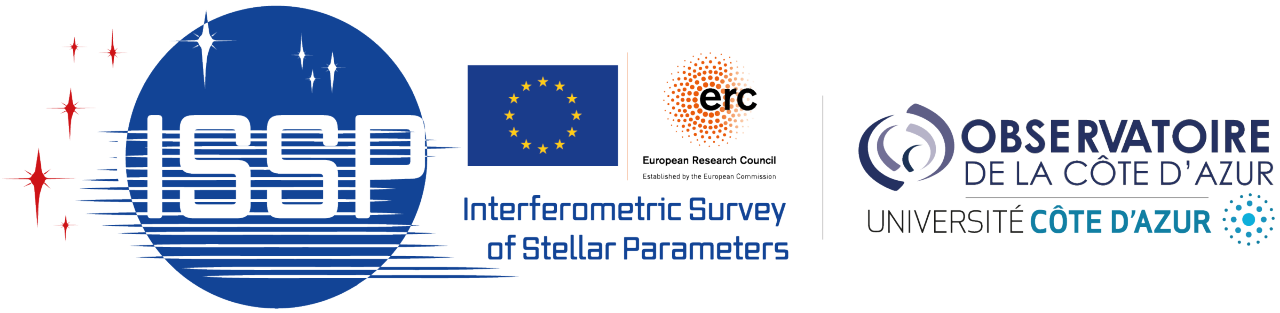
Observatoire De La Côte d'Azur



Characterization of CHARA's Adaptive Optics

Simulations and improvement of SPICA's performance in the visible domain

Pierre GENESLAY

August 21, 2023

Contents

1 Introduction

Adaptive Optics is an essential tool to explore the Universe further and improving the current limits on its performance is critical. In this document, I will summarize the different studies concerning the Adaptive Optics of CHARA.

My main activity is to evaluate the performance of the CHARA TELAO in the visible band and, if possible, to propose upgrades to reach the highest performance for SPICA. I worked in collaboration with Philippe Berio, Denis Mourard, Olivier Lai a specialist of AO in Nice, and in connection with Mattew Anderson and Theo Ten Brumelaar at Mount Wilson. My work started with the simulation of the AO system to evaluate the actual performance (Sec. 2). To do so, I used the Yorick Adaptive Optics simulator (YAO) [1] to create a model as close as possible to the CHARA system. The aim was to obtain some parameters to fit real data.

Throughout this work, I was working on Interaction Matrices. It is fundamental to understand the link between the slopes measured by the *Wave Front Sensor* (WFS) and the command sent to the *Deformable Mirror* (DM). The WFS measures the deformation of the wavefront and the DM corrects it by moving actuators located under the mirror. By analyzing the IM, we can extract the misregistration parameters, they help us to probe the current misalignment of the system from a reference alignment. The actual way of measuring the IM consists in moving the actuator of the DM one by one to change its shape and observe the effect on the sub-aperture array. From the movement of all the actuators, we can extract a map of the slopes of the WFS, this is what we call the *Interaction Matrix* (IM). Our goal was to reproduce this procedure using simulations and to compute a *Synthetic Interaction Matrix* (PSIM) to match it with the measured IM (Sec. 3). The PSIM allows to eliminate the measurement noise.

After the simulations, I tested the PSIM on-sky (Sec. 4). Real data helped to understand the effect of the new method during a night of observations. Data analysis is a direct way to compare the effects of the PSIM and the classical IM. Cubes of data were collected, and different useful variables were extracted. From the telemetry, we retrieved the slopes of the WFS and the commands of the DM. From the SPICA Science detector, we computed the Injection Flux; and from the SPICA Control detector, we derived the Ensquared Energy.

With all these tools and data, the remaining part was to analyze the performance for evaluating the quality of the *Reconstructor* or *Command Matrix*, computed as the inverse of the PSIM. Two methods of inversion were implemented: the *Singular Value Decomposition* (SVD) and the *Minimum Mean Squared Error* (MMSE). In Sec. 5, I will be focusing on the effect of those two types of reconstruction matrices on sky data. To evaluate the performance, I attempted to use the residual variance of the slopes or the Strehl Ratio. The study is continuing...

[1] https://frigaut.github.io/yao/manual.html

2 Simulation of AO with YAO

2.1 Parameters of the simulation

In Yorick Adaptive Optics Simulator, there are hundreds of parameters that help to reproduce an AO system. Different structures can be designed to match the CHARA model. The following list is an accurate description of all possible parameters that affect the simulation :

SIMULATION STRUCTURE

sim.pupildiam defines the size of the pupil in pixels.

sim.debug defines the Debug Level. The value stayed "0".

sim.verbose provides additional details as to what the computer is doing; it is set to one.

TELESCOPE STRUCTURE

tel.diam gives the telescope diameter in meters.

tel.cobs defines the central obstruction, the area where no light can enter. It is a ratio between the diameter of the central obstruction and the telescope diameter.

ATMOSPHERE STRUCTURE

r0 Fried diameter in meter. It defines the strength of the turbulence, it goes from 5cm for harsh conditions to 20cm for excellent ones. The better the r_0 the better the AO correction.

atm.dr0at05mic Normalized Telescope aperture. Its definition is tel.diam/r_0. This expression is a ratio, it has a typical value of λ=0.5µm so for r_0 [0.5µm].

atm.screen creates phase screens. A way to approximate the atmosphere is to create different turbulent volumes and different layers of turbulent phases. Added together, it creates a screen that evolves in function of wind speed, wind direction, and strength of the turbulence for example.

atm.layerfrac defines the weight of the different layers of the atmosphere, the sum of all the layer fractions must be equal to "1".

atm.layerspeed constitutes the different main speeds of the layers, combined with *layerfrac*, it is possible to calculate the mean speed of the turbulence in $m.s^{-1}$.

atm.layeralt computes the different altitudes of the layers. The layers are like thin rugs disposed only at different altitudes. We neglect the width of the layers, there is a discrete number of layers when we do the simulation. Expressed in meters.

atm.winddir defines the directions of the wind, which will affect the shape of the layers during the phase screen creation. Expressed in radians.

WAVEFRONT SENSOR STRUCTURE

EMgain is a typical electron multiplier gain for EMCCD cameras. The gain has the role of amplifying the signal. It increases the Signal-to-Noise Ratio.

wfs.type defines the type of the WFS for the simulation. "curvature"," pyramid"," Zernike", and" Hartmann". Each WFS type has its keywords. For the rest of the list, Only the Shack-Hartmann WFS was used. It is the type of wavefront that is used at CHARA. No other types were relevant to test.

wfs.Wavelength λ is the working wavelength for the WFS in microns.

wfs.gsmag is the magnitude of the star that we observe in the V spectral band.

wfs.skymag is the sky residue for the WFS. The expression is in $mag.arcsec^{-2}$

wfs.shmethod represents the type of method for the Shack-Hartmann. One means Geometric, simple gradient average over sub-apertures. Two means diffractive, full propagation.

wfs.shnxsub is the number of sub-apertures in the telescope's diameter.

wfs.npixpersub is the number of pixels for each aperture. Warning! The true size of the pupil in pixels and the pupil diameter of the simulation is not the same. We use the formula: $wfs.npixpersub = sim.pupildiam/wfs.shnxsub$.

wfs.pixsize represents the size of the sub-aperture CCD pixel in arcsec in the focal plane.

wfs.npixels This number is implicit because we need the true size of the pupil in pixels. It will reshape the image for matching our real pupil size. The simulated pupil may have a power of two pixels needed for the FFT The real pupil of the WFS has another number of pixels of aperture.

wfs.pupoffset is a shifting of the WFS pupil from the plan (0,0). Expressed in meters.

wfs.noise is the enabled noise of the system, read-out noise, and photons noise. The value "0" means no noise and "1" means that we include noise in the simulation.

wfs.ron is the read-out noise expressed in e-.

wfs.darkcurrent defines the quantity of dark current in the simulation. During a take, bad photons can enter the camera and pollute real sky photons. Dark current increases if there is light from the laboratory, heat, or vibrations. This dark current must be widely inferior to one to have a chance to detect few photons from far objects. Expressed in e-/s/pixel.

wfs.biasrmserror is an RMS error on WFS CCD bias in electrons.

wfs.shthreshold This threshold depends on the readout noise. The more the ron is important, the higher the threshold. Expressed in electrons.

wfs.fracIllum defines the fraction of the sub-aperture illuminated to be valid. Always between "0" and "1".

wfs.optthroughput considers the fraction of the flux that is lost during the propagation in the optical systems. It contains the quantum efficiency and the loss of the mirrors.

wfs.fstop defines a field stop. It is used to filter out the noise that is far from the center of each sub-aperture. It is whether "square" shape or "round" shape.

wfs.fssize is the size of the field stop in arcsec, Side for the square shape, and Diameter for the round shape.

DEFORMABLE MIRROR STRUCTURE

dm.type Type of the deformable mirror "stackarray" and "tiptilt" will be used

dm.iffile saves the influence matrix at the end of the simulations.

dm.nxact Number of actuators in the telescope diameter.

dm.pitch Gap between to actuators in pixels ($dm.pitch = sim.pupildiam/(dm.nxact - 1)$).

dm.thresholdresp Threshold below which an actuator will not be kept as valid.

dm.unitpervolt Conversion factor in µm/V for the stack array DM and in arcsec/V for the tip-tilt one.

dm.push4imat Voltage to apply for the Interaction Matrix calibration (in V).

dm.coupling Coupling coefficient applied to neighbor actuators. Between "0" and "1".

dm.elt allows to save huge amounts of RAM for large telescopes. "0" if it's an ELT-like telescope and "1" otherwise.

dm.gain Pre-gain for each DM. TipTiltGain and StackArrayGain can have different values.

dm.irexp forms of the influence function.

- **0** Old ad hoc functional form fitted on actual ZIGO data
- **1** $e^{-\frac{d}{irfact}^{1.5}}$ (we must also set irfact if we use this expression)
- **2** sinc*gaussian

MATRIX STRUCTURE

mat.method reconstructor creation method ("svd", "mmse", "mmse-sparse")

mat.condition For a singular value decomposition, the condition defines the edges of the values that we keep from the eigenvalues. the interval is between maxeigen/condition up to maxeigen*condition. Out of this range, all other eigenvalues are rejected.

mat.file saves the command matrix file and the Interaction matrix file.

TARGET STRUCTURE

target.lambda Image wavelength in microns

target.xposition "Target" X positions in the field of view in arcsec

target.yposition "Target" X positions in the field of view in arcsec

target.dispzoom Display zoom, useful for multi-targets

GUIDE STAR STRUCTURE

gs.zeropoint Photometric flux for a 0-magnitude star. Defined by photons/s.

LOOP STRUCTURE

loop.gain Gain is applied at each iteration of the loop.

loop.framedelay Loop delay dues to calculation and noise. Typically, two frames.

loop.niter Number of iterations of the loop.

loop.ittime Iteration time, in seconds, is the inverse of the frequency of the loop.

loop.startskip Number of iterations that we reject to collect statistics.

loop.leak Leak coefficient from "0" to "1". Zero means no leak. It affects the command sent to the Deformable Mirror.

loop.skipevery In phase screen, skip "skipby" steps every "skipevery" iteration

loop.skipby collects better statistical coverage.

loop.statsevery computes stats every "stats every" iterations.

loop.modalgainfile File with mode gains.

To complete these explanations, a list of the typical parameters injected in the simulator is available in the next table. It gathers all the different values corresponding to the elements introduced before :

Simulation Structure	
sim.pupildiam	128
sim.debug	0
sim.verbose	1
Telescope Structure	
tel.diam	1
tel.cobs	0.25
Atmosphere Structure	
r0	0.125
atm.dr0at05mic	8
atm.screen	"data/screen"
atm.layerfrac	[0.6,0.4]
atm.layerspeed	[10,18]
atm.layeralt	[0,5000]
atm.winddir	[0,0]
Wavefront Sensor Structure	
EMgain	100
wfs.type	"hartmann"
wfs.lambda	0.5
wfs.gsmag	0
wfs.skymag	18.5
wfs.shmethod	2
wfs.shnxsub	7
wfs.npixpersub	18.28
wfs.pixsize	0.7
wfs.npixels	12.86
wfs.pupoffset	0
wfs.noise	1
wfs.ron	0.25
wfs.darkcurrent	0.001
wfs.biasrmserror	0
wfs.shthreshold	1
wfs.fracIllum	0.5
wfs.optthroughput	0.2
wfs.fstop	"square"
wfs.fssize	3.1
EMgain	100
wfs.type	"hartmann"
wfs.lambda	0.5
wfs.gsmag	0
wfs.skymag	18.5
wfs.shmethod	2
wfs.shnxsub	7
wfs.npixpersub	18.28
wfs.pixsize	0.7
wfs.npixels	12.86
wfs.pupoffset	0

wfs.noise	1
wfs.ron	0.25
wfs.darkcurrent	0.001
wfs.biasrmserror	0
wfs.shthreshold	1
wfs.fracIllum	0.5
wfs.optthroughput	0.2
wfs.fstop	“square”
wfs.fssize	3.1
Deformable Mirror Structure	
dm(1).type	“stackarray”
dm(1).iffile	“”
dm(1).nxact	8
dm(1).pitch	18.28
dm(1).thresholdresp	0
dm(1).unitpervolt	1
dm(1).push4imat	1
dm(1).coupling	0.6
dm(1).elt	0
dm(1).gain	1
dm(1).irexp	0
dm(2).type	“tiptilt”
dm(2).iffile	“”
dm(2).unitpervolt	1
dm(2).push4imat	1e-2
dm(2).gain	1
Matrix Structure	
mat.method	”svd”
mat.condition	15
mat.file	“iMat cMat”
Target Structure	
target.lambda	0.7
target.xposition	0
target.yposition	0
target.dispzoom	1
Guide Star Structure	
gs.zeropoint	4.2e9
Loop Structure	
loop.gain	0.41
loop.framedelay	2
loop.niter	10000
loop.ittime	0.00227
loop.startskip	10
loop.leak	0.05
loop.skipevery	1000
loop.skipby	10000
loop.stats every	10
loop.modalgainfile	“simulModeGains.fits”

Table 1: Simulation for 25/04 data VEGA Star V_{mag}=0 r_0=12,5cm.

During all the simulations, among options, a Shack-Hartmann Wave Front Sensor is selected, a Stack-Array Deformable Mirror, and a Tip-Tilt Mirror. The pupil diameter was 128 pixels instead of 90 pixels, the width of the WFS detector because the calculations were easier with a power of 2, a reshape is operated at the end of the simulation. The *Singular Value Decomposition* (SVD) reconstructor was used during this approach because it was the current reconstructor applied on TELAO. Another type of reconstructor will be studied later.

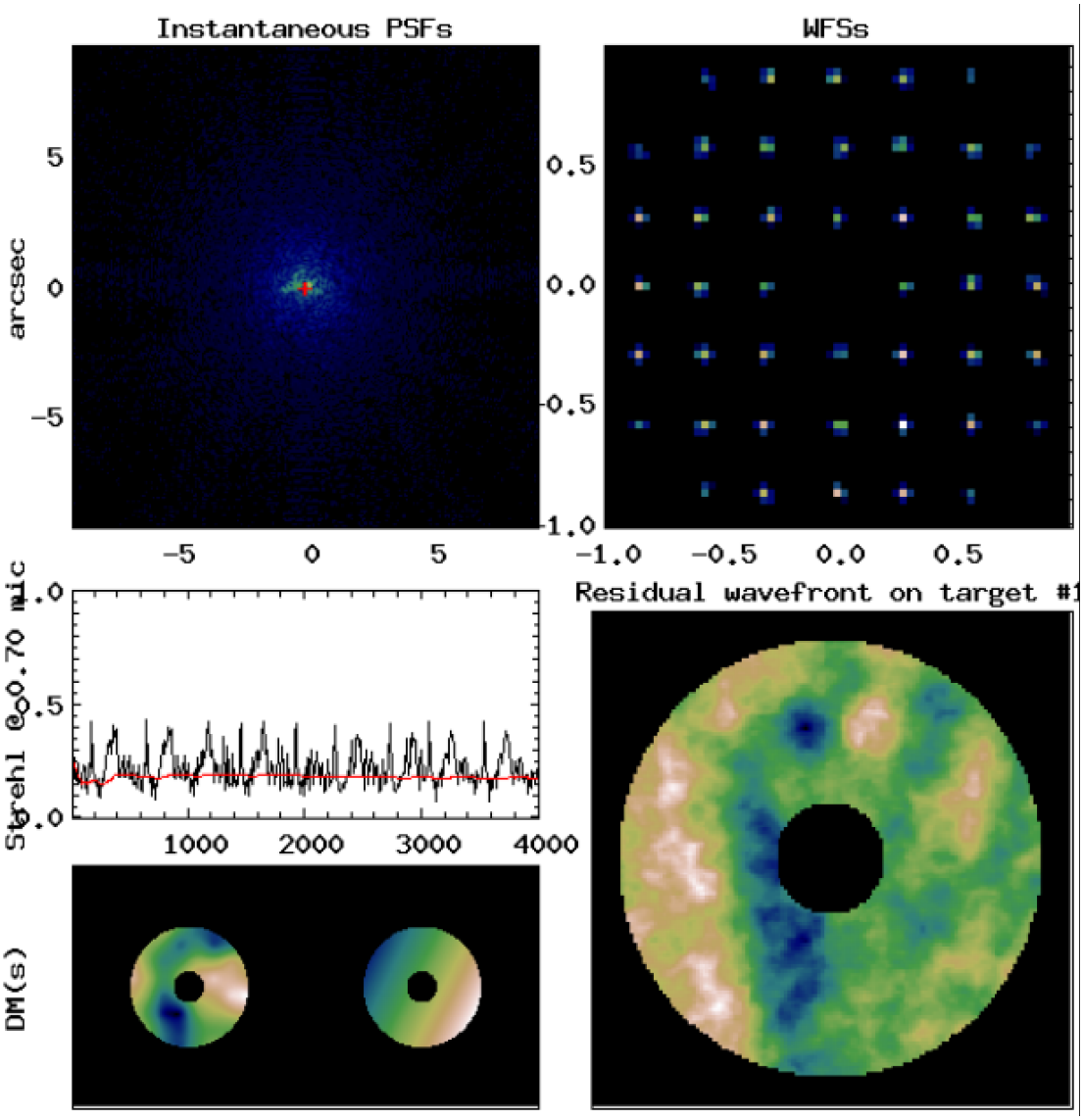


Figure 1: Yorick Adaptive Optics Interface showing the different steps of the AO loop.

Figure 1 above shows the different components of an AO system. At the top left, the instantaneous *Point Spread Function* (PSF) describes the spatial spread of the light intensity from a source point. At the top right, there is the Shack-Hartman WFS with the 36 images in the sub-apertures. The bottom left represents the two DMs; the 60 actuators DM is the left one and the tip-tilt mirror is at the right. After the DM, there is the residual wavefront, which is what shows the image at the bottom right of the screen. To finish, there are the short exposure (black) and the long exposure (red) Strehl Ratio that inform the performance of the AO.

2.2 Matching Yao with TELAO for different inputs (r_0, wind, gain)

Playing with the Wind Speed, the Magnitude of the star, and the Fried diameter, the simulation is adjusted to reproduce the same performance as the telemetry. At the end of the simulation, some important parameters are saved as the commands sent to the DM, the slopes on the WFS, and also the Tip-Tilt command. From the TELAO telemetry, the same multiple sets of data are extracted and compared. If the YAO parameters produced the same results as the real data, it meant that the simulations matched the behavior of the real system. Figure 2 is an illustration of the overlapping of the two histograms, in red the slopes of the WFS extracted from the telemetry of TELAO and in blue the simulation in YAO. As a comment, it appears that the variance was the same and the number of counts was also identical.

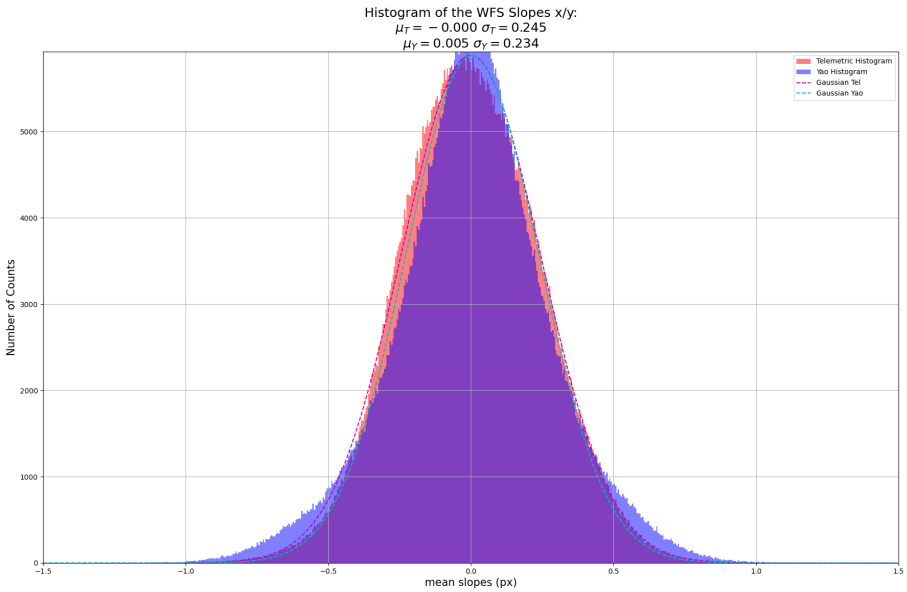


Figure 2: Histogram of the slopes of the WFS from YAO and TELAO.

Similar histograms are produced by the code *compare_telao_yao.py* referred to in part 6. This program helps to match the YAO simulator with real data from TELAO but YAO extracts as well the Strehl Ratio from its simulations, We can deduce if the seeing conditions were appropriate or poor for observations.

2.3 Comparison of simulations for different types of reconstructor

Comparing TELAO and YAO was not sufficient to improve the performance of the system. After further exploration of the YAO parameters, the choice of another reconstructor was considered as the best way to progress. The method for the matrix reconstructor *mat.method*(cf. Table 1 and explanations above) was changed from the "Singular Value Decomposition" method to the "Minimum Mean Square Error" method. In the MMSE method, there was a regularization parameter "a" that influences the performance of the system. The advantage of each method will be discussed later but both of them already appear on the figure 3.

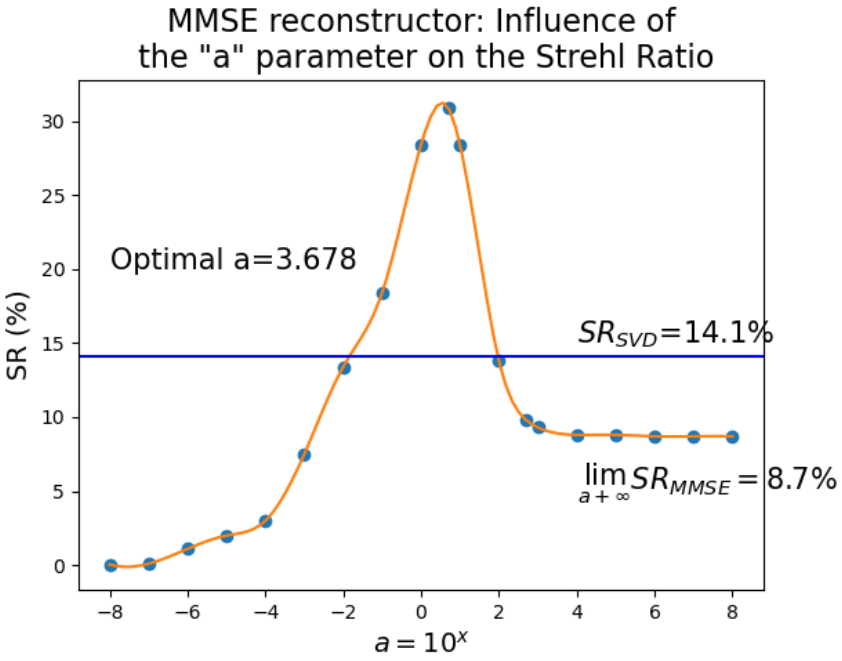


Figure 3: Strehl Ratio according to the Reconstructor.

Figure 3 contains two main pieces of information: first, there is a particular value, a peak which maximizes the SR but also, there is an entire region of "x", for instance between -2 and 2 which corresponds to "a" values between 10^{-2} and 10^{2} where the MMSE method gives better results than the SVD method keeping all the other parameters unchanged. CHARA currently uses the SVD reconstructor.

3 PSIM: New method to create Interaction Matrices

3.1 What is a PSIM ?

The Pseudo Interaction Matrix or PSIM is a new way to think about the link between DM and WFS. Moving the actuator one by one is becoming extremely long for thousands of actuators. To create a PSIM, it is mandatory to know the location of each actuator of the DM and the location of each sub-aperture of the WFS. The program *set_up_psim.py* (cf. part 6) is made to create the actuator location (top left image on the figure 4). After the creation of a phase screen for each actuator, another step was to represent the PSF (cf Sec. 2) which contains the location of the sub-aperture and their size. In the end, a PSIM is created with 60 actuator commands and 72 sub-apertures slopes, which is the current configuration.

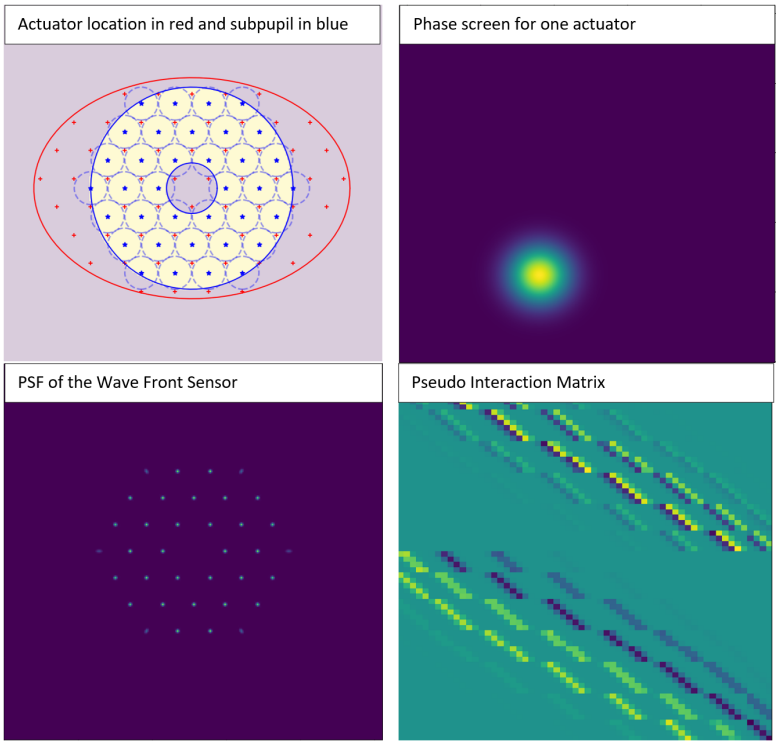


Figure 4: Different creation steps for a PSIM.

3.2 How to create a PSIM ?

The light coming from the star encounters the DM and continues towards a dichroic and three other mirrors before entering into the WFS. A misregistration phenomenon is likely to occur, affecting the IM. There are four main misregistration parameters, figure 5 shows their consequence on the actuator location :

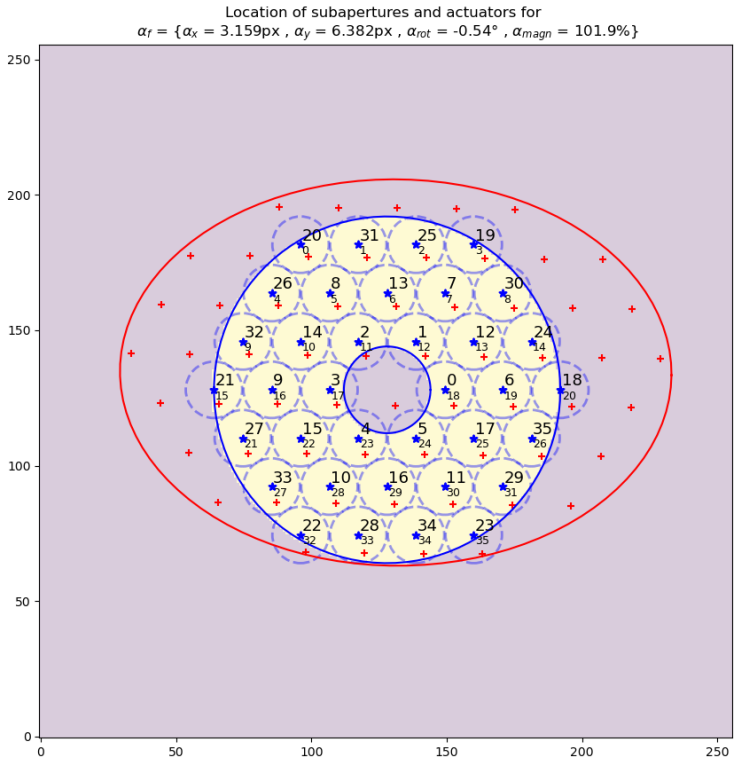


Figure 5: Representation of the actuators of the DM in the WFS plane: visualization of the four parameters (α_x,α_y,α_{rot},α_{magn}) due to the misregistration phenomenon.

The actuator array can be shifted along the x and y direction, it can also be rotated or magnified (or demagnified). Each parameter helps us to understand the alignment between the DM actuator array (in red) and the WFS sub-aperture array (in blue). The goal is then to correct it by moving the WFS which has two degrees of freedom. In the example above, there is an important misregistration for the shift, and an insignificant rotation and magnification

Until now, the search for the different misregistration parameters was done by hand: scan for different values and detect the best one. But an automatic procedure of minimization was set up to accelerate the research. Another program related to the previous one was created: *from_carbon_to_diamond.py*. It uses a sequential least squared algorithm by evaluation of gradient to perform the minimization. Five parameters are computed: the four previously evoked and a scaling factor to adapt the amplitude of the original IM. The default of alignment is kept to implement the PSIM.

The algorithm minimizes the difference between the IM and the PSIM. Below, the two figures show a few slopes in the X direction for the sixty actuators for the PSIM and the IM :

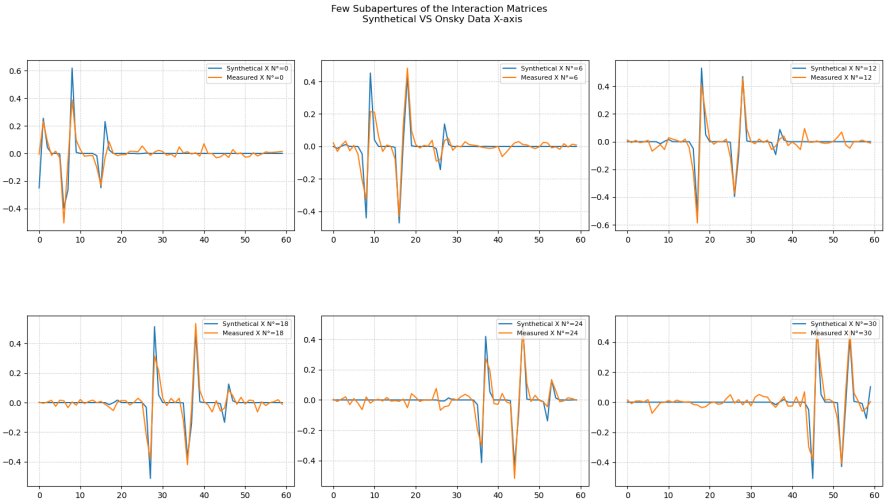


Figure 6: Comparison between slopes in X direction.

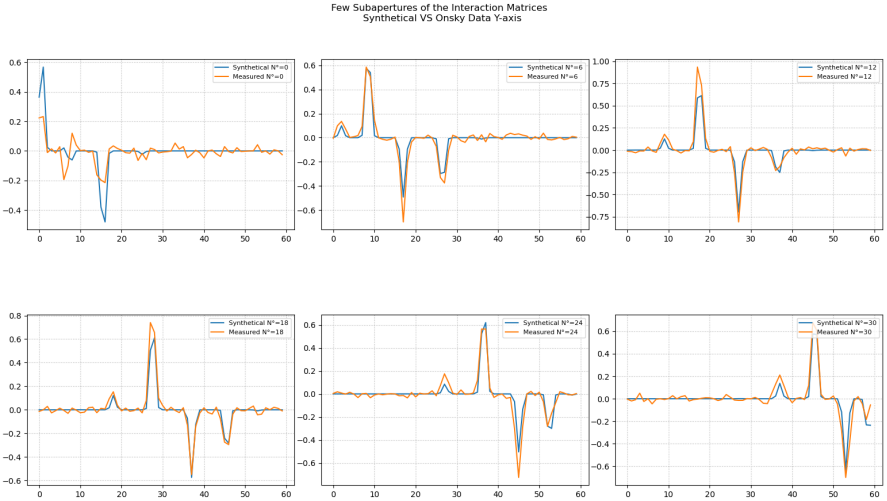


Figure 7: Comparison between slopes in Y direction.

One powerful thing about the PSIM is that we can faithfully reproduce the shape of the IM but without the noise which destabilizes the system. There are still differences even after the minimization, that is the reason why the residue between PSIM and IM is used to evaluate the improvement:

$$\text{residue} = \frac{\sum_{i=1}^{60}\sum_{j=1}^{72}|M_{ij}^{PSIM} - M_{ij}^{TELA}|}{\max\left\{M_{ij}^{TELA}\right\}}$$

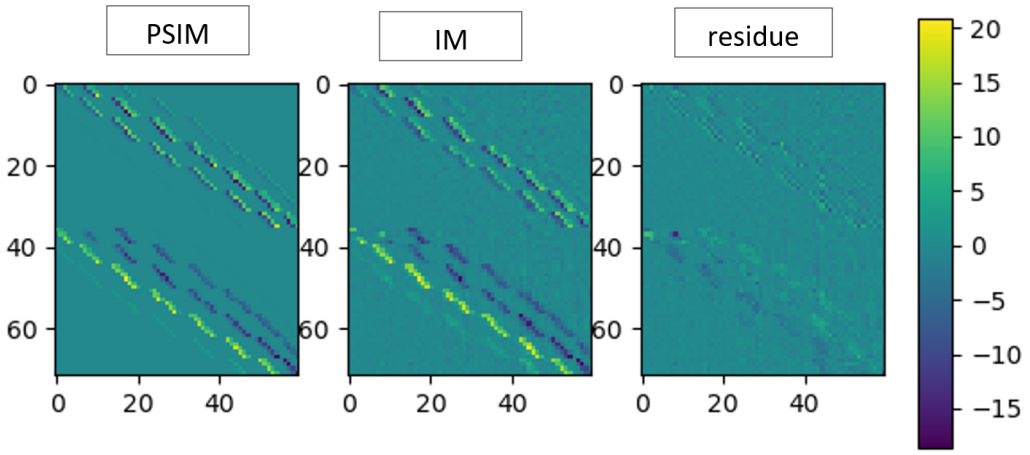


Figure 8: PSIM after optimization compared to the original IM (the residual map on the right represents the error after calculation, the values must be close to zero).

3.3 Comparison between SVD and MMSE reconstructor

After obtaining the IM, the last step is the inversion of this matrix to generate the command matrix. The main issue is the shape of the Matrix, which is not square, and requires either computing the Pseudo-Inverse or using a method such as SVD, which is currently employed and filters several modes:

$$H = UWV^t$$

$$[U]_{size} = N \times M \qquad [W]_{size} = M \times M \qquad [V^t]_{size} = M \times N.$$

$$R_{SVD} = H^{\dagger} = VW^{-1}U^t$$

H is the interaction matrix and R_{SVD} is the command matrix. Also called the "reconstructor".

There is another reconstructor, more complex that can be installed in the system. The Minimum Mean Square Error. It is already available on the YAO simulator:

$$R_{MMSE} = (H^T H + aC)^{-1} H^T$$

C is a regularization matrix and "a" is a regularization parameter that can be adapted. It was already known that the "a" parameter influences the system. Thanks to the program *from_carbon_to_diamond.py*, two types of inversion methods can be compared.Figure 9 shows the residual difference between the original command matrix and the pseudo command matrices. One is the inversion with the SVD method and the other is obtained by changing the "a" parameter during the MMSE reconstruction matrix.

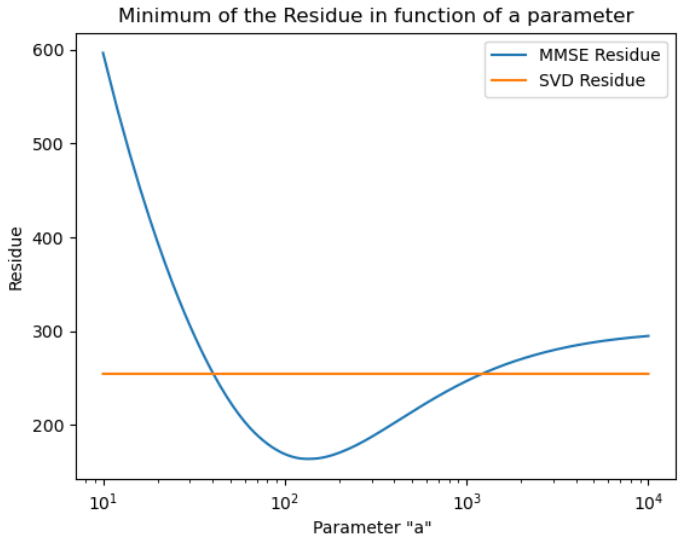


Figure 9: Comparison of the residue for the two types of reconstructor.

As previously mentioned, the residue is identical for the SVD reconstructor and in an entire region (cf. figure 3), the residue for the MMSE is always below the SVD. The lowest residue does not imply that it is the best value of "a" but it improves the accordance with the original CHARA reconstructor. The goal is to mimic the original command matrix as much as possible. Other types of Adaptive Optics systems already used the MMSE reconstructor (GeMs [2]) because it is proven that it improves the performance particularly when the system is affected by noise.

For the different tests on the sky, three values of MMSE "a" parameters were selected: 100, 150, and 200. They correspond to the values in the hollow region of the well (cf. figure 9). Another program comparing the reconstruction matrices with the initial one on CHARA called *psim_vs_im.py* is used. It helped to visualize the different types of configurations:

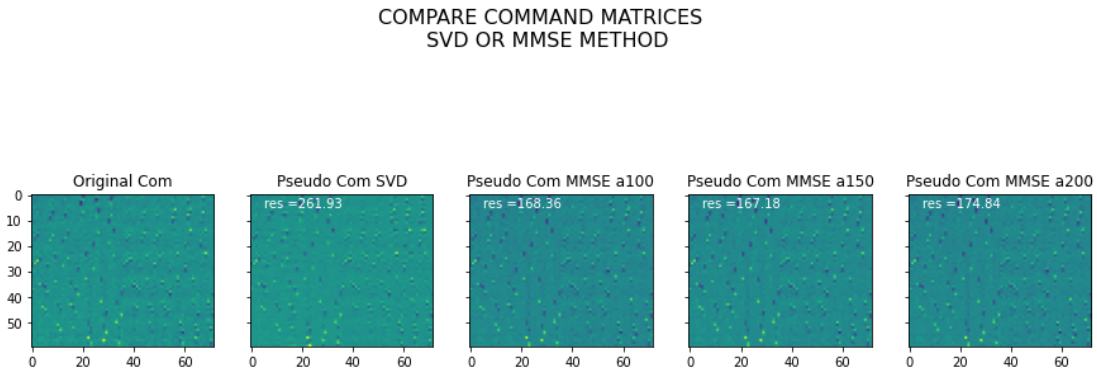


Figure 10: Comparison of the SVD and MMSE reconstructors.

[2]https://ao4elt.edpsciences.org/articles/ao4elt/pdf/2010/01/ao4elt_02010.pdf

4 Tests On-Sky

4.1 Commands and Measures from Telemetries

From the telemetries on CHARA, thanks to the program *compare_telao_yao.py* already discussed, it was easy to obtain the variance of the slopes and the variance of the commands for a certain configuration. The opened loop gave commands close to 0 because the deformable mirror was not activated. The next figure comes to compare the different variances of a telemetry file in a closed loop for the default reconstructor, the SVD, or the MMSE reconstructor for the different values of "a" :

	Default	**SVD**	**MMSE a100**	**MMSE a150**	**MMSE a200**
Slopes (px)	0.3429	0.3385	0.2335	0.3052	0.3351
Commands (µm)	0.3004	0.3374	0.5014	0.4072	0.4463

Table 2: Commands and Slopes variances for different reconstructors on Telescope E1 for HD-120315 star & V_{mag}=1.9 during the night of 04/30/2023.

	Default	**SVD**	**MMSE a100**	**MMSE a150**	**MMSE a200**
Slopes (px)	0.3787	0.3673	0.3353	0.3476	0.3346
Commands (µm)	0.2921	0.3294	0.4592	0.4191	0.3735

Table 3: Commands and Slopes variances for different reconstructors on Telescope E1 for HD-126660 star & V_{mag}=4.1 during the night of 04/30/2023.

After analysis, the reconstructor MMSE seems to be the best to minimize the slopes of the WFS but the single experiment is not enough because it is depending on the observing conditions. The commands on the DM are larger but the most important is to have the less wavefront deformation possible at the end of the loop, and that is true when the variance of the slopes is close to 0.

4.2 Transfer functions from telemetries

The transfer function is a function that represents the internal behavior of a system. It corresponds to the ratio between the close loop signal (active commands on the deformable mirror) and the open loop (no commands). Obtained by the telemetries as before, it is a way to calculate the rejection of the loop, at which frequency the systems can no longer attenuate the signal. The program *compare_tf.py* permits obtaining the new figure that illustrates the transfer function of a CHARA Telemetry with the Amplitude of the attenuation according to the frequency :

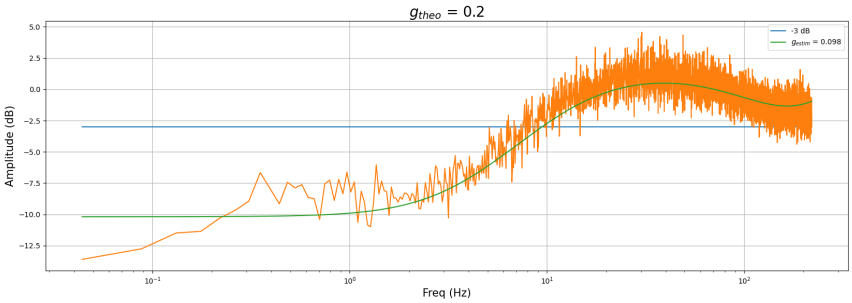


Figure 11: Transfer Function from CHARA's commands of telemetry.

The blue line is the half-power point at -3 dB, it is the point where the output signal dropped half of its peak value, on the figure 11 it is approximately 10 Hz. To understand the behavior of the loop, *transfer_function_telemetry.py* was used. First, the transfer function was plotted for a certain set of data, then the overlap of the real data transfer function with a model was operated to reproduce the shape of the curve. To match the curve, three parameters were mandatory: the loop gain that plays on the amplitude of the rejection, the delay that shifts the curve toward the high or low frequencies, and finally a leak parameter that corresponds to the level of rejection at the low frequencies. The last parameter was a cut-off frequency, it is the frequency where the signal is amplified above and attenuated below :

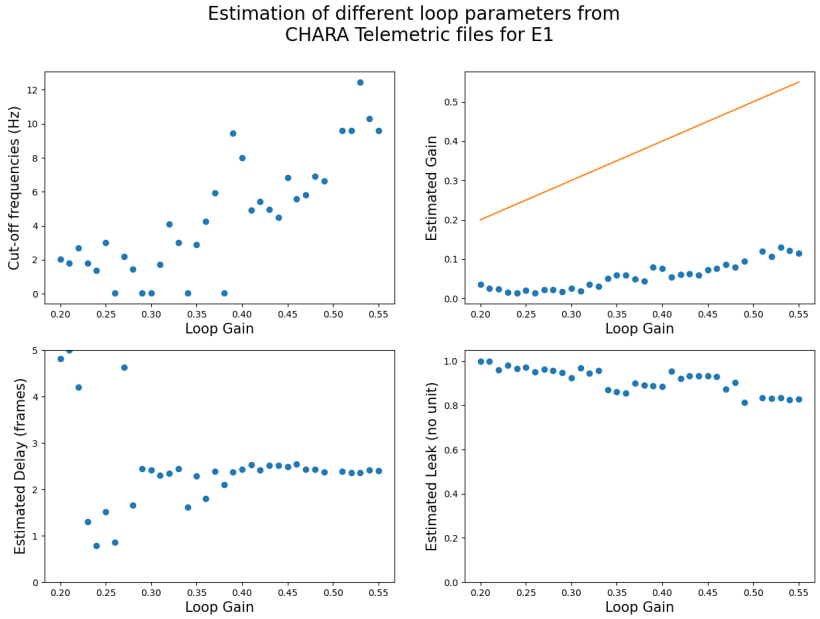


Figure 12: Transfer Function Loop parameters

The figure above shows that the cut-off frequency increases when the loop gain increases, the delay is 2.5 frames for high gains, the leak is around 0.9 and the relationship between the input gain and the estimated gain is not linear. Two explanations are whether the code does not estimate well the parameters or it exists an error during the loop processing.

4.3 Ensquared Energy from the control detector

On SPICA, there is a Control Detector, at the end of a control path mandatory for the alignment. It helps to locate the position of the stars at the entrance of the fibers and with movable mirrors, there is the possibility to re-center and optimize the flux that will be injected later. During the night, we can record cubes of data to do post-processing of the detector images. A way to obtain the performance of the instrument is to calculate the Ensquared Energy of the detector. The first step is the isolate 6 thumbnails each centered on their reference pixel, with a width and height of 30 pixels (same as seen on the screen), then sum up the flux belonging in a square of the size of an FWHM of a fiber, in our case 5 pixels. The last step is to process it for all the frames and plot the histogram. Thanks to the programs *ensquared_energy_one_file.py*, *ensquared_energy_sts.py* and *ensquared_energy_sky.py*, we will obtain a figure as below :

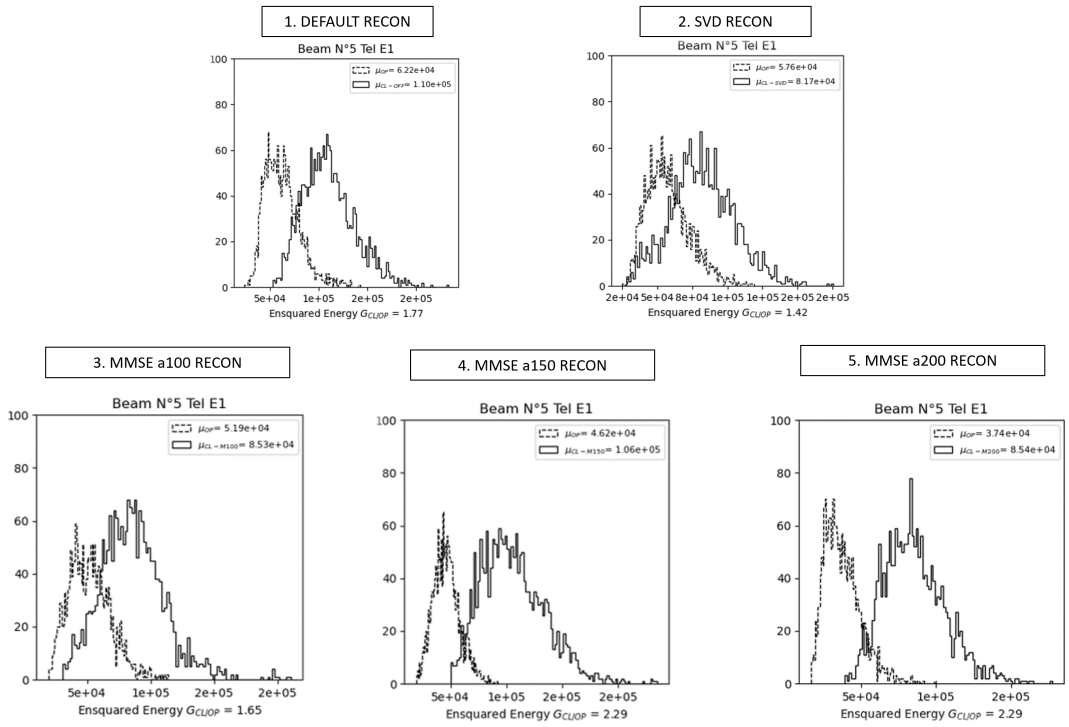


Figure 13: Comparison of ensquared energy between close loop and open loop of the AO for different IM from the Control Detector for HD-120315 star & V_{mag}=1.9.

The histograms give us the *Ensquared Energy* (EE) in open loop / close loop for the different configurations. The absolute value of the mean EE is a good indicator, but the $G_{CL/OP}$ below the graphs (cf. Figure 13) is more relevant because it takes into account the evolution of the seeing during a night. The best configuration is still the MMSE which is equivalent to a=150 or a=200.

4.4 Injected Flux from the science detector

After the control part, there is another detector, the science detector divided into two parts: the interferometric part and the photometric part. Only the photometric part is interesting because it is enough to evaluate the quality of the injection in the fibers. The 6 photometric channels are dispersed and reimaged separately on the detector. During the kappa matrix process (computation of the ratio for each beam between the photometric and the interferometric channel), we also estimate the position of the different photometric peaks. To evaluate the injected flux of the beam, first, an area is defined with the width of the spectral channel and the height of each photometric window, then integrated for all the frames and plotted the histogram. The programs *injected_flux_sts.py*, and *injected_flux_sky.py* have the same design and help to produce the following results :

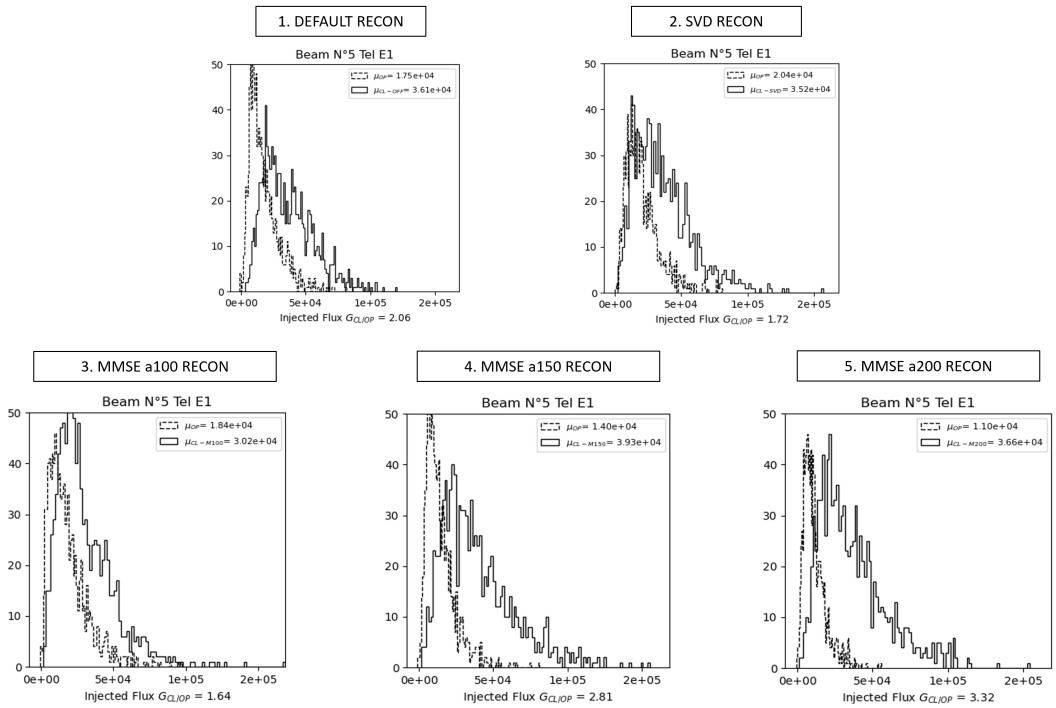


Figure 14: Comparison of injected flux between the close loop and open loop of the AO for different IM from the Science Detector for HD-120315 star & V_{mag}=1.9.

As seen previously, the Injected Flux program has the same structure as the Ensquared Energy program, the aim is to compare the performance of each configuration. The figure shows that the best reconstructor is the MMSE with a=200. Another experiment was led with a different magnitude of star (V_{mag}=4.1) and the results were similar.

For this part, the conclusion was that whatever the magnitude, the quality of the sky, or the type of measurement: commands, slopes, ensquared energy, or injected flux, the MMSE reconstructor seems to improve the performance of the instrument.

5 Performance

5.1 Residual variance by reconstructor

The current system of CHARA has already been discussed in Sec. 2. As a way of improvement, a simulation of a new type of reconstructor matrix is suggested but it remained to observe the results after the implementation on-sky. A method to quantify the performance is to look at the residual slopes projected in mirror space. The next formula shows how to process with R the reconstructor matrix with the size 60×72 and Z the slopes in x and y with the size 10000×72 (10000 corresponds to the number of recorded frames):

$$\Sigma = RZ^T \times (RZ^T)^T = RZ^T \times ZR^T$$

Then, the variance is obtained by the following formula (the variance can be calculated either on close loop or in open loop): $\sigma = \text{mean}(\text{diag}(\Sigma))$. The figure 15 shows the attenuation of the variance described by the value G such as: $G = \frac{\sigma_{CL}}{\sigma_{OL}}$

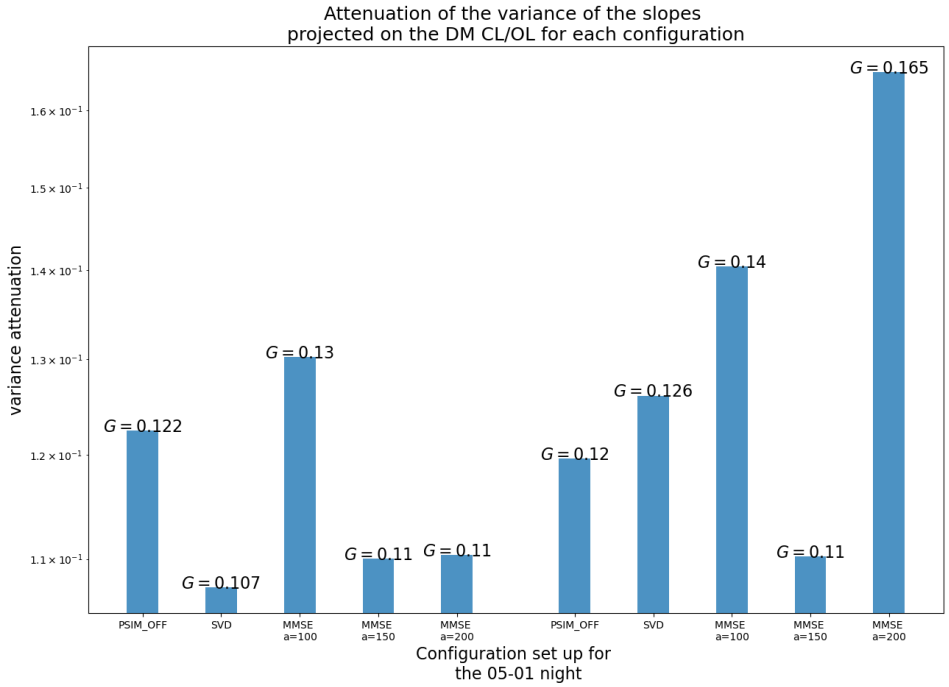


Figure 15: Attenuation Gain for all the configurations. There are two sequences; the left part concerns the magnitude 1.9 star, HD-120315, and the right part is a fainter star with a magnitude 4.1, HD-126660.

For the first star (left), the best configuration is the SVD reconstructor but for a fainter star (right), the MMSE with a=150 gives the best results. The hypothesis could be that the interest of the MMSE method is greater for a fainter star.

5.2 Influence of parameters on "a"

An experiment that could not be led was the influence of the Wind, the r_0, and the magnitude of the "a" value. Simulations were made on YAO[3] about the performance expected by knowing the sky parameters. Tables of data were created representing the ideal values of "a" that need to be included in the formula part 3.3 to optimize the MMSE matrix. One parameter was fixed and the two others varied. The two graphs below show the obtained results :

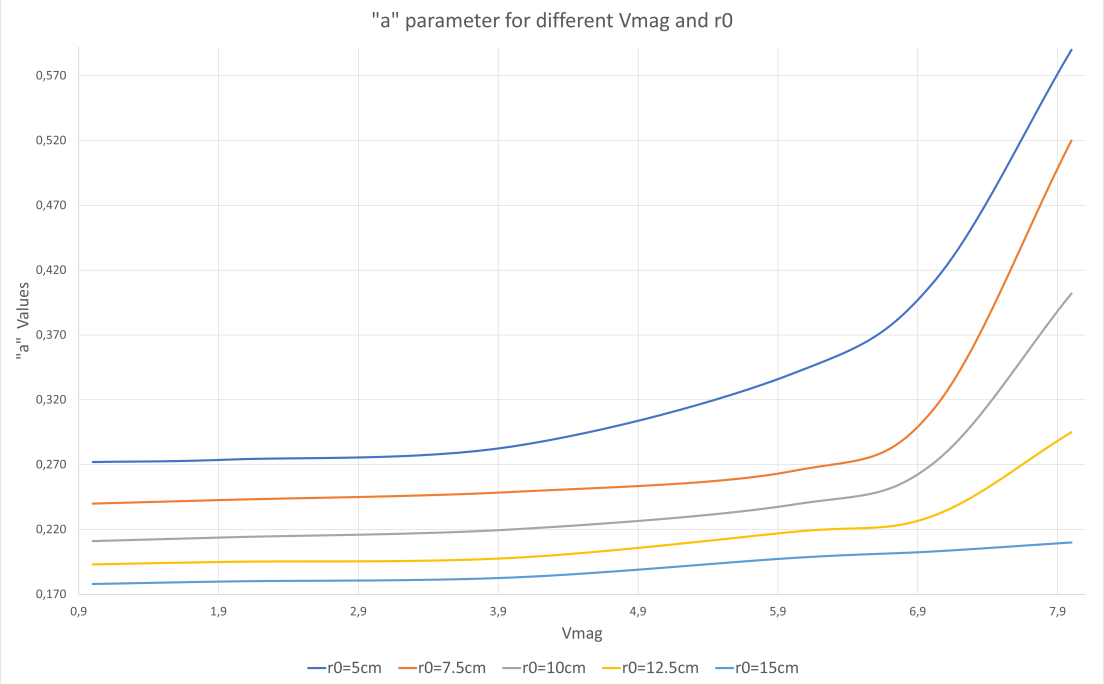


Figure 16: Optimization of "a" parameter according to r_0 and V_{mag} for $W = 6.8m.s^{-1}$.

Two observations can be made. First of all, below $V_{mag} = 4$, for a certain r_0 and $W = 6.8m.s^{-1}$, the "a" parameter is stable, it starts to increase and diverge for high magnitudes. Also, the figure shows a sort of "floors" for the different values of r_0 with the other parameters fixed, and the smaller the r_0, the larger the "a" value. It is understandable because if the seeing is bad, The "a" has to be stronger to compensate for the poor conditions.

The second graph has the same reasoning but for a mean $r_0 = 10cm$. Under a speed of wind of $10m.s^{-1}$, for $V_{mag} < 6$, the optimized "a" is pretty similar and increases. However, for the two other values of wind, the turbulence is higher and the system strongly needs to correct the aberrations, which means increasing the "a" value to produce a more powerful MMSE reconstructor.

[3]There is a scaling factor between the "a" values with the MMSE reconstructor above and the values obtained by YAO. It was difficult to evaluate it because of the input conditions. The interest here was to show the trend but not the absolute values.

The system struggles a lot to compensate for high wind speeds, so it is better to work below $W = 15m.s^{-1}$ which is in any case, a limit for the opening of the domes at CHARA :

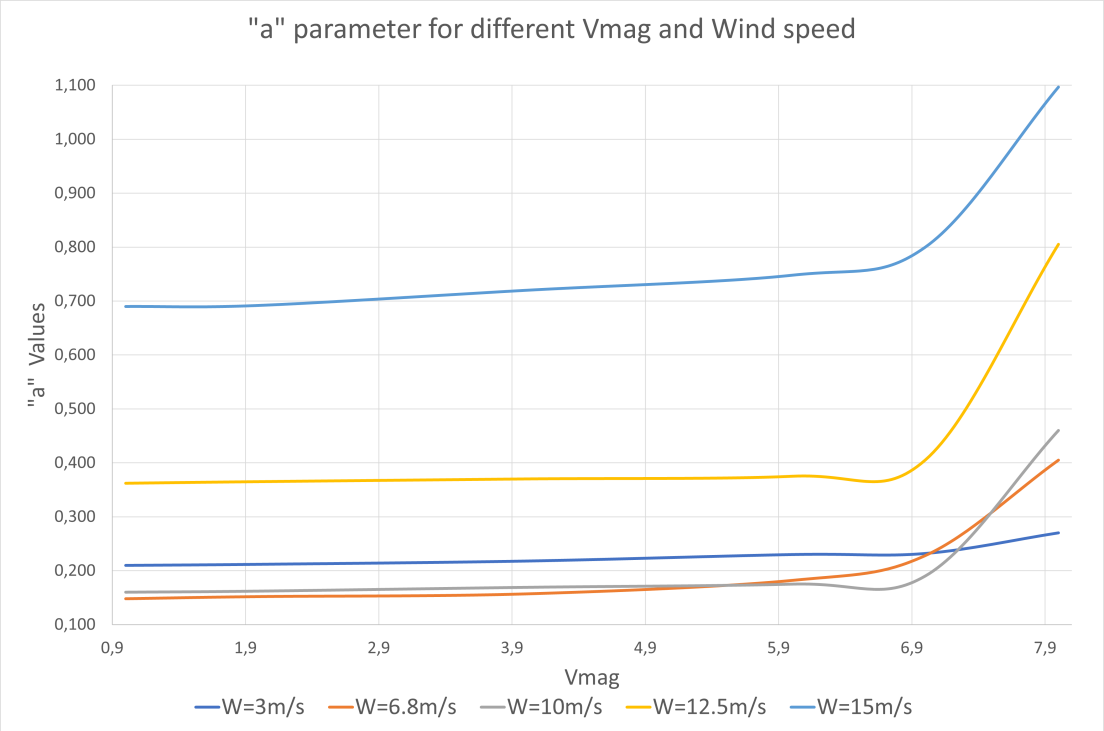


Figure 17: Optimization of "a" parameter according to W and V_{mag} for $r_0 = 10cm$.

5.3 Fried diameter : r_0

There is a method to obtain the Fried diameter r_0 from the telemetry and the influence functions. Let's start with the influence functions. An influence function is an important step to create the interaction matrices but not only. It is also a mathematical function that gives the deformation in µm of the mirror's surface for 1V impulsion of one actuator. It is done for each actuator and it creates a map that is called Influence Function.

A program called *influence_functions.py* (cf. part 6) was created. It allows us to plot the influence function of all telescopes DM for all their actuators. ALPAO supplied the CHARA's influence functions. The code was in Matlab format. Therefore, the new one reads, transforms, and eventually plots all of them. The figure below shows what a deformation of 1V of the 28^{th} actuator looks like on the DM :

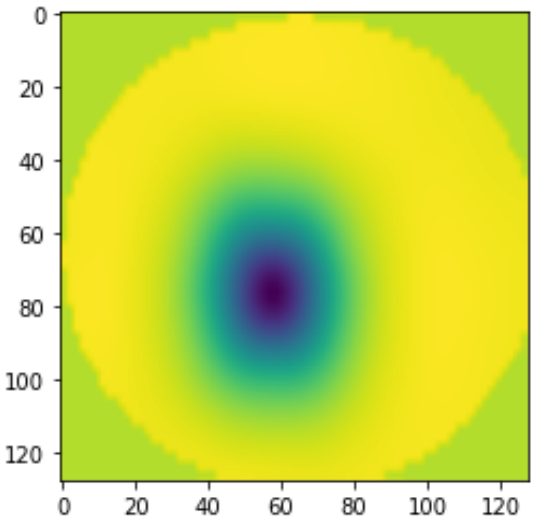


Figure 18: Influence Function for the 28^{th} actuator of the DM.

There is an algorithm computing the r_0 called *estimated_r0_iff_functions.py*. The different steps below summarize the procedure :

1) Extraction of a telemetry command file, the influence functions in 2D, and the Zernike normalized modes in 2D :

$$u_{DM}[60, 1000] \qquad IF[60, 128, 128] \qquad ZM[30, 128, 128]$$

2) Calculate the scalar product of the Zernike modes with the influence functions the influence functions are not orthogonal in phase, so this is just the projection of the influence functions on the Zernike modes) and the influence functions with themselves (this is the geometric covariance matrix Δ_{iff}) :

$$\Delta_{zern}[60, 30] = \sum_{i=1}^{60} \sum_{j=1}^{30} IF_{i,k,k} \times ZM_{j,k,k} \quad \forall k \in \{0, 1, ..., 128\}$$

$$\Delta_{iff}[60, 60] = \sum_{i=1}^{60} \sum_{j=1}^{60} IF_{i,k,k} \times IF_{j,k,k} \quad \forall k \in \{0, 1, ..., 128\}$$

3) Multiplying the projection of the influence functions on the Zernike modes by the inverse of the covariance matrix allows us to obtain the conversion matrix from Zernike modes to commands:

$$Z2C[60, 30] = \Delta_{iff}^{-1} \times \Delta_{zern}$$

4) We invert this to obtain the commands to the Zernike conversion matrix:

$$C2Z[30, 60] = Z2C^{-1}$$

5) From which we compute the Zernike coefficients, from the mirror commands:

$$Zern[30, 10000] = C2Z \times u_{DM}$$

6) We then compute the modal covariance matrix from the Zernike coefficients:

$$\Sigma[30, 30] = Zern \times Zern^T$$

7) And deduce the variance of all the Zernike coefficients:

$$\sigma^2[30, 1] = \text{diag}(\Sigma)$$

In the end, the variance of the coefficient of each mode is compared with the Noll residuals. There is the variance of each Zernike mode for $D/r_0 = 1$. The formula used to deduce the r_0 from a linear adjustment is the following :

$$a_i^2 \times \left(\frac{D}{r_0}\right)^{5/3} = \sigma_i^2 \quad \text{Where } a_i^2 \text{ are the Noll coefficients.}$$

By finding the k (and knowing D), r_0 is easily deduced. This is shown in figure 19 :

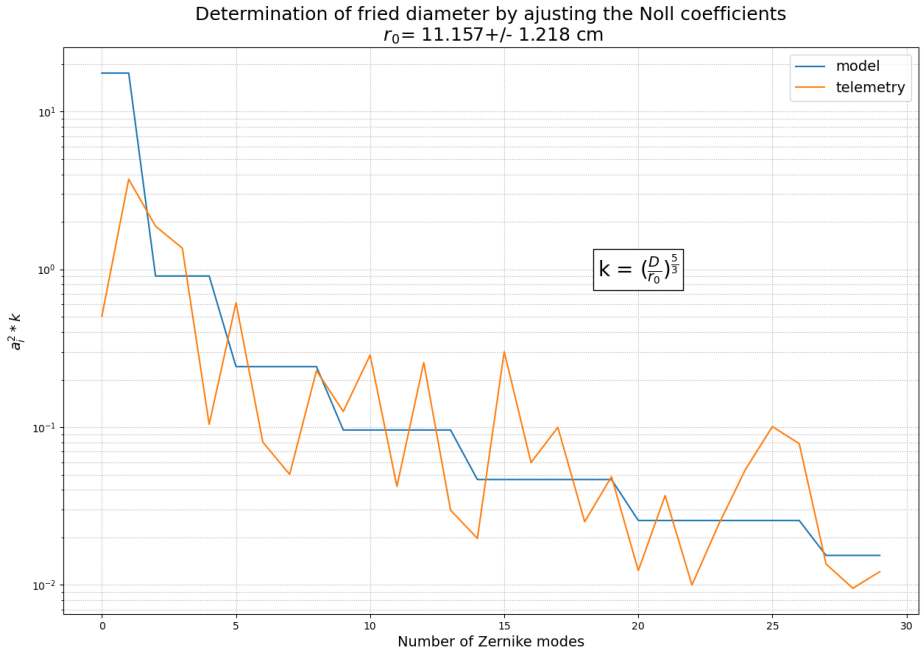


Figure 19: Fried Diameter obtained from the commands sent to the DM.

The procedure to find the r_0 is constantly evolving because it needs some improvement. Just below, there are the results obtained by running the code for different nights of observation:

	E1	**E2**	**W1**	**S1**	**S2**
06/02/2023	28.5±3.2	15.0±2.6	13.2±2.5	24.6±3.2	05.5±0.9
06/03/2023	33.9±3.8	13.4±2.1	13.5±2.3	25.6±2.6	12.9±1.9
06/04/2023	39.1±4.4	14.3±2.3	16.3±3.0	28.8±3.2	15.4±2.2
06/05/2023	31.5±3.1	13.0±1.9	05.3±0.8	21.4±2.1	09.2±1.2
06/21/2023	30.2±3.7	14.1±2.3	11.3±1.8	24.8±3.0	15.2±2.7
06/22/2023	31.4±3.4	14.1±2.2	12.7±2.1	23.6±2.7	15.0±2.6
06/23/2023	28.9±3.3	13.2±2.1	06.2±0.9	26.2±3.2	14.8±2.6
06/24/2023	29.8±3.2	13.7±2.0	13.1±2.1	25.8±3.1	10.0±1.5
07/14/2023	32.2±3.4	14.3±2.3	11.8±1.8	21.8±2.2	15.0±1.8

Table 4: r_0 in (cm) for all telescopes for different sets of telemetry data.

From the table, it is possible to distinguish two ideas. The first is the coherence of the values for each column. The typical r_0 belongs to a very restricted range in the visible band going from 5 to 20 cm on average. What is strange is the second idea. Two telescopes do not respect the other ones; E1 and S1. They seem to have larger values, $\sim$ 30 cm for E1 and $\sim$ 25 cm for S1.

A proposal would be that the influence functions used in the first part of the algorithm were obsolete. They were made at the beginning of the experiments but then, those two mirrors (E1 & S1) underwent some changes that modified their properties. Because they were not computed again, the commands sent to the deformable mirror were diverse and the consequence was that the r_0 was "fake" (as a consequence, it introduced a bias in r_0 on these telescopes). To keep r_0 coherent, an artificial multiplication is applied on the commands of E1 by a factor of 1.8 and S1 by a factor of 1.3. The best way to solve this issue would be to re-calculate all the influence functions for all the DM.

6 Implementation for AO operation

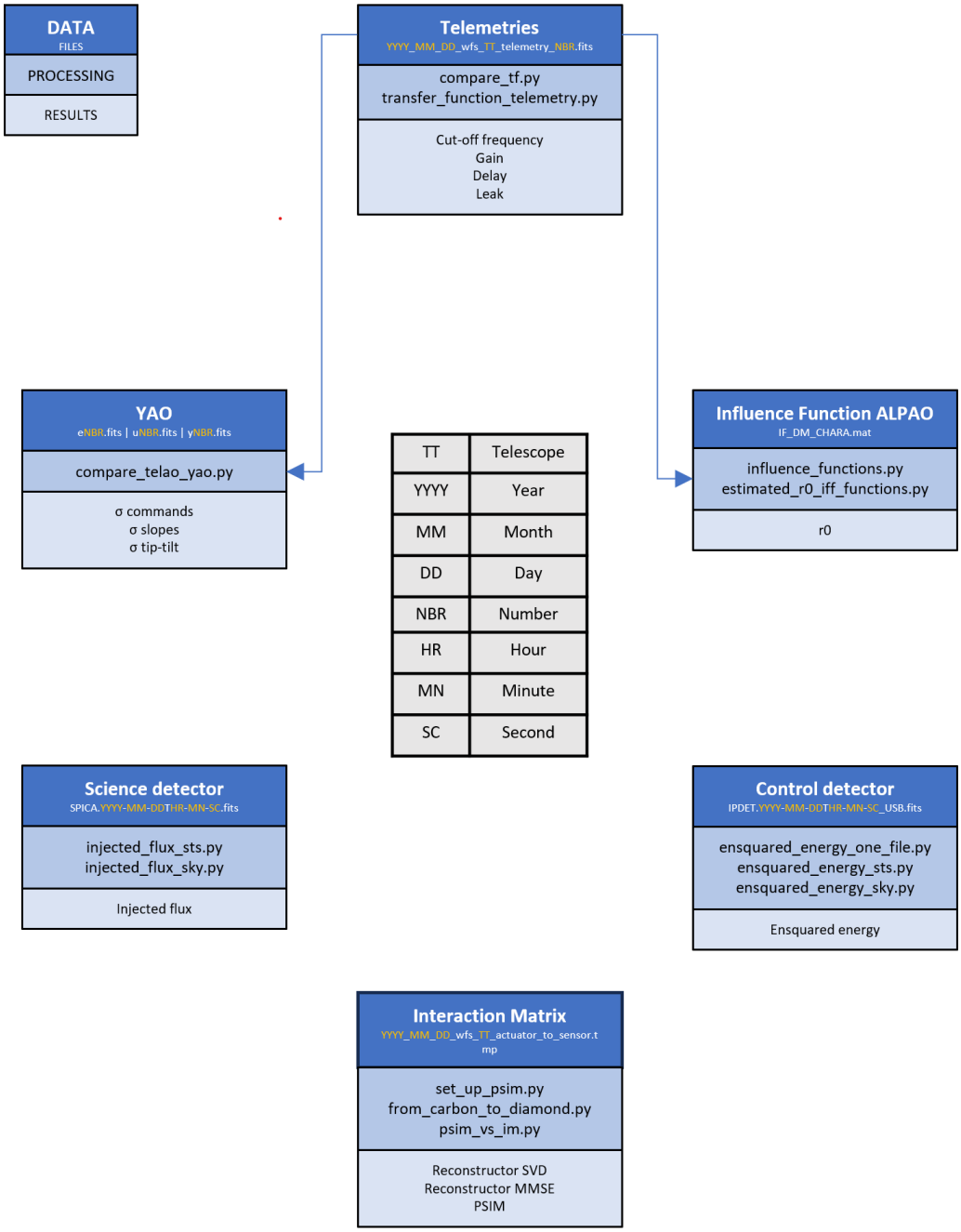


Figure 20: Flow chart of the adaptive optics programs: from input data to output scientific variables.

6.1 Transfer Functions

compare_tf.py: Compares two files and plots their transfer functions, helping to identify the differences between the files.

Inputs : Telemetry file (example on the Figure 6).

Outputs: Transfer Function with the fitted model to acquire gain, leak, and delay during the

measurement.

transfer_function_telemetry.py: From the commands of the DM in CL and OL obtained from Telemetries, calculate the Transfer Function. It helps to understand the loop parameters.

Inputs : Telemetry file (example on the Figure 6).

Outputs: Transfer Function with the fitted model to acquire gain, leak, and delay during the

measurement.

6.2 Compare TELAO / YAO

compare_telao_yao.py: I extract from TELAO (real data) and YAO (simulated data) the same vectors (commands, slopes, tip-tilt) and compare them thanks to histogram plots.

Inputs : Telemetry file (example on the Figure 6).

Outputs : σ commands, σ slopes, σ tip-tilt

6.3 Influence Function ALPAO

influence_functions.py: ALPAO supplied the Transfer function in Matlab format. This program reads, adapts, and reshapes the influence function to match the current system.

Inputs: Influence function Matlab from ALPAO (example on the Figure 6).

Outputs: Influence function adapted to Python.

estimated_r0_iff_functions.py: Helped by the Influence functions already known by CHARA, this program can calculate an estimation of the Fried diameter (r0) by the transformation of the DM commands into Zernike coefficient.

Inputs : Telemetry file (example on the Figure 6).

Outputs: r_0

6.4 Science Detector

injected_flux_sts.py, *injected_flux_sky.py*: From science detector acquisition, it is possible to get the injection flux of every fiber. The goal is to optimize the injection in SPICA's fibers. It exists the STS(Six Telescope Simulator) version or the SKY (real data) version.

Inputs: Telemetry file (example on the Figure 6) and KappaLR.dat (if it is Low Resolution) that gives the x and y coordinate of each photometric channel. These references can evolve.

Outputs: Injected flux

6.5 Control Detector

ensquared_energy_one_file.py: Compares two files and plots their transfer functions, helping to identify the differences between the files.

Inputs: Telemetry file (example on the Figure 6) and reference.dat, a file that contains the x and y coordinate of each box center. It is mandatory to calculate the Ensquared energy.

Outputs: Ensquared energy in %

ensquared_energy_sts.py, *ensquared_energy_sky.py*: From the control detector, the program calculates the energy contained in an FWHM squared around the reference pixel on the detector. The reference pixels are the x and y center of each box on the acquisition camera. The more ensquared energy is contained in that region, the more flux enters the detector. The difference between sky, sts, and "one_file" is the unit: the programs "sky", and "sts" give energy, and "one_file" gives a percentage of ensquared energy.

Inputs: Telemetry file (example on the Figure 6) and reference.dat as before

Outputs: Ensquared energy

6.6 Interaction Matrix

set_up_psim.py: Creates the basic grid to locate each sub-aperture and each actuator position thanks to the misregistration parameters. Without misregistration, the position is the default supplied by CHARA.

Inputs : Telemetry file (example on the Figure 6).

Outputs : It returns functions usable for "*from_carbon_to_diamond.py*".

from_carbon_to_diamond.py: From real IM (CHARA records), we can get the misregistration of the system (misalignment) and by a minimization function, it is possible to create a pseudo matrix with the same misregistration but without noise. After that step, we need to invert the matrix to obtain the reconstructor. It will be injected in the loop to work on the sky.

Inputs : Telemetry file (example on the Figure 6).

Outputs : PSIM, reconstructor MMSE , reconstructor SVD

psim_vs_im.py: This program compares the residues between the default reconstructor and default IM from CHARA with the synthetics ones. It helps to evaluate the performance of each reconstructor.

Inputs : Telemetry file (example on the Figure 6).

Outputs: residues for PSIM, reconstructor MMSE, reconstructor SVD compare to default ones

7 Conclusion

This work is an approach to understanding the adaptive optics of CHARA in the visible band. It shows what the current system is, how it processes and what are the defaults. After my study, other people could with this paper have a base of improvement. There are tools, advice, and tips to succeed. The different codes are commented-on and user-friendly. The figures are explicit and the tables are clear.

The current adaptive optics does not give the best performance yet, whether in term of Strehl Ratio or Injected Flux in the fibers. There is a way to enhance it with the methods I proposed. I already gave a glimpse of analysis on what are the performance and what can be after the implementation of the MMSE reconstructor for example. The detection of the misregistration is a solid tool to probe the alignment of the system and the r_0 has never been calculated from the commands so far.

I would have continued my progress on the "a" parameter. I would have built a real table that links the best "a" value with sky parameters. There are already simulations with YAO but tests on the sky will allow us to have real data. Also, it would be necessary to continue to understand how to increase the accuracy of the r_0 and how to take into account the difference between the telescopes in the computation.

There are still a lot of aspects that I did not discuss in this paper as interaction matrices from sky measurements instead of the internal source, the influence of the LABAO WFS on the AO performance, or finally the Strehl Ratio calculated from SPICA's experiment.

My work here is finished but the story just started. I would like to thank personally Denis Mourard, who held the project and defined the missions and the tests. I thank Philippe Berio who suggested the different improvements and was also available to help me with encoding. I thank Olivier Lai a lot for all the knowledge and advice he gave. Finally, I thank all the people who supported and helped me to accomplish this whole work: Julien Dejongue, Mattew Anderson, Theo Ten Brumelaar, Cyril Pannetier, Fatme Allouche, David Salabert, Stéphane Lagarde, and all others. I wish them all the best for the next years. This subject was astonishing, fascinating, and useful. I hope that my participation influenced positively the project and I am proud to deliver this job.

"This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 101019653)."