Optimizing the Optical Calibration Performance of a Multi-Object Adaptive Optics Instrument

by

Laurie Nhu An Pham
M.Sc., Ecole Nationale Supérieur d'Ingénieurs de Caen, France, 2007

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Mechanical Engineering

Optimizing the Optical Calibration Performance of a Multi-Object Adaptive Optics Instrument

by

Laurie Nhu An Pham
M.Sc., Ecole Nationale Supérieur d'Ingénieurs de Caen, France, 2007

Supervisory Committee

Dr. Colin Bradley, Supervisor
(Department of Mechanical Engineering)

Dr. Carlos Correia, Departmental Member
(Department of Mechanical Engineering)

Supervisory Committee

Dr. Colin Bradley, Supervisor
(Department of Mechanical Engineering)

Dr. Carlos Correia, Departmental Member
(Department of Mechanical Engineering)

ABSTRACT

Multi-Object Adaptive Optics (MOAO) is an adaptive optics technique being developed for Extremely Large Telescopes that will allow simultaneous observation of approximately 20 targets in a several arc-minute field of regard. Raven is an MOAO pathfinder developed by the Adaptive Optics Laboratory of the University of Victoria, in collaboration with the National Research Council of Canada and the Subaru Telescope. It will be the first MOAO instrument on a 8-m class telescope, will demonstrate that MOAO technical challenges such as open-loop control and calibration are achievable on-sky and will deliver science results using three natural guide stars and two science arms on $\sim 3.5'$ field-of-regard. The open-loop approach makes the need for calibration even more crucial.

An important part of the calibration process resides in the misregistration of the wavefront sensors (WFSs) with the deformable mirrors (DMs) because the sensing elements are located before the correcting ones. This problem is solved using a calibration DM seen by all WFSs in the system that permits the open-loop WFS to be registered to the science DMs. The method developed in this thesis registers the position of the DM actuators to the WFSs and gives misregistration values. These results are then used to better align the instrument, to have a better knowledge of the positions of the different optical components and generate new ways to perform the AO correction. Using the registration parameters results, synthetic interaction matrices are created in order to improve the AO correction. Calibration tests are also presented in this thesis. They show complementary tests to the expected requirements to expand the knowledge of the calibration unit behaviour.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Colin Bradley for allowing me to participate in such a big and exciting project. I am always amazed at how an instrument like Raven gets created, funded and built.

Dr. David Andersen for taking the time to help me with my thesis. I greatly appreciated the support.

Dr. Carlos Correia for being part of my committee and for your interest in my work.

Dr. Olivier Lardière for being able to explain to me so many hard-to-grasp concepts with ease. Your sense of humour made the gruelling hours of Raven testing in the laboratory really fun.

Dr. Célia Blain for being so helpful in the Canadian labyrinth, especially the UVic obstacle course. Thank you for listening to my rants and funny stories, and also for sharing so many nap-inducing meals.

Lori Muck for making the administrative part of Raven really easy.

The Raven Team for being such a lovely group of people. Working in a fun and helpful environment like this one is something I value.

I would also like to thank all the people I met while in Canada. Life is not only about work and I am glad to have had so many new experiences and to have discovered so many cultures and countries. I would like to thank all the friends I made from all over the world. Thank you to all the exchangers from 2011 for kickstarting my first year in Canada with an explosion of fun that I still remember very fondly up to this day. You set the bar so high that I knew it could not be repeated the following years.

A big thank you to my duck-duck-duck food fanatics Mariam "I'll come if there's food involved" Ghani and Camille "we need more food" Hoang. Only you could make Canada a place where I eat more duck than when in France. I cannot thank you enough for all the food memories I have and all the cooking tricks you taught me. Have I said "duck" yet ?

Speaking of duck, I also have to thank the best roommate ever: Ashley "Sour Wolf" Field. "I volunteer Sophie as tribute", "That's Woody Harrelson !", "Ephemere" (Angie Pass, never stop being so funny !) and "you look so peaceful when you sleep" are some of the few quotes that always put a smile on my face. Thank you Krista "Skinny Girl" Conle for all the movies we watched together with Mike "Let's climb a tree" Irvine. But no thanks for the awful "The Covenant" and "Priest". Thanks Aidan for being "phat" with me and for all the sweat when Monica'ing. Thanks Kévin for your endless amount of silliness.

And to my friends Tingting, Fanfan and Aymeric that I left in France when starting this big Canadian adventure, a big thank you for just being the same people I always knew every time I come back to visit. I cannot wait to play with your little aliens and predators.

Finally, I'd like to thank "Observatoire de Paris" family. You guys made me want to expand my horizons beyond what I had and it led me to incredible experiences. An extra special thank you to my Persee-Cutters, Julien "10 minutes à perdre" Lozi, Emilie "Kiwi" Lhomé and Sophie "Citron" Jacquinod, for making work my abs everytime I talk to you (I sound so serious here).

There are so many friends that I'm forgetting so if your name starts with a letter between A and Z, thank you !

And last but not least, my Phamily who supports me in everything I do and who believes in me more than I do. I just have the best parents in the world.

DEDICATION

This is dedicated to my best cheerleaders and family:

Papa, Maman and Phiphi.

Chapter 1

Introduction

1.1 Multi-Object Adaptive Optics (MOAO)

The field of view (FoV) of classical adaptive optics (AO) systems is limited by an effect known as anisoplanatism [1]. For a given telescope pointing, the light from a distant object is perturbed by the turbulence in a cylinder of the atmosphere (with a diameter equal to the telescope primary mirror). Light from a nearby distant object will pass through an overlapping, but non-identical cylinder of turbulence on its way to the telescope (figure 1.1). In classical AO systems, one wavefront sensor (WFS) will pick-off light from a single, relatively bright point source, and a deformable mirror (DM) will be commanded to correct the incoming wavefront to null out the wavefront error induced by the turbulence along a single line-of-sight (within a single cylinder). The AO correction for a nearby object will be inferior because it will be viewed through a slightly different cylinder of turbulence. The isoplanatic angle can be described as the angular distance from the guide star at which the wavefront error is below 1 radian, and it is typically about 20 arcseconds.

To enlarge the isoplanatic angle, one can place multiple DMs in series, each conjugated to a different atmospheric altitude. This Multi-Conjugate AO (MCAO) [2, 3] approach can be used to enlarge the FoV to an arcminute or two, but the performance will ultimately still be limited by generalized anisoplanatism. The FoV can be further enlarged by adding more DMs in series, to remove the turbulence generated at even more atmospheric heights. However, the complexity of the MCAO system rises (and the throughput falls) with each additional DM relay.

MOAO [4, 5, 6] is another approach that promises to increase the FOV over which

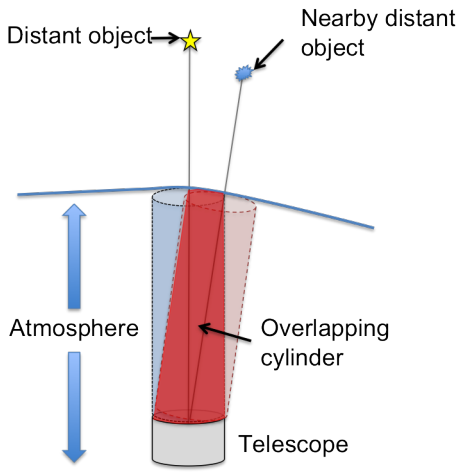


Figure 1.1: Illustration of anisoplanatism. The turbulence in the cylinder associated with a distant object (yellow) is not the same as the one associated with a nearby distant object (blue).

AO corrections can be applied between 5 and 10 arcminutes. An MOAO system selectively corrects for turbulence in the localised region around multiple science targets; i.e. a multiplex advantage is realized. Therefore, if the turbulent volume over the whole telescope FOV can be sensed and modelled (tomographically), then a probe with an embedded DM can be positioned anywhere in the field of regard. This enables optimal turbulence correction at each target's location. Measurement of the turbulent volume utilizes wavefront data from multiple sensors that probe different lines-of-sight through the atmosphere to their respective guide stars. Once the information from these multiple WFSs is combined into a single tomographic model of the turbulence, it is straight-forward to imagine multiple science pick-offs, each incorporating its own DM, feeding multiple integral field spectrographs [7, 8].

MOAO systems are planned for integration with the next generation of extremely large, ground-based optical telescopes (ELTs) . Their large FOV will enable roughly 20 science targets to be imaged and corrected using MOAO [4]. However, the WFSs will be common to all science channels and, therefore, placed upstream of each DM in an open-loop architecture. Such MOAO systems will have a field of regard of a few arcminutes to observe 20 science targets at the same time. The planned instruments for the two projects of ELTs are: IRMOS [9] for the Thirty Meter Telescope [10] [11] and EAGLE [12] for the European Extremely Large Telescope [13].

1.2 The Raven Multi-Object Adaptive Optics Demonstrator

Raven [14] will be the first multi-object adaptive optics instrument on an 8-m class telescope feeding an AO-optimized science instrument, the Subaru InfraRed Camera and Spectrograph (IRCS) [15]. A functional block diagram of the system is shown in figure 1.3 and a top view of the computer-aided design is shown in figure 1.4. The view of the Raven bench is shown in figure 1.5.

Raven consists of 8 main optical subsystems:

- A deployable **Calibration Unit** (CU) [16] which functions as both a telescope simulator and a turbulence generator. It contains an array of off-axis natural guide star (NGS) sources and one on-axis Laser Guide Star (LGS) source. Light from the CU feeds the three open-loop (OL) wavefront sensors, the LGSWFS and two science arms. The three functions of the CU are: 1) to help align other Raven subsystems, 2) to calibrate the AO system (generate interaction matrices and measure field-dependent non-common path aberrations), and 3) to test the MOAO correction with turbulence generated by two rotating phase screens and a ground-conjugated DM (Figure 1.2).

- Three **NGS OL Shack-Hartmann WFSs** that are mounted on x-y translation stages to prevent the pupil from rotating on the WFS lenslet array with respect to the DMs.

- An **on-axis LGSWFS** which will be fed by the Subaru Sodium beacon in order to improve AO correction and sky coverage.

- Two **science pick-off arms** consisting of a mirror mounted on an r $-\theta$ arm followed by a trombone mirror that keeps the optical path length constant.

- A **science relay** for each arm containing a DM which is a custom ALPAO DM with 11x11 actuators over a 25 mm aperture.

- A **figure source and closed-loop (CL) WFS** that are employed to measure the DM shape (using the figure source); calibrate the bench or evaluate MOAO performance; or, allow classical AO system operation.

- A **beam combiner** also contains a K-mirror so that extended objects can be properly aligned onto the slit. After the science relay, light from both arms of the system are combined so that the common beam shares an identical exit pupil and provides two adjacent 4 arcsecond science fields to the single IRCS slit.

- An **acquisition camera** that can be used to determine the telescope pointing and ensure that shadows of the probe arms fall over the NGSs and science targets.

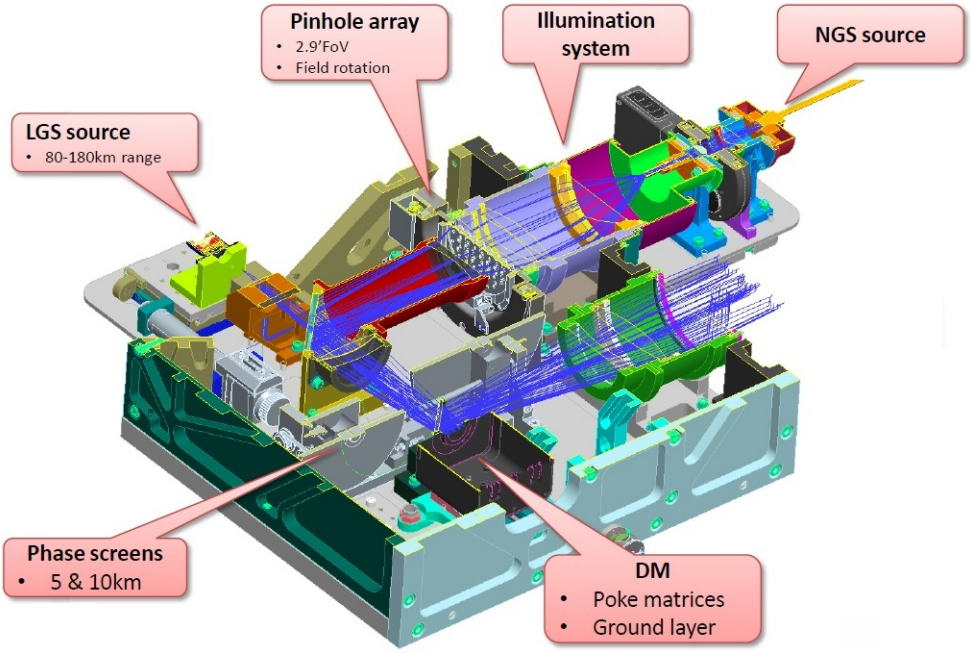


Figure 1.2: Calibration unit system.

The science gain achievable by Raven, in comparison to classical AO systems such as Subaru's AO188 [17], will be modest because Raven only has two science channels. Nevertheless, the 8 m aperture of the Subaru Telescope enables science that is not achievable on smaller telescopes. Raven is capable of delivering high ensquared energy into the IRCS slit. The combined technical and scientific aspects of MOAO, which Raven will demonstrate, will excite the astronomical community and build support for future facility-class MOAO instruments with much larger multiplex advantages on either 8-m class telescopes or extremely large telescopes (ELTs).

MOAO has the potential to deliver near diffraction-limited images to multiple, small patches spread across a large FoR. One challenge of an MOAO system is that it is highly distributed. On Raven, light from one LGS and three NGSs will be sensed within a 3.5 arcmin FoR. Pixels from the OLWFS detectors will be read by the Raven Real Time Computer (RTC) and measured slopes will be used to create a tomographic model of the atmosphere above the observatory. This tomographic model will be sampled in directions defined by the position of the science probes in the patrol field and DM commands will be generated. All of these actions are performed using open-loop control. Accurate knowledge of the science probe placement in the focal plane and the relative alignment of the DMs and WFSs in the pupil plane is required. The RTC also performs the computation to provide the DM commands for the wavefront correction. Other options can be added to it. For example, the wavefront DM fitting can be added.

As a precursor to designing Raven, a broad swath of parameter space was explored in detailed end-to-end simulations [18]. This was necessary in order to determine a system architecture that can realistically meet the proposed performance requirements and deliver useful MOAO-corrected images to the Subaru IRCS spectrograph. As Raven was conceived to be a science-capable, NGS only MOAO system, in addition to a technical demonstration, the AO architecture was designed such that it will deliver the desired performance even when faint guide stars are used. The addition of the single, on-axis Subaru LGS to the NGS constellation improves performance and sky coverage, but does not eliminate the need for good performance with faint NGSs. The system modeling revealed that tomographic errors are the dominant factor limiting the performance of Raven. As a result, the performance will be highly dependent on the total amount of turbulence (and the distribution of turbulence as a function of altitude), and on the asterism of NGSs used to sense the turbulence.

Raven is capable of several operational modes that allow direct comparison of various OL wavefront reconstruction techniques:

- **Closed Loop Mode:** Raven will operate in classical closed-loop AO mode with bright science targets and employing the two CLWFSs.
- **Open Loop (MOAO) Mode:** The OLWFSs feed wavefront slope data to the RTC. The RTC performs a tomographic reconstruction utilizing OLWFS probes and science pick-off location.
- **MOAO and CLWFS Mode:** The CLWFS can also feed frames to the RTC

at a much slower framerate, so that the low-frequency turbulence is corrected in closed-loop directly on the science target. The high-frequency turbulence is corrected in open-loop mode.

- **MOAO and Figure WFS Mode:** The same basic MOAO processes happen with the Figure WFS option as well. In addition, the DM illumination source is turned on and the CLWFSs measure the spots from this source (it is assumed that the science target in this case is much fainter than the DM illumination source). The CLWFSs are read out by the RTC. The slopes from the Figure WFSs can conceivably be used in a CL mode to ensure the DM takes on the shape that is commanded by the OLWFSs.

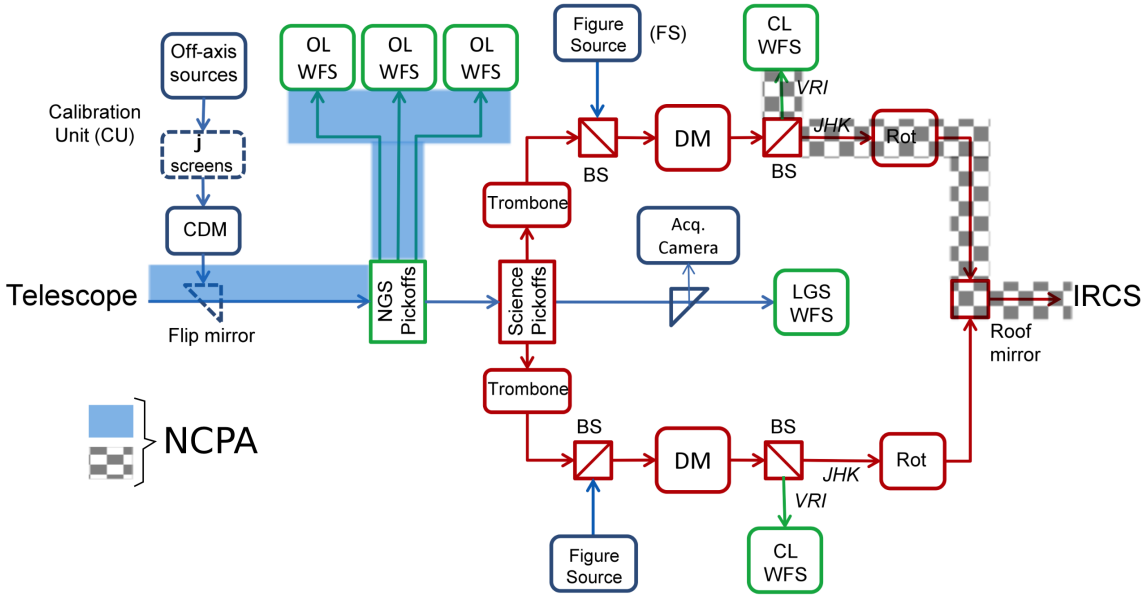


Figure 1.3: Functional optical block diagram of RAVEN. Dashed blocks are deployable. Raven consists of 8 main subsystems: the deployable Calibration Unit, the Open-Loop NGSWFSs, the Science Pick-offs, the Science Relays, the Closed-Loop NGS Truth/Figure WFSs, the Beam Combiner, the LGSWFS and the Acquisition Camera.

1.3 System Calibration Challenges Posed by an MOAO Instrument

To calibrate any AO instrument, misregistrations between the WFS and the DM must be measured and corrected [19]. However, for a MOAO system, the OLWFSs do not

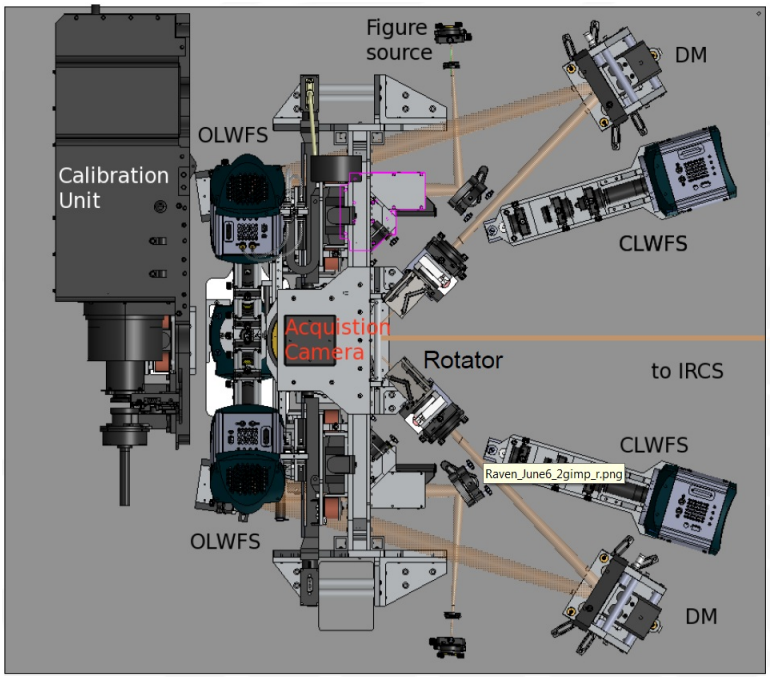


Figure 1.4: Top view of the Raven computer-aided design.

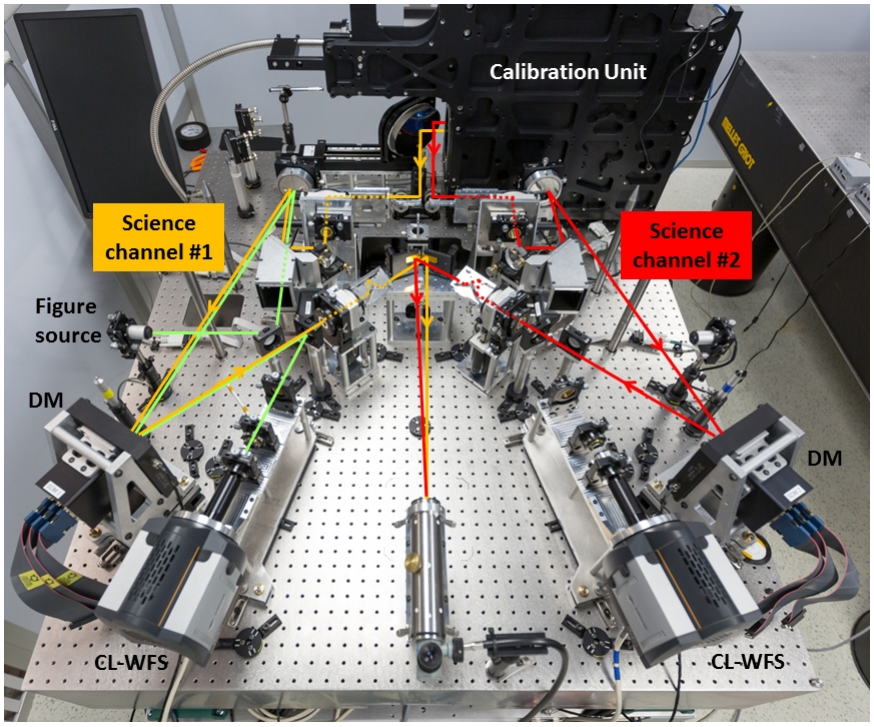


Figure 1.5: Top view of the Raven bench.

see the DM so the classic approach of measuring DM-to-WFS interaction matrices cannot be used. To overcome this issue, Raven employs a CLWFS and a common DM (calibration DM in the CU). With these components, one can measure the open-loop interaction matrix between the science DM and the open-loop wavefront sensors. In order for the RTC to generate the optimal DM shape, the AO system must be carefully calibrated. [20].

Science observations could be up to 8 hours in a single night, during which the natural guide stars and science targets are tracked by their respective pick-off arms. During tracking, the relative positions between science DMs and OLWFSs will vary and affect the interaction matrix structure. Any misregistration between the WFS and science DM axes will alter the original (i.e. static) calibration of the WFS-to-DM registration. Therefore, any subsequent misregistration creates an error in the DM-to-WFS interaction, as embodied in T (the wavefront reconstructor [21]), and results in sub-optimal MOAO performance.

An incorrect DM-to-WFS registration, induced in the Raven calibration process, could introduce significant wavefront error. Misregistration errors, which usually have minor effects in closed-loop AO (slight increase in servolag because it needs more iterations to reach the best correction), lead to static residual wavefront errors in open-loop AO systems due to a relationship between the OLWFS and science DMs registration.

The goal of the calibration is to measure the system's misalignments and aberrations in order to find the best transformation matrix R given in equation (1.1) [20].

$$\overrightarrow{u} = CTR(\overrightarrow{s} - \overrightarrow{s_0}) + \overrightarrow{u_0} \tag{1.1}$$

Where $\overrightarrow{u}$ represents the correct voltages sent to each science DM to obtain a flat wavefront, C is the command matrix, T is the tomographic reconstructor, R is the matrix expressing the OLWFS slopes into a common space, $\overrightarrow{s}$ is the slopes of the 3 wavefront sensors, $\overrightarrow{s_0}$ is the slopes offsets and $\overrightarrow{u_0}$ the command offsets. In this thesis we do not concern ourselves with the tomographic process. We instead focus on how to best determine the open-loop command (C) and transformation matrices (R).

The algorithm proposed herein, for achieving optimal registration, utilizes an experimental measurement of interaction matrices between a DM and a WFS (section 1.3.2). In section 1.3.1, a brief overview of other system aberrations, that must be corrected during calibration, is also presented.

1.3.1 Non-Common Path and Field Dependent Aberrations

Non-common path aberrations (NCPAs) result from the static optical aberrations and optical component spatial misalignments present in the optical path difference between the science output path to IRCS and the CLWFS path (blue and checkered, see figure 1.3). NCPAs are calibrated and removed prior to acquisition.

These aberrations are field-dependent (blue) because of the pick-offs' positions in the telescope field of view. Each WFS pick-off has an offset induced by the telescope optical aberration (see figure 1.6). The dynamic nature of an MOAO system dictates that periodic corrections for these field dependent aberrations must be applied during acquisition. Prior to acquisition, a map of aberration measurements across the field is created. The aberration map supplies offset WFS values that null the measurement with respect to the center of the FOV.

Figure 1.6: Field dependent aberrations in the field of view of a telescope. When changing the pick-off position in the focal plane (left), the field dependent aberration seen depends on that position (right).

1.3.2 Determination of Interaction Matrices

In an MOAO system, there is no direct interaction matrix between the OLWFS and science DM. However, in keeping with the general AO terminology, the expression will be retained herein.

A key CU component is the calibration deformable mirror (CDM). This 277-actuator DM optically links the OLWFS to the science DMs; there is no physical means to directly register the OLWFS with the science DMs.

The indirect process to find the interaction matrix for each science DM is broken down as follows (see figure 1.7):

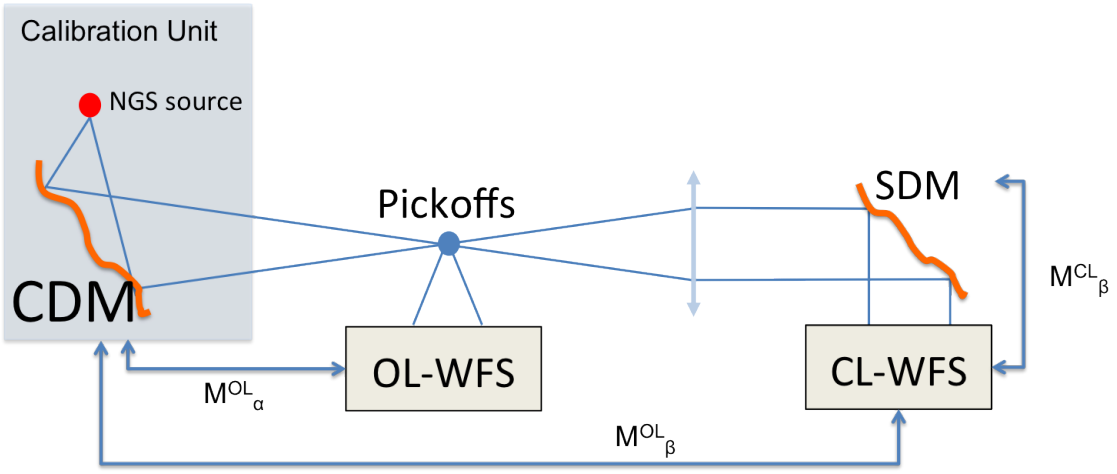


Figure 1.7: Simplified diagram illustrating how the interaction matrices are record for the calibration. The light beam is generated in the calibration unit by the natural guide star source. It is bounces off the calibration DM (CDM). Then 2 pick-offs send the light to the open-loop wavefront sensor (OLWFS) and the science path with a science DM and a closed-loop WFS.

(i) Science DM-to-CLWFS interaction matrix

$$s_\beta = M_\beta^{CL} * u_\beta \tag{1.2}$$

Where:

s_β: slopes of the CLWFS,

M_β^{CL}: interaction matrix between the science DM and CLWFS,

u_β: science DM commands.

(ii) CDM-to-CLWFS interaction matrix

$$s_\beta = M_\beta^{OL} * u_c \tag{1.3}$$

Where:

s_β: slopes of the CLWFS,

M_β^{OL}: interaction matrix between the CLWFS and the CDM,

u_c: CDM command voltages.

(iii) CDM-to-OLWFS interaction matrix

$$s_\alpha = M_\alpha^{OL} * u_c \tag{1.4}$$

Where:

s_α: slopes of the OLWFS,

M_α^{OL}: interaction matrix between the OLWFS and the CDM.

Employing the equations above, the relationship between u_β and s_α is given by:

$$s_\alpha = M_\alpha^{OL} * (M_\beta^{OL})^\dagger * M_\beta^{CL} * u_\beta \tag{1.5}$$

Where, the double product $M_\alpha^{OL} * (M_\beta^{OL})^\dagger * M_\beta^{CL}$ is the SDM-to-OLWFS interaction matrix.

It can be noted that the transformation matrix R can be expressed using these interaction matrices :

$$R^\dagger = M_\alpha^{OL} * (M_\beta^{OL})^\dagger \tag{1.6}$$

An interaction matrix is typically rectangular (its dimension is the number of DM actuators (rows) by the number of valid lenslets of the WFS (columns)) and the pseudo-inverse with singular value decomposition is employed to invert it. In figure 1.8, the circle represents the zone seen by the subapertures of a WFS and the dots represent the actuators.

One issue in this particular case is the position of the DM edge actuators, especially ones outside the pupil. Their effect is not well-measured because their centers are located outside of the zone seen by the WFS. If this effect is not taken into account, these actuators will tend to be commanded by higher voltages because the WFS barely measures the effect of an edge command. This can also result in damaging the DM. A solution to this problem is to set the singular values associated to deformations that cannot be seen by the WFS to zero. The edge actuators are then not used to control high spectral frequencies. This reduces the noise propagated by the command matrix.

We can avoid performing a pseudo-inversion by developing a theoretical and experimental technique to relate the spatial position of the system's DMs actuators with the WFSs subapertures.

1.4 Previous Work

1.4.1 Registration Parameters

In closed-loop AO systems, DM-to-WFS misalignments are accounted for in the interaction matrix created during calibration. Thus, the closed-loop control will flatten the wavefront even if the command is not perfect. In open-loop, however, the misalignments must be identified and quantified prior to their integration in the command matrix because the DM must accurately go to the correct shape with no feedback. There is no published method for applying the interaction matrix in order to locate DM actuator positions on the WFS camera and, subsequently, using these locations to extract misregistration parameters (magnification, translation and rotation). The misregistrations created by these misalignements are particularly important for open-loop tomographic wavefront reconstruction. More transformations could be studied such as barrel deformation but the method will show that the five presented geometrical transformation can provide a good model.

Neichel's paper [22] presents a method to identify the misalignment parameters from an interaction matrix for alignment and tomographic reconstruction. The method is based on registration of misalignment in interaction matrices. It relies on an iterative method of identification of the parameters to relate an estimated matrix to an experimental one.

An alignment technique for a DM and a WFS, presented by Oliker [23], measures misalignments, such as horizontal and vertical translation, magnification and rotation, using a DM with its actuators positioned in a waffle mode. Significant advantages are realized by employing the actuator mapping and misregistration technique:

- The precise spatial location of each DM actuator, as projected onto the OLWFS or CLWFS, is generated.
- A subset of DM actuators can be used to accurately determine the misregistration.
- The method utilizes an interaction matrix, which is always experimentally determined during Raven calibration.

This thesis will use the same algorithm for the horizontal and vertical translation. However, the rotation and magnification misregistration parameters will be estimated by developing other geometrical methods.

1.4.2 Interaction and Transformation Matrices

The transformation matrix presented in equation 1.1 can be expressed as the product of two interaction matrices (see eq. 1.6). These are usually the experimental interaction matrices measured on the instrument. In this thesis, I am proposing a new method to create this transformation matrix synthetically using the instrument registration parameters. This will provide a noise-less theoretical transformation matrix.

Another way to find this transformation matrix has been developed for the CANARY instrument [24] using the Learn method from their Learn and Apply algorithm [25]. The Learn method creates the transformation matrix experimentally without expressing the registration parameters. It links the WFSs data by measuring the covariance matrices between the WFSs. This Learn algorithm is a very powerful tool because all the misalignments are then considered, even the higher orders of parameters that I do not consider. The problem also lies in this advantage that the method proposes: there is no knowledge of the instrument's registration parameters between the optical components. If one wants to model the system, these parameters cannot be taken into account and their contribution is then hard to quantify.

1.5 Thesis Outline

This thesis is based on the following outline:

Chapter 1 presents the context of the thesis with the specific challenges faced by an MOAO system.

Chapter 2 describes the research objectives using the new method to register a DM with a WFS which leads to applications on the Raven instrument.

Chapter 3 presents the algorithms to map the positions of DM actuators on a WFS and extracting the registration parameters.

Chapter 4 gives the methodology to create theoretical interaction matrices and command matrices from experimental measurements. This is a direct application of Chapter 3.

Chapter 5 includes the experiments performed on the Raven instrument. First, there is an overview of tests performed on the calibration unit to assess the performances of its most important components. Then the improvement of the

alignment procedure will be presented in details using the methods developed in Chapter 3. Finally, the performances of synthetic interaction and command matrices compared to experimental matrices will be shown.

Chapter 6 concludes the thesis.

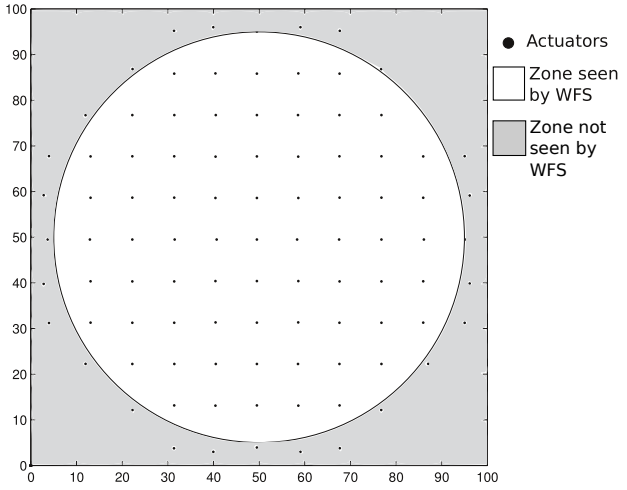


Figure 1.8: Illustration of the science deformable mirror actuators as seen by the OLWFS. The three WFSs are rotated with respect to the science DM by $-90\,^\circ$, $180\,^\circ$ *and* $90\,^\circ$

Chapter 2

Research Objectives

The research objectives include the following steps:

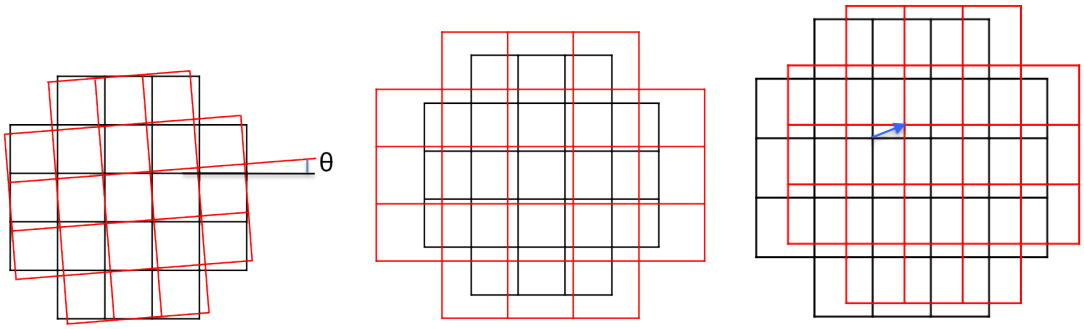
- Analyzing the problems currently preventing the optimal calibration of the open-loop AO system.
- Proposing a new technique for the calibration employing a novel registration method between critical AO components, i.e. the system's science and calibration DMs and the open-loop wavefront sensors.
- Developing a practical Raven system optical calibration tool applicable to any DM and WFS alignment operation : e.g. laboratory or prior to on-sky measurements.

2.1 Actuator Mapping and Misregistration Characterization

There are several applications made possible by our actuator mapping method. The actuator mapping method can be used during alignment of an AO system to accurately center the DM with the WFS or vice-versa. This method can also be used for optical tests, such as the measurement of the pupil size on the WFS (magnification error) and the image distortion. The relative position of the DM actuators to WFS subapertures will help in the calibration of MOAO systems. The usual method to compute a command matrix uses a singular value decomposition to obtain singular values of the matrix. These singular values are then thresholded experimentally to

remove the effect of the deformations that cannot be seen by the WFS. One of the direct applications of our method for determining the relative position of the DM actuators to the WFS subapertures is the optical alignment on the bench. When the actuators' positions are displayed in real-time on the WFS camera, the user can position these actuators to align the DM according to the design. The common alignment procedure is to superimpose the DM center actuator on the WFS center.

Another important application from the position of the DM actuators relative to a WFS is the characterization of the registration. The registration parameters are based on the actuator mapping and give quantitative data for the alignment. A noiseless theoretical transformation matrix can be built from the registration parameters. Even though there will be noise in those registration parameters, we can measure the five required parameters more accurately and faster than we can measure an experimental interaction matrix. The registration parameters considered here are (i) the horizontal and vertical shift, (ii) the rotation angle and (iii) the horizontal and vertical magnification between two sets of actuators' maps. These actuators' maps can be strictly experimental, in which case the parameters will give information on the relative alignment of the WFSs and DMs on the bench. The misregistration parameters will also be utilised by the Raven tomographic reconstructor. That reconstructor needs the measurements and commands expressed in the same space. This will be done using the misalignments parameters between WFSs and DMs.



(a) Rotation *(b) Magnification* *(c) Translation*

Figure 2.1: Illustration of the misregistration parameters. The black grid is the reference, the red grid is the misregistered one.

We have developed a method for estimating the misregistration parameters for the Raven instrument. It is applicable to systems with membrane-style DMs and Shack-Hartmann wavefront sensors. The method relies on the fact that the mirror membrane is continuous and would not work on a segmented DM. A different number of actuators

or a different geometric pattern would work as long as the mirror has a continuous membrane. This method is applicable to Shack-Hartmann wavefront sensors because the positions of the actuators are determined directly by the displacement of the WFS slopes. However, the method could be extended to other types of WFSs.

2.2 Synthetic Interaction and Command Matrices

The registration parameters are the base to create theoretical rotation matrices. They are the matrices that transform the data from the OLWFS space into CLWFS space. They are built using experimental measurements. They are expected to be as good or better than the experimental matrices because they will not contain measurement noise except from the registration parameters identification. In chapter 4, the construction of the synthetic matrices will be presented before their performance comparison to an existing method, called Learn and Apply, transforming WFS data from one space to another in chapter 5.

Chapter 3

Improving the Calibration of an MOAO Instrument

3.1 Actuator Mapping Method

To determine the position of the actuators of a deformable mirror, we can use the interaction matrix between that DM and a wavefront sensor. Although a minimum of 3 actuators are required to compute the 5 registration parameters, we use information from a larger selection of actuators to maximize accuracy. Interaction matrices are used because they are always measured during the calibration process but only information from selected actuators inside the pupil are retained.

Each column of the interaction matrix shown in figure 3.1 represents the coordinates of the WFS slopes during an actuator poke. Since the actuators are poked sequentially, there are as many columns as actuators on the deformable mirror. The upper half of the matrix gives the X coordinates of the slopes whereas the lower half gives their Y coordinates.

Figure 3.2 shows the position of the WFS spots from the interaction matrix when actuator number 107 is poked (blue crosses). When the DM is in its flat position, the spots are located at their reference positions (red crosses). We chose to express slopes in terms of the fraction of the FOV of the WFS subapertures. They are given in pixels so there is a need to convert them into a general coordinates reference. In this case, the selected coordinate system reference is the WFS subaperture but the reference could be chosen arbitrarily because it would result in a simple shift of the coordinates.

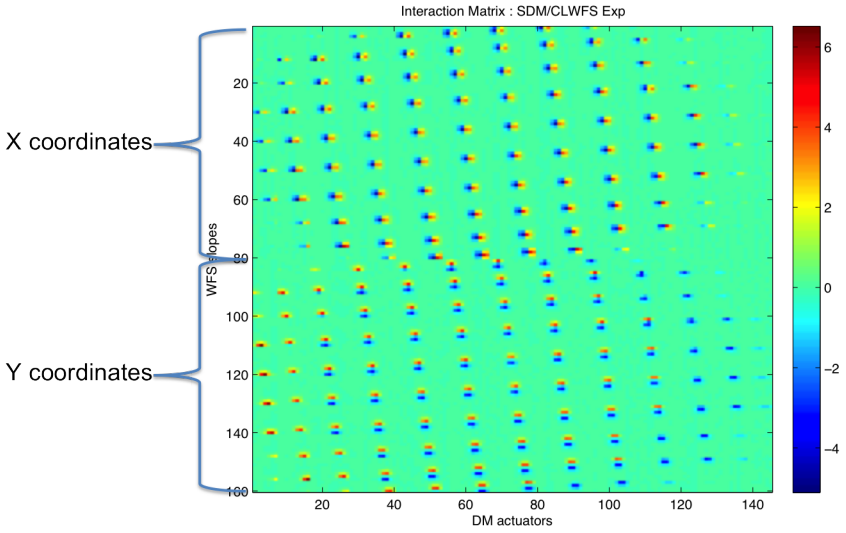


Figure 3.1: Experimental interaction matrix between a CLWFS and a SDM. Each column of the matrix represents the coordinates of the WFS slopes during an actuator poke.

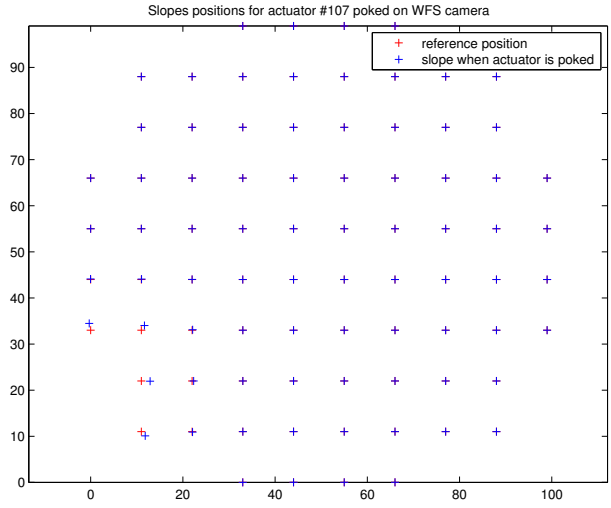


Figure 3.2: Example of WFS reference positions (red) and slopes (blue) when actuator 107 is poked.

Actuator positions are determined from the intersection of slopes resulting from the push of a single actuator. The slopes on the WFS point towards the actuator position. The actuator, the reference position of each slope and the slopes seeing the effect of a poke form a straight line. Thus the intersection of the lines with each other

gives the position of the actuator (blue lines in figure 3.3). This method works when only one actuator is poked, or when there is no overlap between the displacement of the spot of two actuators. In the latter case, their zones of influence should be distinct (see section 5.2). If a subaperture is affected by two actuators, it will not point at an actuator's location and will add noise to the measurement.

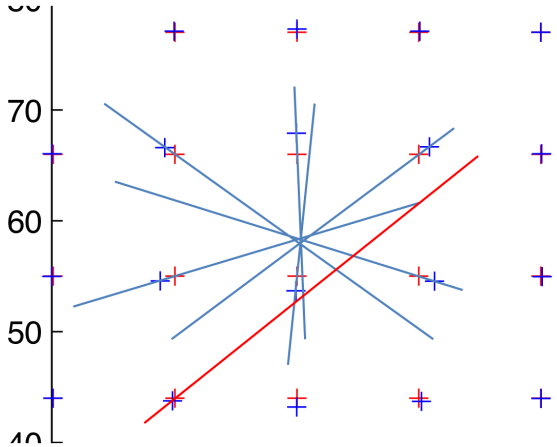


Figure 3.3: Illustration of how the intersection of the slopes gives the actuator position (blue lines). For small spot displacements, the accuracy is lost and intersect the other lines far from the actuators position (red line).

In the experimental determination of the intersection point (see figure 3.3), measurement noise must be removed through the selection of an appropriate filter. WFS noise induces random errors on the slope measurements. In figure 3.3, the red line drawn by the slope on the bottom-left does not pass through the actuator position. This error comes from the noise of the measurement. The threshold chosen to filter the slopes is 10 percent of the maximum spot displacement. If the spot's displacement is less than that value, it will be set as a null displacement and will not be used in the intersection method. The slope errors are not accounted for in the threshold, as a result, an optimization algorithm is performed to find the actuators' positions.

3.1.1 Optimization Method for the Actuators Best Position

The position of each actuator is computed independently and as a result, the algorithm has to be applied as many times as there are actuators. As illustrated in figure 3.3, the slopes do not intersect at a single point. The following optimization algorithm is used to find the actuators' optimal positions given the displacement of the spots.

The optimization method requires a starting point for the search process. The initial guess is chosen with the goal of being as close as possible to the actuators coordinates. This minimizes the computation time as fewer iterations will be needed. When an actuator is poked, the subapertures closest to its position have the greatest spot displacements. If the actuator is located exactly at the center of the subaperture, the corresponding spot should not move because the average slope engendered by that actuator over the subaperture is zero. That is not a problem as surrounding slopes would still point toward the actuator position.

The goal of the algorithm is to minimize the distance between a point and the lines prolonging the slopes. In figure 3.4, the arrows represent the spot displacement from the center of the subaperture. A zone is defined by two dotted lines which have the following parameters: origins are the center of the subapertures and directions are the slope plus or minus ten degrees. This define an arbitrary error zone related to the slope error. Then for the line closest to the considered point, r and σ are defined as: r is the distance between the estimated point and its projection B onto the slope, and σ is the distance between A and B. A is the projection of the guessed point onto the closest error line.

For each non-thresholded slope, r and σ are computed and the function to optimize is equation (3.1) where d is the slope length and n the number of non-thresholded slopes. Multiplying by d^2 gives more weight to spots with longer displacement and reduces the noise. Matlab has the *fminsearch* function that requires a function to minimize and a starting point. With the initial guess and equation (3.1) to minimize, the actuators positions are obtained : they are optimized to be as close as possible to every slopes considered. The optimization process will find the point that is closest to each prolonged slopes and inside the error cone delimited by the error bars. In a case of only 2 slopes measured, the optimized point will be the intersection of the prolonged slopes.

$$\epsilon = \sum_{k=1}^{n} d_k^2 \frac{r_k^2}{\sigma_k^2} \tag{3.1}$$

3.1.2 Disk Filtering : Actuators Visible on WFS

The optimization procedure generates a map of actuator positions. Figure 3.5a presents these positions in red dots and the blue crosses represents the center of the WFS subaperture. Most of the actuators outside of the camera have a position

that is harder to find. This can come from the fact that there are not enough data from the slopes and also because the edge of the mirror membrane is stiffer (so it moves less). The influence functions of the edge actuators are different than other actuators. If the goal is to filter the actuators located outside of the WFS camera, the information in figure 3.5a is enough. The user just has to locate the pupil on the WFS and relate it to the position of the blue crosses.

However, characterising the misregistration parameters requires a more accurate map so the edge actuators are removed automatically through an algorithm that selects an arbitrary zone. That zone is generated via the subapertures of the WFS and keeps all the actuators located within 4.5 subapertures from the center of the WFS. Considering the number of registration modes, not all actuators are needed (only 3 actuators would be enough in theory).

The robustness of this method for measuring WFS to DM registration allows the user to apply it to DMs with different number of actuators and geometries. Figure 3.6 presents the maps of two DMs on the same WFS. On the left (figure 3.6a) is the map for the science DM which has 145 actuators, whereas on the right side (figure 3.6b), the CDM has 277 actuators. The latter has more actuators on the same pupil so they look closer on the WFS. It has to be noted that because of the disk filtering part of the algorithm, the number of actuators on the maps are lower than the total number of actuators.

3.2 Extracting Misregistration Parameters from Maps of Actuators

Once we have measured the relative locations of DM actuators and WFS subapertures, we can extract the misregistration parameters. We compare the actuator map measured by the WFS and the expected manufacturer specified actuator map (figure 3.7). Two actuator maps to compare : a reference map (red) and a model with misregistration (blue).

The transformation from a point A on the misregistered map to a point A_{ref} on the reference map (the DM frame) is given by:

$$A = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} * \begin{pmatrix} x_{Mag} & 0 \\ 0 & y_{Mag} \end{pmatrix} * \left(A_{ref} + \begin{pmatrix} x_{Shift} \\ y_{Shift} \end{pmatrix} \right) \tag{3.2}$$

Where: x_{Mag} and y_{Mag} are the horizontal and vertical magnifications, θ is the rotation angle between the maps, the origin of the rotation is the center of the DM and x_{Shift} and y_{Shift}, the respective horizontal and vertical shifts. Identifying these parameters is the goal of this section. Using equation (3.2), the rotation angle and magnifications are identified first [26].

The rotation angle is computed using the dot product and cross product of all the possible combinations of vectors inside each map. Noting $\overrightarrow{u}$ one possible vector in the reference set and $\overrightarrow{v}$ the corresponding vector in the measured map, the dot product is given in (3.3) and the signed norm of the cross product is given in equation (3.4).

$$\overrightarrow{u}.\overrightarrow{v} = \|u\|.\|v\|.\cos(\widehat{\overrightarrow{u},\overrightarrow{v}}) \tag{3.3}$$

$$\|\overrightarrow{u} \times \overrightarrow{v}\| = \|u\|.\|v\|.\sin(\widehat{\overrightarrow{u},\overrightarrow{v}}) \tag{3.4}$$

Dividing the two previous equations and taking the resulting arctangent, gives the signed rotation angle (equation (3.5)).

$$\theta = \arctan\left(\frac{\|\overrightarrow{u} \times \overrightarrow{v}\|}{\overrightarrow{u}.\overrightarrow{v}}\right) \tag{3.5}$$

Since a rotation preserves the relative distances inside a rotated set, there is no need to apply the inverse rotation to the points before extracting the magnification. This is possible when taking the DM as reference but also because the actuators' order has been recorded during the DM mapping. The DM frame is compared to a theoretical reference DM frame so if there is a magnification in any other direction that is not X nor Y, it will result in a projection on the X and Y axes that will be negligible to X and Y magnification. There is a difference in the direction of the magnification but the method to compute the magnification in each direction is identical:

- $x_{distances}$ and $x_{distances_{REF}}$ the distances of all the possible combinations between two actuators lined up horizontally, for the misregistered map and the reference one, respectively (see figure 3.8a),
- $y_{distances}$ and $y_{distances_{REF}}$ the distances of all the possible combinations between two actuators lined up vertically, for the misregistered map and the reference one, respectively (see figure 3.8b).

Applying the formulae in equations 3.6 and 3.7 gives the horizontal and vertical magnifications, respectively. They are the root mean square of the sum of all the squared distances considered in the misregistered map divided by the sum of all the squared distances considered in the reference map.

$$x_{Mag} = \sqrt{\frac{\sum_{k=1}^{n} x_{distances}^{2}(k)}{\sum_{k=1}^{n} x_{distances_{REF}}^{2}(k)}} \tag{3.6}$$

$$y_{Mag} = \sqrt{\frac{\sum_{k=1}^{n} y_{distances}^{2}(k)}{\sum_{k=1}^{n} y_{distances_{REF}}^{2}(k)}} \tag{3.7}$$

Accordingly to equation (3.2), the data (see figure 3.9a) is unrotated and unscaled. The rotation is performed using the center actuator as the pivot point (see figure 3.9b). The shift is the displacement of the center of gravity of the grid. When it is removed, the experimental map (blue) can be superimposed to the reference map (in red, see figure 3.9c).

These registration parameters are included inside the *getRegistrationParameters* function. It takes an interaction matrix as input and outputs the registration parameters in a *registrationParameters* structure. The Matlab code for the function and the structure are presented in Appendix A.2 and A.1.

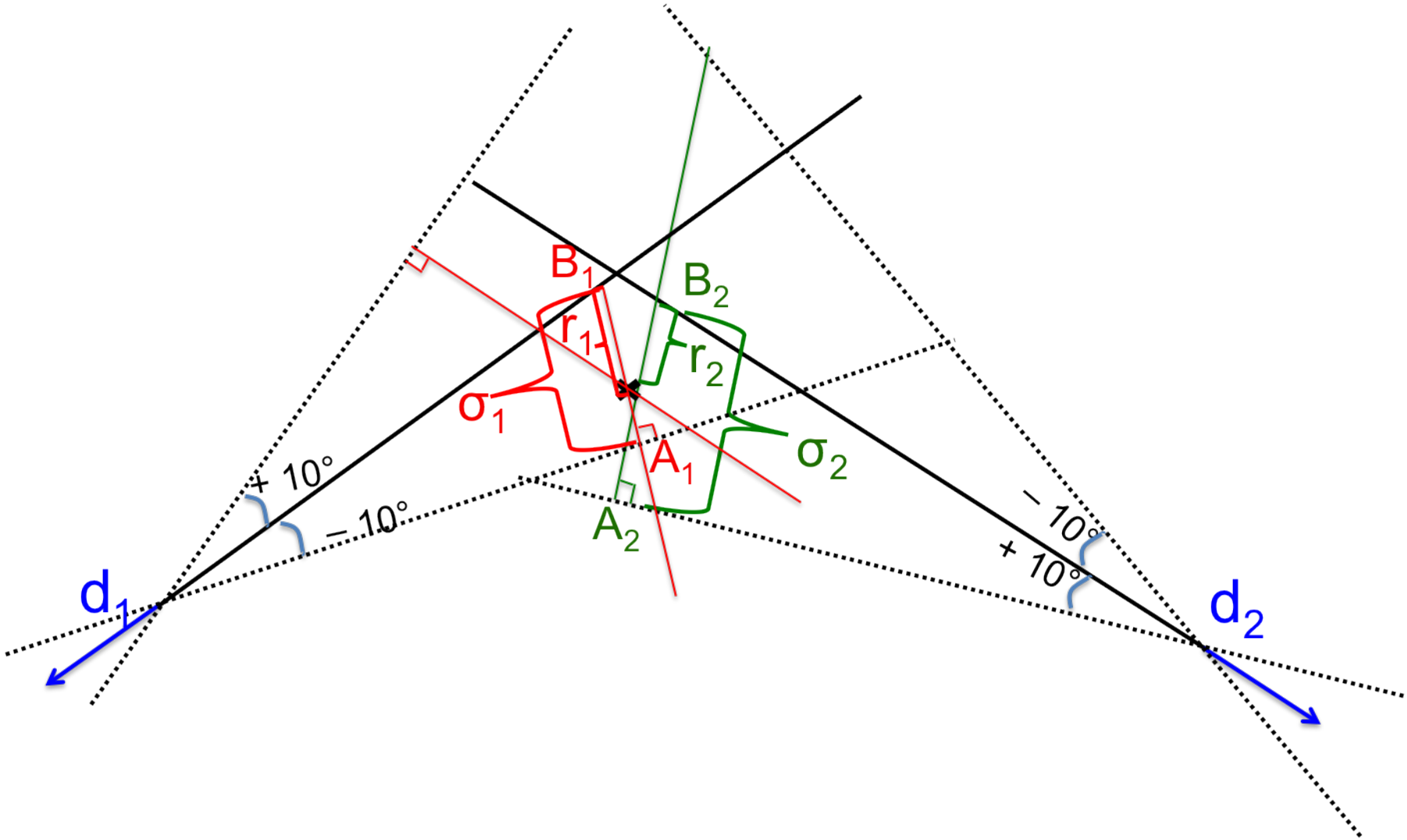
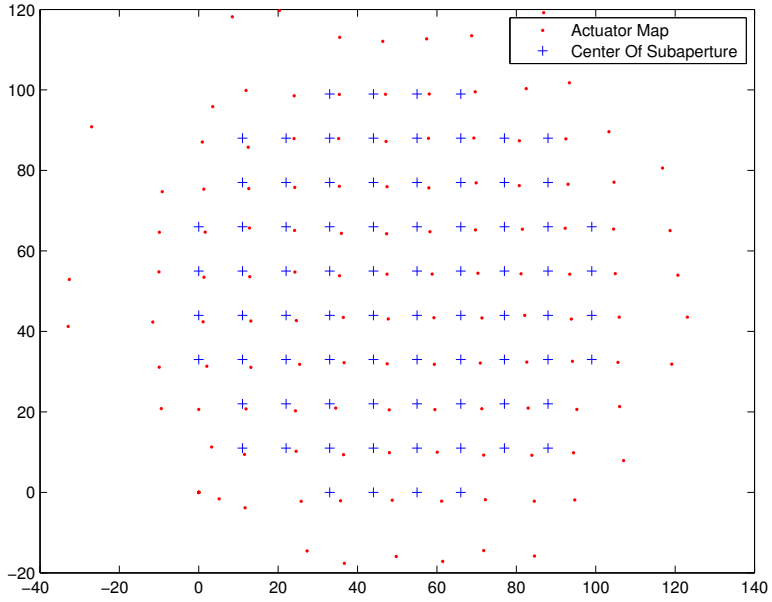
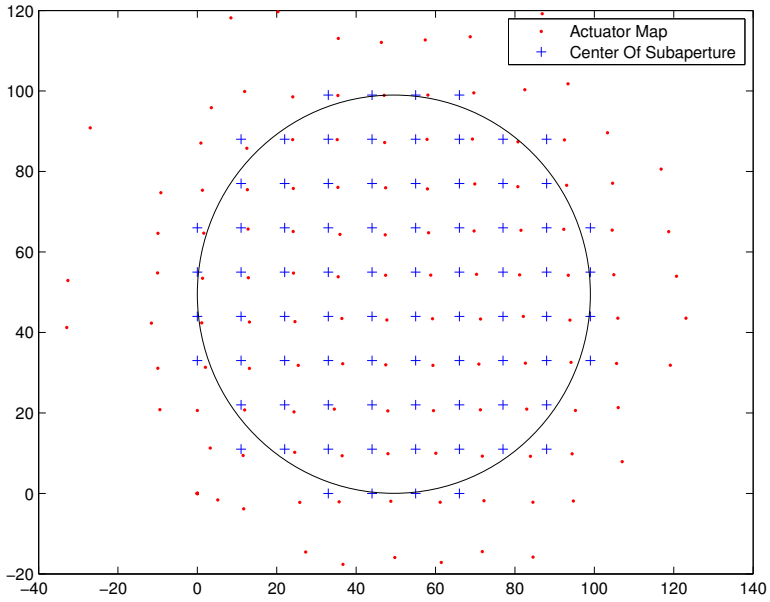


Figure 3.4: Illustration of the optimization function for two slopes. The cross represents the point to optimize.

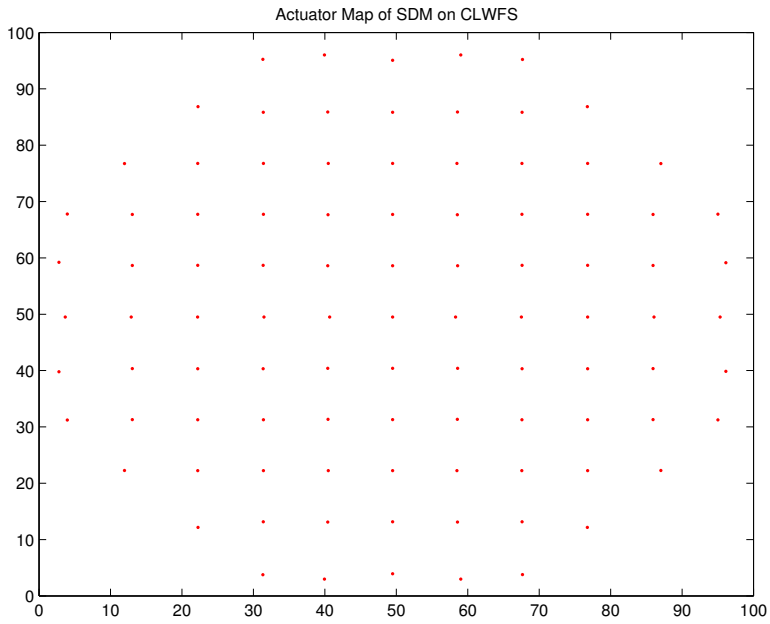


(a) Map of actuators (red dots) compared to the positions of the subaperture centres (blue crosses)

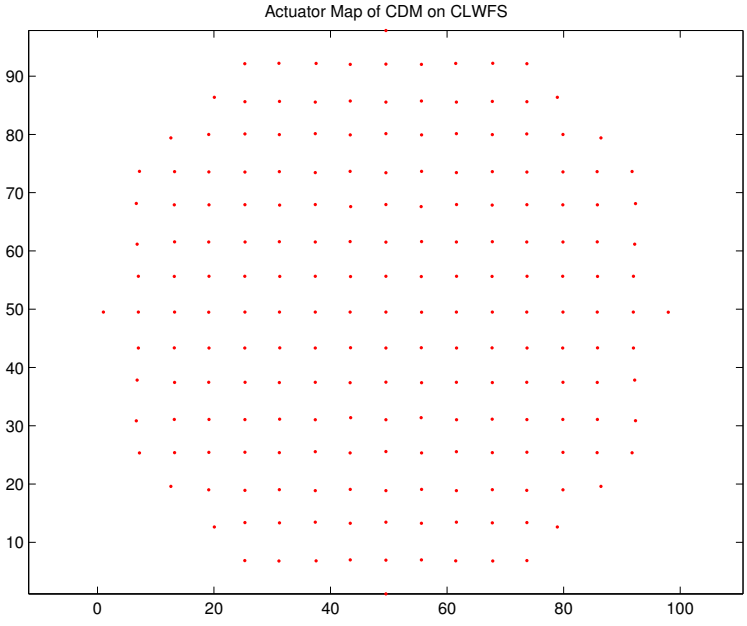


(b) Same as 3.5a but with a filtering disk: actuators positions found outside the circle are discarded in the mapping process.

Figure 3.5: The actuators outside the WFS pupil are hard to locate so they are filtered.



(a) Map of SDM actuators



(b) Map of CDM actuators

Figure 3.6: The actuator mapping is applied to DM with a different number of actuators: 145 for SDM and 277 for CDM.

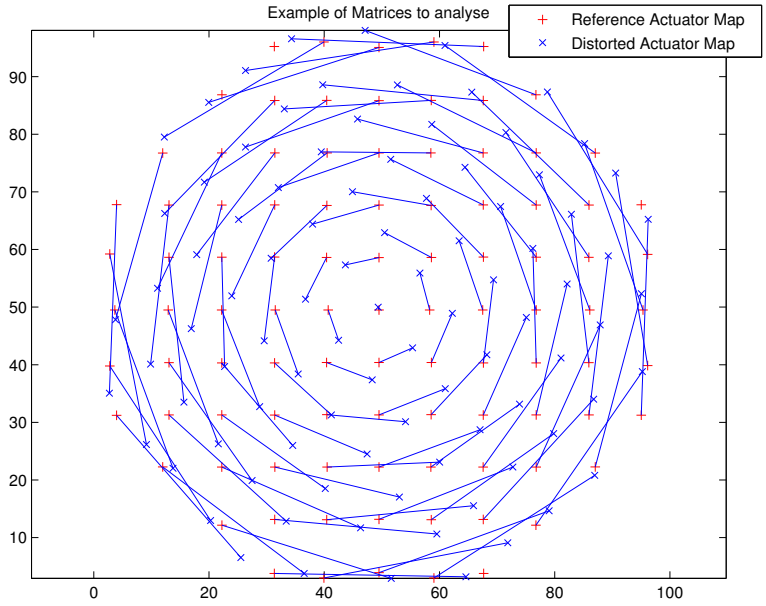
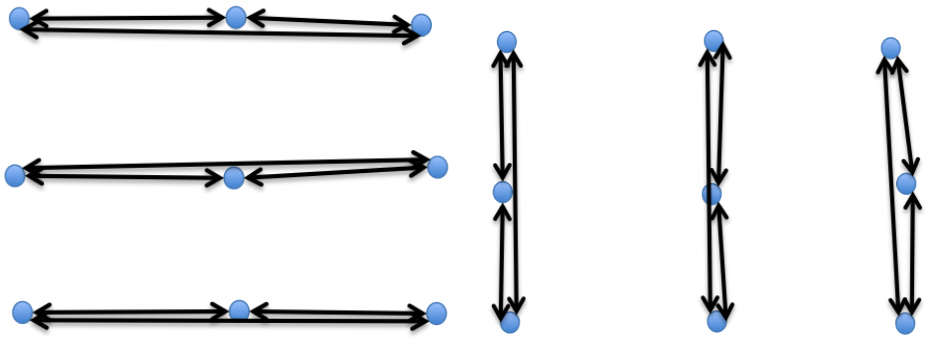


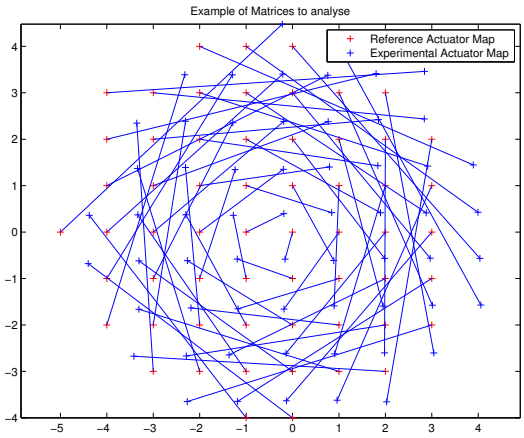
Figure 3.7: Comparison of 2 actuator maps. The red one is the reference. The blue one is rotated by an angle of 40° and shifted by 5 percent of a subaperture on the WFS. The blue lines show the corresponding actuators.



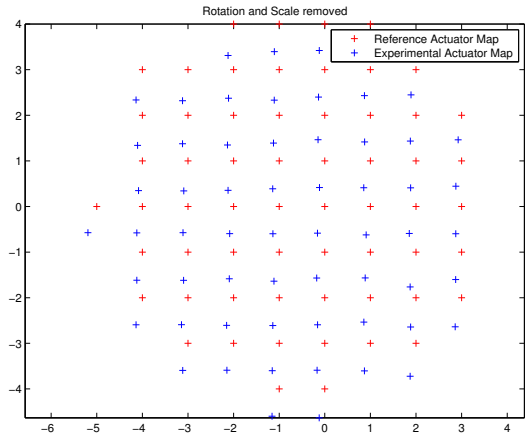
(a) Illustration of the combination of distances for the horizontal magnification

(b) Illustration of the combination of distances for the vertical magnification

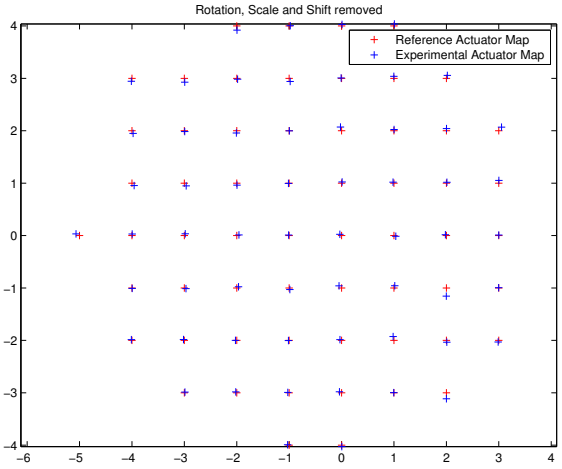
Figure 3.8: Illustration of the combination of distances. The blue dots represent the actuators positions and the arrows represent the distances between all the possible combinations of these points.



(a) Experimental map extracted from a SDM-to-CLWFS interaction matrix (blue crosses) to compare to a reference map (red crosses). The lines indicate the corresponding actuators in each map.



(b) Rotation, scale removed from the data to fit to the reference actuator map before extracting the translation parameter.



(c) Rotation, scale and translation removed from the data to fit to the reference actuator map.

Figure 3.9: Experimental results of the extraction of the registration parameters from a SDM-to-CLWFS interaction matrix.

Chapter 4

Creating Command Matrices with Synthetic Influence Functions

In this chapter, we consider two ways to create synthetic command matrices. The first one uses the interaction matrices described in sections 4.1 and 4.2. The second method, described in section 4.3, is based on the fitting of the DM phase.

4.1 Creating Synthetic Interaction Matrices

Synthetic interaction matrices are created from the registration parameters of experimental matrices and the OOMAO tool. The Object-Oriented Matlab Adaptive Optics (OOMAO [27]) is an extension of the Matlab language. Its library contains a set of classes developed to perform numerical modelling of an AO system and allows the user to propagate a wavefront through the system. This is one of the modelling tools used in the Raven project. The OOMAO simulation tool was not adapted for the generation of synthetic interaction matrices. It had the option to add some misalignments between a DM and a WFS but this was not accurate enough for our purpose. The problem lies in the definition of the phase propagated to the WFS. This is defined by the finite number of pixels on the WFS camera. For Raven modelled with OOMAO, the phase is a matrix of 120x120. Then, when misregistrations are added to the phase, there is a loss of information due to sampling. For example, if a phase needs to be shifted on a vertical or horizontal axis by a fraction of a pixel, interpolations of the new values in the matrix have to be performed. That interpolation adds error when the transformation does not make a pixel correspond to another

pixel of the phase.

Another limitation of that method is that there is no easy way to simulate a DM with actuators outside the pupil. All the DM actuators are defined on the pupil. As a result, for our case of science DMs with 13x13 actuators (with only 11x11 on the pupil), the user has to remember to use the OOMAO model with a 11x11-actuator DM. This does not directly work when the matrices are to be used with the laboratory instrument because of the different size of the matrices.

A different approach is to use OOMAO and simulate the DM poke to the WFS without a DM. The goal is to present the WFS with a series of poked DM actuators in the form of influence functions placed where they would have been if the DM was poked. These influence functions are presented as phase maps. Because OOMAO relies on the phase propagation to compute slopes, this is a realistic method to create a synthetic interaction matrix.

The first step is to compute the DM influence functions as close to reality as possible. To perform this task, we measure the influence functions using a Zygo interferometer. This measurement confirms the manufacturer data: 30% coupling for the CDM and 40% coupling for the SDM. The influence functions also give information on the amplitude corresponding to a normalized manufacturer DM command poke. Then, with that data, an analytic expression of the influence function is computed using a Gaussian function. This function is chosen because it is easy to compute for these tests and should validate the method. This Gaussian then has to be centered on the poked actuator's position as seen by the WFS. I developed a Matlab function, *getSyntheticMatrix*, to compute a synthetic matrix directly from an experimental one (Appendix A.3).

The *getSyntheticMatrix* function use all experimental interaction matrices generated on the Raven instrument and creates synthetic matrices. The size of the interaction matrices is dictated by the DM which is simulated. *getSyntheticMatrix* also uses the influence functions and the number of actuators with their configuration on the mirror. These matrices have the following dimensions: 160x277 for the CDM or 160x145 for the SDM. With this knowledge, we measure the registration parameters to know how the DM is oriented and scaled with respect to the WFS.

The map of valid actuators is created from the DM parameters. The SDM and CDM do not follow the same actuators' placement, nor the same number of actuators. After the grid is created, it has to be scaled, rotated and shifted to appear as the real DM seen on the WFS. The measurements confirm that the CDM has a 30%

coupling and the SDMs have a 40% coupling (figure 4.2). Although the scales from the Zygo interferometer and the synthetic WFS phase are different, due to the higher resolution of the Zygo camera, we can compare the influence functions obtained by both methods.

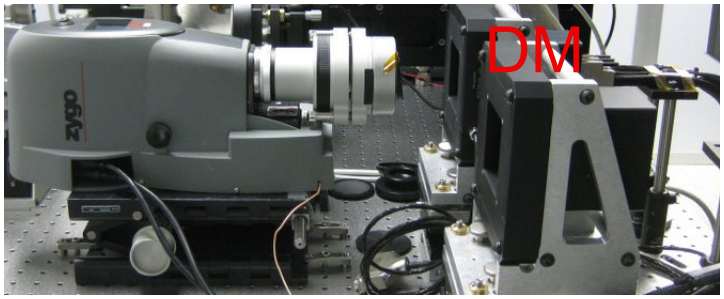
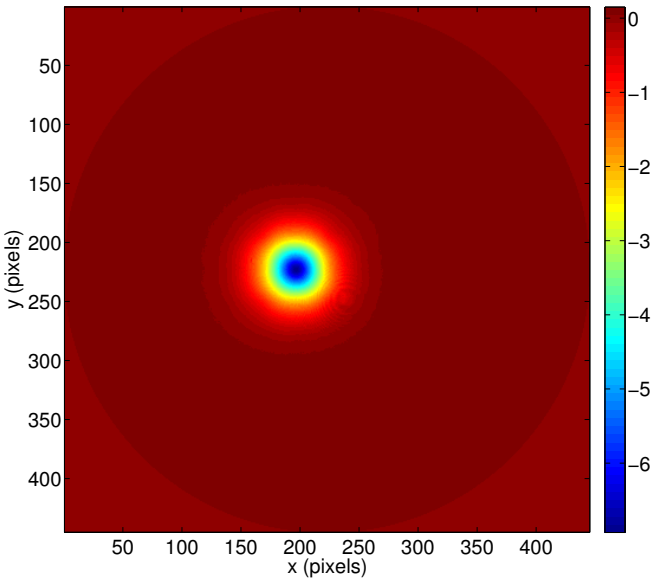


Figure 4.1: Wavefront measurement setup with a Zygo interferometer (left) and a DM (right).

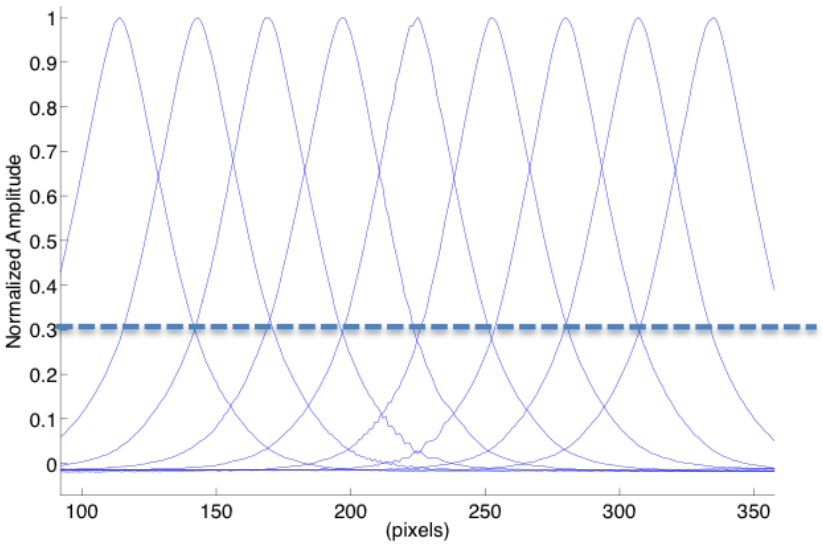
A Gaussian is a simple way to describe an influence function and is fit to validate the method. The coupling and amplitude are taken from the measurements done with the Zygo interferometer. The coupling will affect the width at half maximum. The Gaussian function is defined as follows (eq. 4.1):

$$gaussian(x, y) = Ae^{-(\frac{(x-x_0)^2}{2\sigma_x^2}+\frac{(y-y_0)^2}{2\sigma_y^2})} \tag{4.1}$$

With A the amplitude, x_0 and y_0 the coordinates of the center of the peak, and σ_x and σ_y the spreads of the function. The spread of the function is the same in both directions because the membrane is isotropic, except near the edges. It is set to have the right coupling for either SDM or CDM. The coordinates of the center of the peak are the coordinates of the actuator considered while registering the interaction matrix. We manage to match the coupling but the Gaussian appears to have a broader peak and more narrow wings than the actual influence function. As seen on the graphs, the first intersection with the adjacent influence function for the experimental measurement is 64% while it is 73% for the Gaussian. A different function could be used for a better fit but the Gaussian function is easier to implement, debug and test. The influence functions' amplitudes are computed so they correspond to the actual measured amplitudes. This is done separately for each actuator because the maximum amplitude is not the same, depending on the poked actuator. Edge actuators, for example, have a lower amplitude than other actuators.



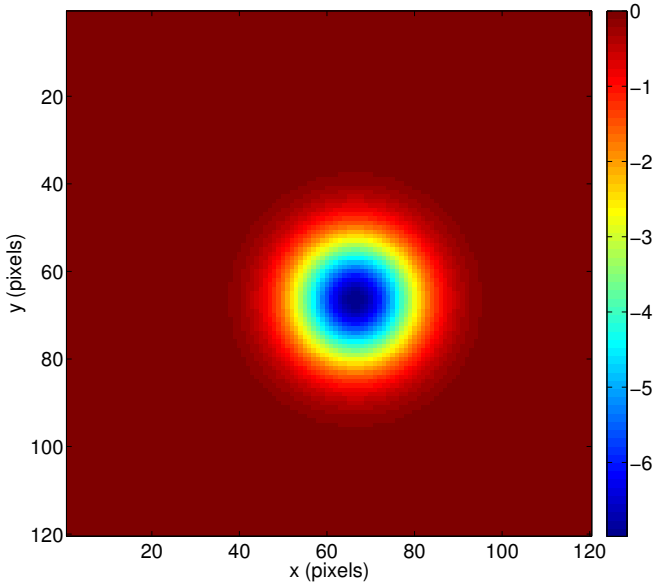
(a) CDM phase on Zygo



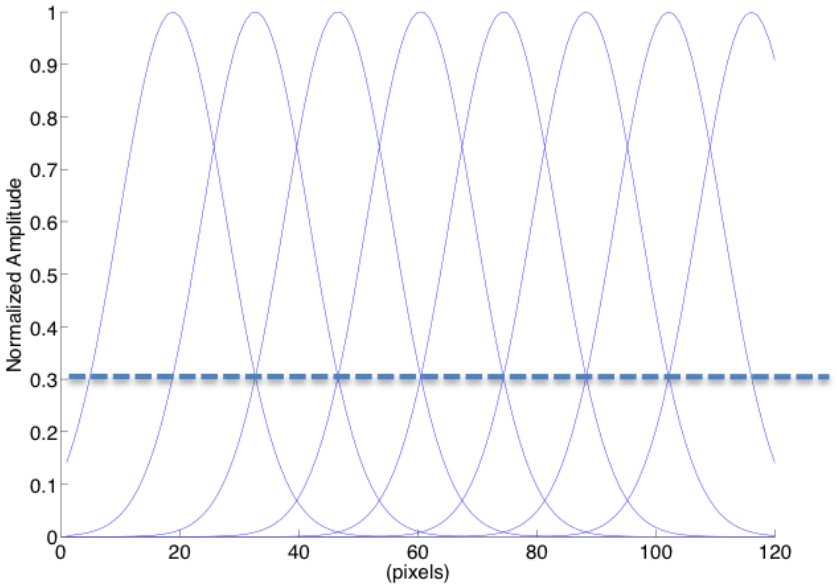
(b) ZYGO CDM influence functions (actuators 135 to 143)

Figure 4.2: (a) Zygo CDM phase and (b) influence functions.

Once the influence functions and the grid of actuators are created, the complete interaction matrix can be modelled. The method presented here uses a basic series of successive actuator pokes while recording the WFS slopes. An interaction matrix



(a) Synthetic CDM Phase generated with Gaussian



(b) Synthetic CDM influence functions (actuators 138 to 145)

Figure 4.3: (a) Synthetic CDM phase and (b) influence functions.

registers the effect of DM commands on a WFS. In this case, the DM is simulated by a series of phase screens propagated from the OOMAO *source* object to the WFS. There are as many phase screens to present to the WFS as there are DM actuators.

Every point on the synthetic grid has to correspond to an actuator on the physical DM and must be correctly oriented and positioned with respect to the WFS. For each phase map presented and propagated to the WFS, the slopes are saved. Slopes measured from successive actuator pushes build interaction matrix column by column.

The creation of synthetic interaction matrices is done with a function that takes the experimental interaction matrix as input and outputs the synthetic matrix. The process gives a synthetic matrix that is qualitatively the same as the experimental one. We still need to validate our results experimentally to show their performance during the correction. Figure 4.4 shows that the resulting interaction matrix is similar to the experimental one. As a result, the next step is to test them in an AO setup.

4.2 Command Matrix Using Singular Value Decomposition

The singular value decomposition (SVD) is a classic method to compute the command matrix using the interaction matrix between the SDM and CLWFS. An interaction matrix presents the conversion matrix between an action on the DM as seen on the WFS. In order to find how to control the DM given the data collected via the WFS, a direct approach is to invert the interaction matrix. However, since the number of DM actuators does not equal the number of WFS subapertures, the interaction matrix is rectangular and a pseudo-inversion is required. This decomposition is performed in Matlab with the *svd()* function. Then singular values are extracted and filtered with a threshold. These singular values are filtered out for better performances. The following code shows how the command matrices are computed :

```
1 [U,S,V] = svd(interactionMatrix); % Singular Value Decomposition
2 singularValues = diag(S); % Collect the singular values
3 nThresholded = 53;  % Number of modes to threshhold
4 iS = diag(1./singularValues(1:end-nThresholded)); % Inversion of ...
     matrix with thresholded singular values
5 [nS,nC] = size(interactionMatrix);
6 iS(nC,nS) = 0; % Fill the matrix with zeros to have the correct ...
     dimensions
7 commandMatrix = V*iS*U';    % Command Matrix
```

Figure 4.5a gives an example of the inversion of the SDM1-to-CLWFS1 interaction matrix. The result is the command matrix (figure 4.5b). The same can be obtained with synthetic matrices (figure 4.6) and the performances of synthetic command matrices are presented in chapter 5.

4.3 Command Matrix Using Phase Fitting

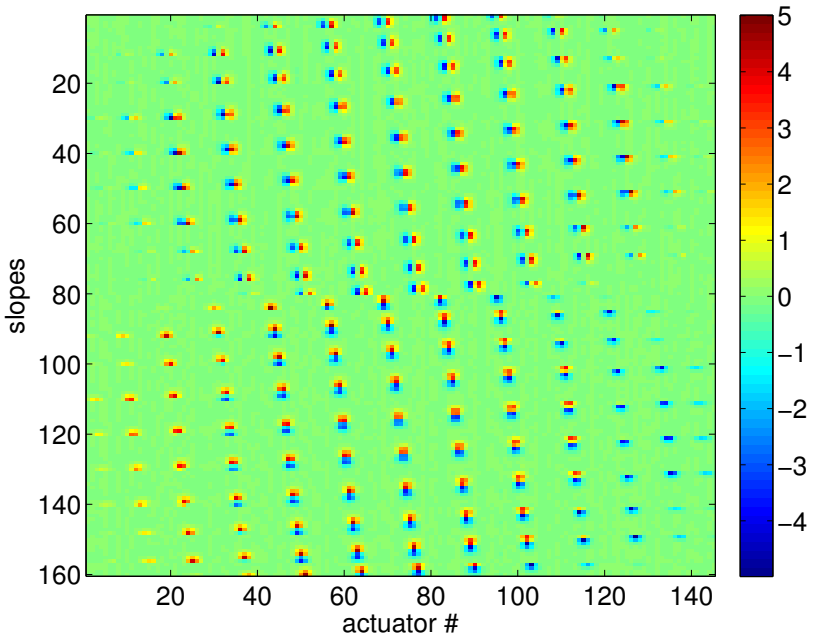
A command matrix generated to fit the distorted phase directly, using the actuators' influence functions, has the advantage of avoiding the singular value decomposition of an interaction matrix.

The creation of the command matrix using a phase fitting method to correct the distorted wavefront is done with the *getCommandMat* function that I developed in Matlab. The phase to correct is expressed into the CLWFS's space. It takes a SDM-to-CLWFS interaction matrix as input and it outputs the command matrix directly as well as the matrix of influence functions that can be used directly in Matlab for phase fitting using the *mldivide* operator on the phase in Matlab. However, the function has to produce the command matrix because the correction process of the Raven sequencer does not handle the *mldivide* Matlab operation needed to fit the phase (the RTC only handles multiplication between matrices). A pseudo-inverse of the influence functions' matrix is then required to perform the phase fitting.

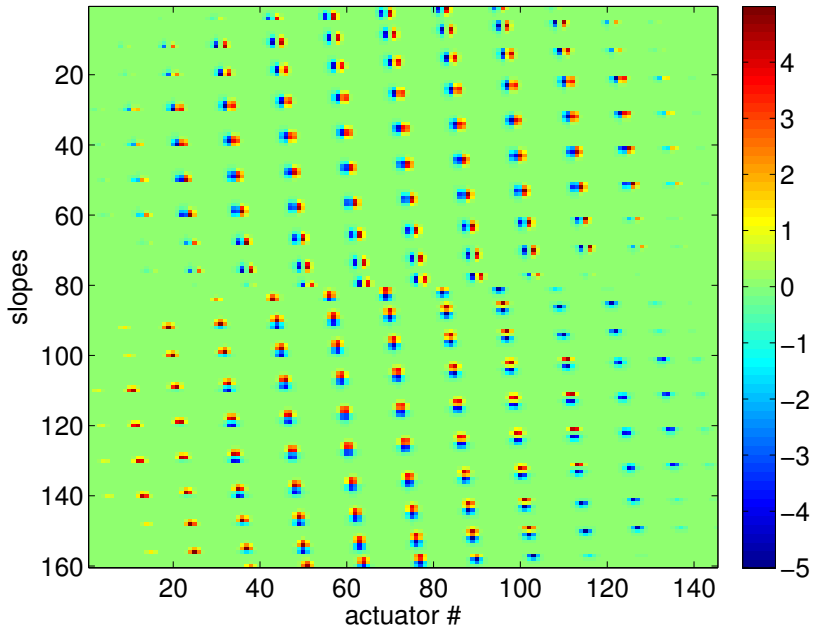
To create the command matrix, a synthetic interaction matrix has to be created first. It follows a similar method as that described in section 4.1 with a few differences. The *getRegistrationParameters* function is used to extract the registration parameters between the SDM and CLWFS. It must be noted that the CLWFSs for each science path are used as references (all the WFS slopes are expressed in CLWFS space). The registration parameters position the influence functions as they would appear on the CLWFS. The set of influence functions will recreate the fitted phase to apply the correction.

The influence functions are also created with a Gaussian function that respects the actuator coupling of the DMs and the amplitude. Again, the influence functions are centered on the actuators. Then the influence functions for each actuators are stored in a matrix where each column of the matrix represents the expression of the actuators' influence functions from 1 to 145 (there are 145 actuators on the SDM). A pseudo-inversion of this matrix will give the phase fitting command matrix. An example of the phase fitting is given in figure 4.7. On the left, there is the non-

corrected phase simulated via OOMAO. This phase is fitted with the DM using a synthetic command matrix (middle) and the residue is presented on the right. This method has not been tested at the time of writing this thesis because the data would need to be in the phase space rather than in the current slope space.

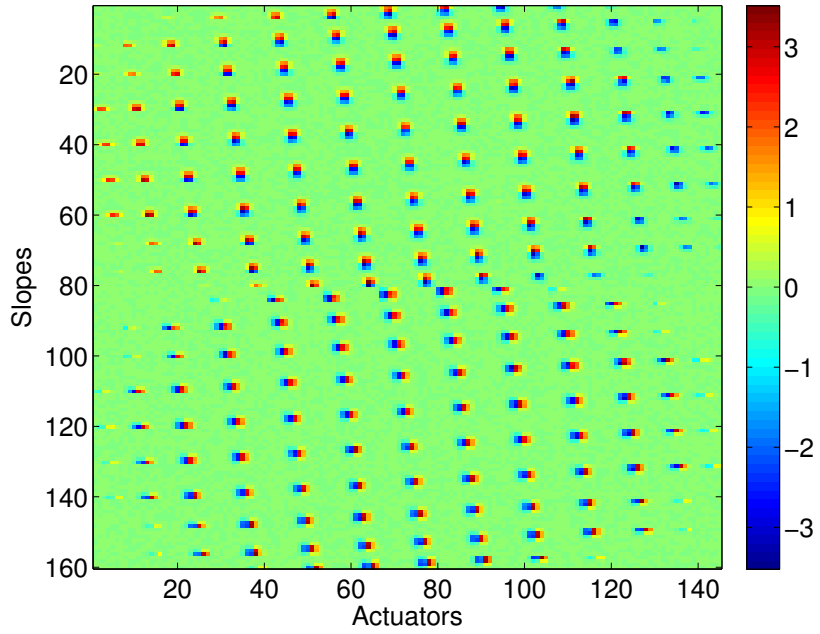


(a) Experimental Interaction Matrix SDM2toCLWFS2.

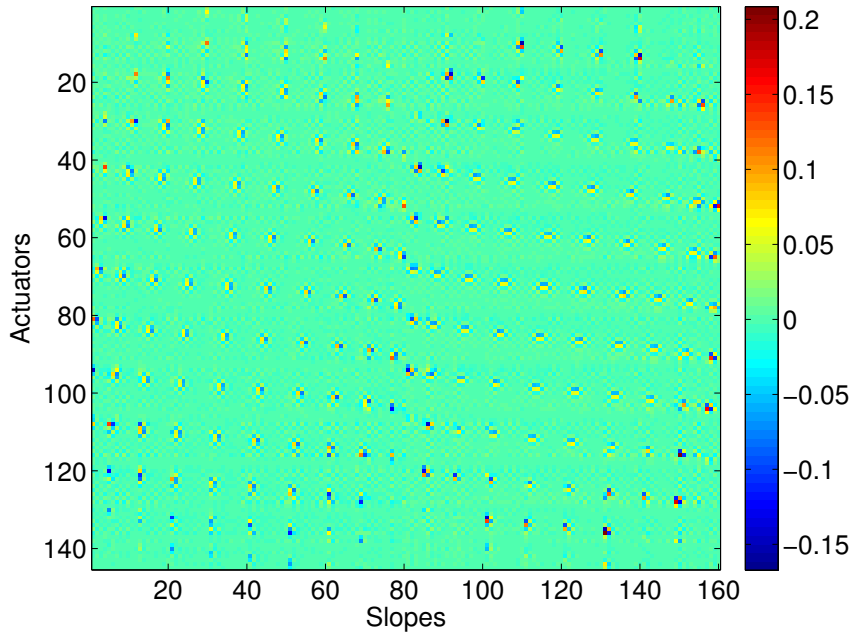


(b) Synthetic Interaction Matrix SDM2toCLWFS2.

Figure 4.4: (a) Experimental and (b) synthetic SDM2-to-CLWFS2 interaction matrices.

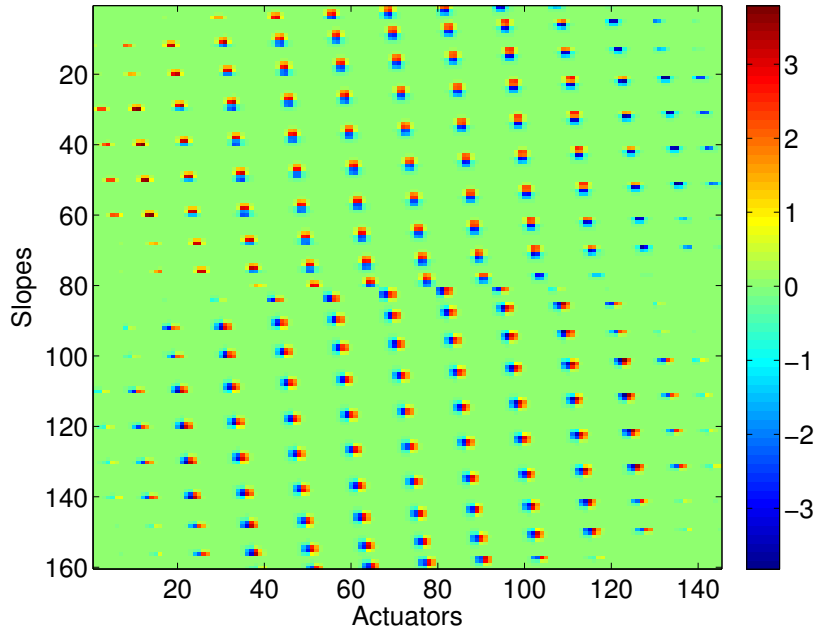


(a) Experimental interaction matrix SDM1-CLWFS1

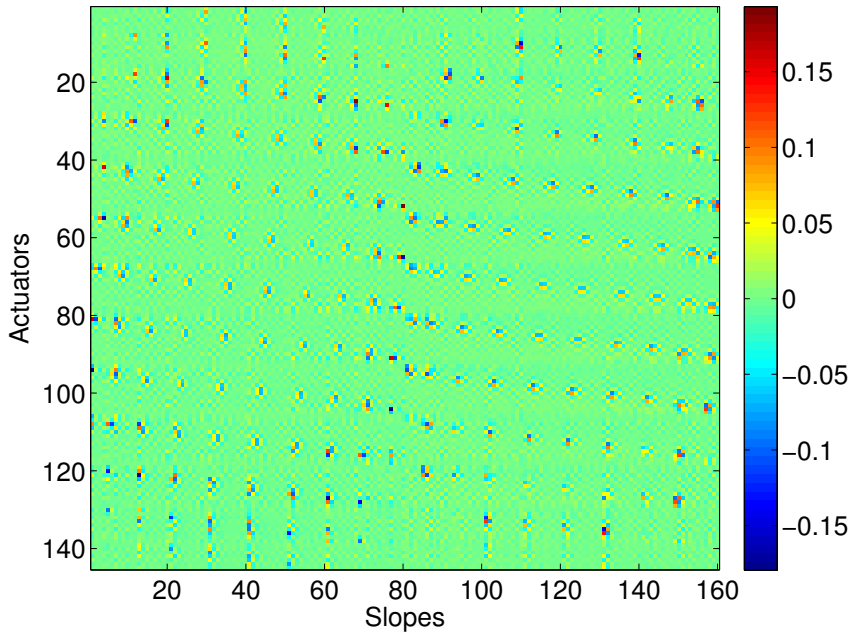


(b) Experimental command matrix obtained after SVD.

Figure 4.5: (a) Experimental interaction and (b) command matrices.

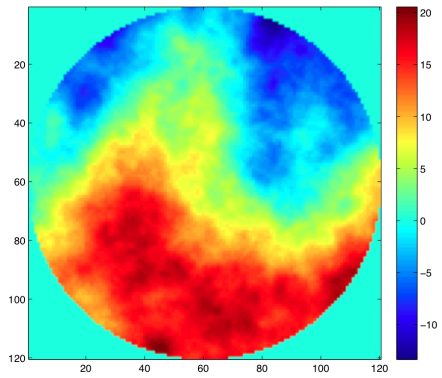


(a) Synthetic interaction matrix SDM1-CLWFS1

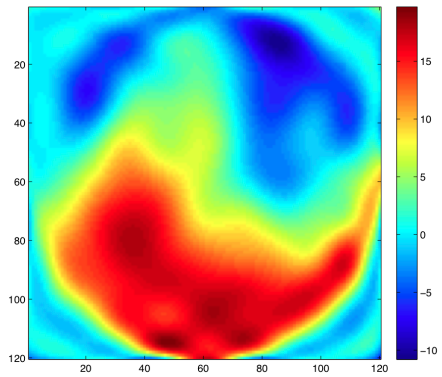


(b) Synthetic command matrix obtained after SVD.

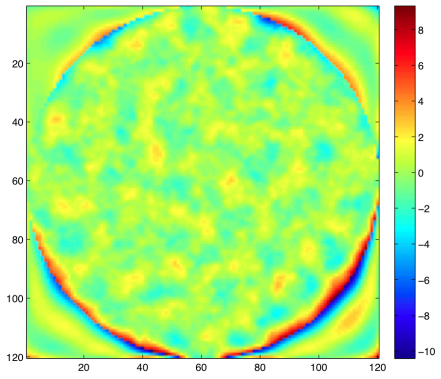
Figure 4.6: (a) Experimental interaction and (b) command matrices.



(a) Aberrated phase simulated on OOMAO.



(b) Phase fitted with the synthetic command matrix.



(c) Residue after correction.

Figure 4.7: Phase fitting.

Chapter 5

Experiments

5.1 Calibration Unit Performance Tests

5.1.1 Calibration Deformable Mirror tests

Before aligning and validating the performance of the Raven AO system, the calibration unit (CU) was studied to expand the knowledge of the CU behaviour beyond the expected requirements [28]. Here, we describe tests of the calibration DM and the source simulating the NGS.

The setup of the tests was as follows: the Zygo interferometer was positioned in front of the DM that was fixed to the table. The five cables sending commands to the DM were attached to a post so they would not pull on its back and change its position during testing. The Zygo interferometer can be controlled remotely through the network (Figure 5.1). Since the Zygo computer cannot be used for anything besides controlling the interferometer and getting data from it, a second computer (Master-PC) is connected to the DM and can access the Zygo-PC through the network. The setup is controlled under Matlab from the Master-PC dedicated to do the tests, collect and analyse data.

Thermal Stability

At the start of the tests, the DM was initialized to a DM best flat position, then measurements were taken throughout a night and the DM shape was analysed. From the beginning to the end of the test, the DM did not receive any new command. After launching the test, there was no one in the room to perturb the measurements during

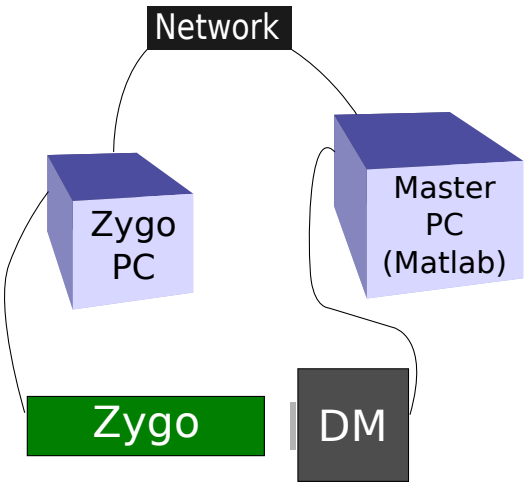


Figure 5.1: Set-up of the Zygo interferometer in front of the DM.

16 hours, so perturbations only came from changes in temperature (see figure 5.2). The tip and tilt (blue line) were caused by the misalignment between the DM and the Zygo interferometer. We removed it to isolate the DM stability measurement from the mechanical alignment of the DM with the interferometer (red line). There is a drastic increase of the wavefront error during the first 20 minutes. Then this wavefront residue stays stable during 15 hours. We see that there is a warm-up period of about 20 minutes. That means that the DM needs time to reach an equilibrium. The 10 nm RMS found on the stability of the system is very good and meets manufacturer's datasheet.

Repeatability Tests

Repeatability tests reflect how closely the DM shape remains the same after receiving repeated commands during a night. These repeatability measurements alternate 11 phase screens (with 1 best flat) in a continuous loop (figure 5.3). There were 10 measurements per hour (every 6 minutes). When no measurements were taken, the DM is changing shape without using the Zygo to get data. That makes the DM stay warm and it simulates the DM normal use when different commands are applied throughout a night of observations to correct atmosphere turbulence. The tip/tilt data includes information about the misalignment between the DM and the Zygo interferometer. Since we are not interested in that information, we removed the tip/tilt information from the data. The repeatability was less than 10 nm RMS after

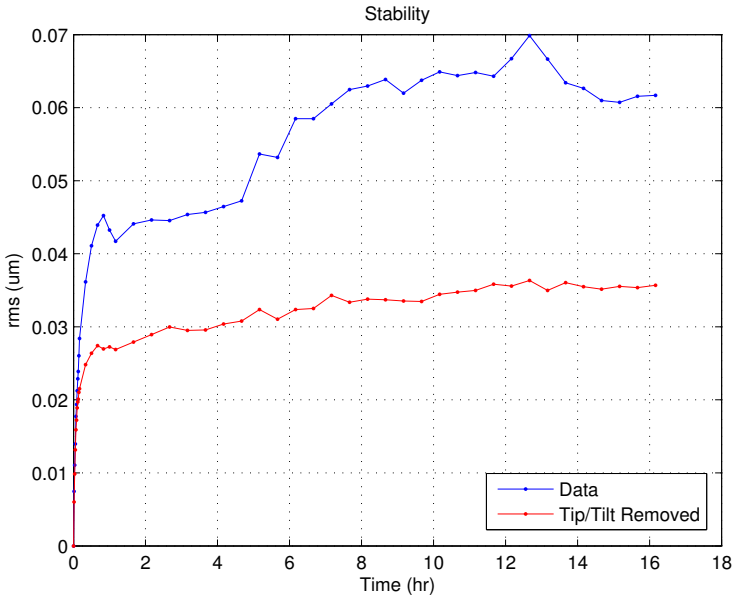


Figure 5.2: Stability test with DM commands sent to get the best flat. Measurements taken during 16 hours.

the first hour of warm-up.When Raven will be used on-sky, the DMs will require warm-up time before system calibrations are made.

5.1.2 Natural Guide Star Source

We wanted to verify the linearity of the NGS source as well as the transmission of the neutral density filters placed in front of it (5.1). A photodiode was placed in front of the output of the CU around the focal point of the central spot. The seeing-limited pinholes on the CU were selected. On the photodiode, a wavelength was chosen for the test and set to 635 nm. The source was set to 100 W at first and the filter wheel was set on the first position. The measurements were taken from the photodiode. Then we turned the filter wheel to position 2 and repeated the operation for each filter. The photodiode had trouble detecting light when using filters 4 to 6. When all the power values were taken, we decreased the source power by 10 W and repeated the measurements. The minimum value of the source is 40 W. Below that value, the halogen lamp lifetime would be reduced. The results are shown in table 5.1. Figure 5.4 shows that the source is linear as expected.

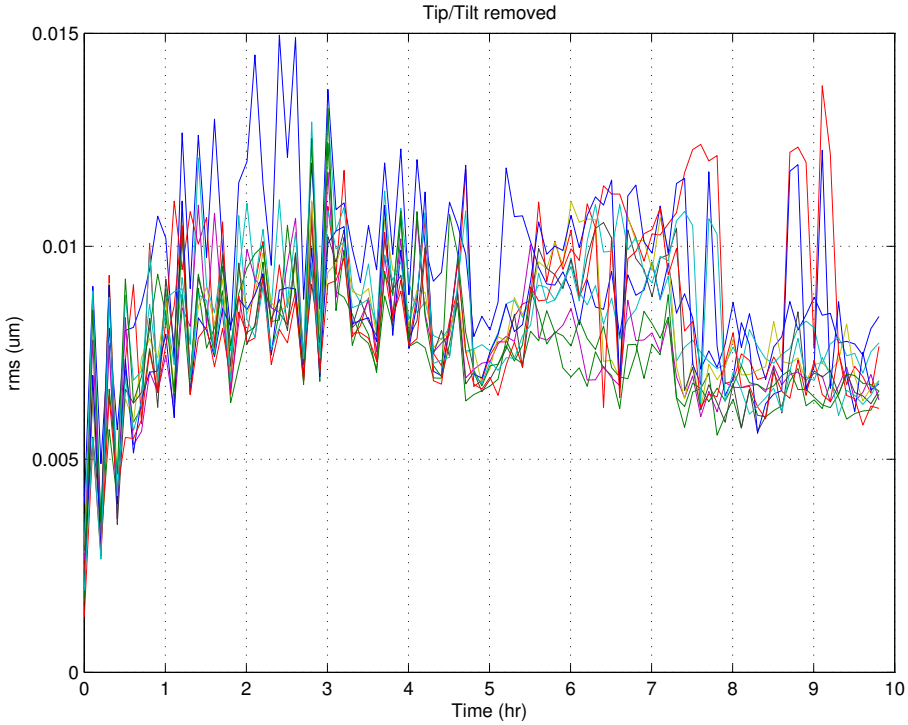


Figure 5.3: Repeatability measurements in function of the time for the 11 phase screens. The tip/tilt is removed so that only the DM error would appear.

Filter\Source (W)	100	90	80	70	60	50	40
1 (nW)	3.60	3.14	2.66	2.20	1.79	1.33	0.91
2 (nW)	1.14	0.98	0.85	0.68	0.55	0.42	0.29
3 (nW)	0.36	0.32	0.26	0.20	0.16	0.12	0.06
4 (nW)	0.04	0.03	0.03	0.02	0.01	0.00	0.00

Table 5.1: Values read on the photodiode placed in front of the CU output.

5.1.3 Neutral Density Filters

Neutral density filters are located in the filter wheel at the NGS source input of the CU. There are 6 of them that can be chosen. The maximum power for the NGS source is 100W and its light is emitted through the calibration unit. We did not place the phase screens in the beam. An Andor camera is used for the tests because it provides more sensitivity than the photodiode that we used for the linearity tests. The camera was cooled to $-40\,^{\circ}\mathrm{C}$ for better performance (low noise). The exposure

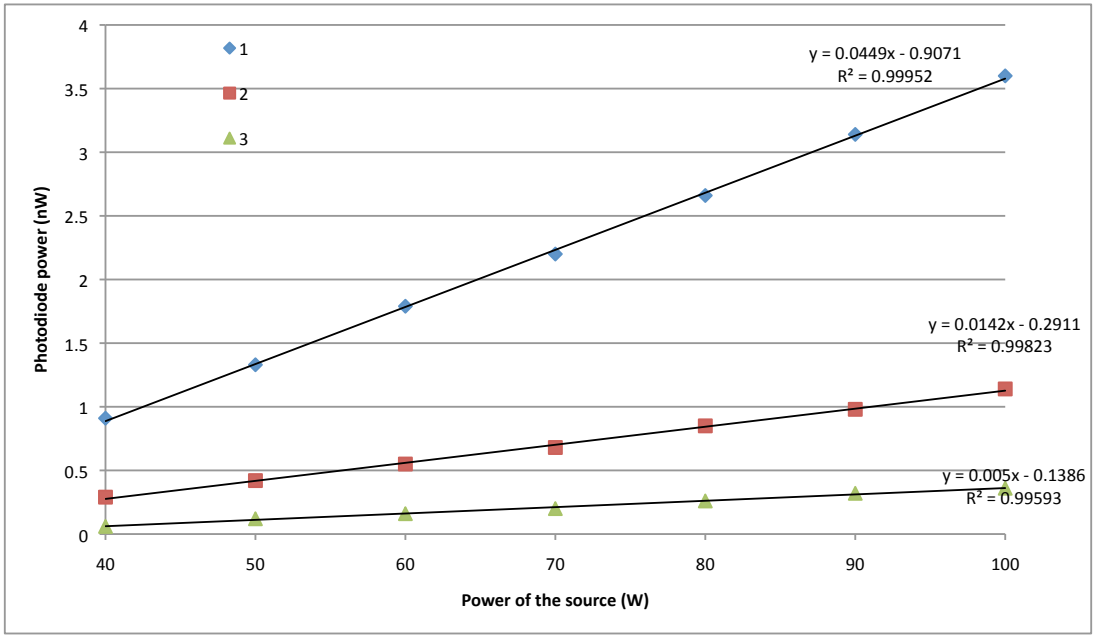


Figure 5.4: Photodiode power in function of the source power for the first 3 filters.

for each image was set up so that the maximum was at midrange (see table 5.2 and figure 5.5). A background image was taken to subtract from the main image and the flat field correction has been applied. Near the top-right corner, a ghost was visible.

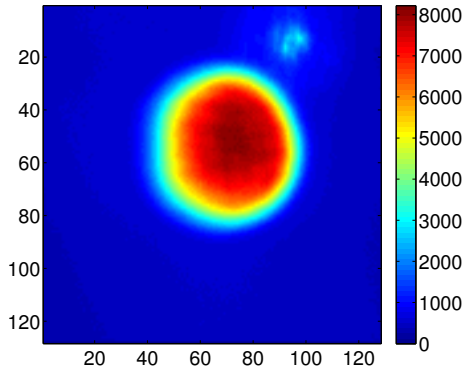
Table 5.3 presents the optical densities (OD) found by comparing the values of the images of the first filter to the other ones (equation 5.1). The sum line indicates that the value used are the sum of all the pixels in the images. The max line is the same method with the maximum of the frames.

Filter ♯	1	2	3	4	5	6
Exposure (sec)	0.004	0.01	0.04	0.2	1	8

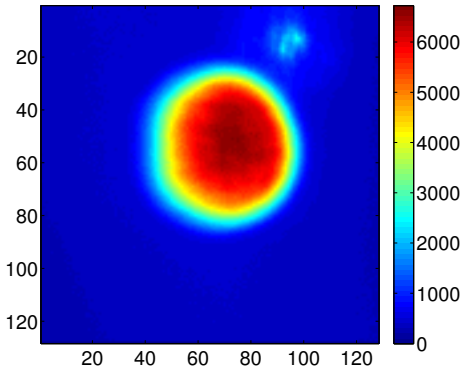
Table 5.2: Exposure times for each filter.

Filter1/Filter ♯	2	3	4	5	6
With sum	0.4893	0.9756	1.6979	2.3856	3.2732
With max	0.4771	0.9542	1.7076	2.4048	3.2982

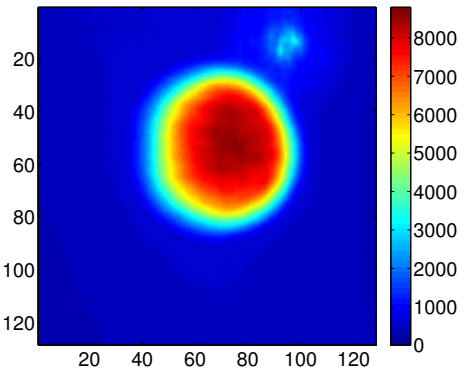
Table 5.3: Density filters test results.



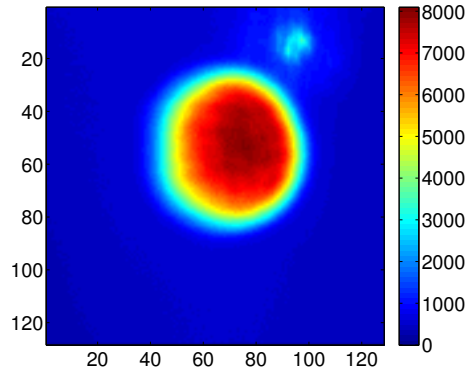
(a) Filter 1, exposure 0.004s



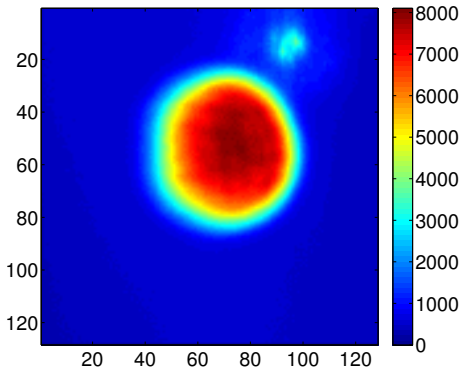
(b) Filter 2, exposure 0.01s



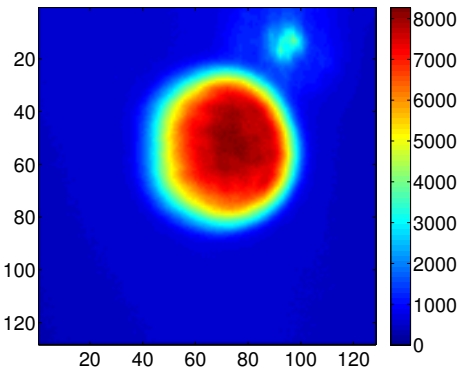
(c) Filter 3, exposure 0.04s



(d) Filter 4, exposure 0.2s



(e) Filter 5, exposure 1s



(f) Filter 6, exposure 8s

Figure 5.5: Images from the NGS source through each filter taken with an Andor camera. The exposure times have been chosen to use the full dynamic range of the camera. The images are defocused to use more pixels for a better averaged result and to avoid having a focused point on potential dead pixels.

$$OD = \frac{\frac{sum1}{exposure1}}{\frac{sum2}{exposure2}} \tag{5.1}$$

5.2 Aligning Science Deformable Mirrors and Open-Loop Wavefront Sensors

The tools we developed in chapter 4 can be applied for aligning the DMs to the WFSs. Thanks to the knowledge of the position of the actuators seen by the WFS, the user can precisely align them with the subapertures. It is computationally inefficient to process all the WFS image data, for each DM actuator push-pull, in order to determine the actuator map from the interaction matrix. The algorithm has been adapted so the misregistration can be estimated with a few commands to a subset of selected actuators. After applying a DM command, an image from the WFS is recorded and information about the actuators' position is then extracted. This is an efficient method that allows alignment in real-time. However, this method is not as precise as the method employing all actuators for generating synthetic interaction matrices.

Application of the method requires that the actuators be spatially distant. These separate zones ensure that the effects of two distinct actuators do not overlap.We use a zone search algorithm to isolate the actuators' effect then the code based on the method described in section 3.1 is applied. The data input is a WFS frame which is equivalent to one interaction matrix column with three poked actuators at the same time instead of one. The goal is to be able to separate the zones corresponding to each actuator.

A threshold sets slopes unaffected by a poke to zero. A map of useful subapertures is then created (figure 5.6). The white cells represent the "1" and the black cells represent the "0". The zones with adjacent "1" cells represent the slopes to consider for the mapping algorithm. If these zones can be identified as in figures 5.7a, 5.7b and 5.7c, the mapping will work as if there was an interaction matrix with three columns, each column being the product of these masks ($Mask(\alpha)$) with the WFS frame measured for this application. In figures 5.7b and 5.7c, there are holes in the middle of useful sections because the actuator's position is in the center of the subaperture, thus the influence function's tangent at the maximum is seen as flat.

The following steps are applied to identify each zones :

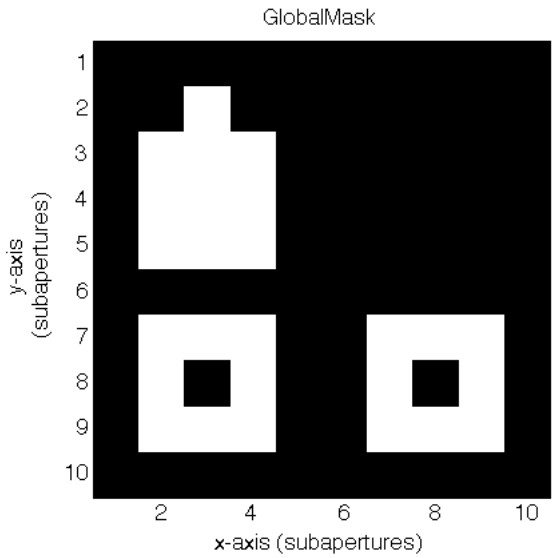
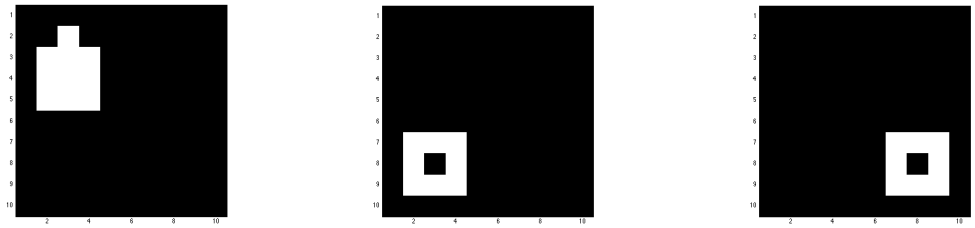


Figure 5.6: GlobalMask: Map of useful subapertures. A white cell corresponds to 1, a black cell to 0.

1. Set the mask counter $\alpha = 1$.
2. Set *GlobalMask* as the map of considered slopes (i.e. non-thresholded).
3. Scan *GlobalMask* until the first "1" is encountered.
   (a) Save its position in $Mask(\alpha)$.
   (b) Check all its adjacent cells in *GlobalMask* for a "1", if there are, save their positions in $Mask(\alpha)$.
   (c) Move to the next "1" in $Mask(\alpha)$.
   (d) Repeat steps 3b and 3c until there are no more adjacent "1" cells.
   (e) If $\alpha <$ number of poked actuators, do $\alpha = \alpha + 1$, subtract $Mask(\alpha)$ to *GlobalMask* and go to 3, otherwise end the process.

When the masks are created (see figure 5.7), multiplying one of them with the WFS frame and applying the actuator mapping function gives the position of the actuator. This has to be repeated as many times as there were poked actuators.

In figure 5.8, three actuators are poked (yellow crosses) and are displayed on the WFS camera. The non-symmetrical triangle shape helps to determine the horizontal and vertical axes of the DM. The position of the central actuator is found from the



(a) Mask 1 *(b) Mask 2* *(c) Mask 3*

Figure 5.7: Illustration of the 3 masks isolating the zones in the WFS frame for each poked actuator. A white cell corresponds to 1, a black cell to 0.

position of the three poked actuators. The user aligns the system by superimposing the yellow circle with the red cross. Raven was not designed to make use of a Fried geometry between the Calibration DM and the WFSs. This design choice was made so that the Calibration DM could introduce a higher spatial order disturbance when used to simulate ground layer turbulence. The three yellow crosses therefore cannot be aligned with the corners of the subapertures. Utilizing the DM actuator position, a more precise alignment procedure is now possible.

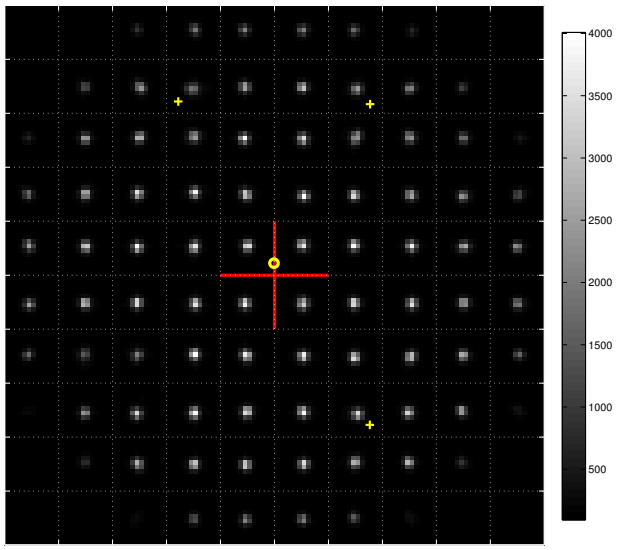


Figure 5.8: WFS display with 3 actuators (yellow crosses) for alignment. The red cross is the center of the WFS. The yellow circle is the deducted center of the DM found using the 3 other actuator measurements.

Once the DMs are aligned with all the WFSs and the interaction matrices recorded,

the registration parameters of the DMs to all WFSs can be computed. They give information on how well the system is aligned and also on the pupil size on the WFS. The reference chosen is a theoretical DM with a Fried configuration so the scale should be of 0.6 subaperture per actuators for the CDM and 1 subaperture per actuators for the SDM. These numbers come from the Raven design. The SDM respects the Fried configuration so there is one actuator per subaperture. Results for the CDM are presented in table 5.4 and results for the SDM are presented in table 5.5. The scales (or magnifications) have been adjusted to fit to the theoretical values through alignment. The Fried geometry is attained with a relative error of 6% for the SDMs' scales due to the 15 ° angle of incidence. The horizontal and vertical displacements are almost perfect with an alignment error of less than 2% of a subaperture. We discovered the CDM-to-CLWFS1, CDM-to-OLWFS2 and CDM-to-OLWFS3 alignment could be fine tuned. These data also show that the angles between the DMs and WFSs are close to the expected design with an absolute error of less than 2 ° to take into account.

DM	WFS	Angle (deg)	xScale	yScale	xShift (subap)	yShift (subap)
CDM	OLWFS1	310.411	0.60	0.60	0.02	-0.01
CDM	OLWFS2	179.874	0.60	0.61	0.02	0.19
CDM	OLWFS3	231.738	0.60	0.60	0.09	0.08
CDM	CLWFS1	270.997	0.68	0.66	0.13	0.16
CDM	CLWFS2	270.052	0.62	0.61	-0.01	-0.02

Table 5.4: Registration parameters between CDM and all WFS. The scale unit is subaperture per actuator pitch.

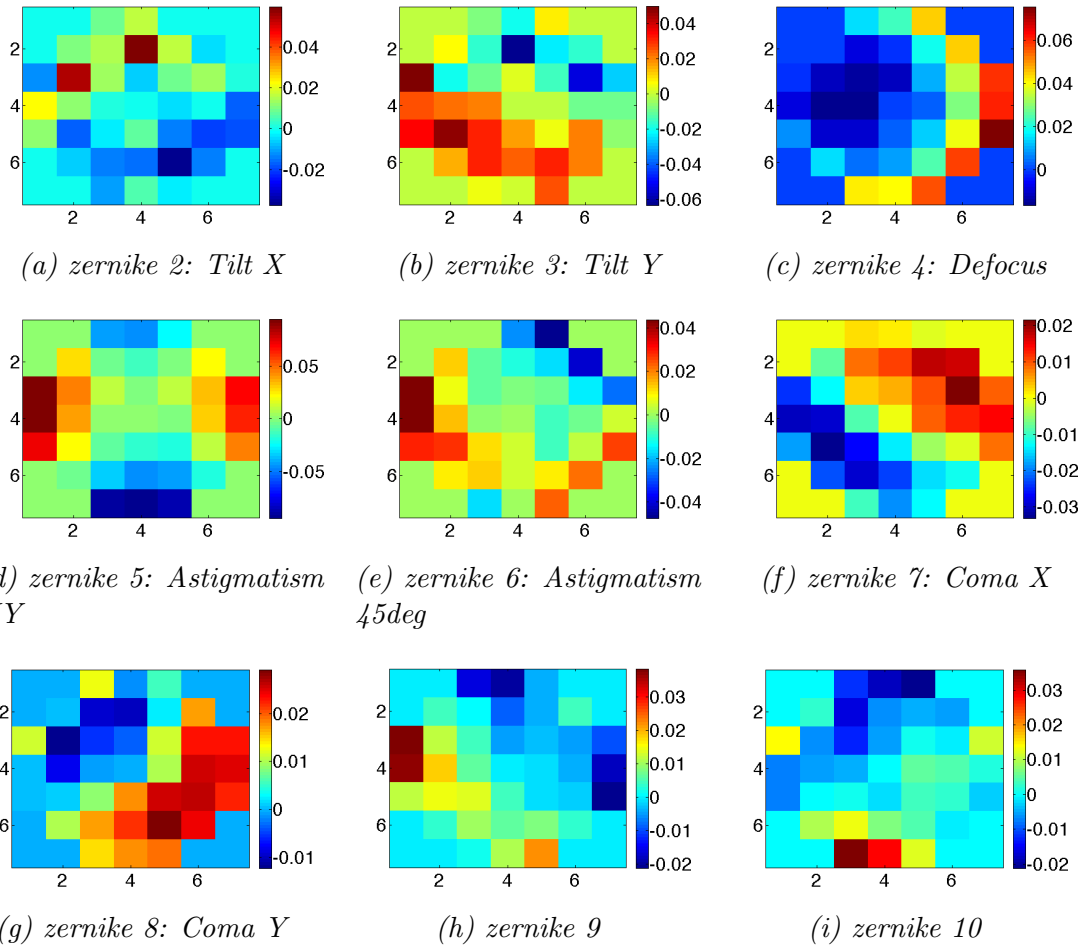
DM	WFS	Angle (deg)	xScale	yScale	xShift (subap)	yShift (subap)
SDM	CLWFS1	270.304	1.06	1.01	0.02	0.02
SDM	CLWFS2	270.224	0.96	0.95	0.04	0.02

Table 5.5: Registration parameters between CDM and CLWFS. The scale unit is subaperture per actuator pitch.

5.3 Non-Common Path Aberrations

Non-common path aberrations are calibrated by setting a slope offset to all WFSs so the slopes are centred during calibration. These NCPAs are measured for each

possible WFS pick-off position of the CU (for each pinhole). Since the measurements are relative to the center, the aberration measured should be the same for all WFS at a given position within the field-of-view. Figure 5.9 represents a map of the Zernike coefficients for each pinhole position of the calibration unit. The center pinhole has a null coefficient because it is the reference: the aberrations are measured relatively to the center of the telescope. This is an illustration of what the Zernike modes look like when the slopes are projected onto the Zernike polynomials. Once these aberrations are measured, they can be subtracted from the slope values as an offset. These slope offsets change as a function of the position of the pick-offs in the field of view so they have to be updated during observations.



(a) zernike 2: Tilt X *(b) zernike 3: Tilt Y* *(c) zernike 4: Defocus*

(d) zernike 5: Astigmatism XY *(e) zernike 6: Astigmatism 45deg* *(f) zernike 7: Coma X*

(g) zernike 8: Coma Y *(h) zernike 9* *(i) zernike 10*

Figure 5.9: The Zernike coefficients are presented on maps of pinhole positions from the calibration unit.

5.4 Synthetic Interaction Matrices Results

5.4.1 Open-Loop Configuration Setup

The accuracy of the synthetic interaction matrices is measured using an open-loop configuration. The calibration unit can act as a telescope simulator and provide the guide stars, science objects and turbulence. The CDM can create a ground layer turbulence. It is important to note that this is a ground-layer perturbation so all the WFS see the same perturbation. This allows the pick-offs to be placed at different positions in the field of view and do the measurement in parallel.

Each Raven pick-off arm was placed on a diffraction-limited spot from the calibration unit to be closer to the final configuration. The diffraction-limited spots simulate the stars to be observed. All the interaction matrices are then recorded (except for the LGSWFS that was not used at the time of these tests). An OLWFS was used to measure the turbulence and its slopes are used to compute the correction to apply to one of the science DMs. Finally, the CLWFS associated with the science DM gave the amount of residual wavefront error after correction. We use this data to evaluate the performance of the synthetic matrices.

Raven is operated using Matlab commands. We first initialize the Raven instrument and give control to the user. It turns the lights on to simulate the stars and initialize the drivers. The Matlab object *raven* is controlled by the *user* to perform the calibration process.

```
% Initialization of the Raven instrument
raven = lib.drivers.rtcRaven([1 1 1 1 1 1]) % Start Raven
me = lib.user.user(raven)   % Start User
```

The following instructions set the instrument for the measurements of the interaction matrices. They reset all the slopes and commands to the factory or default values, center the pupils on the WFS and cancel the static errors between the WFSs (aberrations).

```
raven.resetAll
me.resetAllSlopeOffsets(raven,'factory',0)
raven.centerAll
```

Then the interaction matrices are measured: the DMs can use the Hadamard method [29] or a sine method. These methods have been tested experimentally and have proven to work well with the chosen parameters. The Hadamard method is based on the Hadamard matrices which are matrices with orthogonal rows of -1 and $+1$. These matrices, used to send commands to the DM, produce interaction matrices with less noise than the usual poke method. The zonal poke method only uses one actuator at a time so a lot of potential information is lost. For example, the subapertures far enough from the poked actuator only have a noise signal. The Hadamard method makes sure that all the slopes are effectively used.

A sine method is used to compute the SDM-to-CLWFS interaction matrices. The method consists of sending a sine-wave poke with a different frequency on each actuator. Knowing the frequency corresponding to each poke, the displacements of the slopes can be identified and analysed to compute the interaction matrix. The results are interaction matrices with a form that is independent of the method chosen to compute them. The values of 0.01 for the CDM and 0.03 for the SDM are the voltage commands sent to the actuators to record the interaction matrices. They have been chosen experimentally to give good interaction matrices. The poke displacement is comparable with the turbulence.

```
% Command lines to record interaction matrices
me.rotMatMeasure(raven,me.pokeMat.cdmHadamard,.01,3)     % CDM
me.sdmSinePoke(raven,1,0.03)     % SDM 1
me.sdmSinePoke(raven,2,0.03)     % SDM 2
```

After the instrument is calibrated and the aberrations are removed, misalignements and registration of the interaction matrices, the open-loop tests are launched using the following command line :

```
me.openLoop(raven,1,1,1,'synt','mmse')
```

The arguments of the *me.openLoop* method are :

1. *raven* object : instance given to the Raven instrument in Matlab,
2. Number associated with the OLWFS,
3. Number associated with the science arm number (correction with SDM 1 or 2),

4. Set to 1 for complete open-loop application.

Once all the experimental interaction matrices are recorded, the synthetic ones are created using the *getSyntheticMatrix* function (figure 5.10). The left column has the interaction matrices between the CDM and all the WFSs (except the LGSWFS that was not in use at the time of the tests). The patterns are different because of the different angles of rotation between the CDM and these WFS. In the right column, the SDM-to-CLWFS interaction matrices are presented. They both have a 90° angle between the SDM and CLWFS which explain why they have the same pattern. The synthetic matrices are very similar to the experimental ones but not exactly identical. The experimental matrices are presented in appendix B.

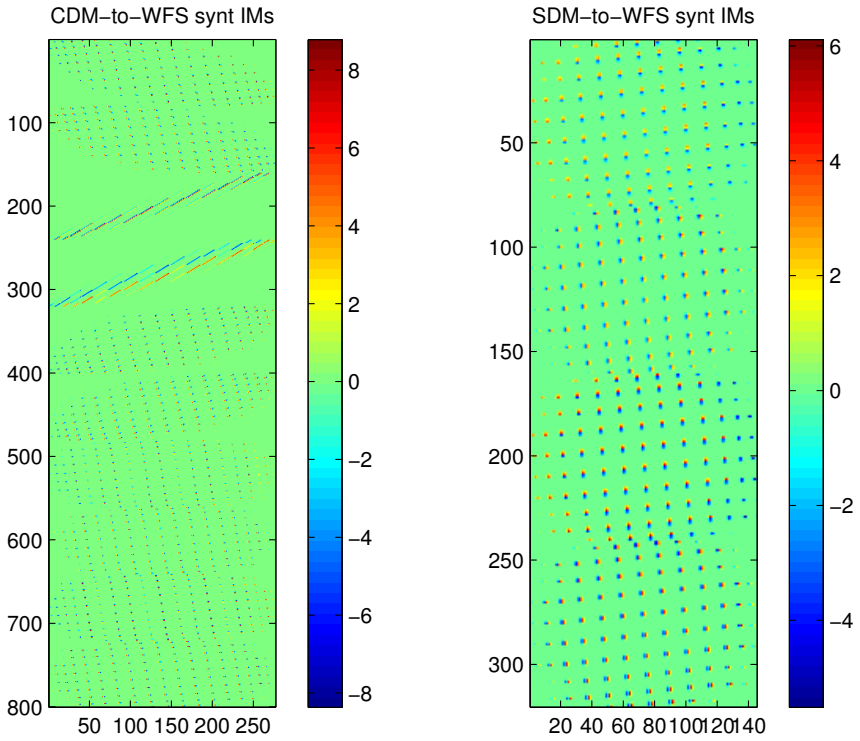


Figure 5.10: Synthetic Interaction Matrices.

5.4.2 Homogenization of OLWFS Measurements

In this section, experimental tests compare the use of synthetic interaction matrices with experimental interaction matrices to perform the transformation of OLWFS

slopes into CLWFS slope space. There are two different methods based on experimental data implemented on the Raven bench to perform that transformation. The first one relies on equation 1.5 in section 1.3. It uses the interaction matrices between the CDM and the OLWFS and CLWFS, the CDM being an intermediary between the two WFS. The transformation is performed on the OLWFS slopes and are converted into CLWFS slopes so the SDM can correct the aberrated wavefront with :

$$s_\beta = M_\beta^{OL} * (M_\alpha^{OL})^\dagger * s_\alpha \tag{5.2}$$

Where:

s_β: slopes of the OLWFS expressed into the CLWFS space,

M_β^{OL}: interaction matrix between the CLWFS and the CDM,

M_α^{OL}: interaction matrix between the OLWFS and the CDM,

s_α: slopes of the OLWFS.

The product of the two matrices transform the OLWFS slopes into the CLWFS space, which allows the correction to be applied on the science DM to flatten the wavefront as seen by the CLWFS (and the science camera). This product is the transformation matrix R from equation 1.1. These interaction matrices are replaced with synthetic ones to compare with the experimental ones. In this configuration, the registration parameters between the CDM and WFSs are known, which is not the case for the second method.

The second method is based on the Learn part of the Learn and Apply method [25]. This method calibrates experimentally the instrument for the misregistrations without expressing them using a Minimum Mean Square Error (MMSE). A series of linearly independent measurements are taken to compute the transformation matrix between two WFSs.

Again, in order to keep the problem simple, without including any tomographic considerations, the system used to compare the effect of synthetic matrices is an open-loop system. It uses *(i)* the CDM to create a ground layer of turbulence so that all the WFSs can see the same perturbation,*(ii)* an OLWFS to measure the turbulence, *(iii)* a SDM to correct the turbulence seen by the OLWFS and *(iv)* a CLWFS to measure the wavefront correction.

The results are presented in figure 5.11. The tests were not run in parallel but the WFS saw the same sequence of perturbation from the CDM in a time sequence looping over time. The root mean square residue using the transformation matrices for the experimental SVD, the MMSE (Learn) and synthetic SVD are, respectively, $0.42\mu m$, $0.42\mu m$ and $0.39\mu m$. These values should be lower but since we are comparing their relative values here, the results are still relevant. This shows that the method works and the correction is possible with synthetic matrices and performed slightly better than the experimental matrices. These are very encouraging results because the synthetic matrices can be improved by using curves that would better fit the influence functions and testing difference thresholds for the SVD. It gives us an improved understanding of the registration of the system compared to the completely experimental Learn and Apply method. A hybrid approach combining both methods may lead to better results.

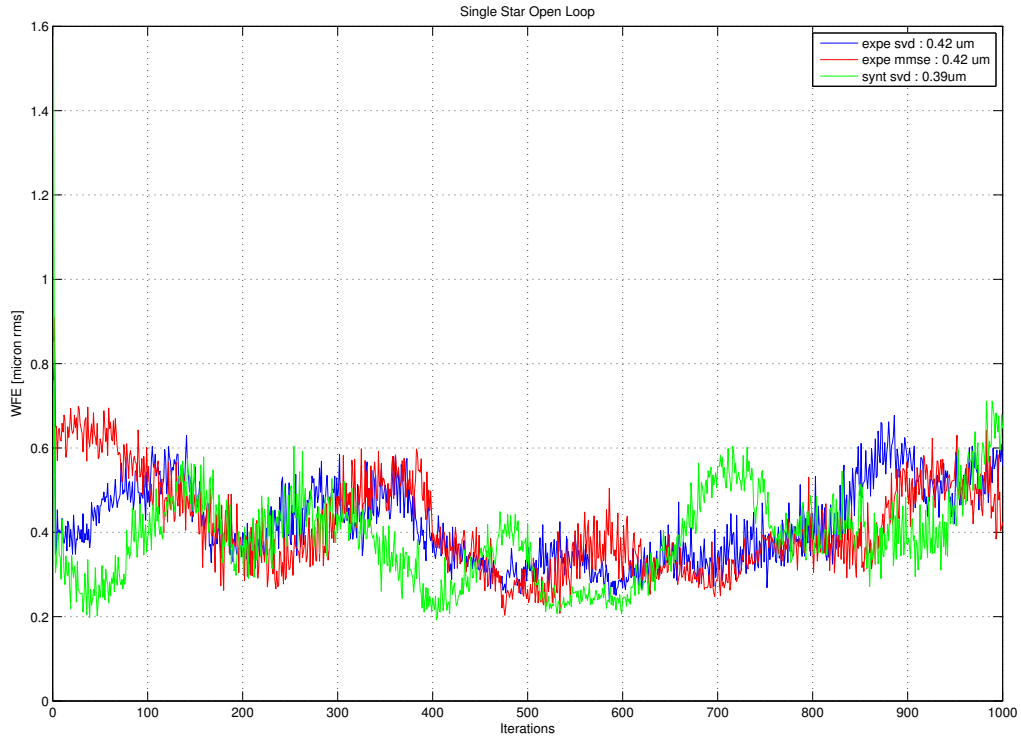


Figure 5.11: Open-Loop correction of a ground layer turbulence generated by the CDM. The correction is performed using the data from OLWFS2 and corrected on SDM1 with the residue measured by CLWFS1

5.4.3 Comparing Experimental To Synthetic Command Matrices

In this section, we compared the performance of experimental interaction matrices to synthetic ones. The synthetic interaction matrices were created following the method described in chapter 4. The experimental command matrix created by the pseudo-inversion is replaced by a synthetic command matrix created the same way. The result is the wavefront residue seen by the CLWFS located after the SDM selected for the tests. In this case, SDM 1 was used.

The results are shown in figure 5.12. They present the wavefront error (WFE) in function of the number of iterations (time). The lower this number, the better the correction. The curves are not taken at the same time so they do not reflect the correction of the turbulence at the same point. However, the rms values for both configurations can be compared and they are very similar : 0.48 microns rms for the experimental command matrix and 0.46 microns rms for the synthetic command matrix. The synthetic matrix had a better correction than the experimental. We note that the residue value for both tests should be lower but that their relative values can be compared.

This section validates the potential use of synthetic command matrices as it does not improve the correction drastically but shows that the method works. It has to be noted that the synthetic matrices have been created using a Gaussian function which represents the influence functions well but more complicated and accurate fitting functions could be developed.

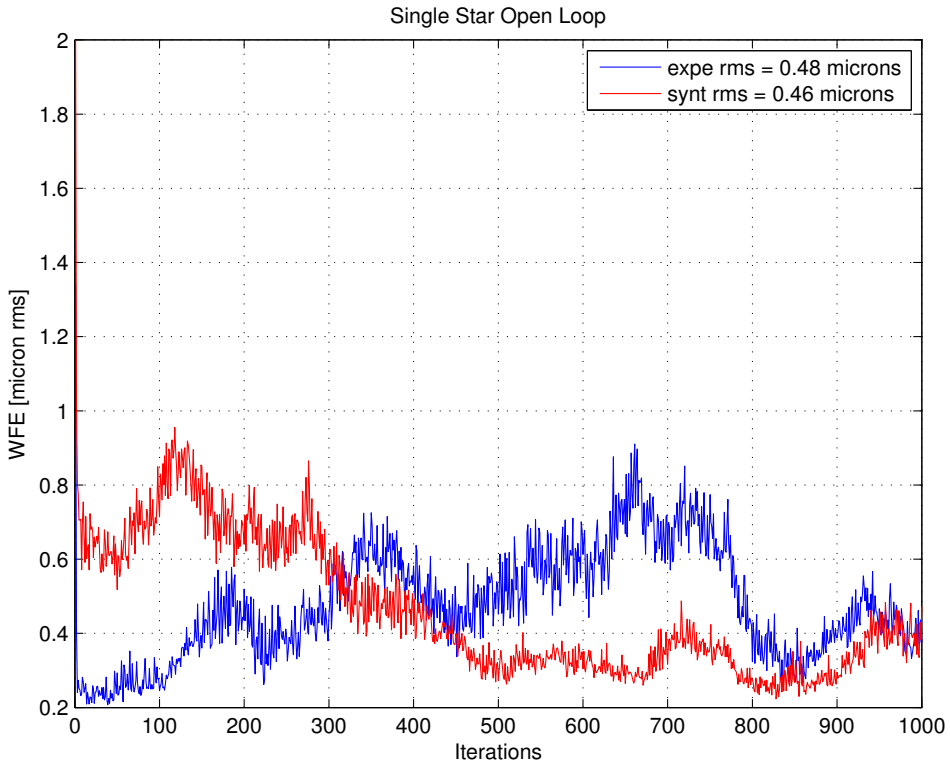


Figure 5.12: Comparison of experimental and synthetic command matrix performances.

Chapter 6

Conclusions

The future of ground telescopes is coming with the construction of extremely large telescopes. They pose a new problem because of their large field of view of a few arcminutes. The problem of anisoplanatism is still very limiting because of the need to observe more science objects in a same direction within the telescope's FOV. MOAO is a system under development for these ELTs and will enable about 20 science targets to be studied in parallel. The open-loop nature of the system requires a more critical knowledge of the system for a better calibration.

Raven is a multi-object adaptive optics demonstrator and the first MOAO instrument on an 8-m class telescope feeding an AO-optimized science instrument, the Subaru InfraRed Camera and Spectrograph (IRCS). With its two science channels, it will validate the use of MOAO with science observations.

An important part of the calibration process resides in the misregistration of the WFSs with the DMs because the sensing elements are located before the correcting ones. This problem is solved using a calibration DM seen by all WFSs in the system that permits the OLWFS to be registered to the science DMs. The method developed in this thesis registers the position of the DM actuators to the WFSs. It uses the deformable mirror continuous glass sheet property which makes the slopes point to the direction of the poked actuator. Then, using a series of actuator pokes, a map of actuators is created and the misregistration values between the DM and the WFS are computed.

The actuator mapping gave us enough information to provide a powerful tool for alignment, especially to achieve a Fried geometry with the SDMs and CLWFS, with a relative error of 6% only for the magnification. The horizontal and vertical displacements are accurate at 2% of a subaperture and the rotation angles have

absolute errors of less than 2°.

I have presented a new method to perform the transformation of matrices into a new reference space using synthetic matrices. The open-loop correction using synthetic interaction matrices gave better results for the transformation matrices (expressing the OLWFS slopes into CLWFS slopes) and the command matrices. These are very encouraging results because the influence functions can have a better fit than a Gaussian the different thresholds can be tested for the SVD.

These results are used to better align the instrument, to have a better knowledge of the positions of the different optical components and to generate new ways to perform the AO correction on Raven.

Bibliography

[1] David L. Fried. Anisoplanatism in adaptive optics. *J. Opt. Soc. Am.*, 72(1):52–52, Jan 1982.

[2] Dustin C. Johnston and Byron M. Welsh. Analysis of multiconjugate adaptive optics. *J. Opt. Soc. Am. A*, 11(1):394–408, Jan 1994.

[3] Edward A. Laag, S. Mark Ammons, Donald T. Gavel, and Renate Kupke. Multiconjugate adaptive optics results from the laboratory for adaptive optics MCAO/MOAO testbed. *J. Opt. Soc. Am. A*, 25(8):2114–2121, Aug 2008.

[4] F. Hammer et al. FALCON: a concept to extend adaptive optics corrections to cosmological fields. *Proceedings SPIE*, 5382:727–736, 2004.

[5] Raven Team. *Raven Conceptual Design Review*, 2011.

[6] Gendron, E., Vidal, F., Brangier, M., Morris, T., Hubert, Z., Basden, A., Rousset, G., Myers, R., Chemla, F., Longmore, A., Butterley, T., Dipper, N., Dunlop, C., Geng, D., Gratadour, D., Henry, D., Laporte, P., Looker, N., Perret, D., Sevin, A., Talbot, G., and Younger, E. Moao first on-sky demonstration with canary. *A&A*, 529:L2, 2011.

[7] G. Courtes. An Integral Field Spectrograph (IFS) for Large Telescopes. In C. M. Humphries, editor, *IAU Colloq. 67: Instrumentation for Astronomy with Large Optical Telescopes*, volume 92 of *Astrophysics and Space Science Library*, page 123, 1982.

[8] Peter J. McGregor, John Hart, Peter G. Conroy, Murray L. Pfitzner, Gabe J. Bloxham, Damien J. Jones, Mark D. Downing, Murray Dawson, Peter Young, Mark Jarnyk, and Jan Van Harmelen. Gemini near-infrared integral field spectrograph (NIFS), 2003.

[9] David R. Andersen, Stephen S. Eikenberry, Murray Fletcher, William Gardhouse, Brian Leckie, Jean-Pierre Vran, Don Gavel, Richard Clare, Rafael Guzman, Laurent Jolissaint, Roger Julian, and William Rambold. The MOAO system of the IRMOS near-infrared multi-object spectrograph for TMT. In *Proc. SPIE*, volume 6269, pages 62694K–62694K–12, 2006.

[10] David Crampton, Luc Simard, and David Silva. Early light tmt instrumentation. In *Proc. SPIE*, volume 7014, pages 70141D–70141D–13, 2008.

[11] Larry Stepp. Thirty meter telescope project update. In *Proc. SPIE*, volume 8444, pages 84441G–84441G–16, 2012.

[12] Jean-Gabriel Cuby, Simon Morris, Thierry Fusco, Matthew Lehnert, Philip Parr-Burman, Gerard Rousset, Jean-Philippe Amans, Stephen Beard, Ian Bryson, Mathieu Cohen, Nigel Dipper, Chris Evans, Marc Ferrari, Eric Gendron, Jean-Luc Gimenez, Damien Gratadour, Peter Hastings, Zoltan Hubert, Emmanuel Hugot, Pascal Jagourel, Philippe Laporte, Vincent Lebrun, David Le Mignant, Fabrice Madec, Richard Myers, Benoit Neichel, Tim Morris, Clelia Robert, Hermine Schnetler, Mark Swinbank, Gordon Talbot, William Taylor, Francois Vidal, Sbastien Vivs, Pascal Vola, Niraj Welikala, and Martyn Wells. Eagle: a moao fed multi-ifu nir workhorse for e-elt. In *Proc. SPIE*, volume 7735, pages 77352D–77352D–15, 2010.

[13] S. Ramsay, S. D'Odorico, M. Casali, J. C. Gonzlez, N. Hubin, M. Kasper, H. U. Kufl, M. Kissler-Patig, E. Marchetti, J. Paufique, L. Pasquini, R. Siebenmorgen, A. Richichi, J. Vernet, and F. M. Zerbi. An overview of the e-elt instrumentation programme. In *Proc. SPIE*, volume 7735, pages 773524–773524–15, 2010.

[14] Rodolphe Conan, Colin Bradley, Olivier Lardière, Célia Blain, Kim Venn, David Andersen, Luc Simard, Jean-Pierre Véran, Glen Herriot, David Loop, Tomonori Usuda, Shin Oya, Yutaka Hayano, Hiroshi Terada, and Masayuki Akiyama. Raven: a harbinger of multi-object adaptive optics-based instruments at the subaru telescope. In *Proc. SPIE*, volume 7736, pages 77360T–77360T–14, 2010.

[15] Alan T. Tokunaga, Naoto Kobayashi, James Bell, Gregory K. Ching, Klaus-Werner Hodapp, Joseph L. Hora, Doug Neill, Peter M. Onaka, John T. Rayner, Louis Robertson, David W. Warren, Mark Weber, and Tony T. Young. Infrared

camera and spectrograph for the subaru telescope. In *Proc. SPIE*, volume 3354, pages 512–524, 1998.

[16] Jean-Francois Lavigne, Frédéric Lamontagne, Min Wang, Mathieu Tremblay, Olivier Lardiére, David Andersen, Reston Nash, Patrice Côté, Geneviève Anctil, Colin Bradley, and Francois Châteauneuf. Design of the calibration unit for the MOAO demonstrator raven. In *AO4ELT2 Proceedings*, 2012.

[17] Yosuke Minowa, Yutaka Hayano, Shin Oya, Makoto Watanabe, Masayuki Hattori, Olivier Guyon, Sebastian Egner, Yoshihiko Saito, Meguro Ito, Hideki Takami, Vincent Garrel, Stephen Colley, Taras Golota, and Masanori Iye. Performance of Subaru adaptive optics system AO188. *Proceedings SPIE*, 7736, 2010.

[18] David R. Andersen, Kate J. Jackson, Célia Blain, Colin Bradley, Carlos Correia, Meguru Ito, Olivier Lardière, and Jean-Pierre Véran. Performance Modeling for the RAVEN Multi-Object Adaptive Optics Demonstrator. *Publications of the Astronomical Society of the Pacific*, 124(915):pp. 469–484, 2012.

[19] Markus Kasper, Enrico Fedrigo, Douglas P. Looze, Henri Bonnet, Liviu Ivanescu, and Sylvain Oberti. Fast calibration of high-order adaptive optics systems. *J. Opt. Soc. Am. A*, 21(6):1004–1008, Jun 2004.

[20] Laurie Pham, Olivier Lardière, and David Andersen. Calibration Algorithm Development for the RAVEN MOAO test bed. In *AO4ELT2 Proceedings*, 2012.

[21] Kate Jackson et al. Tomographic wavefront error estimation and measurement for raven, a multi-object adaptive optics demonstrator. *Proceedings SPIE*, 2012.

[22] Benoit Neichel, Amelie Parisot, Cyril Petit, Thierry Fusco, and François Rigaut. Identification and calibration of the interaction matrix parameters for AO and MCAO systems. *SPIE proceedings*, 8447(1):84475N–84475N–15, September 2012.

[23] Michael D. Oliker. Alignment techniques for DM, Lenslet, and WFS camera at the SOR. *Proceedings SPIE*, 3126:595–604, 1997.

[24] Richard M. Myers, Zoltn Hubert, Timothy J. Morris, Eric Gendron, Nigel A. Dipper, Agla Kellerer, Stephen J. Goodsell, Grard Rousset, Eddy Younger, Michel

Marteaud, Alastair G. Basden, Fanny Chemla, C. Dani Guzman, Thierry Fusco, Deli Geng, Brice Le Roux, Mark A. Harrison, Andrew J. Longmore, Laura K. Young, Fabrice Vidal, and Alan H. Greenaway. CANARY: the on-sky NGS/LGS MOAO demonstrator for eagle. In *Proc. SPIE*, volume 7015, pages 70150E–70150E–9, 2008.

[25] Fabrice Vidal, Eric Gendron, and Gérard Rousset. Tomography approach for multi-object adaptive optics. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 27(11):A253–64, November 2010.

[26] Berthold K. P. Horn, H.M. Hilden, and Shariar Negahdaripour. Closed-form Solution of Absolute Orientation using Orthonormal Matrices. *Journal of the Optical Society America*, 5(7):1127–1135, 1988.

[27] Rodolphe Conan. *Object-Oriented Matlab Adaptive Optics User Guide*, 2010.

[28] Olivier Lardière. *Calibration Unit Design Requirements*, 2011.

[29] Serge Meimon, Thierry Fusco, and Cyril Petit. The slope-oriented hadamard scheme for in-lab or on-sky interaction matrix calibration. In *AO4ELT2 Proceedings*, 2012.

Appendix A

Matlab Codes

A.1 Function registrationData

```
% Structure containing the registration parameters
classdef registrationData
   properties
      angleInRadians;
      angleInDeg;
      angleStd;
      xScale;
      yScale;
      xShiftInPitch;
      yShiftInPitch;
      rmsResidualError;
   end
end
```

A.2 Function getRegistrationParameters

```
%% [regisParam, actMapXRef, actMapYRef, unRotScaleShiftMatX, ...
    unRotScaleShiftMatY] = ...
    getRegistrationParameters(interactionMatrix, method)
% INPUT : The interaction matrix between the DM and WFS the user ...
    wants to
```

```
3  % extract the registration parameters from (translation, scale, ...
       rotation).
4  %           interactionMatrix
5  %           method 'weighted' or 'opti' (using the center of gravity
6  %           determination method or optimization method to find ...
       actuators'
7  %           positions.
8  %
9  % OUPUT :   The registration parameters in the registrationData ...
       object :
10 % regisParam.
11 %           actMapXRef, actMapYRef are the coordinates of the ...
       reference map
12 %           unRotScaleShiftMatX, unRotScaleShiftMatY are the ...
       coordinates of
13 %           the un-rotated, un-scaled, un-shifted experimental ...
       map. These
14 %           are used to find and plot the residual map.
15 %
16 % DEPENDENCY :  getActuatorMap.m (that uses projectionOfGuess.m)
17 %               getRotationAngle.m
18 %               getScale.m
19 %               getShift.m
20 %               utilities.m (OOMAO library)
21 %
22 % EXAMPLES :
23 % * To get the registration data between a DM and a WFS using their
24 % interaction matrix :
25 % regisParam = getRegistrationParameters(interactionMatrix);
26 %
27 % * To plot the residual map (residue between the reference map ...
       and the
28 % un-rotated, un-scaled, un-shifted matrix):
29 % [regisParamCdmOlwfs1, XRef,YRef,XExp,YExp] = ...
       getRegistrationParameters(interactionCdmOlwfs1);
30 % maskCommonActu = logical(XExp.*YExp);
31 % figure, clf,
32 % ...
       quiver(XRef(maskCommonActu),YRef(maskCommonActu),(XRef(maskCommonActu)-XExp(ma
33 % title('Residual registration CDM-to-OLWFS 1')
34
35
```

```
36 function [regisParam, actMapXRef, actMapYRef, ...
       unRotScaleShiftMatX, unRotScaleShiftMatY, maskActuInsidePupil] ...
       = getRegistrationParameters(interactionMatrix, method)
37 regisParam = registrationData;
38 %% Initialization
39 nActuatorCdm = 19;
40 %Reshape the mask of the valid actuators (277 valid actuators on ...
       the CDM)
41 maskValidActuatorCdm = logical(utilities.piston(nActuatorCdm));
42 maskValidActuatorCdm(2,5) = false;
43 maskValidActuatorCdm(3,4) = false;
44 maskValidActuatorCdm(4,3) = false;
45 maskValidActuatorCdm(5,2) = false;
46 maskValidActuatorCdm(2,15) = false;
47 maskValidActuatorCdm(3,16) = false;
48 maskValidActuatorCdm(4,17) = false;
49 maskValidActuatorCdm(5,18) = false;
50 maskValidActuatorCdm(18,15) = false;
51 maskValidActuatorCdm(17,16) = false;
52 maskValidActuatorCdm(16,17) = false;
53 maskValidActuatorCdm(15,18) = false;
54 maskValidActuatorCdm(18,5) = false;
55 maskValidActuatorCdm(17,4) = false;
56 maskValidActuatorCdm(16,3) = false;
57 maskValidActuatorCdm(15,2) = false;
58
59 nActuatorSdm = 13;
60 %Reshape the mask of the valid actuators
61 maskValidActuatorSdm = logical(utilities.piston(nActuatorSdm));
62 maskValidActuatorSdm(1,4) = true;
63 maskValidActuatorSdm(1,10) = true;
64 maskValidActuatorSdm(4,1) = true;
65 maskValidActuatorSdm(4,end) = true;
66 maskValidActuatorSdm(10,1) = true;
67 maskValidActuatorSdm(10,end) = true;
68 maskValidActuatorSdm(end,4) = true;
69 maskValidActuatorSdm(end,10) = true;
70 % Get the size of interaction matrix : number of actuators across
71 %  the pupil and load the correct mask of valid actuators
72
73 [nSlopes,nActu] = size(interactionMatrix);
74 if nActu == 277
```

```
75      nActuAcross = nActuatorCdm;
76      maskValidActuator = maskValidActuatorCdm;
77      u = (-9:1:9);
78      regisParam.pitch = 1.5; % in mm
79  elseif nActu == 145
80      nActuAcross = nActuatorSdm;
81      maskValidActuator = maskValidActuatorSdm;
82      u = (-6:1:6);
83      regisParam.pitch = 2.5; % in mm
84  elseif nActu == 97
85      nActuAcross = 11;
86      maskValidActuator = logical(utilities.piston(nActuAcross));
87      u = (-5:1:5);
88      regisParam.pitch = 2.5; % in mm
89  end
90  %% Load the experimental map to compare
91  if strcmpi(method,'opti')
92      % Experimental Actuator Map
93      [actMapX, actMapY, maskActuInsidePupil] = ...
            getActuatorMap(interactionMatrix,maskValidActuator);
94  else
95      [actMapX, actMapY] = ...
            getActuatorMapCoG(interactionMatrix,maskValidActuator);
96  end
97  % Create the Reference Actuator Map
98  [tmpMapXRef, tmpMapYRef] = meshgrid(u,u);
99  actMapXRef = zeros(nActuAcross);
100 actMapYRef = zeros(nActuAcross);
101 maskCommonActu = logical(actMapX.*actMapY);
102 actMapXRef(maskCommonActu) = tmpMapXRef(maskCommonActu);
103 actMapYRef(maskCommonActu) = tmpMapYRef(maskCommonActu);
104
105 %figure, plot(actMapXCoG(:),actMapYCoG(:),'+')
106 %hold on, plot(actMapX(:), actMapY(:),'r+'), hold off
107 %% Rotation
108 [regisParam.angleInRadians,regisParam.angleStd] = ...
       getRotationAngle(actMapXRef, actMapYRef, actMapX, actMapY);
109 regisParam.angleInDeg = regisParam.angleInRadians*180/pi;
110 %% Scale
111 [regisParam.xScale, regisParam.yScale] = getScale(actMapXRef, ...
       actMapYRef, actMapX, actMapY,nActuAcross);
112 %% Remove the rotation to find the translation
```

```
indexCenter = (size(maskCommonActu,2)+1)/2;
% create a matrix of these points, which will be useful in future ...
    calculations
v = [actMapX(:)';actMapY(:)'];
% Point which will be the center of rotation
x_center = actMapX(indexCenter,indexCenter);
y_center = actMapY(indexCenter,indexCenter);
% create a matrix which will be used later in calculations
centerMatrix = repmat([x_center; y_center], 1, length(actMapX).^2);
% define the rotation matrix
theta = -regisParam.angleInRadians; % pi/3 radians = 60 degrees
R = [cos(theta) -sin(theta); sin(theta) cos(theta)];

unRotatedMat = R*(v - centerMatrix) + centerMatrix;
unRotMatX = zeros(size(maskCommonActu));
unRotMatY = zeros(size(maskCommonActu));
tmp = unRotatedMat(1,:);
unRotMatX(maskCommonActu==1) = tmp(maskCommonActu==1);
tmp2 = unRotatedMat(2,:);
unRotMatY(maskCommonActu==1) =  tmp2(maskCommonActu==1);
clear tmp tmp2
% Unscale matrix
unRotScaledMatX = zeros(size(unRotMatX));
unRotScaledMatY = zeros(size(unRotMatY));
unRotScaledMatXtmp = (unRotMatX - repmat(x_center,  ...
    size(actMapX)))/regisParam.xScale + repmat(x_center,  ...
    size(actMapX));
unRotScaledMatX(maskCommonActu==1) = ...
    unRotScaledMatXtmp(maskCommonActu==1);
unRotScaledMatYtmp = (unRotMatY - repmat(y_center,  ...
    size(actMapY)))/regisParam.yScale + repmat(y_center,  ...
    size(actMapY));
unRotScaledMatY(maskCommonActu==1) =  ...
    unRotScaledMatYtmp(maskCommonActu==1);
%% Shift
[regisParam.xShiftInPitch, regisParam.yShiftInPitch] = ...
    getShift(actMapXRef, actMapYRef, unRotScaledMatX, ...
    unRotScaledMatY,maskCommonActu);
% regisParam.xShiftInMm = regisParam.xShiftInPitch*regisParam.pitch;
% regisParam.yShiftInMm = regisParam.yShiftInPitch*regisParam.pitch;

%% Remove the Shift from the matrix and plot the result
```

```
146 unRotScaleShiftMatX = zeros(size(actMapX));
147 unRotScaleShiftMatY = zeros(size(actMapY));
148 unRotScaleShiftMatX = ...
       unRotScaledMatX-repmat(regisParam.xShiftInPitch,size(unRotScaledMatX));
149 unRotScaleShiftMatY = ...
       unRotScaledMatY-repmat(regisParam.yShiftInPitch,size(unRotScaledMatY));
150 unRotScaledMatXtmp = ...
       unRotScaledMatX-repmat(regisParam.xShiftInPitch,size(unRotScaledMatX));
151 unRotScaledMatYtmp = ...
       unRotScaledMatY-repmat(regisParam.yShiftInPitch,size(unRotScaledMatY));
152 unRotScaleShiftMatX(maskCommonActu) = ...
       unRotScaledMatXtmp(maskCommonActu);
153 unRotScaleShiftMatY(maskCommonActu) = ...
       unRotScaledMatYtmp(maskCommonActu);
154 regisParam.rmsResidualError = ...
       sqrt(sum(sum(maskCommonActu.*((actMapXRef-unRotScaleShiftMatX).^2 ...
       + ...
       (actMapYRef-unRotScaleShiftMatY).^2)))/sum(sum(maskCommonActu)));
155 %% REMOVE THE POINTS THAT CLEARLY HAVE TOO MUCH NOISE (Distance ...
       more than % of pitch)
156 normDisplace = zeros(size(actMapXRef));
157 normDisplace = ...
       maskCommonActu.*sqrt((actMapXRef-unRotScaleShiftMatX).^2 + ...
       (actMapYRef-unRotScaleShiftMatY).^2);
158 %tmpMask2 = ...
       sqrt((actMapXRef(maskCommonActu)-unRotScaleShiftMatX(maskCommonActu)).^2+(actM
159 filterMask = true(size(actMapXRef));
160 threshold = 2 * regisParam.rmsResidualError;
161 filterMask(normDisplace>threshold) = 0;
162 newMask = logical(maskCommonActu.*filterMask);
163 %% RUN WITH NEW MASK :
164 [tmpMapXRef, tmpMapYRef] = meshgrid(u,u);
165 actMapXRef = zeros(nActuAcross);
166 actMapYRef = zeros(nActuAcross);
167 maskCommonActu = newMask;
168 actMapXRef(maskCommonActu) = tmpMapXRef(maskCommonActu);
169 actMapYRef(maskCommonActu) = tmpMapYRef(maskCommonActu);
170 %% Rotation
171 [regisParam.angleInRadians,regisParam.angleStd] = ...
       getRotationAngle(actMapXRef, actMapYRef, actMapX, actMapY);
172 regisParam.angleInDeg = regisParam.angleInRadians*180/pi;
173 %% Scale
```

```
174 [regisParam.xScale, regisParam.yScale] = getScale(actMapXRef, ...
       actMapYRef, actMapX, actMapY,nActuAcross);
175 %% Remove the rotation to find the translation
176 indexCenter = (size(maskCommonActu,2)+1)/2;
177 % create a matrix of these points, which will be useful in future ...
       calculations
178 v = [actMapX(:)';actMapY(:)'];
179 % Point which will be the center of rotation
180 x_center = actMapX(indexCenter,indexCenter);
181 y_center = actMapY(indexCenter,indexCenter);
182 % create a matrix which will be used later in calculations
183 centerMatrix = repmat([x_center; y_center], 1, length(actMapX).^2);
184 % define the rotation matrix
185 theta = -regisParam.angleInRadians; % pi/3 radians = 60 degrees
186 R = [cos(theta) -sin(theta); sin(theta) cos(theta)];
187
188 unRotatedMat = R*(v - centerMatrix) + centerMatrix;
189 unRotMatX = zeros(size(maskCommonActu));
190 unRotMatY = zeros(size(maskCommonActu));
191 tmp = unRotatedMat(1,:);
192 unRotMatX(maskCommonActu==1) = tmp(maskCommonActu==1);
193 tmp2 = unRotatedMat(2,:);
194 unRotMatY(maskCommonActu==1) =  tmp2(maskCommonActu==1);
195 clear tmp tmp2
196 % Unscale matrix
197 unRotScaledMatX = zeros(size(unRotMatX));
198 unRotScaledMatY = zeros(size(unRotMatY));
199 unRotScaledMatXtmp = (unRotMatX - repmat(x_center,  ...
       size(actMapX)))/regisParam.xScale + repmat(x_center,  ...
       size(actMapX));
200 unRotScaledMatX(maskCommonActu==1) = ...
       unRotScaledMatXtmp(maskCommonActu==1);
201 unRotScaledMatYtmp = (unRotMatY - repmat(y_center,  ...
       size(actMapY)))/regisParam.yScale + repmat(y_center,  ...
       size(actMapY));
202 unRotScaledMatY(maskCommonActu==1) =  ...
       unRotScaledMatYtmp(maskCommonActu==1);
203 %% Shift
204 [regisParam.xShiftInPitch, regisParam.yShiftInPitch] = ...
       getShift(actMapXRef, actMapYRef, unRotScaledMatX, ...
       unRotScaledMatY,maskCommonActu);
205 % regisParam.xShiftInMm = regisParam.xShiftInPitch*regisParam.pitch;
```

```
206 % regisParam.yShiftInMm = regisParam.yShiftInPitch*regisParam.pitch;
207
208 %% Remove the Shift from the matrix and plot the result
209 unRotScaleShiftMatX = zeros(size(actMapX));
210 unRotScaleShiftMatY = zeros(size(actMapY));
211 unRotScaledMatXtmp = ...
       unRotScaledMatX-repmat(regisParam.xShiftInPitch,size(unRotScaledMatX));
212 unRotScaledMatYtmp = ...
       unRotScaledMatY-repmat(regisParam.yShiftInPitch,size(unRotScaledMatY));
213 unRotScaleShiftMatX(maskCommonActu) = ...
       unRotScaledMatXtmp(maskCommonActu);
214 unRotScaleShiftMatY(maskCommonActu) = ...
       unRotScaledMatYtmp(maskCommonActu);
215
216 regisParam.rmsResidualError = ...
       sqrt(sum(sum(maskCommonActu.*((actMapXRef-unRotScaleShiftMatX).^2 ...
       + ...
       (actMapYRef-unRotScaleShiftMatY).^2)))/sum(sum(maskCommonActu)));
217 end
```

A.3 Function getSyntheticMatrix

```
1  function generatedMatrix = getSyntheticMatrix(interactionMatrix)
2  % Mask of valid actuators defined by the constructor data for the CDM
3  maskValidActuatorCdm = logical(utilities.piston(19));
4  maskValidActuatorCdm(2,5) = false;
5  maskValidActuatorCdm(3,4) = false;
6  maskValidActuatorCdm(4,3) = false;
7  maskValidActuatorCdm(5,2) = false;
8  maskValidActuatorCdm(2,15) = false;
9  maskValidActuatorCdm(3,16) = false;
10 maskValidActuatorCdm(4,17) = false;
11 maskValidActuatorCdm(5,18) = false;
12 maskValidActuatorCdm(18,15) = false;
13 maskValidActuatorCdm(17,16) = false;
14 maskValidActuatorCdm(16,17) = false;
15 maskValidActuatorCdm(15,18) = false;
16 maskValidActuatorCdm(18,5) = false;
17 maskValidActuatorCdm(17,4) = false;
```

```
18 maskValidActuatorCdm(16,3) = false;
19 maskValidActuatorCdm(15,2) = false;
20
21 % Mask of valid actuators defined by the constructor data for the SDM
22 maskValidActuatorSdm = logical(utilities.piston(13));
23 maskValidActuatorSdm(1,4) = true;
24 maskValidActuatorSdm(1,10) = true;
25 maskValidActuatorSdm(4,1) = true;
26 maskValidActuatorSdm(4,end) = true;
27 maskValidActuatorSdm(10,1) = true;
28 maskValidActuatorSdm(10,end) = true;
29 maskValidActuatorSdm(end,4) = true;
30 maskValidActuatorSdm(end,10) = true;
31
32 coupling30 = 8; % To create Gaussian with coupling of 30% (have ...
      to multiply by scale)
33 coupling40 = 8.9;
34
35 [nSlopes,nActu] = size(interactionMatrix);
36 [regisP, XRef,YRef,XExp,YExp] = ...
      getRegistrationParameters(interactionMatrix,'opti');
37
38 if nActu == 277
39     nActuAcross = 19;
40     maskValidActuator = maskValidActuatorCdm;
41     coupling = coupling30;
42     amplitude = -1;
43     u = (-9:1:9);
44     sourceWavelength = photometry.R;
45 elseif nActu == 145
46     nActuAcross = 13;
47     maskValidActuator = maskValidActuatorSdm;
48     coupling = coupling40;
49     amplitude = 1;
50     u = (-6:1:6);
51     sourceWavelength = photometry.H;
52 elseif nActu == 97
53     nActuAcross = 11;
54     maskValidActuator = logical(utilities.piston(nActuAcross));
55     coupling = coupling40;
56     amplitude = 1;
57     u = (-5:1:5);
```

```
58      sourceWavelength = photometry.H;
59  end
60
61  % WFS initialization
62  nLenslet = 10;  %nLensletxnLenslet lenslets
63  nPx = 12;       %nPx pix/subap
64
65  % Initialization of WFS
66  telescopeSize = 8;  % in meters
67  fieldOfViewInArcMinutes = 2;
68  samplingTimeForCam = 1/500;
69  tel = telescope(telescopeSize,...           %  Telescope
70      'fieldOfViewInArcMin',fieldOfViewInArcMinutes,...
71      'samplingTime',samplingTimeForCam,...
72      'resolution',nLenslet*nPx);
73  source1 = source('wavelength',sourceWavelength);   % SCIENCE SOURCE
74
75  wfs = shackHartmann(nLenslet,nLenslet*nPx);
76  wfs.validLenslet = logical(utilities.piston(nLenslet));
77  source1 = source1.*tel*wfs; % Propagation to the wfs
78  wfs.referenceSlopes = wfs.slopes; %set the reference slopes
79
80  % 1) APPLY THE TRANSFORMATION TO A MESHGRID (OF ACTUATORS)
81  % Magnification
82  inputXScale = regisP.xScale;
83  inputYScale = regisP.yScale;
84  inputTheta = pi/2-regisP.angleInRadians; % Rotation angle between ...
       DM-to-WFS in radians
85  % Shift : value in pixel (12 pixels by subaperture)
86  inputXShift = regisP.yShiftInPitch;%*regisP.xScale*nPx; % ...
       (transform from subaperture to pixels)
87  inputYShift = regisP.xShiftInPitch;%*regisP.yScale*nPx;
88
89  % Create the Reference Actuator Map
90  [actMapXRef, actMapYRef] = meshgrid(u,u);
91  actMapYRef = flipud(actMapYRef);
92  actMapXRef(maskValidActuator == 0) = 0;
93  actMapYRef(maskValidActuator == 0) = 0;
94
95  indexCenter = length(actMapXRef)/2+0.5;
96  x_center = actMapXRef(indexCenter,indexCenter); % Assuming the ...
       maps are square
```

```
97  y_center = actMapYRef(indexCenter,indexCenter);
98  % To make sure the rotation center is the center of the matrix
99  centerMatrix = repmat([x_center; y_center], 1, ...
        length(actMapXRef).^2);
100
101 M = [inputXScale*cos(inputTheta) -inputYScale*sin(inputTheta); ...
        inputXScale*sin(inputTheta) inputYScale*cos(inputTheta)];
102 clear tmp
103 B = repmat([inputXShift*inputXScale; ...
        inputYShift*inputYScale],1,length(actMapXRef).^2);
104 tmpMat = M * ([actMapXRef(:)'; actMapYRef(:)']  - centerMatrix) + ...
        centerMatrix + B;
105
106 % 2) APPLY MASK OF VALID ACTUATORS TO THAT MESHGRID
107 actMapX = zeros(size(actMapXRef));
108 actMapY = zeros(size(actMapYRef));
109 tmp = tmpMat(1,:);
110 actMapX(maskValidActuator) = tmp(maskValidActuator);
111 tmp2 = tmpMat(2,:);
112 actMapY(maskValidActuator) =  tmp2(maskValidActuator);
113
114 % In Phase Coordinates : 120x120
115 actMapX(maskValidActuator) = tmp(maskValidActuator)*nPx + 60;
116 actMapY(maskValidActuator) =  tmp2(maskValidActuator)*nPx + 60;
117 xCoord = actMapX(maskValidActuator);
118 yCoord = actMapY(maskValidActuator);
119
120 % 4) APPLY THAT DISPLACEMENT TO PHASE ("POKE ACTUATOR" without DM)
121     phaseSize = 1:120;
122     slope = zeros(2*wfs.nValidLenslet,nActu);
123     allPhase = zeros(120,120,nActu);
124     sigmaX = coupling*inputXScale;  % NB for Gaussian function : ...
            FWTM = 2*sqrt(2*log(2))*sigma
125     sigmaY = coupling*inputYScale;
126 for k = 1:nActu
127     % Apply new phase here (120x120)
128     xCenter = xCoord(k);%60+(k-1)*12;
129     yCenter = yCoord(k);%60+(k-1)*12;
130     gaussian = @(x,y) ...
            amplitude*exp(-((x-xCenter).^2/(2*sigmaX.^2) + ...
            (y-yCenter).^2/(2*sigmaY.^2)));
131     gaussianValue = zeros(120);
```

```
132     for j = 1:120   % On the phase size
133         gaussianValue(j,:) = gaussian(j,phaseSize);
134     end
135     allPhase(:,:,k) = gaussianValue;
136     source1.reset
137     +wfs;
138     source1.amplitude = 1;
139     source1.phase = gaussianValue;  % Set the poked actuator phase
140     source1=source1*wfs;    % Propagate the phase (WFS sees the ...
            poked actuator)
141     slope(:,k) = wfs.slopes;  % Save that slope
142 end
143
144 % Adapt the range of the synthetic matrix to the experimental one ...
      using a
145 % minimization technique.
146 range = fminsearch(@(x) ...
      abs(sum((x*slope(:)).^2)-sum(interactionMatrix(:).^2)),1);
147 generatedMatrix = slope*range;
148 end
```

A.4 Function getCommandMat

```
1  function [invertedInfluenceFun, influenceFunOnDm, ...
       maskValidActuator] = getCommandMat(interactionMatrix,dmNumber)
2  % Generate the command matrix corresponding to the registration ...
       parameters found between the DM and WFS.
3  % Then use :
4  % commandVectorPinv(maskValidActuator)= ...
       invertedInfluenceFun*phaseToFit;
5  % to generate the command vector and then apply the correction ...
       with :
6  % reconstructedphasePinv = ...
       influenceFunOnDm(:,:)*commandVectorPinv(maskValidActuator);
7
8  [nSlopes,nActu] = size(interactionMatrix);
9  [regisP, XRef,YRef,XExp,YExp] = ...
       getRegistrationParameters(interactionMatrix,'opti');
10
```

```
maskValidActuatorCdm = logical(utilities.piston(19));
maskValidActuatorCdm(2,5) = false;
maskValidActuatorCdm(3,4) = false;
maskValidActuatorCdm(4,3) = false;
maskValidActuatorCdm(5,2) = false;
maskValidActuatorCdm(2,15) = false;
maskValidActuatorCdm(3,16) = false;
maskValidActuatorCdm(4,17) = false;
maskValidActuatorCdm(5,18) = false;
maskValidActuatorCdm(18,15) = false;
maskValidActuatorCdm(17,16) = false;
maskValidActuatorCdm(16,17) = false;
maskValidActuatorCdm(15,18) = false;
maskValidActuatorCdm(18,5) = false;
maskValidActuatorCdm(17,4) = false;
maskValidActuatorCdm(16,3) = false;
maskValidActuatorCdm(15,2) = false;

maskValidActuatorSdm = logical(utilities.piston(13));
maskValidActuatorSdm(1,4) = true;
maskValidActuatorSdm(1,10) = true;
maskValidActuatorSdm(4,1) = true;
maskValidActuatorSdm(4,end) = true;
maskValidActuatorSdm(10,1) = true;
maskValidActuatorSdm(10,end) = true;
maskValidActuatorSdm(end,4) = true;
maskValidActuatorSdm(end,10) = true;

coupling30 = 8; % To create Gaussian with coupling of 30% (have ...
    to multiply by scale)
coupling40 = 8.9;

if nActu == 277
    nActuAcross = 19;
    maskValidActuator = maskValidActuatorCdm;
    coupling = coupling30;
    amplitude = -1;
    u = (-9:1:9);
elseif nActu == 145
    nActuAcross = 13;
    maskValidActuator = maskValidActuatorSdm;
    coupling = coupling40;
```

```
52      amplitude = 1;
53      u = (-6:1:6);
54  elseif nActu == 97
55      nActuAcross = 11;
56      maskValidActuator = logical(utilities.piston(nActuAcross));
57      coupling = coupling40;
58      amplitude = 1;
59      u = (-5:1:5);
60  end
61
62  nPx = 12;       %nPx pix/subap
63
64  % 1) APPLY THE TRANSFORMATION TO A MESHGRID (OF ACTUATORS)
65  % Magnification
66  inputXScale = regisP.xScale;
67  inputYScale = regisP.yScale;
68  inputTheta = pi/2-regisP.angleInRadians; % Rotation angle between ...
        DM-to-WFS in radians
69  % Shift : value in pixel (12 pixels by subaperture)
70  inputXShift = regisP.yShiftInPitch;
71  inputYShift = regisP.xShiftInPitch;
72
73  % Create the Reference Actuator Map
74  [actMapXRef, actMapYRef] = meshgrid(u,u);
75  actMapYRef = flipud(actMapYRef);
76  actMapXRef(maskValidActuator == 0) = 0;
77  actMapYRef(maskValidActuator == 0) = 0;
78
79  indexCenter = length(actMapXRef)/2+0.5;
80  x_center = actMapXRef(indexCenter,indexCenter); % Considering the ...
        maps are square
81  y_center = actMapYRef(indexCenter,indexCenter);
82  % To make sure the rotation center is the center of the matrix
83  centerMatrix = repmat([x_center; y_center], 1, ...
        length(actMapXRef).^2);
84
85  M = [inputXScale*cos(inputTheta) -inputYScale*sin(inputTheta); ...
        inputXScale*sin(inputTheta) inputYScale*cos(inputTheta)];
86  clear tmp
87  B = repmat([inputXShift*inputXScale; ...
        inputYShift*inputYScale],1,length(actMapXRef).^2);
```

```
88  tmpMat = M * ([actMapXRef(:)'; actMapYRef(:)']  - centerMatrix) + ...
        centerMatrix + B;
89
90  % 2) APPLY MASK OF VALID ACTUATORS TO THAT MESHGRID
91  actMapX = zeros(size(actMapXRef));
92  actMapY = zeros(size(actMapYRef));
93  tmp = tmpMat(1,:);
94  actMapX(maskValidActuator) = tmp(maskValidActuator);
95  tmp2 = tmpMat(2,:);
96  actMapY(maskValidActuator) =  tmp2(maskValidActuator);
97
98  % In Phase Coordinates : 120x120
99  actMapX(maskValidActuator) = tmp(maskValidActuator)*nPx + 60;
100 actMapY(maskValidActuator) =  tmp2(maskValidActuator)*nPx + 60;
101 xCoord = actMapX(maskValidActuator);
102 yCoord = actMapY(maskValidActuator);
103
104 % Range of phase for SDM 103 or 104 to adapt the amplitude of the ...
        phase and calibrate the DM poke were computed Zygo data :
105 % sdmData = load('20120801_DSdm103.mat');rangeSdm103 = ...
        max(sdmData.D) - min(sdmData.D);mean(rangeSdm103)
106 if dmNumber == 103
107     amplitude = 5.9098;
108 else
109     amplitude = 7.5610;
110 end
111
112 % 4) APPLY THAT DISPLACEMENT TO PHASE ("POKE ACTUATOR" without DM)
113 phaseSize = 1:120;
114 % slope = zeros(2*wfs.nValidLenslet,nActu);
115 influenceFunOnDm = zeros(120*120,nActu);
116 sigmaX = coupling*inputXScale;  % NB for Gaussian function : FWTM ...
        = 2*sqrt(2*log(2))*sigma
117 sigmaY = coupling*inputYScale;
118 for k = 1:nActu
119     % Apply new phase here (120x120)
120     xCenter = xCoord(k);%60+(k-1)*12;
121     yCenter = yCoord(k);%60+(k-1)*12;
122     gaussian = @(x,y) ...
            amplitude*exp(-((x-xCenter).^2/(2*sigmaX.^2) + ...
            (y-yCenter).^2/(2*sigmaY.^2)));
123     gaussianValue = zeros(120);
```

```
124     for j = 1:120   % On the phase size
125         gaussianValue(j,:) = gaussian(j,phaseSize);
126     end
127     influenceFunOnDm(:,k) = gaussianValue(:);
128 end
129 invertedInfluenceFun = pinv(influenceFunOnDm(:,:));
130 
131 end
```

Appendix B

Experimental Interaction Matrices

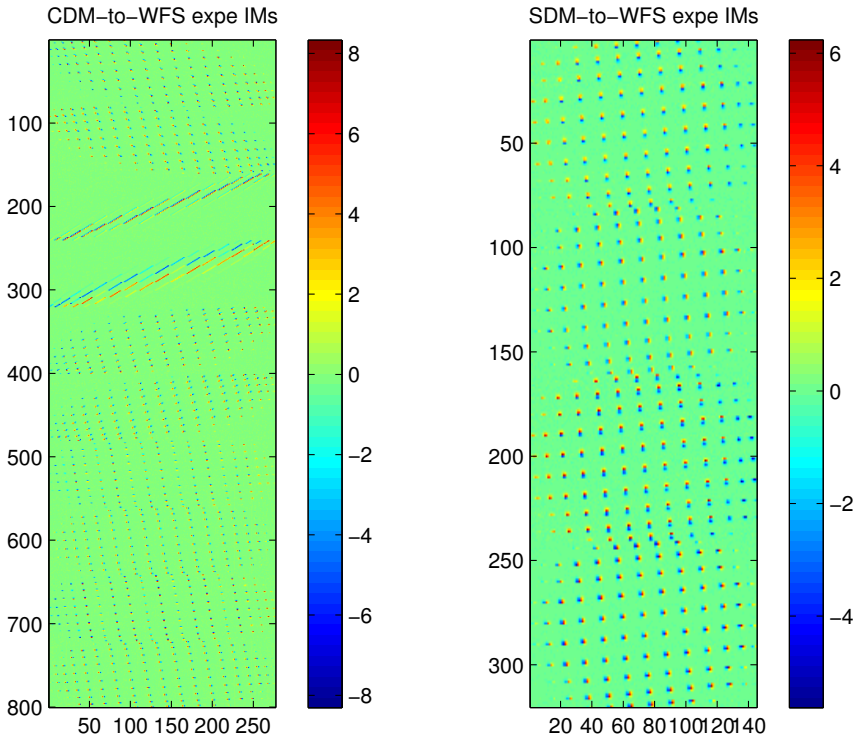


Figure B.1: Experimental Interaction Matrices.