlos M. Correia, Johann Kolb, and Maria Inês Carvalho. Estimation of atmospheric turbulence parameters from Shack-Hartmann wavefront sensor measurements. *M.N.R.A.S* , 483(1):1192–1201, Feb 2019.

[2] R. Conan and C. Correia. Object-oriented Matlab adaptive optics toolbox. In *Adaptive Optics Systems IV*, volume 9148 of *Proc. SPIE* , page 91486C, August 2014.

[3] C. Correia, R. Conan, and O. Beltramo-Martin et al. OOMAO – Object-Oriented Matlab Adaptive Optics. In *Sixth International Conference on Adaptive Optics for Extremely Large Telescopes*, page 70, November 2019.

[4] R Flicker. PSF reconstruction for Keck AO. Technical report, W.M. Keck Observatory, 65-1120 Mamalahoa Hwy, Waimea, HI 96743, United-States, 2008.

[5] E. Gendron and P. Léna. Astronomical adaptive optics. 1: Modal control optimization. *Astron. & Astrophys.* , 291:337–347, November 1994.

[6] L. Jolissaint, S. Ragland, J. Christou, and P. Wizinowich. Determination of the optical turbulence parameters from the adaptive optics telemetry: critical analysis and on-sky validation. *Applied Optics*, 57:7837, September 2018.

[7] L. Jolissaint, S. Ragland, and P. Wizinowich. Adaptive Optics Point Spread Function Reconstruction at W. M. Keck Observatory in Laser Natural Guide Star Modes : Final Developments. In *Adaptive Optics for Extremely Large Telescopes IV (AO4ELT4)*, page E93, October 2015.

[8] Douglas J. Laidlaw, James Osborn, Timothy J. Morris, Alastair G. Basden, Olivier Beltramo-Martin, Timothy Butterley, Eric Gendron, Andrew P. Reeves, Gérard Rousset, Matthew J. Townson, and Richard W. Wilson. Optimizing the accuracy and efficiency of optical turbulence profiling using adaptive optics telemetry for extremely large telescopes. *M.N.R.A.S* , 483(4):4341–4353, Mar 2019.

[9] O. A. Martin, C. M Correia, E Gendron, G Rousset, F Vidal, T. J. Morris, A. G. Basden, R. M. Myers, Y. H. Ono, B. Neichel, and T. Fusco. William Herschel Telescope site characterization using the MOAO pathfinder CANARY on-sky data. In *Adaptive Optics V from proc. SPIE*, 2016.

[10] P. Martinez, J. Kolb, M. Sarazin, and A. Tokovinin. On the Difference between Seeing and Image Quality: When the Turbulence Outer Scale Enters the Game. *The Messenger*, 141:5–8, September 2010.

[11] R. J. Noll. Zernike polynomials and atmospheric turbulence. *JOSA*, 66:207–211, March 1976.

[12] F. J. Rigaut, J.-P. Véran, and O. Lai. Analytical model for Shack-Hartmann-based adaptive optics systems. In D. Bonaccini & R. K. Tyson, editor, *Adaptive Optical System Technologies*, volume 3353 of *Proc. SPIE* , pages 1038–1048, September 1998.

[13] G. Rousset, J. Primot, and J. C. Fontanella. Visible wavefront sensor development. *LEST Foundation, Technical Report*, 28:17–33, 1987.

[14] S. Thomas, T. Fusco, A. Tokovinin, M. Nicolle, V. Michau, and G. Rousset. Comparison of centroid computation algorithms in a Shack-Hartmann sensor. *M.N.R.A.S* , 371:323–336, September 2006.

[15] J.-P. Veran, F. Rigaut, H. Maitre, and D. Rouan. Estimation of the adaptive optics long-exposure point-spread function using control loop data. *Journal of the Optical Society of America A*, 14:3057–3069, November 1997.

[16] A. Ziad, E. Aristidi, A. Agabi, J. Borgnino, F. Martin, and E. Fossat. First statistics of the turbulence outer scale at Dome C. *Astron. & Astrophys.* , 491:917–921, December 2008.