Feedforward Compensation of Hysteresis and Creep in Adaptive Optics Devices

H.K. Bijlsma

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Feedforward Compensation of Hysteresis and Creep in Adaptive Optics Devices

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

H.K. Bijlsma

January 2, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Feedforward Compensation of Hysteresis and Creep in Adaptive Optics Devices

by

H.K. Bijlsma

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering

Dated: January 2, 2017

Supervisor(s):

Prof.dr.ir. M. Verhaegen

Dr. P. Pozzi

Reader(s):

ir. J. W. Spronck

Dr.-Ing. S. Wahls

Dr. S. Bonora

Abstract

Adaptive optics are often operated in wavefront-sensorless control fashion, since this requires no sensors. These devices often make use of piezoelectric actuators, which exhibit creep and hysteresis. In open loop operation, hysteresis and creep limit the performance. Hysteresis limits the precision of the compensation and creep reduces the performance during extended periods of time. In research, many feedforward compensation methods for these effects have been developed. In this thesis, a dynamic Generalized Prandtl-Ishlinskii model is constructed, consisting of a Generalized Prandtl-Ishlinskii model to compensate for the hysteresis and a feedforward zero phase error tracking controller to compensate the creep. The dynamic Generalized Prandtl-Ishlinskii model is tested on a deformable mirror and lens. The single-input and single-output performance is evaluated for step signals, a sawtooth signal and a more realistic signal. The multiple-input and multiple-output performance is tested for a realistic signal as well. It is found that the dynamic Generalized Prandtl-Ishlinskii model reduces the normalized root mean square of the tracking error by 38.5% on the mirror and by 53.8% on the lens for multiple-input and multiple-output operation. This reduction is larger compared to a static Generalized Prandtl-Ishlinskii model. At the end of the thesis, propositions for future improvements of the identification of the model are made.

Table of Contents

1 Introduction **3**

2 Background **5**

2-1 Adaptive optics . . . 5

2-1-1 Wavefront aberrations . . . 6

2-1-2 Shack-Hartmann sensor . . . 7

2-1-3 Applications . . . 8

2-1-4 Adaptive optics control techniques . . . 8

2-2 Piezoelectric Actuators . . . 9

2-3 Adaptive optics devices . . . 10

2-3-1 Lens . . . 10

2-3-2 Mirror . . . 11

2-4 Hysteresis . . . 12

2-5 Creep . . . 14

2-6 Previous research . . . 15

3 Methods **17**

3-1 Setup . . . 17

3-2 Reconstruction of the Zernikes . . . 18

3-3 Model . . . 20

3-4 Model inversion . . . 21

3-4-1 Inversion of the hysteresis model . . . 22

3-4-2 Inversion of the dynamic model . . . 22

3-5 Identification . . . 23

3-5-1 Influence functions . . . 23

3-5-2 Identification procedure . . . 23

3-5-3 Dataset for identification . . . 25

3-6 Compensation . . . 26

3-6-1 Single actuator . . . 27

3-6-2 Multiple actuators . . . 28

3-7 MATLAB files . . . 29

4 Results **31**
4-1 Identified models . . . 31
4-2 Compensation performance . . . 34
4-2-1 Influence of dynamics on the hysteresis identification . . . 34
4-2-2 Variables . . . 34
4-2-3 Response to simple signals . . . 35
4-2-4 Single-input and single-output . . . 37
4-2-5 Multiple-input and multiple-output . . . 39
4-2-6 Reproducibility . . . 40
4-3 Discussion . . . 41

5 Conclusion and recommendations **43**

A Zernikes in Cartesian coordinates **49**

B SISO results **51**

C MATLAB code **55**
C-1 calccentroid . . . 55
C-2 calimirror . . . 55
C-3 createinvmodel . . . 58
C-4 derzer . . . 60
C-5 Driver_ mirror . . . 63
C-6 fit_ dyn_ PI . . . 65
C-7 fitgrid . . . 66
C-8 invPI . . . 66
C-9 LocReg . . . 67
C-10 makegrid . . . 67
C-11 measzer_ mirror . . . 68
C-12 mols . . . 69
C-13 simPI . . . 70
C-14 simPI_ asym . . . 71
C-15 simPI_ asymi . . . 71
C-16 Testcomp_ mirror . . . 72

D Glossary **79**
List of Acronyms . . . 79
List of Symbols . . . 79

List of Figures

2-1 Working principle of Adaptive Optics (AO) [1] 5
2-2 Wavefronts in a lens system with a point source 6
2-3 Distorted wavefront of a point source 6
2-4 Effect of distortion of the wavefront on imaging of a point source 7
2-5 Working principle of a Shack-Hartmann (SH) sensor [3] 8
2-6 Feedforward control 8
2-7 Location of the actuators of the lens 11
2-8 Drawing of the cross section of the lens, adapted from [6] 11
2-9 Different wavefronts produced by the lens [6] 11
2-10 Schematic drawing of a Piezoelectric Deformable Mirror (PDM) [18] 12
2-11 Placement of te actuators of the mirror [18] 12
2-12 Hysteresis curve of actuator 19 of the mirror 13
2-13 The mathematical behavior of the operator of the Prandtl-Ishlinskii (PI) model $F_r[v](t)$ 14
2-14 Creep in a piezoelectric actuator [23] 15

3-1 Picture of the setup with the lens. Photo by Martino Quintavalla (Istituto di Fotonica e Nanotecnologie), used with permission 17
3-2 A schematic view of the setup with the lens. Adapted from schematic drawing provided by Martino Quintavalla (Istituto di Fotonica e Nanotecnologie), used with permission 17
3-3 Picture of the setup with the mirror 18
3-4 A schematic view of the setup with the mirror 18
3-5 The geometrical properties of the image of the SH sensor with the subaperture of spot A_k in yellow 19
3-6 The integration of the hysteresis and dynamics model 21

3-7 The dataset used for identification 26
3-8 Used steps signal for evaluation of Single-Input and Single-Output (SISO) compensation 27
3-9 Used sawtooth signal for evaluation of SISO compensation 27
3-10 The reference used for testing the SISO compensation 28
3-11 Flow scheme Multiple-Input and Multiple-Output (MIMO) compensation 29

4-1 The fit of the identified dynamic Generalized Prandtl-Ishlinskii (GPI) model to the identification dataset for actuator 19 of the mirror with 5 operators 33
4-2 Simulated step response of the fitted transfer function compared to the step response of actuator 1 of the mirror. Chosen order is 2. 33
4-3 Simulated step response of the fitted transfer function compared to the step response of actuator 11 of the mirror. Chosen order is 1. 33
4-4 The effect of (not) taking into account dynamics during identification on the reference tracking of a static GPI model with 5 operators on actuator 19 of the mirror. 34
4-5 e_{NRMS} for MIMO actuation of the lens for different amounts of operators 35
4-6 e_{NRMS} for MIMO actuation of the mirror for different amounts of operators . . . 35
4-7 Responses of the lens compensated with the different models to a signal composed of steps 36
4-8 Responses of the mirror compensated with the different models to a signal composed of steps 36
4-9 Responses of the lens compensated with the different models to a sawtooth signal 36
4-10 Responses of the mirror compensated with the different models to a sawtooth signal 36
4-11 Hysteresis loop of the lens compensated with different models for a sawtooth signal 37
4-12 Hysteresis loop of the mirror compensated with different models for a sawtooth signal 37
4-13 Reference compared to the output for a system compensated with a dynamic GPI model and no compensation during SISO actuation of actuator 1 of the lens . . . 39
4-14 Reference compared to the output for a system compensated with a dynamic GPI model and no compensation during SISO actuation of actuator 19 of the mirror . 39
4-15 Reference compared to the output for a system compensated with a dynamic GPI model, a static GPI model and no compensation during MIMO actuation on the lens 40
4-16 Reference compared to the output for a system compensated with a dynamic GPI model and no compensation during MIMO actuation on the mirror 40
4-17 Reproducibility of the lens system for SISO actuation with compensation with a dynamic GPI model with 6 operators 40
4-18 Reproducibility of the lens system for MIMO actuation with compensation with a dynamic GPI model with 6 operators 40
4-19 Reproducibility of the mirror system for SISO actuation with compensation with a dynamic GPI model with 5 operators 41
4-20 Reproducibility of the mirror system for MIMO actuation with compensation with a dynamic GPI model with 5 operators 41

List of Tables

2-1 Technical properties of the mirror PDM30-19 [18] 12

3-1 MATLAB functions and their uses 30

4-1 Variance Accounted For (VAF) values for identification of the dynamic GPI models for each actuator of the lens for the identified model with 6 operators and a dynamic model 32

4-2 VAF values for identification of the dynamic GPI models for each actuator of the mirror for the identified model with 5 operators and a dynamic model 32

4-3 Hysteresis percentage h in $\%$ for different models 37

4-4 Mean of the Normalized RMS values of the error for all actuators of the mirror . 38

4-5 Mean of the Normalized RMS values of the error for all actuators of the lens . . 38

4-6 Mean of the Normalized RMS values of the error for all actuators of the lens for the dataset multiplied by minus 1 38

A-1 Zernike polynomials in Cartesian coordinates 50

B-1 e_{NRMS} of single actuators of the lens with a GPI model without dynamics compared to a GPI model with dynamics and no compensation 52

B-2 e_{NRMS} of single actuators for the dataset multiplied by minus 1 of the lens with a GPI model without dynamics compared to a GPI model with dynamics and no compensation 53

B-3 VAF of a dynamic GPI model with 6 operators of the mirror on a validation dataset 54

B-4 VAF of a dynamic GPI model with 6 operators of the lens on a validation dataset 54

Preface

I would like to express my gratitude to my supervisors Hans Verstraete and Paolo Pozzi for the daily support. Also, I would like to thank Michel Verhaegen for his guiding questions. Finally, i'd like to thank Stefano Bonora for supplying the lens and Martino Quintavalla for helping with the measurements.

Chapter 1

Introduction

In this thesis, methods for compensating hysteresis and creep are tested on a deformable lens and a deformable mirror. This has been done with the goal to improve the performance of the devices. Many methods for hysteresis and creep compensation have been described in literature. The focus of this thesis is on Feedforward (FF) methods, since FF methods do not require sensors to perform their function. The following research questions will be answered in this report:

- To what extent can a Generalized Prandtl-Ishlinskii (GPI) model in series with a dynamic model compensate hysteresis and creep in a deformable lens?
- To what extent can a GPI model in series with a dynamic model compensate hysteresis and creep in a deformable mirror?
- What is the advantage of adding a dynamic model in series with a GPI model for compensation of hysteresis and creep in a deformable lens and a deformable mirror?
- Are there differences in the compensation of hysteresis and creep between the deformable lens and deformable mirror?

The outline of the thesis is as follows:

- **Chapter 2 Background** explains some fundamentals of Adaptive Optics (AO) and its applications. Also, some details are given on the lens and mirror used in the experiments. Then, information is given on compensation methods for hysteresis and creep. Finally, the state of the art in hysteresis and creep compensation in AO systems is summarized.
- **Chapter 3 Methods** describes the experimental setups used and shows how measurements with the Shack-Hartmann (SH) sensor are converted to Zernikes. The hysteresis and dynamics models used are described and the model identification and compensation methods are introduced.

- **Chapter 4 Results** shows the identified models and their characteristics. Subsequently, the results of the compensation are shown for both the operation of one actuator at the time (Single-Input and Single-Output (SISO)) as well as for operation of multiple actuators at the same time (Multiple-Input and Multiple-Output (MIMO)). The results are compared to the same experiments without compensation and a static hysteresis compensation method. This is done for both a deformable lens and a deformable mirror.
- **Chapter 5 Conclusion and recommendations** draws conclusions and recomentations based on the results.

At the back of the thesis, all used symbols and abbreviations are listed.

Chapter 2

Background

In this chapter, the principle of Adaptive Optics (AO) is introduced and Piezoelectric Actuators (PEAs) are discussed. Then, the devices used are described, with particular focus on the hysteresis and creep behaviors of PEAs. Finally, previous research on the compensation of hysteresis and creep is summarized.

2-1 Adaptive optics

The field of AO deals with the compensation of disturbances in the wavefront of optical systems. This is done by manipulating the incoming wavefront in such a way that it precisely compensates the disturbances in the incoming wavefront, see Figure 2-1.

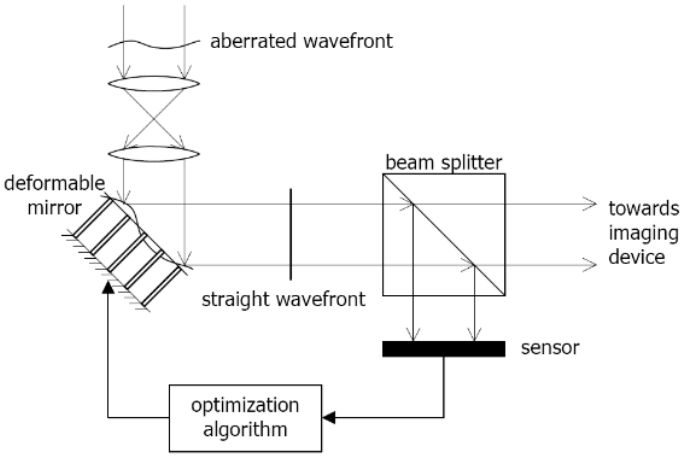

Figure 2-1: Working principle of AO [1]

2-1-1 Wavefront aberrations

Some properties of light can be explained by expressing them as waves. Therefore, light from a coherent and monochromatic source can be described by its amplitude and phase. The wavefront is defined as the surface on which light has the same phase.

For example, in the case of a point source emitting light at a given wavelength, a spherical wavefront is emitted see Figure 2-2, due to the constant speed of light in a homogeneous medium. Based on the laws of refraction, a lens can change the shape of the wavefront. This principle can be used to generate an image, by converting diverging spherical wavefronts to converging ones.

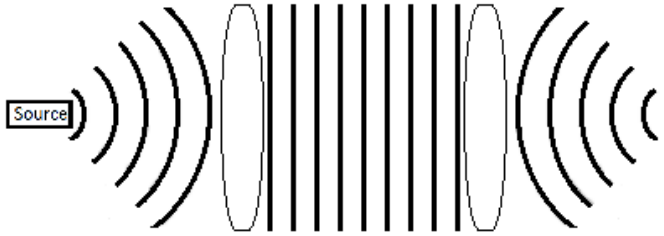

Figure 2-2: Wavefronts in a lens system with a point source

However, in many applications imaging is hindered by the fact that the wavefront is not spherical. This can be caused by inhomogeneities in the medium through which the light propagates, see Figure 2-3.

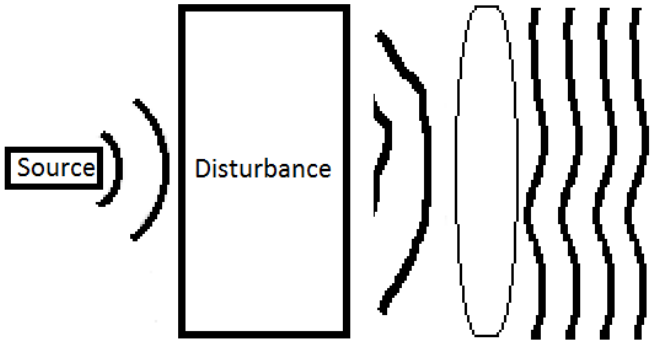

Figure 2-3: Distorted wavefront of a point source

These distortions of the wavefront will cause the image of the point source to become blurry, see Figure 2-4. Distortions of the wavefront are called aberrations.

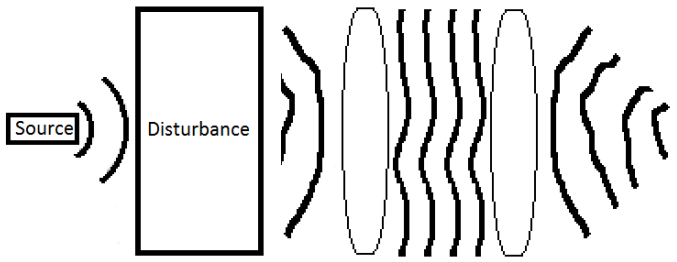

Figure 2-4: Effect of distortion of the wavefront on imaging of a point source

Aberrations of the wavefront can be expressed as series of Zernike polynomials Z_u^v [2]. These are orthogonal polynomials, expressed on the unit disk, which are defined as follows:

$$Z_u^v(\rho,\theta) = \sqrt{2(u+1)}R_u^v(\rho,\theta)\cos v\theta \qquad \text{even terms when } v \neq 0 \qquad (2\text{-}1)$$

$$Z_u^v(\rho,\theta) = \sqrt{2(u+1)}R_u^v(\rho,\theta)\sin v\theta \qquad \text{odd terms when } v \neq 0 \qquad (2\text{-}2)$$

$$Z_u^v(\rho,\theta) = \sqrt{(u+1)}R_u^v(\rho,\theta) \qquad \text{when } v = 0 \qquad (2\text{-}3)$$

with

$$R_u^v(\rho,\theta) = \sum_{s=0}^{\frac{u-v}{2}} \frac{(-1)^s(u-s)!}{s!(\frac{u+v}{2}-s)!(\frac{u-v}{2}-s)!}\rho^{u-2s} \qquad (2\text{-}4)$$

and the radius ρ and angle θ which define a point on the unit disk. Zernikes can also be expressed in Cartesian coordinates. These expressions are given in Appendix A. There, the numbering of the Zernikes that is used in this report is also given.

2-1-2 Shack-Hartmann sensor

In the setup, a Shack-Hartmann (SH) sensor is used to measure the wavefront (aberrations). The working principle of a SH sensor is shown in Figure 2-5. In [3] a more detailed description of the working principle is given. The SH sensor is formed by an array of microlenses and a camera detector, and needs a point source to operate. Due to wave propagation, the light incident on a single lenslet will focus on the detector at a distance which is equal to the focal length F of the lenslets, with a lateral displacement proportional to the average gradient of the wavefront over the lenslet surface. From the average gradients measured on all lenslets, the shape of the wavefront can be reconstructed. This procedure will be explained in more detail in Section 3-2.

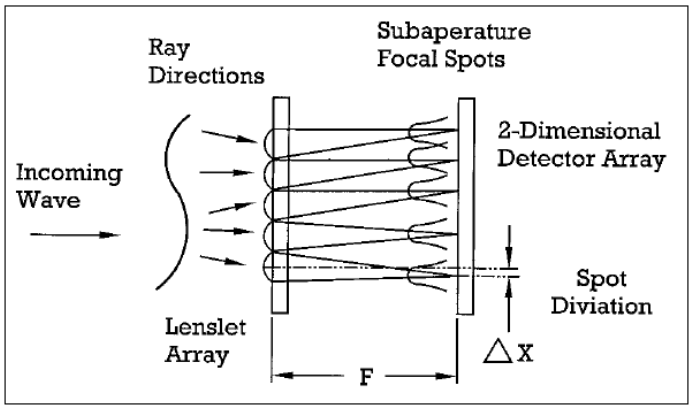

Figure 2-5: Working principle of a SH sensor [3]

2-1-3 Applications

The main application of AO is astronomy, where the presence of point-like bright stars enables the use of a SH sensor for real time correction of the atmospheric turbulence. However, there are many other applications of AO. In [4], applications in multiphoton, confocal, structured illumination, light sheet-microscopy and STED microscopy are mentioned. In [5] applications in three-dimensional displays and zoom systems are mentioned. The lens that is used for experiments in this thesis has been used for OCT [6].

2-1-4 Adaptive optics control techniques

Generally, there are two kinds of control used in AO. These are feedback and Feedforward (FF) control. Feedback (FB) control is control that makes use of the signals from sensors (as shown in Figure 2-1) to drive the output to a given reference. FF control is control which constructs the input from the reference (so without sensors), in such a way that the output y tracks the reference ref (see Figure 2-6).

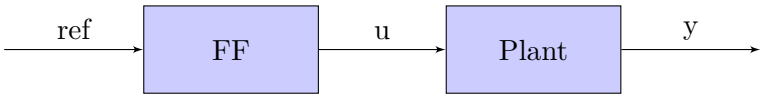

Figure 2-6: Feedforward control

In AO, several correction techniques have been developed to apply control without the use of a wavefront sensor. A motivation for this is that guide stars, which provide the signal that the SH sensor uses, are not always available. Other motivations are lower costs, a simpler system and a higher signal to noise ratio. This higher signal to noise ratio is achieved by using all photons for imaging, instead of using part of them for wavefront sensing. This leads to a shorter exposure time and/or a better image quality.

Without a wavefront sensor, aberrations can not be measured directly. Therefore, the extent to which they are compensated by the AO is unknown. In order to deal with this, different

metrics have been developed to provide information about the correction performance, but do not require a wavefront sensor [7, 8, 9, 10]. This signal can serve as feedback signal for control.

Wavefront sensorless techniques can be divided in different categories:

- Model free
  By taking measurements and applying a derivative-free optimization algorithm on the input u, a metric is maximized [11, 12], see Equation 2-5. Examples of optimization algorithms that can be used are simulated annealing or the SPGD algorithm [13] and an example of a metric is intensity I.

$$u = \arg\max_{u} I(u) \tag{2-5}$$

- Model based
  First, an estimate of the current aberrations in Zernike modes z_{est} is made on basis of measurements. When this is completed, control is applied which compensates those aberrations, see Equation 2-6. In order to be able to do this, the influence functions of the control on the aberrations have to be measured during calibration of the system. Subsequently, the estimate of the aberrations is updated and the new estimated optimal control is applied. The advantage of this technique is that the optimization is faster compared to model free methods [12, 14].

$$u = f^{-1}(z_{est}) \text{ where } f(u) = z \text{ is the model of the AO device} \tag{2-6}$$

In this thesis, FF methods for hysteresis and creep compensation in wavefront sensor-less AO applications are developed. To evaluate the performance of the methods, a wavefront sensor is used. However, this sensor is not longer necessary when these compensation methods are used in a system.

2-2 Piezoelectric Actuators

In AO, the most common adaptive elements used are deformable mirrors [6]. However, recently a deformable lens has been developed as well [15]. Replacing a deformable mirror with a deformable lens offers the advantage that a lens is much easier (and therefore cheaper) to implement in an existing system. However, deformable mirrors have the advantages that they have a higher optical efficiency and no chromatic aberration [4]. These AO devices mostly make use of PEAs, because of their fast response and high resolution [16].

PEAs are actuators made of a piezoelectric material, which exhibits strain depending on the voltage applied. The direction in which this strain occurs is determined by the manufacturing process. The general model that is used for PEAs is in matrix form [17]:

$$[\delta] = [s^E][\sigma] + [d]'[E] \tag{2-7}$$

$$[D] = [d][\sigma] + [\epsilon^T][E] \tag{2-8}$$

with as inputs the stress σ and the electric field E. The outputs are the displacement δ and the dielectric displacement D. The other variables are the compliance s^E while a constant electric field is applied, the piezoelectric constant d and the permittivity ϵ^T while a constant mechanic force is applied.

However, this linear model is unable to describe the nonlinear effects and dynamics, including creep, in PEAs. Two of these nonlinear effects that occur in PEAs are hysteresis and saturation. These dynamics and nonlinear effects cause tracking errors [1, 16] which can degrade the performance of the system in which they are used. In AO, these will result in errors in the wavefront correction which is produced with the AO device [1]. Various methods to deal with the hysteresis are described in literature [16]. Most of the compensation techniques make use of (inverse) models of these (non)linear effects. The methods used in this thesis to compensate the hysteresis and the dynamics are explained in Section 3-3.

2-3 Adaptive optics devices

In this section, the physical structure of the lens [15] and mirror [18] are investigated. Also, the mechanics are briefly discussed.

2-3-1 Lens

The lens used in the experiments consists of two transparent solid surfaces with a transparent fluid in between [6]. A piezoelectric ring is placed on the outside of both solid layers. Both rings are divided in 9 equal sections, which can be actuated separately. This gives a total of 18 actuators. The location of these actuators in the solid layers is shown in Figure 2-7. The cross section of the lens is shown in Figure 2-8. The piezoelectric ring is made as a bending actuator. An applied voltage makes the inside of the piezoelectric ring bend up or down and therefore displaces the layer of liquid in between. The difference between the two solid layers is that one side is constrained in the movement, while the other side is connected to an elastomer foam. This causes the actuators on the different solid layers to have different influences on the wavefront. Unlike previously reported lenses implantations [15], the one reported can generate aberrations up to the fourth order [15], see Figure 2-9.

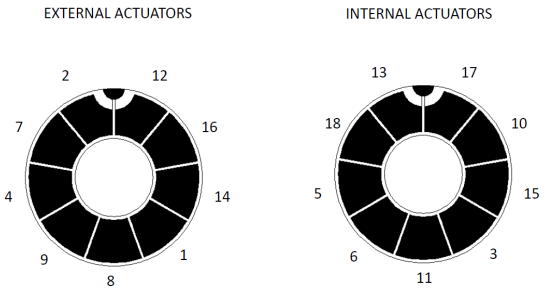

Figure 2-7: Location of the actuators of the lens

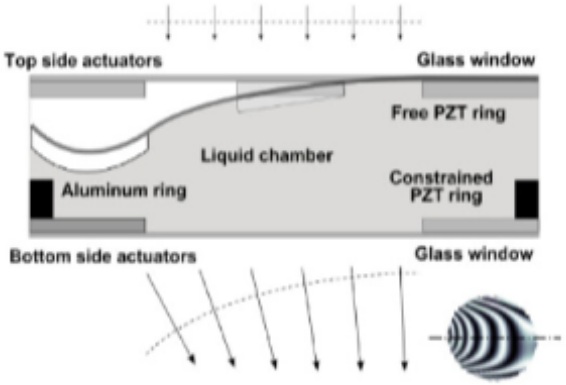

Figure 2-8: Drawing of the cross section of the lens, adapted from [6]

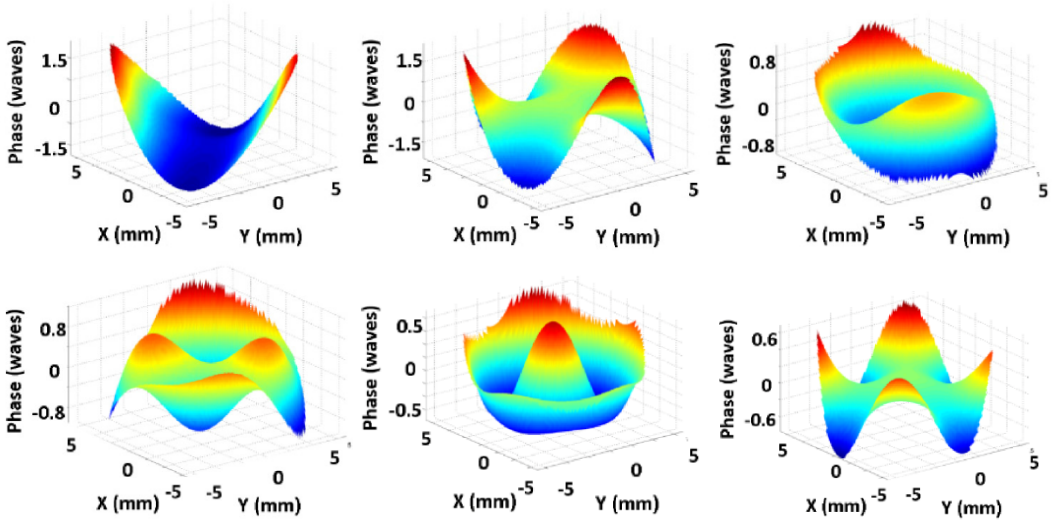

Figure 2-9: Different wavefronts produced by the lens [6]

2-3-2 Mirror

The mirror used in the experiments is a Piezoelectric Deformable Mirror (PDM) from OKOTech (30mm 19ch PDM [18]). Technical data on this mirror can be found in Table 2-1. It consists of a base on which PEAs are placed. On these, a flexible faceplate is mounted, see Figure 2-10. By applying a voltage to the PEAs, the shape of the faceplate can be changed. The actuators of the mirror used for the experiments are placed in a hexagonal pattern, shown in Figure 2-11. According to its specifications, it can correct aberrations up to the fourth order [18].

Aperture shape	circular 30 mm in diameter
Mirror coating	Metal or Metal + dielectric
Actuator voltages	0 . . . + 300V (with respect to the ground electrode)
Number of electrodes	19
Actuator capacitance C_a	5 nF
Main initial aberration	sphere
Initial Root Mean Square (RMS) deviation	less than 1 μm from reference sphere
Maximum stroke	8μm at +400V
Actuator pitch	7 mm
Weight	320 g

Table 2-1: Technical properties of the mirror PDM30-19 [18]

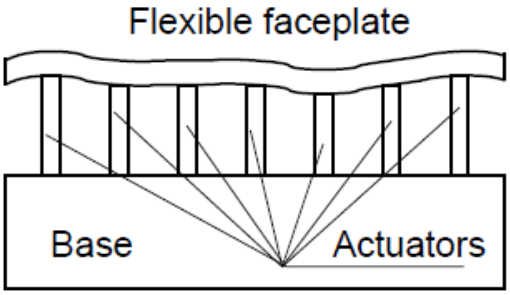

Figure 2-10: Schematic drawing of a PDM [18]

Figure 2-11: Placement of te actuators of the mirror [18]

2-4 Hysteresis

In this section the hysteresis phenomenon and its properties are investigated. Compensation methods for hysteresis are then discussed. Subsequently, different hysteresis models are explained and their properties and implementation are discussed.

The hysteresis curve is defined as the curve of the output y as function of the input u. An example of a hysteresis curve of an actuator in the mirror is shown in Figure 2-12. Hysteresis can also be expressed as a percentage h, which is the maximum size of the opening of the hysteresis curve divided by the full stroke:

$$h = \frac{\max(y_d(u) - y_a(u))}{\max(y) - \min(y)} \text{ for any } u \tag{2-9}$$

where y_d is the output at the interval where $\dot{u} < 0$ and y_a is the output at the interval where $\dot{u} > 0$.

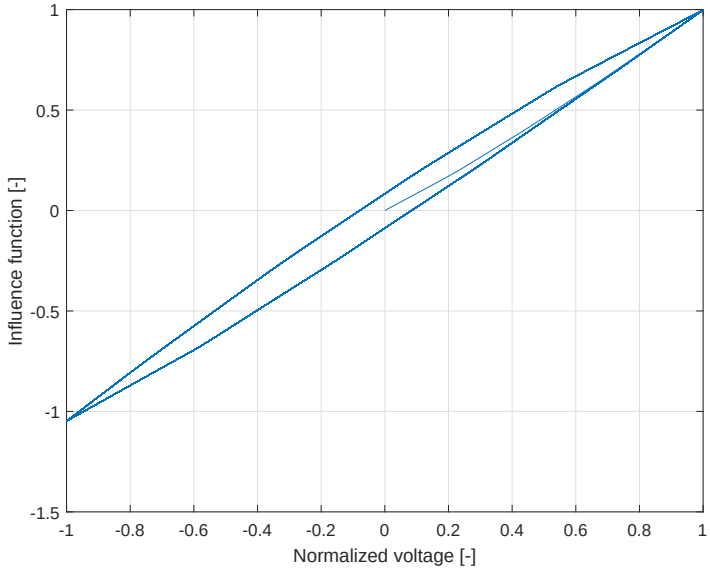

Figure 2-12: Hysteresis curve of actuator 19 of the mirror

Prandtl-Ishlinskii model

The Prandtl-Ishlinskii (PI) model is composed of play operators F_r, and is defined as follows [16]:

$$y(t) = p_0 v(t) + \int_0^\infty p(r) F_r[v](t) dr \tag{2-10}$$

The definition of the play operator F_r is:

$$F_r[v](0) = f_r(v(0), 0) \tag{2-11}$$
$$F_r[v](t) = f_r(v(t), F_r[v](t_i)) \tag{2-12}$$
$$f_r(v, F_r[v]) = max(v - r, min(v + r, F_r[v])) \tag{2-13}$$

where $p(r)$ is an arbitrary function determining the weight of the operators and t_i is the previous time instant.

A geometric illustration of the play operator is shown in Figure 2-13

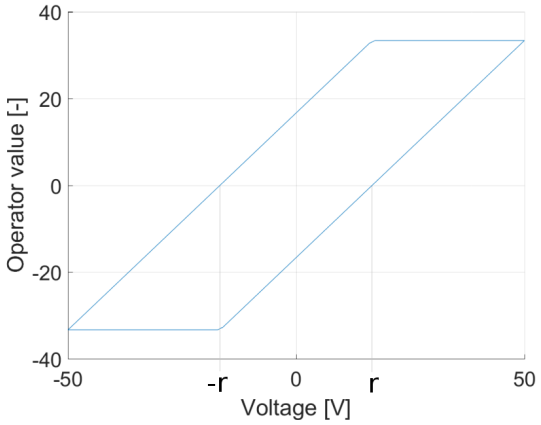

Figure 2-13: The mathematical behavior of the operator of the PI model $F_r[v](t)$

The PI has the advantages that it is invertible [19] and that it can represent asymmetric hysteresis curves by turning it into a Generalized Prandtl-Ishlinskii (GPI) model [20].

The infinite integral complicates the implementation of the PI model [16]. Therefore, the integral is mostly expressed as a sum of a finite amount of operators with the threshold and weight functions expressed as vectors [21], see Equation 2-14.

$$\int_0^\infty p(r)F_r[v](t)dr \approx \sum_{i=1}^{n} p_i F_{r_i}[v](t) \tag{2-14}$$

However, this leads to a less smooth hysteresis curve, since a finite amount of functions are now used to approximate the hysteresis. The smoothness and fit of the hysteresis can be improved by using more terms, but requires a higher computational cost. [22]

2-5 Creep

Creep is the phenomenon causing a small drift-like behavior in the output of the system, while the input is constant. This occurs in the low frequency range. The difference between creep and drift is that the direction and size of the creep can be predicted based on the history of the input [23], while drift has an unpredictable behavior. PZT actuators are known to exhibit creep [23]. An example of creep in a PEA is given in Figure 2-14.

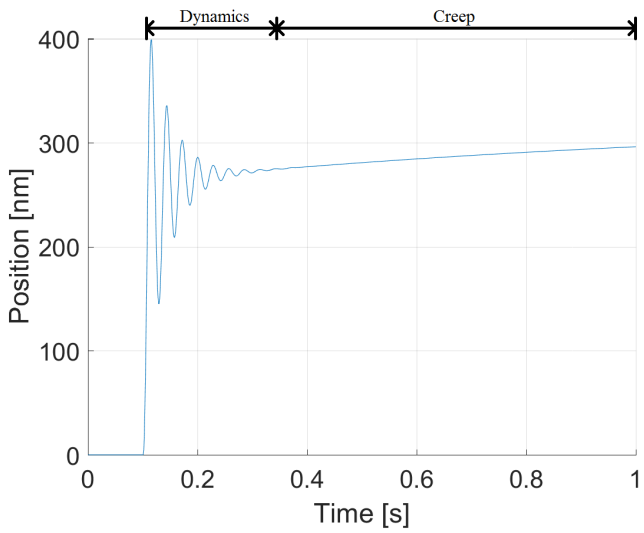

Figure 2-14: Creep in a piezoelectric actuator [23]

In applications in which FB is applied, the inaccuracies which creeps leads to are small, because it is measured and the FB controller compensates for it [24]. However, when only FF is applied and the dynamics of the creep are not considered, the creep is not measured and can lead to inaccuracies [25].

Different logarithmic models have been developed for creep [23, 25]. Also, linear dynamics can model creep as well [26]. Due to its simplicity, the linear model will be used in this thesis and the creep will be modeled by a transfer function.

2-6 Previous research

Previous research has been done on the compensation of creep and hysteresis of an actuator in a deformable mirror [27]. Also, inverse hysteresis compensation has been applied in control with both FF and FB on a deformable mirror [28]. Adaptive identification [29, 30] and identification based on least squares [31] are mentioned as methods to identify the hysteresis and creep models. Different kinds of datasets are used for this purpose. [31] makes use of random signals with no spectral density above a specific frequency, while most rely on a sinusoid with a varying amplitude, for example [26, 32]. However, the frequency of this sinusoid should be chosen carefully such that the dynamics or creep do not influence the identification of the hysteresis [32]. For the identification of creep, step functions can be used [32]. Furthermore, some research identifies the hysteresis, dynamics and creep from one dataset [31], while other research [26] first identifies a hysteresis compensator and then identifies the creep on a dataset which is already compensated for the hysteresis. The dynamics are then identified on a dataset compensated for hysteresis and creep.

Chapter 3

Methods

In this chapter, the setup is described and the model structure of the compensator is explained. Then, the identification of this model is discussed and the way in which the compensation is evaluated is presented. At the end of the chapter an overview of the MATLAB codes employed is reported.

3-1 Setup

The hysteresis compensation method is tested on an adaptive lens [15] and a deformable mirror from OKOTech (30mm 19ch PDM [18]). A picture of the test setups can be seen in Figure 3-1 and Figure 3-3. Schematic overviews of these setups are shown in Figures 3-2 and 3-4.

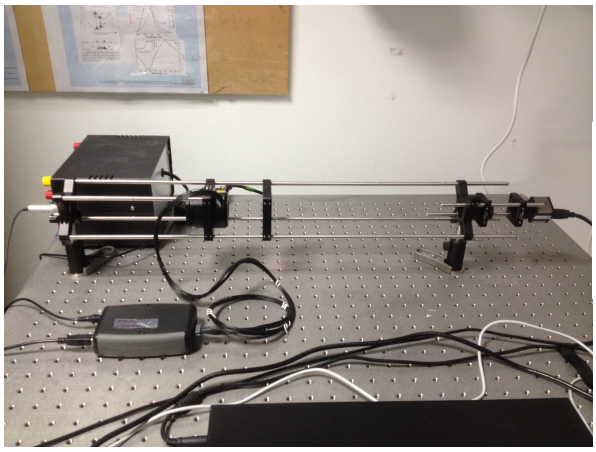

Figure 3-1: Picture of the setup with the lens. Photo by Martino Quintavalla (Istituto di Fotonica e Nanotecnologie), used with permission

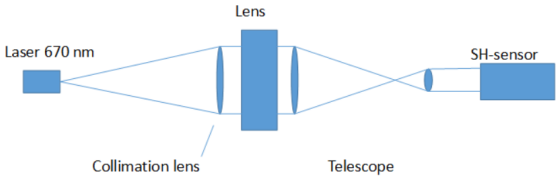

Figure 3-2: A schematic view of the setup with the lens. Adapted from schematic drawing provided by Martino Quintavalla (Istituto di Fotonica e Nanotecnologie), used with permission

Figure 3-3: Picture of the setup with the mirror

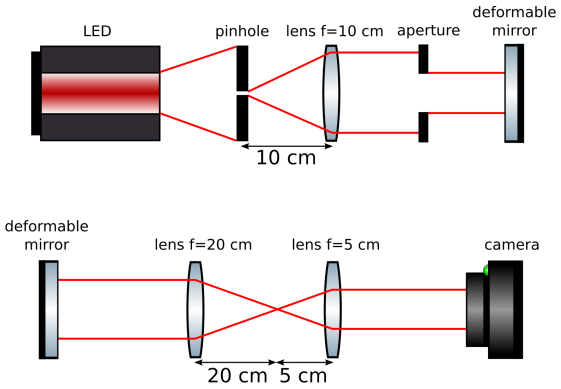

Figure 3-4: A schematic view of the setup with the mirror

In the mirror setup, a pinhole of 30 μm is placed in front of the source to provide a point like light source. In order to make the wavefront flat, a lens is placed after the pinhole. This lens has a focal length of 10 cm and is equal to its distance to the pinhole. Then, an aperture is used which limits the size of the beam to 2.5 cm such that only the active part of the mirror is shined upon. After the light beam has reflected off the mirror (path below in Figure 3-4), a telescope is used to make sure that the beam with a diameter of 2.5 cm is reduced to a diameter of 0.5 cm, which can be registered by the Shack-Hartmann (SH) sensor. A similar setup is used for the lens.

3-2 Reconstruction of the Zernikes

The camera in the SH sensor measures the intensity as a number between 0 and 255. The image that is captured is thresholded so that all intensity values of $I_{min_{centroid}}$ or less are set to 0. Then, the centroids I_x and I_y of the spots (recognized as a square with at least one pixel with a brightness value of $I_{min_{spot}}$ or higher during calibration), are determined as:

$$I_x = \frac{\sum_x (x \sum_y I(x,y))}{\sum_x \sum_y I(x,y)} \tag{3-1}$$

$$I_y = \frac{\sum_y (y \sum_x I(x,y))}{\sum_x \sum_y I(x,y)} \tag{3-2}$$

The size of the edge of the square that is taken into account to calculate the centroid is $r_{centroid}$. Also, the minimum distance between spots that are detected is $r_{centroid_{min}}$. The parameters $I_{min_{centroid}}$, $I_{min_{spot}}$, $r_{centroid}$ and $r_{centroid_{min}}$ are setup dependent and should be determined every time a new setup is used.

Then, reference positions are calculated for these centroids by fitting a grid of equidistant points while the actuators of the Adaptive Optics (AO) device are at mid stroke. Subsequently, the displacement of these centroids I_x and I_y compared to their reference position can be used

to determine the Zernike expansion of the wavefront. This reconstruction is explained in the rest of this section.

The displacement of the centroids S are related to the Zernikes z as follows:

$$S = Z_{xy} z \tag{3-3}$$

with

$$Z_{xy} = \begin{bmatrix} \left(\frac{dZ_1}{dx}\right)_1 & \left(\frac{dZ_2}{dx}\right)_1 & \cdots & \left(\frac{dZ_N}{dx}\right)_1 \\ \left(\frac{dZ_1}{dy}\right)_1 & \left(\frac{dZ_2}{dy}\right)_1 & \cdots & \left(\frac{dZ_N}{dy}\right)_1 \\ \left(\frac{dZ_1}{dx}\right)_2 & \left(\frac{dZ_2}{dx}\right)_2 & \cdots & \left(\frac{dZ_N}{dx}\right)_2 \\ \left(\frac{dZ_1}{dy}\right)_2 & \left(\frac{dZ_2}{dy}\right)_2 & \cdots & \left(\frac{dZ_N}{dy}\right)_2 \\ \vdots & \vdots & \ddots & \vdots \\ \left(\frac{dZ_1}{dx}\right)_k & \left(\frac{dZ_2}{dx}\right)_k & \cdots & \left(\frac{dZ_N}{dx}\right)_k \\ \left(\frac{dZ_1}{dy}\right)_k & \left(\frac{dZ_2}{dy}\right)_k & \cdots & \left(\frac{dZ_N}{dy}\right)_k \end{bmatrix} \tag{3-4}$$

Where $\left(\frac{dZ_N}{dx}\right)_k$ and $\left(\frac{dZ_N}{dy}\right)_k$ are the average derivatives of the Zernike term N to respectively x or y over the subaperture of spot k [33], see Figure 3-5.

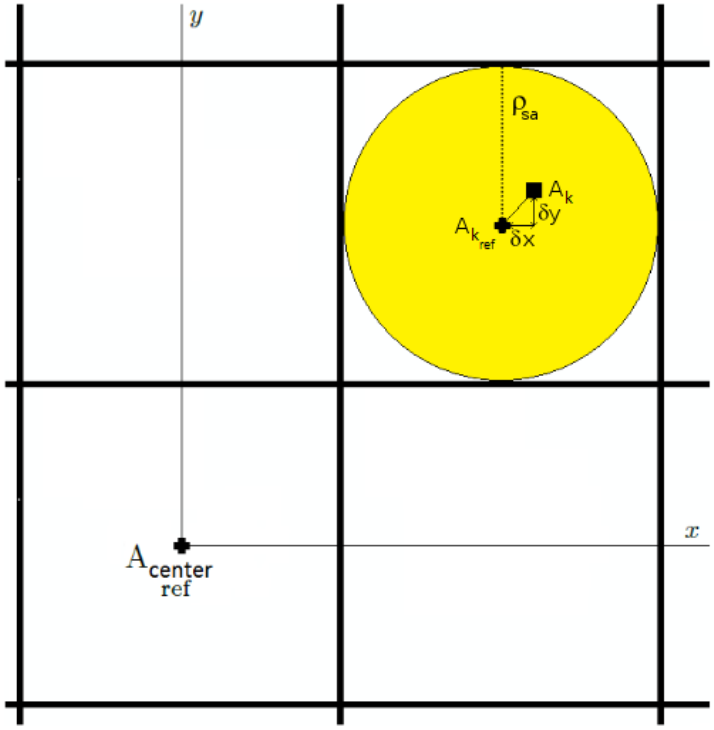

Figure 3-5: The geometrical properties of the image of the SH sensor with the subaperture of spot A_k in yellow

In Equations 2-1 to 2-3 the Zernikes are defined in polar coordinates. These can also be represented in Cartesian coordinates, see Appendix A. From these the derivatives of the Zernike terms with respect to dx and dy can be obtained for a spot at position $A_{k_{ref}}$. These are [33]:

$$\left(\frac{dZ_i}{dx}\right)_{A_k} = \int_{x_{A_{k_{ref}}}-\rho_{sa}}^{x_{A_{k_{ref}}}+\rho_{sa}} \int_{-\sin\left(\cos^{-1}\left(\frac{x-x_{A_{k_{ref}}}}{\rho_{sa}}\right)\rho_{sa}\right)+y_{A_{k_{ref}}}}^{\sin\left(\cos^{-1}\left(\frac{x-x_{A_{k_{ref}}}}{\rho_{sa}}\right)\rho_{sa}\right)+y_{A_{k_{ref}}}} \frac{dZ_i}{dx}\ dydx \tag{3-5}$$

$$\left(\frac{dZ_i}{dy}\right)_{A_k} = \int_{x_{A_{k_{ref}}}-\rho_{sa}}^{x_{A_{k_{ref}}}+\rho_{sa}} \int_{-\sin\left(\cos^{-1}\left(\frac{x-x_{A_{k_{ref}}}}{\rho_{sa}}\right)\rho_{sa}\right)+y_{A_{k_{ref}}}}^{\sin\left(\cos^{-1}\left(\frac{x-x_{A_{k_{ref}}}}{\rho_{sa}}\right)\rho_{sa}\right)+y_{A_{k_{ref}}}} \frac{dZ_i}{dy}\ dydx \tag{3-6}$$

where ρ_{sa} is the radius of the subaperture and $x_{A_{k_{ref}}}$ and $y_{A_{k_{ref}}}$ are the x and y position of the reference spot.

Solving Equation 3-3 in least squares sense gives:

$$z = (Z_{xy}^T Z_{xy})^{-1} Z_{xy}^T S = BS \ \ [m] \ \text{ with} \tag{3-7}$$

$$S = \frac{1}{F}\begin{bmatrix} \delta x_1 & \delta y_1 \\ \vdots & \vdots \\ \delta x_k & \delta y_k \end{bmatrix} \ \ (\text{see Figure } 2-5 \text{ and } 3-5) \tag{3-8}$$

The procedure in which the matrix B relating the displacement of the centroids to the Zernike modes causing them is calculated during the calibration of the AO device.

During the experiments, 27 Zernikes (up to the 6th order and not including the piston) are measured. Since it is the purpose in this thesis to let the induced aberrations follow a reference, the aberrations that are present in the setup are not relevant. Therefore these are determined at the start of each measurement by taking the mean of 10 measurements while the actuators are at mid stroke, and subtracted from every measurement.

3-3 Model

The dynamic hysteresis compensation model consists of two parts: A Generalized Prandtl-Ishlinskii (GPI) model [20] H to model the hysteresis and a transfer function G to model the dynamics (including creep). The description of the hysteresis model H is:

$$H(u(t)) = g_4 + p_0 g(u(t)) + \sum_{j=1}^{N} p_j(r) F_{r_j}[g(u)](t) \tag{3-9}$$

Where the play operators (see Figure 2-13 for the behavior of play operators) are defined as:

$$F_{r_j}[g(u)](0) = f_{r_j}(g(u(0)), g(0)) \tag{3-10}$$
$$F_{r_j}[g(u)](t) = f_{r_j}(g(u(t)), F_{r_j}[g(u)](t_i)) \tag{3-11}$$
$$f_{r_j}(g(u), F_{r_j}[g(u)]) = max(g(u) - r_j, min(g(u) + r_j, F_{r_j}[g(u)])) \tag{3-12}$$

In order to account for asymmetry of the hysteresis curve, the polynomial function $g(u)$ is introduced as:

$$g(u) = g_1 u^2 + g_2 u + g_3 \tag{3-13}$$

Then a constant g_4 is added to the model to make sure that the output y is 0 for an input u of 0:

$$g_4 = -H(g(0)) \tag{3-14}$$

The dynamics are modeled by the transfer function $G(q)$:

$$G(q) = \frac{b_1 q^n + \cdots + b_{n-2} q + b_{n-1}}{a_1 q^n + \cdots + a_{n-1} q + a_n} \tag{3-15}$$

The models are integrated as shown in Figure 3-6. This results in the mathematical description:

$$y = G\left(q\right) * H\left(g\left(u\left(q\right)\right)\right) \tag{3-16}$$

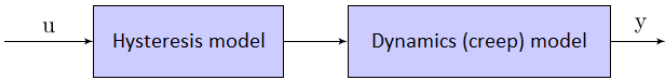

Figure 3-6: The integration of the hysteresis and dynamics model

In this thesis, a dynamic GPI model refers to a GPI model in series with a dynamic model and a static GPI model refers to a GPI model alone.

3-4 Model inversion

In order to perform compensation of the hysteresis and the dynamics, both the GPI model and the dynamic model have to be inverted. In this section, it is described how an inverse of these models can be estimated. The full inverse model is:

$$u(q) = H^{-1}\left(G_{ZPET}(q) * y(q)\right) \tag{3-17}$$

In the next two sections, it is explained how H^{-1} and $G_{ZPET}(q)$ are constructed.

3-4-1 Inversion of the hysteresis model

When the GPI model is given as in Equation 3-9, the inverse can be constructed according to [20]:

$$H^{-1}(y(k)) = g^{-1}\left(p_0^{-1}(u(k) - g_4) + \sum_{j=1}^{m} \hat{p}_j F_{\hat{r}_j}[u(k) - g_4]\right) \tag{3-18}$$

This inverse H^{-1} is also a hysteresis model with thresholds $\hat{r}_j$ and weights $\hat{p}_j$:

$$\hat{r}_j = p_0 r_j + \sum_{i=1}^{j-1} p_i(r_j - r_i) \tag{3-19}$$

$$\hat{p}_j = -\frac{p_j}{(p_0 + \sum_{i=1}^{j} p_i)(p_0 + \sum_{i=1}^{j-1} p_i)} \tag{3-20}$$

3-4-2 Inversion of the dynamic model

An approximate inverse of the transfer function can be constructed with Zero Phase Error Tracking (ZPET) Feedforward (FF) control [34]. This is done by using the poles P of the transfer function of the system as zeros and the zeros Z_s of the original system (Equation 3-21) as poles.

$$G(q) = K\frac{(q - Z_{s_1})(q - Z_{s_2})\dots(q - Z_{s_n})(q - Z_{us_1})(q - Z_{us_2})\dots(q - Z_{us_n})}{(q - P_1)(q - P_2)\dots(q - P_n)} \tag{3-21}$$

However, in case a zero is located outside the unit circle (Z_{us}), it not used as a pole, as it would make the system unstable. Instead, this zero is kept as zero multiplied by -1. This results in the transfer function in Equation 3-22, which has the same phase as the perfect inverse:

$$G_{inv_{phase}}(q) = \frac{(q - P_1)(q - P_2)\dots(q - P_n)(q + Z_{us_1})(q + Z_{us_2})\dots(q + Z_{us_n})}{(q - Z_{s_1})(q - Z_{s_2})\dots(q - Z_{s_n})} \tag{3-22}$$

Also, in order to make sure that the inverse has more poles than zeros, η high frequency poles are added, see Equation 3-23.

$$G_{inv_{proper}}(q) = \frac{G_{inv_{phase}}(q)}{(q - P_{\omega n_1})(q - P_{\omega n_2})\dots(q - P_{\omega n_\eta})} \tag{3-23}$$

Finally, the transfer function is multiplied by a factor such that the dc-gain of the inverse transfer function is equal to 1 divided by the dc-gain of the original transfer function. The final transfer function that will be used as inverse is:

$$G_{ZPET}(q) = \frac{G_{inv_{proper}}(q)}{G_{proper}(1)G(1)} \tag{3-24}$$

3-5 Identification

The identification is performed with a specifically designed dataset, described in Section 3-5-3. An algorithm that is based on least squares is used to identify a dynamic Prandtl-Ishlinskii (PI) model and serves as an initial guess of the parameters of the GPI model. The values of the transfer function are then determined by fitting a transfer function to a step response. Subsequently, the parameters of the GPI model are fitted while the transfer function is kept constant. This is done to prevent the dynamics and creep from influencing the identified hysteresis. In section 4-2-1 the effects of neglecting the dynamics in this estimation are shown. As mentioned before, the weights of the regular PI model are used as initial guess.

3-5-1 Influence functions

In order to simplify the modeling of the system, it is assumed that the effect of each actuator on the system is not influenced by the input on the other actuators. Also, it is assumed that the influence of an actuator on the wavefront is linear, and can be described as a series of Zernike coefficients. The vector of coefficients is called the influence function of the actuator.

The determination of the influence functions ζ is shown in Equation 3-25. $z(u_j)$ is a vector containing 27 Zernikes (up to the 6th order and not including the piston) and u_j is the voltage on the j'th actuator. In order to reduce the influence of noise on the determination of the influence function, the Zernikes $z(u_j)$ are measured for 31 successive instances and the mean value is used in the calculation. This is done for both the maximum and minimum voltage, see Figure 3-7. After determining the influence function, its magnitude can be considered the output y of the actuator (such that the system becomes Single-Input and Single-Output (SISO) when only one actuator is used). The determination of the magnitude of the influence function of the j'th actuator y_j is done by solving the least squares problem in Equation 3-26.

$$\zeta_j = 0.5 \left(\frac{z(u_{j_{min}})}{u_{j_{min}}} + \frac{z(u_{j_{max}})}{u_{j_{max}}} \right) \tag{3-25}$$

$$\min_{y_j} \ ||\zeta_j y_j(k) - z(k)||_2 \tag{3-26}$$

3-5-2 Identification procedure

First, the procedure in [31] is used to identify the parameters of a symmetric PI model. The values found are used to provide a starting point for the optimization of the GPI model in series with a dynamic model. The thresholds r of the symmetric PI model are calculated with Equation 3-27 and 3-28. This formula is used, because it provides an equidistant spread of the threshold values r_i.

$$r_0 = 0 \tag{3-27}$$

$$r_i = \frac{\max(u)i}{N+1} \tag{3-28}$$

The system is then represented as a linear regression model.

$$
\begin{aligned}
y(k) =& \phi(k)\Theta \\
=& \left[\begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-n) \end{bmatrix}^T, \begin{bmatrix} w_1(k-1) \\ w_1(k-2) \\ \vdots \\ w_1(k-n) \end{bmatrix}^T, \begin{bmatrix} w_2(k-1) \\ w_2(k-2) \\ \vdots \\ w_2(k-n) \end{bmatrix}^T \cdots \begin{bmatrix} w_N(k-1) \\ w_N(k-2) \\ \vdots \\ w_N(k-n) \end{bmatrix}^T \right] \\
\times & \left[\begin{bmatrix} a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix}^T, \begin{bmatrix} b_1p_1 \\ b_2p_1 \\ \vdots \\ b_{n-1}p_1 \end{bmatrix}^T, \begin{bmatrix} b_1p_2 \\ b_2p_2 \\ \vdots \\ b_{n-1}p_2 \end{bmatrix}^T \cdots \begin{bmatrix} b_1p_N \\ b_2p_N \\ \vdots \\ b_{n-1}p_N \end{bmatrix}^T \right]^T
\end{aligned} \tag{3-29}
$$

where the coefficients a and b are from the transfer function:

$$G(q) = \frac{b_1q^n + \cdots + b_{n-2}q + b_{n-1}}{a_1q^n + \cdots + a_{n-1}q + a_n} \text{ with } a_1 = 1 \tag{3-30}$$

and w contains the outputs of the play operators as:

$$w_j = f_{r_j}(u, F_{r_j}[u_i]) = max(u - r_j, min(u + r_j, F_{r_j}[u_i])) \tag{3-31}$$

and p are the weights of the PI model

Equation 3-29 is in matrix form:

$$Y = \Phi\Theta + \Xi \tag{3-32}$$

with

$$Y = \begin{bmatrix} y(1) \\ \vdots \\ y(M) \end{bmatrix}, \Phi = \begin{bmatrix} \phi(1) \\ \vdots \\ \phi(M) \end{bmatrix}, \Xi = \begin{bmatrix} \xi(1) \\ \vdots \\ \xi(M) \end{bmatrix} \tag{3-33}$$

where ξ is noise.

The parameters Θ minimizing the difference between the measured output y_{meas} and simulated output y_{sim} can be found from:

$$\min_{\Theta} \ ||Y - \Phi\Theta||_2 \tag{3-34}$$

which is a least squares problem. Several ways to solve this minimization problem can be found in [35]. However, by solving Equation 3-34, only the values of the multiplication of the coefficients b with the weights p are known and not the coefficients themselves, see Equation 3-29. The least squares solution which gives the solution of the weights and coefficients of

Equation 3-35 can be found by constructing Ψ and taking the singular value decomposition, see Equation 3-36.

$$\Psi = \begin{bmatrix} b_1 p_1 & \dots & b_1 p_N \\ \vdots & \ddots & \vdots \\ b_{n-1} p_1 & \dots & b_{n-1} p_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{n-1} \end{bmatrix} \begin{bmatrix} p1 & \dots & p_N \end{bmatrix} = bp \tag{3-35}$$

$$SVD(\Psi) = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{3-36}$$

$$b = U_1 \tag{3-37}$$

$$p = V_1 \Sigma_1 \tag{3-38}$$

This procedure does provide coefficients for a transfer function. However, it is found that identifying a transfer function from a step response with the MATLAB command tfest is more accurate for the lens. Therefore, this MATLAB command is used instead. The coefficients found in this procedure are used as initial weights p.

Now the GPI model in series with a dynamic model is identified. This is done with an optimization algorithm searching the solution of the optimization problem in Equation 3-39.

$$\Theta = \arg\min_{\Theta} J(\Theta, u) \text{ with} \tag{3-39}$$

$$J(\Theta, u) = \frac{\sum_{k=1}^{M} (y_{sim}(\Theta, u) - y_{meas})^2}{\sum_{k=1}^{M} y_{sim}(\Theta, u)^2} \tag{3-40}$$

$$y_{sim}(\Theta, u) = G(H(\Theta, g(\Theta, u)), \Theta) \tag{3-41}$$

For this purpose, the command *fminunc* is used in MATLAB (with the default BFGS Quasi-Newton algorithm). The parameter g_4 is not included in this optimization, but is determined by its definition instead, shown in Equation 3-14.

3-5-3 Dataset for identification

The signal used as input for identifying the dynamic GPI model is shown in Figure 3-7. All inputs to the AO devices in this thesis are scaled, where 1 normalized voltage corresponds to the max voltage and -1 normalized voltage to the minimal voltage. This signal is chosen since both the dynamics and the hysteresis have to be identified. It can be seen that it contains three different parts:

- In order to determine the influence function of every actuator, steps from maximum to minimum amplitude ($u_{j_{min}} = -1$ normalized voltage) and from minimum to maximum amplitude ($u_{j_{max}} = 1$ normalized voltage) are applied to the actuator at the beginning of the identification dataset. This is done at the beginning since in that case the influence of creep is minimal. The dynamics are identified from this part of the identification dataset as well.

- At the second part of the identification dataset, positive and negative steps in the voltage are alternated, with amplitudes ranging from -1 normalized voltage to 1 normalized voltage in steps of 0.05 normalized voltage. This is done to cover a large part of the amplitude range of the actuator.

- At the end of the dataset, the whole amplitude range of the actuator is covered and the dynamics at many frequencies are excited. For this purpose, a frequency sweep with frequencies from 0.01 Hz to 1 Hz plus a sine with a frequency of 0.08 Hz with both an amplitude of 0.5 normalized voltage is used.

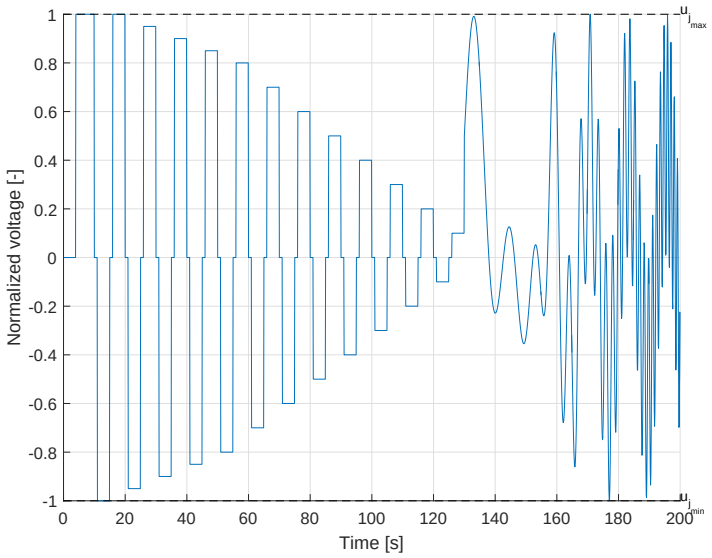

Figure 3-7: The dataset used for identification

3-6 Compensation

The performance of the compensation is evaluated for SISO and for Multiple-Input and Multiple-Output (MIMO) actuation. The performance of the hysteresis compensation is evaluated by actuating the system with the response of the inverse system to a reference and then computing the normalized Root Mean Square (RMS) of the difference between the reference and the output. A reference is given for all Zernikes up to the third order, excluding piston and including spherical aberration. The other Zernikes that are measured are only measured in order to make sure that higher order Zernikes that occur do not influence the values of the estimated lower order Zernikes. SISO performance is tested for a steps and sawtooth signal and both SISO and MIMO are tested for a more realistic reference. This more realistic reference contains multiple frequencies and amplitudes, which makes sure that the compensation of the dynamics can be evaluated as well. At last, the compensation method is compared to the conventional compensation method: A GPI model without dynamics.

3-6-1 Single actuator

The SISO compensation is tested with a reference consisting of steps of 10 seconds and a sawtooth signal. The experiment with the reference consisting of steps can be used to check whether the dynamics are compensated well. The performance of the hysteresis compensation can be checked with the sawtooth signal. The used reference signals are shown in Figures 3-8 and 3-9.

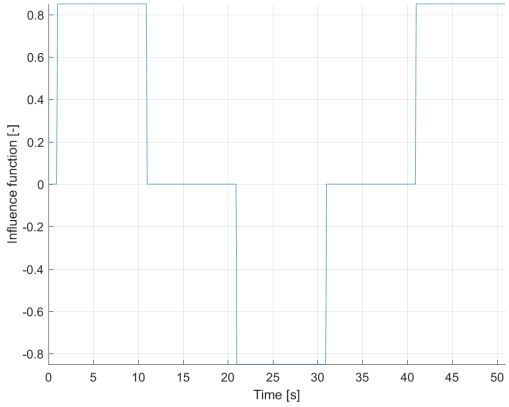

Figure 3-8: Used steps signal for evaluation of SISO compensation

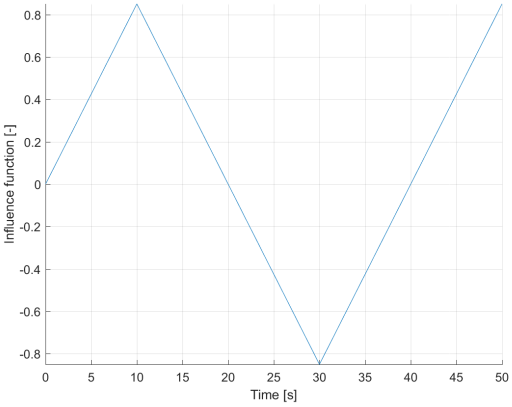

Figure 3-9: Used sawtooth signal for evaluation of SISO compensation

In order to evaluate the performance of the compensation for a more realistic reference a new reference trajectory is generated. This starts with a dynamic part which resembles the optimization of the input. At the end of the dataset, a static reference is given, resembling the found optimum. The signal is constructed as follows: First 10 samples at mid stroke are applied. Then, 201 samples of random numbers τ between -1 and 1 and a constant random value τ for the remaining 190 samples are filtered with a 6th order Butterworth filter with a cutoff frequency of 0.5 Hz and applied to the actuator, see Equation 3-42 and Figure 3-10. This reference signal is then scaled such that actuator inputs do not exceed the device limitations. The performance measure that is used is the RMS error normalized by the RMS value of the reference, given in Equation 3-43.

$$ref_{y_j} = \left[0, \cdots, 0, f_{butter}(1.25\tau(201), 0.5\tau(1) \left[1, \cdots, 1\right])\right]^T \tag{3-42}$$

$$e_{NRMS} = \frac{\sqrt{\sum_{k=1}^{M}(ref_y(k) - y_{meas}(k))^2}}{\sqrt{\sum_{k=1}^{M} ref_y^2(k)}} \tag{3-43}$$

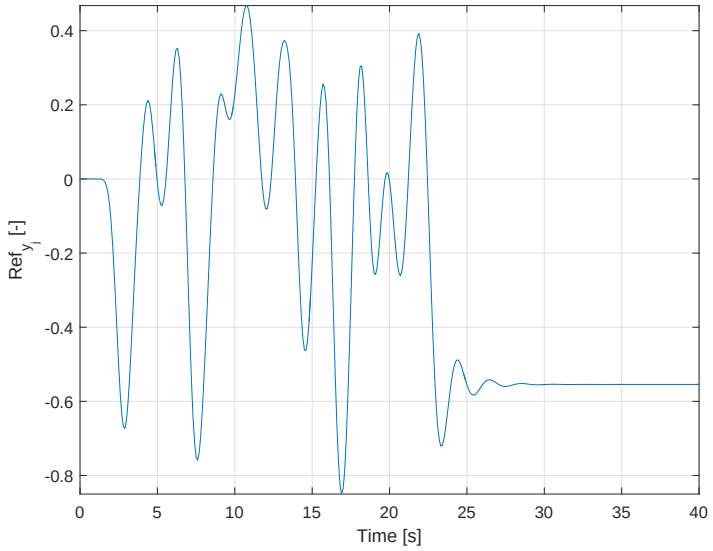

Figure 3-10: The reference used for testing the SISO compensation

3-6-2 Multiple actuators

In order to evaluate the performance when multiple actuators (MIMO performance) are used, the same procedure (Equation 3-42) as for the single actuators is used to create a realistic reference signal ref_z for all Zernikes up to the third order, excluding piston and including spherical aberration. Then, the required magnitudes of the influence functions of each actuator ($refy_j$) are calculated with the minimization given in Equation 3-44, in which Q is the cost of a tracking error and R is the cost of control. The solution of the minimization can be formulated as the constrained least squares problem in Equation 3-45, where l is the maximum absolute value of the voltage on the actuators. This magnitude is used as input for the inverse compensator, which then gives the input for the actuators of the lens. The whole process is shown in the flowchart in Figure 3-11.

$$\min_y((ref_z - z(y)))^T Q((ref_z - z(y))) + y^T R y \tag{3-44}$$

$$min_y \left(\begin{bmatrix} \sqrt{Q}[\zeta_1 & \dots & \zeta_j] \\ & \sqrt{R} & \end{bmatrix} \begin{bmatrix} y \end{bmatrix} - \begin{bmatrix} \sqrt{Q}\ ref_z \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right)^2 \tag{3-45}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \\ -1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 \end{bmatrix} \begin{bmatrix} y \end{bmatrix} \leq \begin{bmatrix} l & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & l \\ l & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & l \end{bmatrix}$$

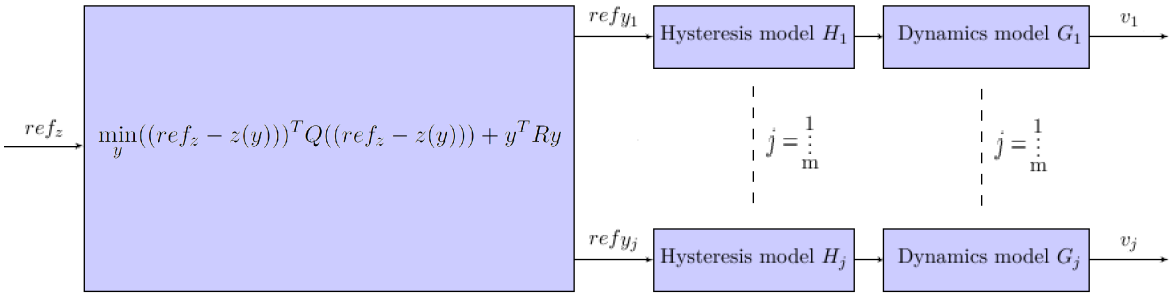

Figure 3-11: Flow scheme MIMO compensation

The used performance measure is given in Equation 3-46 and is the mean value of the RMS of the tracking error normalized by the RMS of the reference of all actuators.

$$e_{NRMS} = \frac{\sum_{i=1}^{N_{Zernikes}} \frac{\sqrt{\sum_{k=1}^{M}(z_{meas}(k,i) - z_{ref}(k,i))^2}}{\sqrt{\sum_{k=1}^{M} z_{ref}(k,i)^2}}}{N_{Zernikes}} \tag{3-46}$$

3-7 MATLAB files

In order to perform the methods in this chapter, some MATLAB functions have been created. These MATLAB functions with their uses are given in Table 3-1 All the code of these function can be found in Appendix C.

MATLAB function	Use
calccentroid	Calculates the centroid of the intensity in a region of interest as described in Secton 3-2
fit_dyn_PI	Fits a (dynamic) (G)PI model for every actuator of the AO device with a dataset
measzer	Reconstruction of the Zernikes as explained in section 3-2
derzer	Calculates the average derivatives in Equation 3-4
cali	Performs the calibration of the AO device
simPI	Simulates a PI model
simPIasym	Simulates a GPI model
simPIasymi	Simulates an inverse GPI model
invPI	Inverts a PI model
createinvmodel	Identifies a (dynamic) (G)PI model
driver	Runs the identification dataset on the AO device
mols	Runs the algorithm in Section 3-5 used to identify a dynamic PI model and serves as an initial guess of the parameters of the GPI model
testcomp	Runs a compensation test on the AO device
LocReg	Solves the minimization in Equation 3-45
makegrid	Creates a grid with equidistant points
fitgrid	Calculates the mismatch between the fitted grid and the observed centroid locations

Table 3-1: MATLAB functions and their uses

Chapter 4

Results

In this chapter, the obtained results are discussed. First, the properties of the identified (dynamic) Generalized Prandtl-Ishlinskii (GPI) models are shown. Then, the remaining normalized Root Mean Square (RMS) error is given for both Single-Input and Single-Output (SISO) and Multiple-Input and Multiple-Output (MIMO) operation of the system with a dynamic GPI model compensator. These values are compared to using a static GPI to study the added value of adding a dynamic model. Finally, the results are commented in the discussion.

4-1 Identified models

The fit of the identified dynamic GPI model to the dataset for identification is shown in Figure 4-1. The characteristics of the models that are identified for an actuator are shown in Figures 2-12 and 4-2. In the hysteresis curve in Figure 2-12 it can be seen that the identified dynamic GPI model for actuator 1 of the mirror exhibits hysteresis. Hysteresis curves for other actuators look similar. The transfer functions of the actuators are not the same for all of them. Either a first order or a second order transfer function is fitted for the mirror and a second or third order transfer function for the lens. The order is decided on basis of visual inspection of the step responses of the fitted transfer functions and the real step responses, see for example Figures 4-2 and 4-3.

The Variance Accounted For (VAF) (see Equation 4-1) values of the identified dynamic GPI models of the actuators of the lens are shown in Table 4-1. In Table 4-2 the VAF values of the identified dynamic GPI models of the actuators of the mirror are shown.

$$\text{VAF} = 1 - \frac{\sum_{k=1}^{M}(y_{meas}(k) - y(k))^2}{\sum_{k=1}^{M} y_{meas}(k)^2} \tag{4-1}$$

It can be seen that the VAF for some actuators is higher than for others. This is caused by the amount of noise on the measurements. The magnitude of this noise varies for different Zernikes modes. Since the influence function of every actuator is not influenced to the same

extent by noise on an individual Zernike mode, the noise levels varies for different actuators. The parameters of the Shack-Hartmann have a lot of influence on this noise. This is discussed further in the discussion in Section 4-3.

Actuator	1	2	3	4	5
VAF	0.9994	0.9986	0.9922	0.9993	0.9993
Actuator	6	7	8	9	10
VAF	0.9994	0.9978	0.9985	0.9991	0.9994
Actuator	11	12	13	14	15
VAF	0.9994	0.9981	0.9989	0.9989	0.9988
Actuator	16	17	18	mean	
VAF	0.9991	0.9992	0.9993	0.9986	

Table 4-1: VAF values for identification of the dynamic GPI models for each actuator of the lens for the identified model with 6 operators and a dynamic model

Actuator	1	2	3	4	5
VAF	0.9979	0.9978	0.9305	0.9891	0.9530
Actuator	6	7	8	9	10
VAF	0.9974	0.9993	0.9997	0.9880	0.9843
Actuator	11	12	13	14	15
VAF	0.9960	0.9994	0.9994	0.9983	0.9921
Actuator	16	17	18	19	mean
VAF	0.9972	0.9992	0.9799	0.9996	0.9894

Table 4-2: VAF values for identification of the dynamic GPI models for each actuator of the mirror for the identified model with 5 operators and a dynamic model

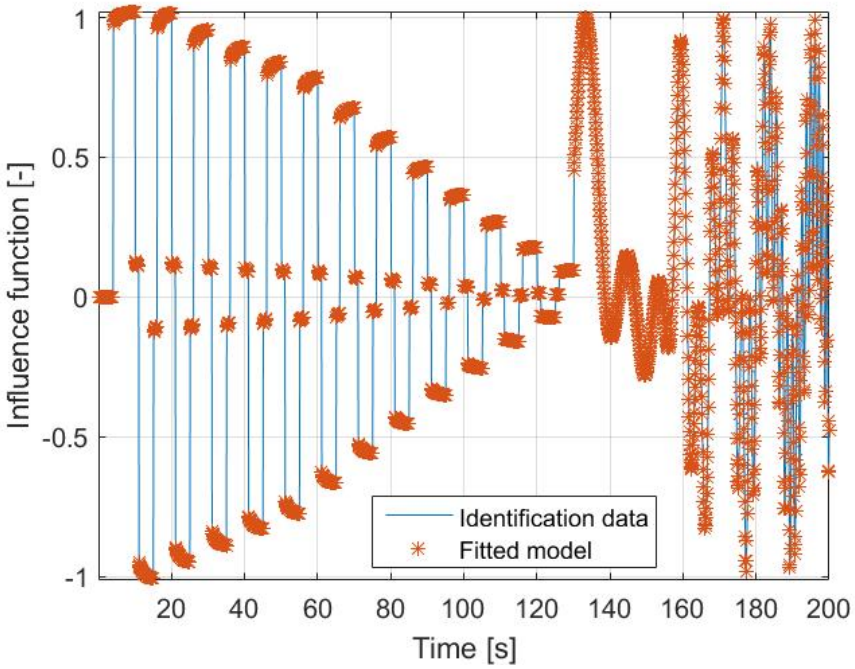

Figure 4-1: The fit of the identified dynamic GPI model to the identification dataset for actuator 19 of the mirror with 5 operators

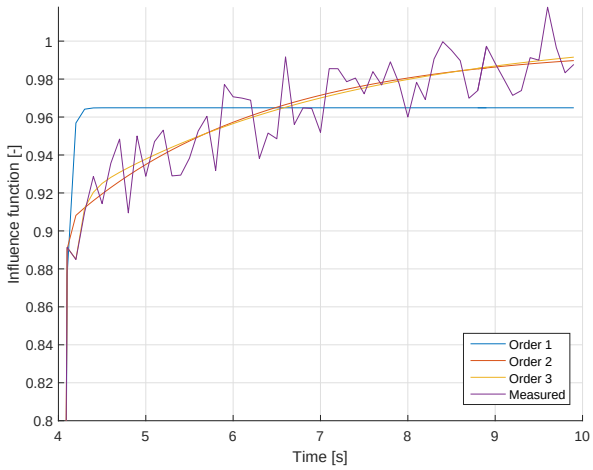

Figure 4-2: Simulated step response of the fitted transfer function compared to the step response of actuator 1 of the mirror. Chosen order is 2.

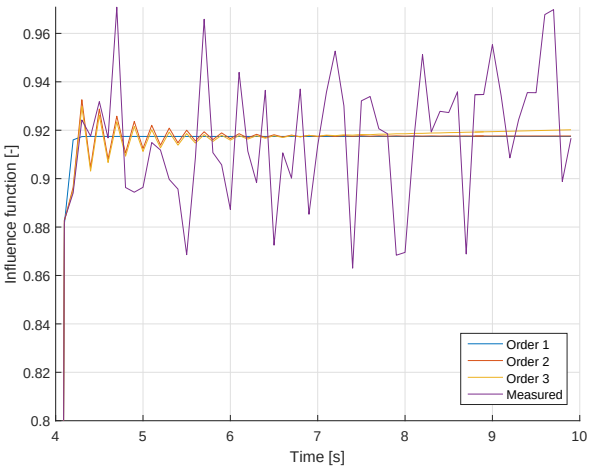

Figure 4-3: Simulated step response of the fitted transfer function compared to the step response of actuator 11 of the mirror. Chosen order is 1.

The SISO results can be used as a validation dataset for the identified models. The VAF of the fitted model on these datasets is given in Appendix B.

4-2 Compensation performance

In this section, the measured compensation performance is shown for both single actuators and MIMO actuation. First, a comparison of models obtained with and without consideration of the dynamics during identification is presented. Then, with MIMO experiments, different values of parameters for the model are tested to evaluate which give the best results. The section is concluded with SISO results.

4-2-1 Influence of dynamics on the hysteresis identification

The importance of taking the dynamics into account during the identification of the hysteresis can be seen by looking at the hysteresis percentage h. This percentage is 15.29% for the model in which the dynamics are not taken into account and 7.52% when they are taken into account for actuator 19 of the mirror. Two experiments with both these models are compared in Figure 4-4. In this Figure, the reference tracking of a static GPI models is shown while the dynamics have been taken into account during the identification of one of them and not during the other. It can be clearly seen that if the dynamics are not taken into account, hysteresis is overestimated.

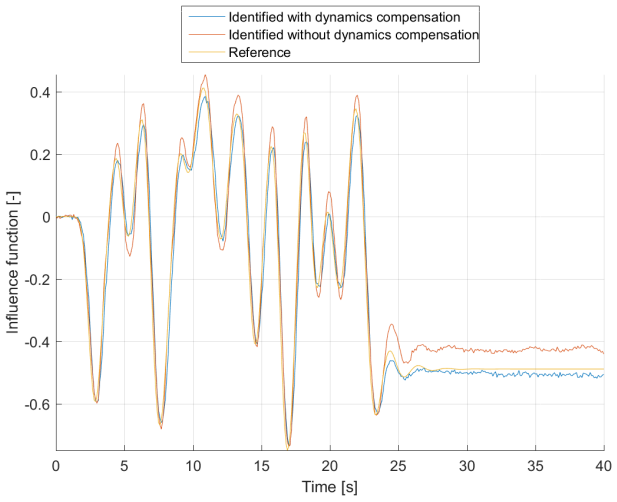

Figure 4-4: The effect of (not) taking into account dynamics during identification on the reference tracking of a static GPI model with 5 operators on actuator 19 of the mirror.

4-2-2 Variables

Before identifying the model, the amount of operators of the GPI model has to be specified. Also, a choice should be made whether or not a dynamic model will be included. In this section, the influence of these choices is examined.

Increasing the amount of operators can make the model more accurate. However, a high amount of parameters can also result in overfitting, which can decrease the compensation

performance. Another reason for performance decrease when increasing the amount of parameters is the fact that there is no guarantee that the global minimum of the cost function is found. Therefore, increasing the complexity of the model increases the chance that the optimization gets stuck in a (worse) local minimum and decreases the likelihood that a good solution is found. Fitted models have been tested for multiple operator values and with or without a dynamic model. The tracking error as defined in Equation 3-46 is shown for multiple parameter values for MIMO actuation in Figures 4-5 and 4-6. Note that for the model with 1 operator g is set to be: $g(u) = u$. This is essentially is a static gain. Combining this with a dynamic model allows compensation by only inverting the transfer function. The combination with a static gain results in essentially no compensation at all. These numbers can be used to compare the compensated results.

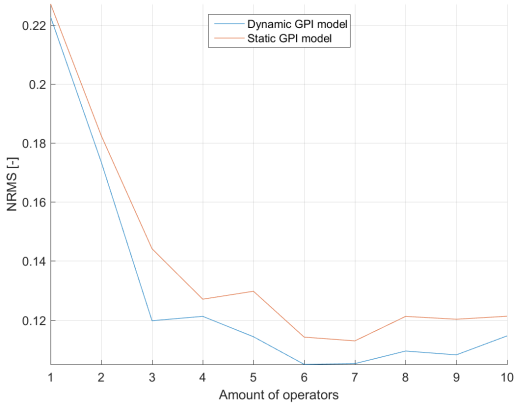

Figure 4-5: e_{NRMS} for MIMO actuation of the lens for different amounts of operators

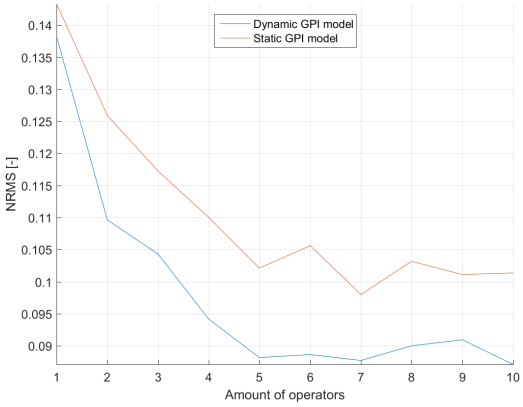

Figure 4-6: e_{NRMS} for MIMO actuation of the mirror for different amounts of operators

4-2-3 Response to simple signals

The responses to steps are shown in Figures 4-7 and 4-8. It can be seen that for the mirror, the dynamics are reduced by the dynamic GPI model compared to the other models. However, for the lens it can be observed that the dynamics for a positive step are reduced, but for a negative step, it overcompensates the dynamics. It should be noted that the dynamics are compensated less well for most actuators, which is expected to be due to identification problems. Those will be discussed further in Section 4-3. A steady state error is observed for positive step, which was also present in sawtooth signal. This does not occur every time, but more often than not. This will be discussed further in Section 4-3.

The responses to a sawtooth signal are shown in Figures 4-9 and 4-10. It can be seen that the systems without a compensator deviate much more from the reference compared to the other models. However, by visual inspection there is no clear difference in the quality of the reference tracking between the static and dynamic GPI model.

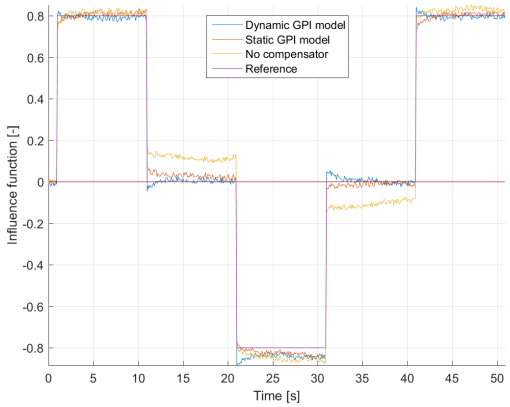

Figure 4-7: Responses of the lens compensated with the different models to a signal composed of steps

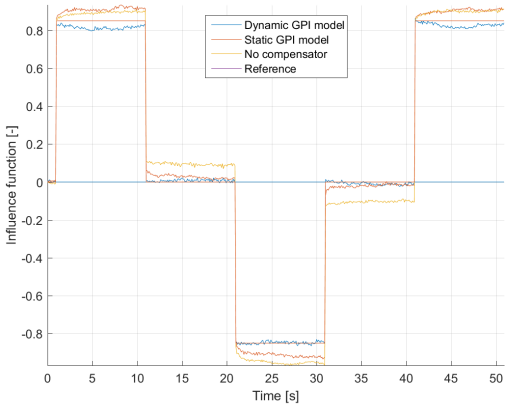

Figure 4-8: Responses of the mirror compensated with the different models to a signal composed of steps

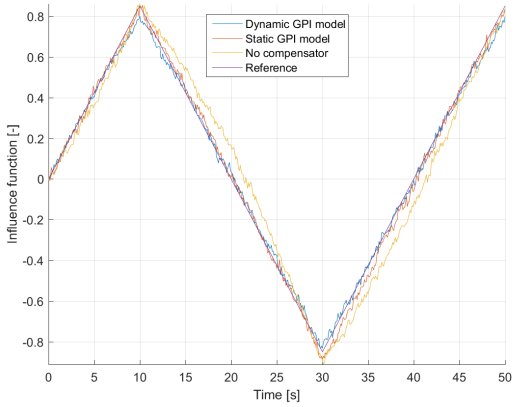

Figure 4-9: Responses of the lens compensated with the different models to a sawtooth signal

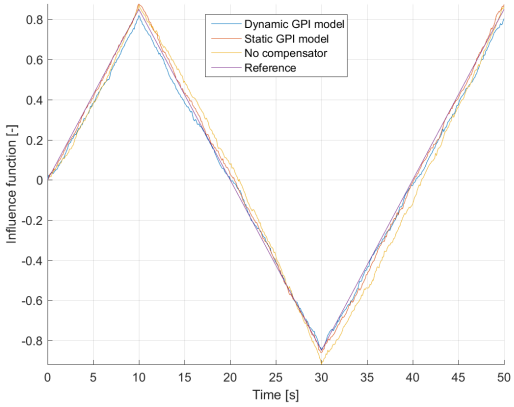

Figure 4-10: Responses of the mirror compensated with the different models to a sawtooth signal

The input-output plot of the response to the sawtooth signal in Figures 4-9 and 4-10 is shown in Figure 4-11 and 4-12. It can be seen that the hysteresis is decreased for both the static as well as the dynamic GPI model. The hysteresis percentage h is shown for these experiments in Table 4-3.

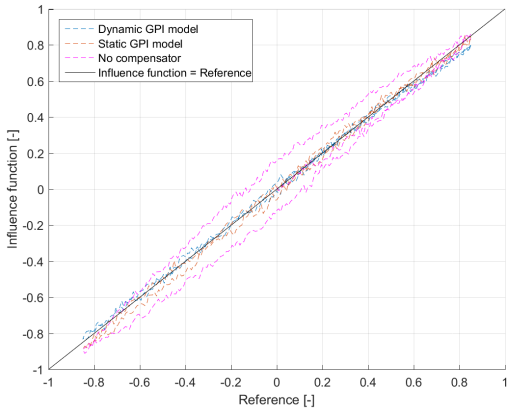

Figure 4-11: Hysteresis loop of the lens compensated with different models for a sawtooth signal

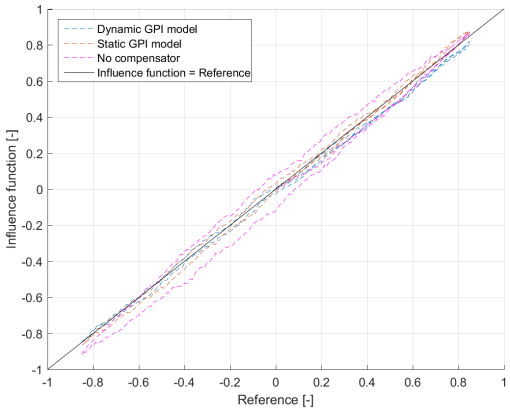

Figure 4-12: Hysteresis loop of the mirror compensated with different models for a sawtooth signal

Actuator	Dynamic GPI	Static GPI	No compensation
Lens actuator 8	6.70	6.33	18.3
Mirror actuator 19	4.06	4.48	11.8

Table 4-3: Hysteresis percentage h in $\%$ for different models

4-2-4 Single-input and single-output

The results for a single actuator are shown in Figure 4-13 and 4-14. The values of the normalized RMS error as given in Equation 3-43 are given in Table 4-4, 4-5 and Figure 4-6. It can be seen that actuators with a model with a higher VAF (see Appendix B) also have a better performance. Individual numbers on the SISO performance of the actuators can be found in Appendix B. The results are split in different time intervals, where the performance from 0-25 seconds can be seen as the performance during dynamics. The performance from 25 to 30 seconds can be seen as a measure of the steady state error for static operation. 0-30 seconds gives an overall measure of both dynamic operation and static operation. It can be seen that the GPI model removes the tracking error due to hysteresis from the system. This is mostly visible for the constant voltage at the end of the dataset, where the dynamic and static GPI model outperform the dynamic model and the system without compensator. Furthermore, it can be seen that the dynamic GPI model has less tracking error at the peaks of the dynamic part of the dataset compared to a static GPI model. For most actuators, this results in a smaller tracking error, but not for all. However the mean of the tracking errors of all actuators decreases by adding a dynamic model.

The results of the SISO compensation are shown in Table 4-4, 4-5 and 4-6. Differences were observed in the lens between the response to a negative step and a positive step of the input, see the Figure 4-15 later in this section. Therefore the SISO operation of the lens is tested for a randomly generated dataset according to Section 3-5-3 and its negative counterpart. It can be seen that the results for the dataset with the positive step (multiplied by -1) are slightly better. This was to be expected, since the dynamics are identified on a step response

to a positive step in the input. Overall, the mean of the SISO performance of all actuators improves by adding a dynamical model. Dynamic compensation brings improvement in the SISO performance in $\frac{18}{19}$ of all the actuators in mirror, compared to $\frac{10}{18}$ for the lens. For different actuators, steady state offsets of different sizes are observed for SISO operation. One thing that is observed is that the output at $t > 25s$ of the dynamic GPI model is always smaller compared to the output of the static GPI model. Therefore, for individual actuators, the normalized RMS tracking error is sometimes larger for the dynamic GPI model compared to the static GPI model. In general, the steady state error for the lens increases by using a dynamic model. This increase is larger for the normal dataset compared to the dataset multiplied by -1. Due to the better reference tracking during $t \leq 25s$, the dynamic GPI model has a smaller normalized RMS tracking error if the mean is taken over all actuators.

Time interval	Dynamic GPI	Static GPI	No compensation
0-25 sec	0.0565	0.0677	0.1439
25-30 sec	0.0296	0.0611	0.1274
0-30 sec	0.0449	0.0684	0.1428

Table 4-4: Mean of the Normalized RMS values of the error for all actuators of the mirror

Time interval	Dynamic GPI	Static GPI	No compensation
0-25 sec	0.0458	0.0693	0.2224
25-30 sec	0.0494	0.0328	0.1740
0-30 sec	0.0491	0.0514	0.1948

Table 4-5: Mean of the Normalized RMS values of the error for all actuators of the lens

Time interval	Dynamic GPI	Static GPI	No compensation
0-25 sec	0.0451	0.0716	0.2025
25-30 sec	0.0433	0.0426	0.1037
0-30 sec	0.0454	0.0570	0.1511

Table 4-6: Mean of the Normalized RMS values of the error for all actuators of the lens for the dataset multiplied by minus 1

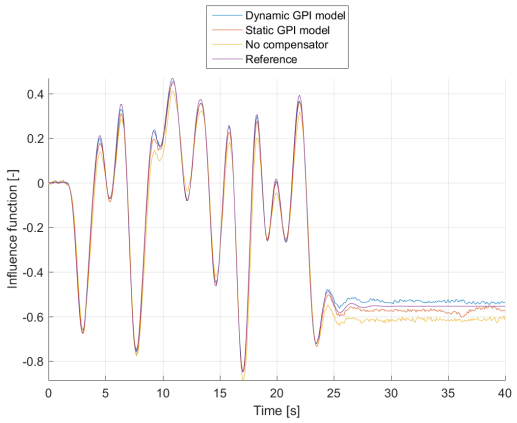

Figure 4-13: Reference compared to the output for a system compensated with a dynamic GPI model and no compensation during SISO actuation of actuator 1 of the lens

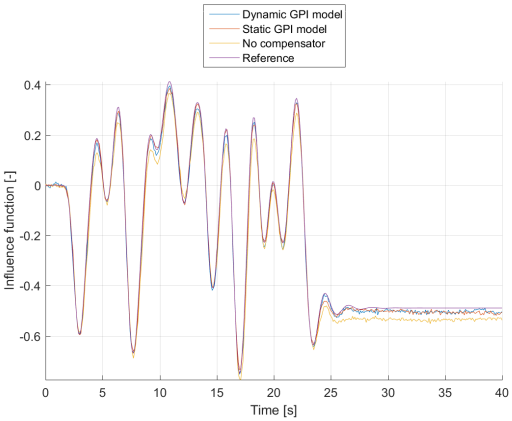

Figure 4-14: Reference compared to the output for a system compensated with a dynamic GPI model and no compensation during SISO actuation of actuator 19 of the mirror

4-2-5 Multiple-input and multiple-output

In order to illustrate the effect of the compensator, the output of the compensated system is compared with a static GPI model and no compensation at all. The used static GPI model is the same as the dynamic GPI model, but the transfer functions that are used for the dynamic models are static gains. The results for MIMO application are shown in Figures 4-15 and 4-16. It can be seen here that during the peaks in the dynamic part of the dataset the dynamic GPI model performs better than the static one. Not all Zernikes are as accurate as the one shown in the Figures 4-15 and 4-16. Especially the tip-tilt has some steady state error. From Figures 4-5 and 4-6 it can be seen that overall, adding a GPI model reduces the tracking error.

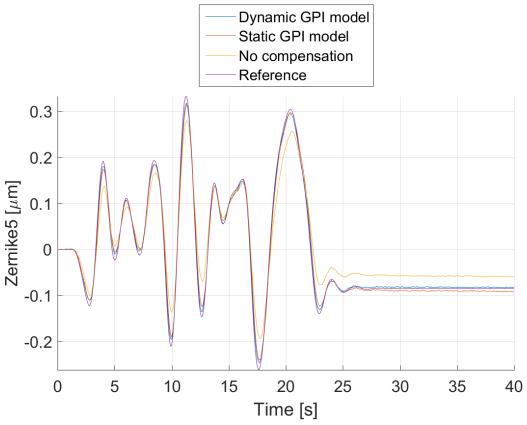

Figure 4-15: Reference compared to the output for a system compensated with a dynamic GPI model, a static GPI model and no compensation during MIMO actuation on the lens

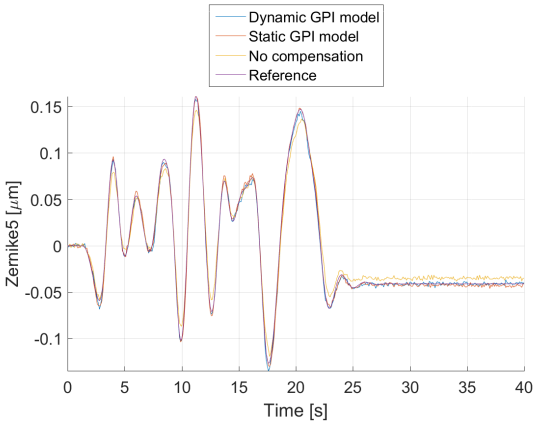

Figure 4-16: Reference compared to the output for a system compensated with a dynamic GPI model and no compensation during MIMO actuation on the mirror

4-2-6 Reproducibility

In order to show the reproducibility of the system, multiple measurements are made with the same settings, both for SISO and MIMO operation. The results are shown in Figures 4-17 to 4-20. It can be seen that the response of the mirror is reproducible. However, the lens measurements show some variance in the both the SISO experiments as well as in the MIMO experiments for the Zernikes 1, 6 and 7 (see Appendix A for the numbering of the Zernikes).

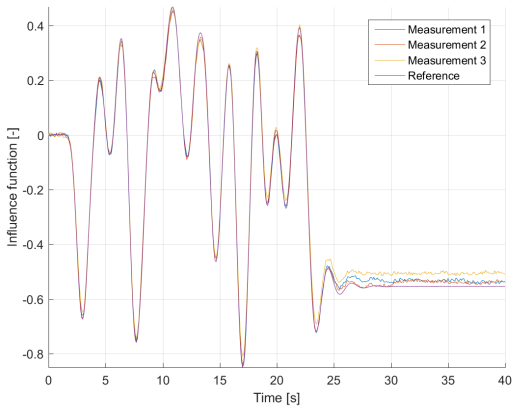

Figure 4-17: Reproducibility of the lens system for SISO actuation with compensation with a dynamic GPI model with 6 operators

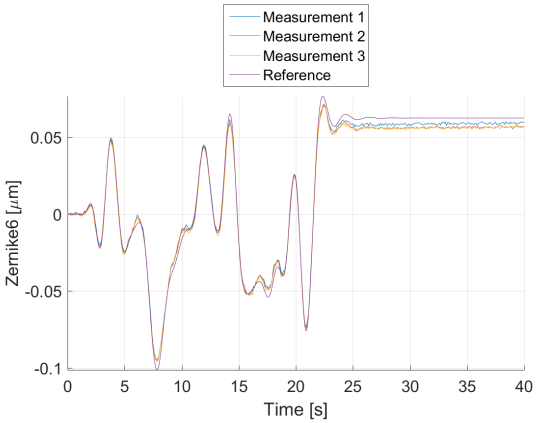

Figure 4-18: Reproducibility of the lens system for MIMO actuation with compensation with a dynamic GPI model with 6 operators

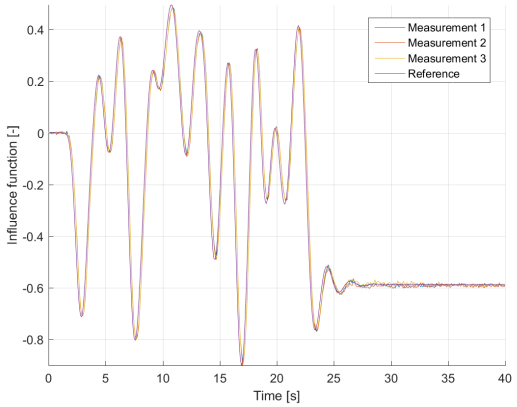

Figure 4-19: Reproducibility of the mirror system for SISO actuation with compensation with a dynamic GPI model with 5 operators

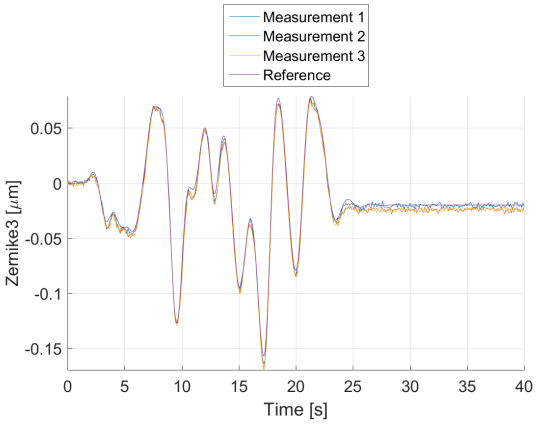

Figure 4-20: Reproducibility of the mirror system for MIMO actuation with compensation with a dynamic GPI model with 5 operators

4-3 Discussion

The fact that not taking in account the dynamics during identification results in an overestimation of the hysteresis is as expected. This is due to the fact that dynamics cause phase delay, which has a similar shape as hysteresis in the input-output plot. It should be noted that no special attention has been taken to minimize this influence by choosing a identification frequency which does not contain a lot of dynamics. Instead, the identification dataset described in Section 3-5-3 is used. However this method would be useful in case no frequency can be found on which the dynamics have a small enough influence or when this frequency is not previously known.

In the proposed models, there is no influence between the actuators. Also, it is assumed that the second order polynomial g (see Equation 3-13) can approximate the asymmetry of the hysteresis. Furthermore, it is assumed that the hysteresis and dynamics are the same for all Zernikes in the same influence function. However, in reality these assumptions might not hold. These influences could be modeled, but this would significantly increase the model complexity. This would require a more refined identification and it would increase the computational cost of identification and compensation. Also, many high frequency dynamics can not be measured at 10 Hz, the limiting frequency of the wavefront sensor employed. When these can be measured accurately, the performance of the compensators can most likely be improved.

Adding a dynamic model does not reduce the tracking error for all actuators for SISO operation. Also, adding the dynamic model does increase the hysteresis percentage h of the lens for some actuators for the lens. A possible explanation is the observed difference between the response to a positive step and negative step. This modeling error of the dynamics will also impact the hysteresis identification, which could explain some of the remaining hysteresis errors in the system.

It is observed that the amount of noise of the measured Zernikes is influenced by the chosen parameters of the camera and the Zernike reconstruction algorithm ($I_{min_{centroid}}$, $I_{min_{spot}}$,

$r_{centroid}$ and $r_{centroid_{min}}$). The alignment of the setup influences this noise as well. This noise influenced the identification of the dynamics and is one of the reasons that this identification did not work well. More sophisticated and robust methods exist to identify dynamics, for example [36, 37]. However, those require datasets which consist of more data than only a step response. A step response was chosen here to identify the dynamics, since this response is not influenced by the hysteresis according to the assumed model. However, previous research [26] has indicated that identifying the dynamics from a hysteresis compensated system is possible. Perhaps a longer dataset would make the dynamics identification better, as done in [26]. However, it has also been shown that identifying the hysteresis with the procedure described in this thesis results in better hysteresis compensation compared to not taking the dynamics into account. Therefore, it is advised to use this method to identify a dynamics model on a dataset which is compensated by a hysteresis model identified in the procedure in this report.

It is also observed that not all Zernikes are as well compensated as others. Especially the tip and tilt show some steady state errors. These could be issues with the hysteresis compensation, but could also be issues in the alignment.

Dynamic compensation brings improvement in $\frac{18}{19}$ of all the actuators in mirror, compared to $\frac{10}{18}$ for the lens. A possible explanation for this is that besides the problems with the identification of the dynamics mentioned before, the dynamics of the lens exhibit nonlinearity as well, which can not be modeled by a transfer function. This can be seen by the fact that the increase of the steady state error by adding a dynamic model is larger for the normal dataset compared to the dataset multiplied by -1. This is expected, since the dynamics are identified on a positive step only. Therefore, it is recommended to explore the use of some nonlinear dynamic models in the future for modeling of the dynamics of the lens. It should be noted that for no compensation, the steady state error at $25 - 30$ seconds is larger, see Table 4-5 and 4-6. This could be explained by the fact that there is more creep for a positive step, since creep leads to a higher output for the same input, while hysteresis does the opposite. This larger creep for a positive step is also confirmed by the response to the steps signal, see Figure 4-7.

The steady state error that is observed for a positive step in the sawtooth signal might indicate a modeling error or a mistake in modeling the virgin curve of the hysteresis. Another cause could be that the assumption that the asymmetry can be described by a second order polynomial does not hold. For now the cause remains unknown and this could be studied in the future.

Chapter 5

Conclusion and recommendations

In this report, experiments are described from which the following conclusions can be drawn:

- It is possible to compensate for hysteresis and creep with a model based on a Generalized Prandtl-Ishlinskii (GPI) model and a transfer function. This has been verified on both a deformable lens as well as a deformable mirror for both Single-Input and Single-Output (SISO) operation and Multiple-Input and Multiple-Output (MIMO) operation.
- On both of these devices, the dynamic hysteresis compensator decreased the tracking error more compared to the conventional static hysteresis compensation for the SISO and for the MIMO case.
- Dynamics compensation decreases the tracking error during dynamic parts, but increases the steady state error for the lens. For the mirror, the error is reduced during both parts.
- A difference was observed between the response to a negative step and a positive step of the input in the lens: The creep is larger for a positive step. This causes modeling errors in both the dynamic model and hysteresis model.

Therefore, the following recommendations are made:

- When an application requires a smaller tracking error than the device with hysteresis and creep can provide, it is recommended to consider using a GPI model with a dynamic model to compensate for this hysteresis and creep.
- Future research is recommended on the application of a compensator based on a GPI model and a transfer function on systems with higher frequency dynamics.
- Future research is recommended into the steady state error observed for a positive step.

- Future research in finding a more robust way to fit the model in order to enable compensation with models with more parameters and thereby increasing the performance even further.
- Find a better way to identify the dynamics of the lens. A suggestion is to identify those from a system compensated with a static GPI model [26] and using a nonlinear dynamics model should be considered.

Bibliography

[1] H. Kroese, *Feedback control of a piezo deformable mirror for a wavefront-sensorless AO setup*. PhD thesis, TU Delft, Delft University of Technology, 2009.

[2] J. Wang and D. E. Silva, "Wave-front interpretation with zernike polynomials," *Applied optics*, vol. 19, no. 9, pp. 1510–1518, 1980.

[3] B. C. Platt and R. Shack, "History and principles of shack-hartmann wavefront sensing," *Journal of Refractive Surgery*, vol. 17, no. 5, pp. S573–S577, 2001.

[4] M. J. Booth, "Adaptive optical microscopy: the ongoing quest for a perfect image," *Light: Science & Applications*, vol. 3, no. 4, p. e165, 2014.

[5] H.-C. Lin, M.-S. Chen, and Y.-H. Lin, "A review of electrically tunable focusing liquid crystal lenses," *Transactions on Electrical and Electronic Materials*, vol. 12, no. 6, pp. 234–240, 2011.

[6] S. Bonora, Y. Jian, P. Zhang, A. Zam, E. N. Pugh, R. J. Zawadzki, and M. V. Sarunic, "Wavefront correction and high-resolution in vivo oct imaging with an objective integrated multi-actuator adaptive lens," *Optics express*, vol. 23, no. 17, pp. 21931–21941, 2015.

[7] D. P. Biss, R. H. Webb, Y. Zhou, T. G. Bifano, P. Zamiri, and C. P. Lin, "An adaptive optics biomicroscope for mouse retinal imaging," in *MOEMS-MEMS 2007 Micro and Nanofabrication*, pp. 646703–646703, International Society for Optics and Photonics, 2007.

[8] M. A. Vorontsov, "Decoupled stochastic parallel gradient descent optimization for adaptive optics: integrated approach for wave-front sensor information fusion," *JOSA A*, vol. 19, no. 2, pp. 356–368, 2002.

[9] J. Fienup and J. Miller, "Aberration correction by maximizing generalized sharpness metrics," *JOSA A*, vol. 20, no. 4, pp. 609–620, 2003.

[10] D. Débarre, E. J. Botcherby, T. Watanabe, S. Srinivas, M. J. Booth, and T. Wilson, "Image-based adaptive optics for two-photon microscopy," *Optics letters*, vol. 34, no. 16, pp. 2495–2497, 2009.

[11] H. R. Verstraete, S. Wahls, J. Kalkman, and M. Verhaegen, "Model-based sensor-less wavefront aberration correction in optical coherence tomography," *Optics letters*, vol. 40, no. 24, pp. 5722–5725, 2015.

[12] H. Yang, Z. Zhang, and J. Wu, "Performance comparison of wavefront-sensorless adaptive optics systems by using of the focal plane," *International Journal of Optics*, vol. 2015, 2015.

[13] S. Zommer, E. Ribak, S. Lipson, and J. Adler, "Simulated annealing in ocular adaptive optics," *Optics letters*, vol. 31, no. 7, pp. 939–941, 2006.

[14] M. J. Booth, "Adaptive optics in microscopy," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1861, pp. 2829–2843, 2007.

[15] S. Bonora, "Deformable lens structure for adaptive optics devices," Aug. 6 2015. WO Patent App. PCT/IB2015/050,609.

[16] G.-Y. Gu, L.-M. Zhu, C.-Y. Su, H. Ding, and S. Fatikow, "Modeling and control of piezo-actuated nanopositioning stages: a survey," 2014.

[17] A. Meitzler, H. Tiersten, A. Warner, D. Berlincourt, G. Couqin, and F. Welsh III, "Ieee standard on piezoelectricity," 1988.

[18] G. Vdovin, O. Soloviev, M. Loktev, and V. Patlan, *Oko guide to adaptive optics*. OKOTechnologies, 2013.

[19] M. Al Janaideh, *Generalized Prandtl-Ishlinskii hysteresis model and its analytical inverse for compensation of hysteresis in smart actuators*. PhD thesis, Concordia University Montreal, Quebec, Canada, 2009.

[20] M. Al Janaideh, S. Rakheja, J. Mao, and C.-Y. Su, "Inverse generalized asymmetric prandtl-ishlinskii model for compensation of hysteresis nonlinearities in smart actuators," in *Networking, Sensing and Control, 2009. ICNSC'09. International Conference on*, pp. 834–839, IEEE, 2009.

[21] M. Al Janaideh, S. Rakheja, and C.-Y. Su, "An analytical generalized prandtl–ishlinskii model inversion for hysteresis compensation in micropositioning control," *Mechatronics, IEEE/ASME Transactions on*, vol. 16, no. 4, pp. 734–744, 2011.

[22] G. V. Webb, D. C. Lagoudas, and A. J. Kurdila, "Hysteresis modeling of sma actuators for control applications," *Journal of Intelligent Material Systems and Structures*, vol. 9, no. 6, pp. 432–448, 1998.

[23] H. Jung and D.-G. Gweon, "Creep characteristics of piezoelectric actuators," *Review of Scientific Instruments*, vol. 71, no. 4, pp. 1896–1900, 2000.

[24] H. Jung, J. Y. Shim, and D. Gweon, "New open-loop actuating method of piezoelectric actuators for removing hysteresis and creep," *Review of Scientific Instruments*, vol. 71, no. 9, pp. 3436–3440, 2000.

[25] Y. Liu, J. Shan, and N. Qi, "Creep modeling and identification for piezoelectric actuators based on fractional-order system," *Mechatronics*, vol. 23, no. 7, pp. 840–847, 2013.

[26] M. Rakotondrabe, C. Clévy, and P. Lutz, "Complete open loop control of hysteretic, creeped, and oscillating piezoelectric cantilevers," *Automation Science and Engineering, IEEE Transactions on*, vol. 7, no. 3, pp. 440–450, 2010.

[27] X. Wang, V. Pommier-Budinger, A. Reysset, and Y. Gourinat, "Simultaneous compensation of hysteresis and creep in a single piezoelectric actuator by open-loop control for quasi-static space active optics applications," *Control Engineering Practice*, vol. 33, pp. 48–62, 2014.

[28] H. Song, R. Fraanje, G. Schitter, G. Vdovin, and M. Verhaegen, "Controller design for a high-sampling-rate closed-loop adaptive optics system with piezo-driven deformable mirror," *European Journal of Control*, vol. 17, no. 3, pp. 290–301, 2011.

[29] J. Minase, T.-F. Lu, B. Cazzolato, and S. Grainger, "Adaptive identification of hysteresis and creep in piezoelectric stack actuators," *The International Journal of Advanced Manufacturing Technology*, vol. 46, no. 9-12, pp. 913–921, 2010.

[30] B. Mokaberi and A. A. Requicha, "Compensation of scanner creep and hysteresis for afm nanomanipulation," *Automation Science and Engineering, IEEE Transactions on*, vol. 5, no. 2, pp. 197–206, 2008.

[31] C. Qi, F. Gao, H.-X. Li, S. Li, X. Zhao, and Y. Dong, "An incremental hammerstein-like modeling approach for the decoupled creep, vibration and hysteresis dynamics of piezoelectric actuator," *Nonlinear Dynamics*, vol. 82, no. 4, pp. 2097–2118, 2015.

[32] L. Liu, K. K. Tan, C. S. Teo, S.-L. Chen, and T. H. Lee, "Development of an approach toward comprehensive identification of hysteretic dynamics in piezoelectric actuators," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1834–1845, 2013.

[33] G.-m. Dai, "Modified hartmann-shack wavefront sensing and iterative wavefront reconstruction," in *1994 Symposium on Astronomical Telescopes & Instrumentation for the 21st Century*, pp. 562–573, International Society for Optics and Photonics, 1994.

[34] M. Tomizuka, "Zero phase error tracking algorithm for digital control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, no. 1, pp. 65–68, 1987.

[35] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *International Journal of control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[36] D. Croft and S. Devasia, "Vibration compensation for high speed scanning tunneling microscopy," *Review of Scientific Instruments*, vol. 70, no. 12, pp. 4600–4605, 1999.

[37] G. Schitter and A. Stemmer, "Identification and open-loop tracking control of a piezoelectric tube scanner for high-speed scanning-probe microscopy," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 3, pp. 449–454, 2004.

Appendix A

Zernikes in Cartesian coordinates

Zernike number	n	m	$Z_n^m(x,y)$
0	0	0	$\sqrt{2}$
1	1	-1	$2y$
2	1	1	$2x$
3	2	-2	$\sqrt{6}2yx$
4	2	0	$\sqrt{3}(-1+2y^2+2x^2)$
5	2	2	$\sqrt{6}(-y^2+x^2)$
6	3	-3	$2\sqrt{2}(-y^3+3yx^2)$
7	3	-1	$2\sqrt{2}(-2y+3y^3+3yx^2)$
8	3	1	$2\sqrt{2}(-2x+3x^3+3y^2x)$
9	3	3	$2\sqrt{2}(x^3-3y^2x)$
10	4	-4	$\sqrt{10}(-4y^3x+4yx^3)$
11	4	-2	$\sqrt{10}(-6yx+8y^3x+8yx^3)$
12	4	0	$\sqrt{5}(1-6y^2-6x^2+6y^4+12y^2x^2+6x^4)$
13	4	2	$\sqrt{10}(3y^2-3x^2-4y^4+4x^4)$
14	4	4	$\sqrt{10}(y^4-6y^2x^2+x^4)$
15	5	-5	$2\sqrt{3}(y^5-10y^3x^2+5yx^4)$
16	5	-3	$2\sqrt{3}(4y^3-12yx^2-5y^5+10y^3x^2+15yx^4)$
17	5	-1	$2\sqrt{3}(3y-12y^3-12yx^2+10y^5+20y^3x^2+10yx^4)$
18	5	1	$2\sqrt{3}(3x-12x^3-12y^2x+10x^5+20y^2x^3+10y^4x)$
19	5	3	$2\sqrt{3}(-4x^3+12y^2x+5x^5-10y^2x^3-15y^4x)$
20	5	5	$2\sqrt{3}(x^5-10y^2x^3+5y^4x)$
21	6	-6	$\sqrt{14}(6y^5x-20y^3x^3+6yx^5)$
22	6	-4	$\sqrt{14}(20y^3x-20yx^3-24y^5x+24yx^5)$
23	6	-2	$\sqrt{14}(30x^5y+60x^3y^3+30xy^5-40x^3y+40xy^3+12xy)$

Zernike number	n	m	$Z_n^m(x,y)$
24	6	0	$\sqrt{7}(-1+12y^2+12x^2-30y^4-60y^2x^2-30x^4$ $+20y^6+60y^4x^2+60y^2x^4+20x^6)$
25	6	2	$\sqrt{14}(-6y^2+6x^2+20y^4-20x^4-15y^6-15y^4x^2+15y^2x^4+15x^6)$
26	6	4	$\sqrt{14}(-5y^4+30y^2x^2-5x^4+6y^6-30y^4x^2-30y^2x^4+6x^6)$
27	6	6	$\sqrt{14}(-y^6+15y^4x^2-15y^2x^4+x^6)$

Table A-1: Zernike polynomials in Cartesian coordinates

Appendix B

SISO results

Actuator	Dynamic GPI model	Static GPI model	No compensation
1	0.0375	0.0494	0.1352
2	0.0428	0.0426	0.1710
3	0.0431	0.0773	0.2469
4	0.0916	0.0442	0.1604
5	0.0596	0.0487	0.2274
6	0.0780	0.0511	0.2278
7	0.0463	0.0408	0.1784
8	0.0336	0.0697	0.1725
9	0.0367	0.0526	0.1120
10	0.0528	0.0598	0.2483
11	0.0307	0.0483	0.2178
12	0.0514	0.0434	0.1626
13	0.0355	0.0562	0.2302
14	0.0501	0.0318	0.1451
15	0.0397	0.0633	0.2503
16	0.0532	0.0362	0.1520
17	0.0546	0.0592	0.2338
18	0.0471	0.0506	0.2353
mean	0.0491	0.0514	0.1948

Table B-1: e_{NRMS} of single actuators of the lens with a GPI model without dynamics compared to a GPI model with dynamics and no compensation

Actuator	Dynamic GPI model	Static GPI model	No compensation
1	0.0431	0.0639	0.1681
2	0.0760	0.0438	0.1136
3	0.0441	0.0944	0.1684
4	0.0618	0.0387	0.1487
5	0.0483	0.0562	0.1696
6	0.0425	0.0549	0.1741
7	0.0713	0.0463	0.1311
8	0.0506	0.0701	0.1217
9	0.0540	0.0471	0.1207
10	0.0220	0.0630	0.1773
11	0.0250	0.0614	0.1670
12	0.0505	0.0487	0.1104
13	0.0348	0.0668	0.1850
14	0.0457	0.0387	0.1180
15	0.0462	0.0627	0.1700
16	0.0411	0.0489	0.1322
17	0.0322	0.0596	0.1698
18	0.0289	0.0601	0.1742
mean	0.0454	0.0570	0.1511

Table B-2: e_{NRMS} of single actuators for the dataset multiplied by minus 1 of the lens with a GPI model without dynamics compared to a GPI model with dynamics and no compensation

Actuator	VAF
1	0.9997
2	0.9995
3	0.9937
4	0.9962
5	0.9959
6	0.9994
7	0.9996
8	0.9997
9	0.9970
10	0.9906
11	0.9966
12	0.9995
13	0.9996
14	0.9997
15	0.9937
16	0.9982
17	0.9991
18	0.9909
19	0.9992
Mean	0.9972

Table B-3: VAF of a dynamic GPI model with 6 operators of the mirror on a validation dataset

Actuator	VAF
1	0.9985
2	0.9980
3	0.9980
4	0.9901
5	0.9960
6	0.9930
7	0.9977
8	0.9989
9	0.9986
10	0.9969
11	0.9990
12	0.9971
13	0.9987
14	0.9973
15	0.9983
16	0.9969
17	0.9967
18	0.9976
Mean	0.9971

Table B-4: VAF of a dynamic GPI model with 6 operators of the lens on a validation dataset

Appendix C

MATLAB code

C-1 calccentroid

```
function [x,y]=calccentroid(I_ROI,c_x,c_y)
% Calculates centroid of region of interest
% Inputs:
%   I_ROI = measured intensity in ROI
%   c_x = x coordinates of ROI
%   c_y = y coordinates of ROI
% Outputs:
%   x = x coordinate of centroid
%   y = y coordinate of centroid

x=sum(c_x*sum(I_ROI,1)')/sum(sum(I_ROI));
y=sum(c_y*sum(I_ROI,2))/sum(sum(I_ROI));
```

C-2 calimirror

```
function calimirror(t_end,newcali)
% This function calibrates the lens
% with a exponentially decaying sinusoid
% Inputs:
%   t_end =  duration of calibration (min 10 seconds)
%            preferably 50 seconds or more
%   newcali = toggle 1 to create new calibration file cali.mat

N_act=20; % Amount of actuators
idedactuators=[2:20]; % Used actuators
dt=0.1; % Actuating frequency
t=0:dt:t_end; % Time vector

n=16; % Amount of different x positions in grid
```

```
15 m=15; % Amount of different y positions in grid
16
17 pitch_camera=5.2*1e-6; % from specsheet [m]
18 pitch_lenslets=300*1e-6; % from specsheet [m]
19 sa_radius_m=(300e-6)/2; % pitch lenslets/2
20 pixsize=5.2e-6; %pixel size camera
21
22 I_min_centroid=20;
23 I_min_spot=50;
24 r_centroid=15; % Radius of spot
25
26 %sinusoid with amplitude 1 followed by an
27 % exponentially decaying sinusoid and zeros
28 v=zeros(length(t),N_act);
29 for k=idedactuators
30     v(:,k)=[[sin(4/3*pi*t(1:30))]';[0.89.^(t(31:end-1)...
31     -1).*sin(4/3*pi*t(31:end-1))]';0];
32 end
33
34 %% Apply exponentially decaying sinusoid
35 tic
36 for i=1:length(t);
37 %% Apply voltages
38 % please change when using another device
39     edac40('write',1,1.8*v(i,:)+1.8);
40     while toc<dt*i
41     end
42 end
43
44 %% Take image
45 if newcali==1
46     % please change when using another device
47     image = ueye('capture', 1);
48
49 %% Finds local maxima and constructs a new calibration image
50
51     % Filter image
52     mask=image>I_min_centroid;
53     imagesub=double(image).*mask;
54     maximage=255;
55     i=0;
56
57     centers=zeros(1024,1280);
58
59     % While loop that detects all spots with a brightness
60     % above I_min_spot with ROI inside the image
61     while double(maximage)>I_min_spot
62         i=i+1;
63         [maximagesub,imax]=max(imagesub);
64         [maximage,imax2]=max(maximagesub);
65         if ((imax<(r_centroid+1)|imax>length(imagesub(:,1))-...
66             (r_centroid+1))|(imax2<(r_centroid+1)|...
67             imax2>length(imagesub(1,:))-(r_centroid+1)))
68             % If the ROI falls outside the acquired image,
69             % the spot is not taken into account
70             imagesub(max(imax(imax2)-r_centroid,1):min(imax(imax2),end)+...
71             r_centroid,max(imax2-r_centroid,1):min(imax2+r_centroid,end))=...
```

```
72              zeros(size(imagesub(max(imax(imax2)-r_centroid,1):min(...
73              imax(imax2),end)+r_centroid,max(imax2-r_centroid,1):min(...
74              imax2+r_centroid,end))));
75              i=i-1;
76          else
77              [centrs(i,1),centrs(i,2)]=calccentroid(imagesub...
78              (imax(imax2)-r_centroid:imax(imax2)+r_centroid,imax2-...
79              r_centroid:imax2+r_centroid),imax2-r_centroid:imax2+...
80              r_centroid,imax(imax2)-r_centroid:imax(imax2)+r_centroid);
81              cvmax(i,1:3)=[double(imax2),double(imax(imax2)),...
82              double(maximage)];
83              imagesub(imax(imax2)-r_centroid:imax(imax2)+r_centroid,...
84              imax2-r_centroid:imax2+r_centroid)=zeros(2*r_centroid+1);
85              centers(round(centrs(i,1)),round(centrs(i,2)))=255;
86          end
87      end
88
89  %% Determine reference coordinates
90      pitch=pitch_lenslets/pitch_camera; %pitch of lenslets in pixels
91      % Determine spot in the middle
92      theta0=[(max(centrs(:,1))+min(centrs(:,1)))/2,...
93      (max(centrs(:,2))+min(centrs(:,2)))/2];
94      % Find parameters of the grid with equally spaced spots that
95      % matches the found spots in an optimal way
96      theta=fminsearch(@(theta)fitgrid(theta,pitch,centrs,n,m),theta0);
97      % Create grid with found parameters
98      centrsref=makegrid(theta,pitch,centrs,n,m);
99
100 %% Toggle on to view the fitted reference positions to
101 % the observed spots
102 %      figure
103 %      hold on
104 %      plot(centrs(:,1),centrs(:,2),'*')
105 %      plot(centrsref(:,1),centrsref(:,2),'*')
106
107 %% Calculate the derivates of all the Zernike modes at all the
108     % normalized reference locations
109     for i=1:length(centrsref)
110         radius_spot(i)=norm(centrsref(i,:)-theta0)*...
111         pixsize+sa_radius_m;
112     end
113     pupil_radius_m=max(radius_spot);
114
115 %% Fill matrix E with zernike derivates to dx and dy
116     for k_subap=1:length(centrsref);
117         E([k_subap,k_subap+length(centrsref)],:)=...
118         derzer((centrsref(k_subap,1)-theta0(1))*...
119         pixsize/pupil_radius_m,(centrsref(k_subap,2)-theta0(2))*...
120         pixsize/pupil_radius_m,sa_radius_m/pupil_radius_m)...
121         /pupil_radius_m;
122     end
123
124 %% Calculate matrix B which relates the measured zernikes z and
125 % the observed displacements of the spots S as follows: z=B*S
126     B = pinv(E(:, 2:end));
127
128     % Save coordinates of spots in grid to cali.mat (reference
```

```
129     % position for no zernike abberations) and setup
130     % Guang-Ming Dai's reconstruction matrix
131     save('cali.mat','centrsref','B')
132 end
```

C-3 createinvmodel

```
1  function [invp,invr,p,r,vaf,offset,Hestinv,Hest]=...
2      createinvmodel(u,y,dt,N_op,n,wn,order_act)
3  % This function calculates the model and inverse model with
4  % input and output data
5  % Inputs:
6  %    u = input
7  %    y = output
8  %    N_op = amount of operators
9  %    n = toggle 0 for static model, toggle 1 for dynamic model
10 %    wn = frequency at which an extra pole is
11 %         added to make the inverse
12 %    order_act = order of identified model
13 % Outputs:
14 %    invp = weights of inverse model
15 %    invr = thresholds of inverse model
16 %    p = weights of model
17 %    r = thresholds of model
18 %    vaf = residual of the cost function
19 %    offset = parameters of the polynomial of the GPI model
20 %    Hestinv = transfer function of inverse model
21 %    Hest = transfer function of model
22
23 x0est=zeros(1,N_op); % Initial value of operators
24
25 r=max(abs(u))*(0:N_op-1)/(N_op); %thresholds
26 %% Calculate operator values
27 xest(:,1)=u;
28 xest(1,2:N_op)=max(u(1)-r(2:end),min(u(1)+r(2:end),x0est(2:end)));
29 for k=2:length(u);
30     xest(k,2:end)=max(u(k)-r(2:end),...
31     min(u(k)+r(2:end),xest(k-1,(2:end))));
32 end
33
34 %% Determine weights and transfer function
35 [~,~,p]=mols(xest,y,max(order_act,1));
36 opt=tfestOptions('InitMethod','all');
37 Hest=tfest(iddata(y(1:100),u(1:100),dt),order_act,order_act,...
38 'Ts',dt,'InputDelay',0,opt);
39
40 %% Refine estimate
41
42 options=optimset('MaxfunEvals',20000,'MaxIter',20000);
43 theta=fminunc(@(theta)errormodel(theta,u,y), [p;0;1;0]',options);
44
45 if n==0;
46     G=fminunc(@(G)optgain(G,theta,u,y),1,options);
47     [~,y_est]=optgain(G,theta,u,y); % Calculate y_est
48     p=p*G; % Include the static gain in p
```

```
49 else
50     [~,y_est]=errormodel(theta,u,y); % Calculate y_est
51 end
52 vaf=(1-sum((y-y_est).^2)/sum(y.^2));
53
54 %% Function which gives the error with as input the weights p and
55 % the polynomial offset
56 function [e,y_est]=errormodel(theta,u,y)
57     p=theta(1:N_op)';
58     if N_op>1
59         offset=theta(end-2:end)';
60     else
61         offset=[0,1,0]; % If only 1 operator is used, set g(u)=u
62     end
63     % Calculate operator values
64     xest(:,1)=polyval(offset,u);
65     xest(1,2:N_op)=max(polyval(offset,u(1))-r(2:end),min(...
66     polyval(offset,u(1))+r(2:end),x0est(2:end)+offset(end)));
67     for k=2:length(u);
68         xest(k,2:end)=max(polyval(offset,u(k))-r(2:end),min(...
69         polyval(offset,u(k))+r(2:end),xest(k-1,(2:end))));
70     end
71     offset(end+1)=-xest(1,:)*p; % Calculate g_4
72     % Calculate output
73     y_est=xest*p+offset(end);
74     if order_act>0
75     y_est=lsim(Hest,y_est,0:dt:dt*(length(y_est)-1));
76     end
77     e=((y_est-y)'*(y_est-y)/(y'*y));
78 end
79
80 %% Function to compute a static gain G instead of
81 % a the transfer function Hest
82 function [e,y_est]=optgain(G,theta,u,y)
83     p=theta(1:N_op)';
84     if N_op>1
85         offset=theta(end-2:end)';
86     else
87         offset=[0,1,0]; % If only 1 operator is used, set g(u)=u
88     end
89     % Calculate operator values
90     xest(:,1)=polyval(offset,u);
91     xest(1,2:N_op)=max(polyval(offset,u(1))-r(2:end),min(...
92     polyval(offset,u(1))+r(2:end),x0est(2:end)+offset(end)));
93     for k=2:length(u);
94         xest(k,2:end)=max(polyval(offset,u(k))-r(2:end),min(...
95         polyval(offset,u(k))+r(2:end),xest(k-1,(2:end))));
96     end
97     % Calculate output
98     offset(end+1)=-xest(1,:)*(p*G); % Calculate g_4
99     y_est=xest*(p*G)+offset(end);
100    e=((y_est-y)'*(y_est-y)/(y'*y));
101 end
102
103 %% Plot fit to hysteresis curve
104 % figure
105 % plot(u,y)
```

```
% hold on
% plot(u,y_est)

%% Plot fit to identification data
figure(1)
plot(y)
hold on
plot(y_est,'*')
axis('tight')
grid on
set(gca,'FontSize', 12);
xlabel('Time [s]')
ylabel('Influence function [-]')
legend('Identification data','Fitted model','Location','best')

%% Invert PI model
if N_op>1
    [invp,invr]=invPI(p',r);
else
    invp=1/p;
    invr=r;
end

%% Create inverse transfer function (ZPET)
if not(n==0)
    Hest=tf(Hest.num,Hest.den,dt);
    [Z,P,~] = tf2zp(Hest.num{1,1},Hest.den{1,1});
    Zinv=P;
    Pinv=[];
    for i=1:length(Z)
        if abs(Z(i))<1
           Pinv=[Pinv;Z(i)];
        else
            Zinv=[Zinv;-Z(i)];
            disp('RHP zero fitted')
        end
    end
    Hestinv=zpk(Zinv,Pinv,1,dt);
    Hestinv=Hestinv*c2d(tf(1,[1/wn,1]),dt)^...
    (length(Zinv)-length(Pinv));
    Hestinv=Hestinv*dcgain(1/Hest)/dcgain(Hestinv);
end
end
```

C-4 derzer

```
function e=derzer(X,Y,r_sa)
% This function calculates the derivate Z_n to
% dx and dy for the position
% Outputs
%   e = vector with derivates to to zernike terms to dx and dy
% Inputs
%   X = x coordinate of reference spot
%   Y = y coordinate of reference spot
%   r_sa = radius of subaperture
```

```
10
11 %% Integration limits for y as function
12 % of x when the area to be integrated
13 % is a circle with radius r_sa
14 ymin=@(x)-sin(acos(x/r_sa)).*r_sa;
15 ymax=@(x)sin(acos(x/r_sa)).*r_sa;
16
17 %% Derivates to dx
18 ex1=@(x,y)0+0*(x+X)+0*(y+Y);
19 ex2=@(x,y)0+0*(x+X)+0*(y+Y);
20 ex3=@(x,y)2.0+0*(x+X)+0*(y+Y);
21 ex4=@(x,y)sqrt(6.0).*(y+Y).*2.0;
22 ex5=@(x,y)sqrt(3.0).*(x+X).*4.0;
23 ex6=@(x,y)sqrt(6.0).*(x+X).*2.0;
24 ex7=@(x,y)sqrt(2.0).*(x+X).*(y+Y).*1.2e1;
25 ex8=@(x,y)sqrt(2.0).*(x+X).*(y+Y).*1.2e1;
26 ex9=@(x,y)sqrt(2.0).*((x+X).^2.*9.0+(y+Y).^2.*3.0-2.0).*2.0;
27 ex10=@(x,y)sqrt(2.0).*((x+X).^2.*3.0-(y+Y).^2.*3.0).*2.0;
28 ex11=@(x,y)sqrt(1.0e1).*((x+X).^2.*(y+Y).*1.2e1-(y+Y).^3.*4.0);
29 ex12=@(x,y)sqrt(1.0e1).*((y+Y).*-6.0+(x+X).^2.*(y+Y).*2.4e1+...
30 (y+Y).^3.*8.0);
31 ex13=@(x,y)sqrt(5.0).*((x+X).*-1.2e1+(x+X).*(y+Y).^2.*2.4e1+...
32 (x+X).^3.*2.4e1);
33 ex14=@(x,y)-sqrt(1.0e1).*((x+X).*6.0-(x+X).^3.*1.6e1);
34 ex15=@(x,y)-sqrt(1.0e1).*((x+X).*(y+Y).^2.*1.2e1-(x+X).^3.*4.0);
35 ex16=@(x,y)sqrt(3.0).*((x+X).*(y+Y).^3.*2.0e1-(x+X).^3.*(y+Y)...
36 .*2.0e1).*-2.0;
37 ex17=@(x,y)sqrt(3.0).*((x+X).*(y+Y).*-2.4e1+(x+X).*(y+Y).^3.*...
38 2.0e1+(x+X).^3.*(y+Y).*6.0e1).*2.0;
39 ex18=@(x,y)sqrt(3.0).*((x+X).*(y+Y).*-2.4e1+(x+X).*(y+Y).^3.*...
40 4.0e1+(x+X).^3.*(y+Y).*4.0e1).*2.0;
41 ex19=@(x,y)sqrt(3.0).*((x+X).^2.*(y+Y).^2.*6.0e1-(x+X).^2.*...
42 3.6e1+(x+X).^4.*5.0e1-(y+Y).^2.*1.2e1+(y+Y).^4.*1.0e1+3.0).*2.0;
43 ex20=@(x,y)sqrt(3.0).*((x+X).^2.*(y+Y).^2.*3.0e1+(x+X).^2.*1.2e1-...
44 (x+X).^4.*2.5e1-(y+Y).^2.*1.2e1+(y+Y).^4.*1.5e1).*-2.0;
45 ex21=@(x,y)sqrt(3.0).*((x+X).^2.*(y+Y).^2.*-3.0e1+(x+X).^4.*...
46 5.0+(y+Y).^4.*5.0).*2.0;
47 ex22=@(x,y)sqrt(1.4e1).*((x+X).^2.*(y+Y).^3.*-6.0e1+(x+X).^4.*...
48 (y+Y).*3.0e1+(y+Y).^5.*6.0);
49 ex23=@(x,y)-sqrt(1.4e1).*((x+X).^2.*(y+Y).*6.0e1-(x+X).^4.*...
50 (y+Y).*1.2e2-(y+Y).^3.*2.0e1+(y+Y).^5.*2.4e1);
51 ex24=@(x,y)sqrt(1.4e1).*(y+Y).*(((x+X).^2+(y+Y).^2).^2.*1.5e1-...
52 (x+X).^2.*2.0e1-(y+Y).^2.*2.0e1+6.0).*2.0-sqrt(1.4e1).*(x+X).*...
53 (y+Y).*((x+X).*4.0e1-(x+X).*((x+X).^2+(y+Y).^2).*6.0e1).*2.0;
54 ex25=@(x,y)sqrt(7.0).*((x+X).*2.4e1+(x+X).^3.*(y+Y).^2.*2.4e2-...
55 (x+X).*(y+Y).^2.*1.2e2+(x+X).*(y+Y).^4.*1.2e2-(x+X).^3.*...
56 1.2e2+(x+X).^5.*1.2e2);
57 ex26=@(x,y)sqrt(1.4e1).*((x+X).*1.2e1+(x+X).^3.*(y+Y).^2.*6.0e1-...
58 (x+X).*(y+Y).^4.*3.0e1-(x+X).^3.*8.0e1+(x+X).^5.*9.0e1);
59 ex27=@(x,y)-sqrt(1.4e1).*((x+X).^3.*(y+Y).^2.*1.2e2-(x+X).*...
60 (y+Y).^2.*6.0e1+(x+X).*(y+Y).^4.*6.0e1+(x+X).^3.*2.0e1-(x+X).^5.*3.6e1);
61 ex28=@(x,y)sqrt(1.4e1).*((x+X).^3.*(y+Y).^2.*-6.0e1+(x+X).*...
62 (y+Y).^4.*3.0e1+(x+X).^5.*6.0);
63
64 %% Derivates to dy
65 ey1=@(x,y)0+0*(x+X)+0*(y+Y);
66 ey2=@(x,y)2.0+0*(x+X)+0*(y+Y);
```

```
ey3=@(x,y)0+0*(x+X)+0*(y+Y);
ey4=@(x,y)sqrt(6.0).*(x+X).*2.0;
ey5=@(x,y)sqrt(3.0).*(y+Y).*4.0;
ey6=@(x,y)sqrt(6.0).*(y+Y).*-2.0;
ey7=@(x,y)sqrt(2.0).*((x+X).^2.*3.0-(y+Y).^2.*3.0).*2.0;
ey8=@(x,y)sqrt(2.0).*((x+X).^2.*3.0+(y+Y).^2.*9.0-2.0).*2.0;
ey9=@(x,y)sqrt(2.0).*(x+X).*(y+Y).*1.2e1;
ey10=@(x,y)sqrt(2.0).*(x+X).*(y+Y).*-1.2e1;
ey11=@(x,y)-sqrt(1.0e1).*((x+X).*(y+Y).^2.*1.2e1-(x+X).^3.*4.0);
ey12=@(x,y)sqrt(1.0e1).*((x+X).*-6.0+(x+X).*(y+Y).^2.*2.4e1+...
(x+X).^3.*8.0);
ey13=@(x,y)sqrt(5.0).*((y+Y).*-1.2e1+(x+X).^2.*(y+Y).*2.4e1+...
(y+Y).^3.*2.4e1);
ey14=@(x,y)sqrt(1.0e1).*((y+Y).*6.0-(y+Y).^3.*1.6e1);
ey15=@(x,y)-sqrt(1.0e1).*((x+X).^2.*(y+Y).*1.2e1-(y+Y).^3.*4.0);
ey16=@(x,y)sqrt(3.0).*((x+X).^2.*(y+Y).^2.*-3.0e1+(x+X).^4.*5.0+...
(y+Y).^4.*5.0).*2.0;
ey17=@(x,y)sqrt(3.0).*((x+X).^2.*(y+Y).^2.*3.0e1-(x+X).^2.*...
1.2e1+(x+X).^4.*1.5e1+(y+Y).^2.*1.2e1-(y+Y).^4.*2.5e1).*2.0;
ey18=@(x,y)sqrt(3.0).*((x+X).^2.*(y+Y).^2.*6.0e1-(x+X).^2.*...
1.2e1+(x+X).^4.*1.0e1-(y+Y).^2.*3.6e1+(y+Y).^4.*5.0e1+3.0).*2.0;
ey19=@(x,y)sqrt(3.0).*((x+X).*(y+Y).*-2.4e1+(x+X).*...
(y+Y).^3.*4.0e1+(x+X).^3.*(y+Y).*4.0e1).*2.0;
ey20=@(x,y)sqrt(3.0).*((x+X).*(y+Y).*-2.4e1+(x+X).*...
(y+Y).^3.*6.0e1+(x+X).^3.*(y+Y).*2.0e1).*-2.0;
ey21=@(x,y)sqrt(3.0).*((x+X).*(y+Y).^3.*2.0e1-(x+X).^3.*...
(y+Y).*2.0e1).*2.0;
ey22=@(x,y)sqrt(1.4e1).*((x+X).^3.*(y+Y).^2.*-6.0e1+(x+X).*...
(y+Y).^4.*3.0e1+(x+X).^5.*6.0);
ey23=@(x,y)sqrt(1.4e1).*((x+X).*(y+Y).^2.*6.0e1-(x+X).*(y+Y).^4.*...
1.2e2-(x+X).^3.*2.0e1+(x+X).^5.*2.4e1);
ey24=@(x,y)sqrt(1.4e1).*(x+X).*(((x+X).^2+(y+Y).^2).^2.*1.5e1-...
(x+X).^2.*2.0e1-(y+Y).^2.*2.0e1+6.0).*2.0-sqrt(1.4e1).*(x+X).*...
(y+Y).*((y+Y).*4.0e1-(y+Y).*((x+X).^2+(y+Y).^2).*6.0e1).*2.0;
ey25=@(x,y)sqrt(7.0).*((y+Y).*2.4e1+(x+X).^2.*(y+Y).^3.*2.4e2-...
(x+X).^2.*(y+Y).*1.2e2+(x+X).^4.*(y+Y).*1.2e2-(y+Y).^3.*...
1.2e2+(y+Y).^5.*1.2e2);
ey26=@(x,y)-sqrt(1.4e1).*((y+Y).*1.2e1+(x+X).^2.*(y+Y).^3.*...
6.0e1-(x+X).^4.*(y+Y).*3.0e1-(y+Y).^3.*8.0e1+(y+Y).^5.*9.0e1);
ey27=@(x,y)-sqrt(1.4e1).*((x+X).^2.*(y+Y).^3.*1.2e2-(x+X).^2.*...
(y+Y).*6.0e1+(x+X).^4.*(y+Y).*6.0e1+(y+Y).^3.*2.0e1-(y+Y).^5.*3.6e1);
ey28=@(x,y)-sqrt(1.4e1).*((x+X).^2.*(y+Y).^3.*-6.0e1+(x+X).^4.*...
(y+Y).*3.0e1+(y+Y).^5.*6.0);

%% Derivates to dx integrated over a circle with
% radius r_sa and center (X,Y)
e(1,1)=integral2(ex1,-r_sa,r_sa,ymin,ymax);
e(1,2)=integral2(ex2,-r_sa,r_sa,ymin,ymax);
e(1,3)=integral2(ex3,-r_sa,r_sa,ymin,ymax);
e(1,4)=integral2(ex4,-r_sa,r_sa,ymin,ymax);
e(1,5)=integral2(ex5,-r_sa,r_sa,ymin,ymax);
e(1,6)=integral2(ex6,-r_sa,r_sa,ymin,ymax);
e(1,7)=integral2(ex7,-r_sa,r_sa,ymin,ymax);
e(1,8)=integral2(ex8,-r_sa,r_sa,ymin,ymax);
e(1,9)=integral2(ex9,-r_sa,r_sa,ymin,ymax);
e(1,10)=integral2(ex10,-r_sa,r_sa,ymin,ymax);
e(1,11)=integral2(ex11,-r_sa,r_sa,ymin,ymax);
```

```
124 e(1,12)=integral2(ex12,-r_sa,r_sa,ymin,ymax);
125 e(1,13)=integral2(ex13,-r_sa,r_sa,ymin,ymax);
126 e(1,14)=integral2(ex14,-r_sa,r_sa,ymin,ymax);
127 e(1,15)=integral2(ex15,-r_sa,r_sa,ymin,ymax);
128 e(1,16)=integral2(ex16,-r_sa,r_sa,ymin,ymax);
129 e(1,17)=integral2(ex17,-r_sa,r_sa,ymin,ymax);
130 e(1,18)=integral2(ex18,-r_sa,r_sa,ymin,ymax);
131 e(1,19)=integral2(ex19,-r_sa,r_sa,ymin,ymax);
132 e(1,20)=integral2(ex20,-r_sa,r_sa,ymin,ymax);
133 e(1,21)=integral2(ex21,-r_sa,r_sa,ymin,ymax);
134 e(1,22)=integral2(ex22,-r_sa,r_sa,ymin,ymax);
135 e(1,23)=integral2(ex23,-r_sa,r_sa,ymin,ymax);
136 e(1,24)=integral2(ex24,-r_sa,r_sa,ymin,ymax);
137 e(1,25)=integral2(ex25,-r_sa,r_sa,ymin,ymax);
138 e(1,26)=integral2(ex26,-r_sa,r_sa,ymin,ymax);
139 e(1,27)=integral2(ex27,-r_sa,r_sa,ymin,ymax);
140 e(1,28)=integral2(ex28,-r_sa,r_sa,ymin,ymax);
141
142 %% Derivates to dy integrated over a circle with
143 % radius r_sa and center (X,Y)
144 e(2,1)=integral2(ey1,-r_sa,r_sa,ymin,ymax);
145 e(2,2)=integral2(ey2,-r_sa,r_sa,ymin,ymax);
146 e(2,3)=integral2(ey3,-r_sa,r_sa,ymin,ymax);
147 e(2,4)=integral2(ey4,-r_sa,r_sa,ymin,ymax);
148 e(2,5)=integral2(ey5,-r_sa,r_sa,ymin,ymax);
149 e(2,6)=integral2(ey6,-r_sa,r_sa,ymin,ymax);
150 e(2,7)=integral2(ey7,-r_sa,r_sa,ymin,ymax);
151 e(2,8)=integral2(ey8,-r_sa,r_sa,ymin,ymax);
152 e(2,9)=integral2(ey9,-r_sa,r_sa,ymin,ymax);
153 e(2,10)=integral2(ey10,-r_sa,r_sa,ymin,ymax);
154 e(2,11)=integral2(ey11,-r_sa,r_sa,ymin,ymax);
155 e(2,12)=integral2(ey12,-r_sa,r_sa,ymin,ymax);
156 e(2,13)=integral2(ey13,-r_sa,r_sa,ymin,ymax);
157 e(2,14)=integral2(ey14,-r_sa,r_sa,ymin,ymax);
158 e(2,15)=integral2(ey15,-r_sa,r_sa,ymin,ymax);
159 e(2,16)=integral2(ey16,-r_sa,r_sa,ymin,ymax);
160 e(2,17)=integral2(ey17,-r_sa,r_sa,ymin,ymax);
161 e(2,18)=integral2(ey18,-r_sa,r_sa,ymin,ymax);
162 e(2,19)=integral2(ey19,-r_sa,r_sa,ymin,ymax);
163 e(2,20)=integral2(ey20,-r_sa,r_sa,ymin,ymax);
164 e(2,21)=integral2(ey21,-r_sa,r_sa,ymin,ymax);
165 e(2,22)=integral2(ey22,-r_sa,r_sa,ymin,ymax);
166 e(2,23)=integral2(ey23,-r_sa,r_sa,ymin,ymax);
167 e(2,24)=integral2(ey24,-r_sa,r_sa,ymin,ymax);
168 e(2,25)=integral2(ey25,-r_sa,r_sa,ymin,ymax);
169 e(2,26)=integral2(ey26,-r_sa,r_sa,ymin,ymax);
170 e(2,27)=integral2(ey27,-r_sa,r_sa,ymin,ymax);
171 e(2,28)=integral2(ey28,-r_sa,r_sa,ymin,ymax);
172
173 %% Divide by integrated area such that
174 % average derivative is calculated
175 e=e./(pi.*r_sa.^2);
```

C-5 Driver_ mirror

```
% This file creates an identification dataset
% for an actuator of the lens
% Output format: 'testdata',[date],'act',[actuatornumber],'.mat'
% Parameters in output file:
%   t = time vector
%   v = voltage
%   z = measured zernikes
close all
clear all

N_act=20; % Amount of actuators
idedactuators=[2:20]; % Used actuators

%% Open camera
% please change when using another device
ueye('open',1, 1);
ueye('configure', 1, 0, [0,0,1280,1024], -1);
ueye('settiming', 1, 0.4, 30, 35);

%% Apply voltages
% please change when using another device
edac40('open',1,'169.254.159.198');
edac40('write',1,1.8*zeros(20,1)+1.8);
%%

N_zer=6; % Max order of measured zernikes

for k=idedactuators; % Actuator number to be run
    %% Calibrate
    calimirror(50,1); % Calibrate lens

    dt=0.1; % sample time
    t_end=200-dt; % duration of experiment
    t=0:dt:t_end;

    %% Actuate and measure
    v=zeros(length(t),N_act);
    % Create frequency sweep + low frequency sinusoid
    v(1301:end,k)=(chirp(t(1:end-1300),0.01,t(end-1300),1,...
    'quadratic')+sin(4/25*pi*t(1:end-1300)))/2;
    v(:,k)=[zeros(40,1);ones(60,1);zeros(10,1);-ones(40,1);...
    zeros(10,1);ones(40,1);zeros(10,1);-0.95*ones(40,1);...
    zeros(10,1);0.95*ones(40,1);zeros(10,1);-0.9*ones(40,1);...
    zeros(10,1);0.9*ones(40,1);zeros(10,1);-0.85*ones(40,1);...
    zeros(10,1);0.85*ones(40,1);zeros(10,1);-0.8*ones(40,1);...
    zeros(10,1);0.8*ones(40,1);zeros(10,1);-0.7*ones(40,1);...
    zeros(10,1);0.7*ones(40,1);zeros(10,1);-0.6*ones(40,1);...
    zeros(10,1);0.6*ones(40,1);zeros(10,1);-0.5*ones(40,1);...
    zeros(10,1);0.5*ones(40,1);zeros(10,1);-0.4*ones(40,1);...
    zeros(10,1);0.4*ones(40,1);zeros(10,1);-0.3*ones(40,1);...
    zeros(10,1);0.3*ones(40,1);zeros(10,1);-0.2*ones(40,1);...
    zeros(10,1);0.2*ones(40,1);zeros(10,1);-0.1*ones(40,1);...
    zeros(10,1);0.1*ones(40,1);v(1301:end,k)/max(abs(...
    v(1301:end,k)))]; % Add steps in beginning of the dataset
    z=zeros(length(t),sum(2:N_zer+1));

    %% Run identification
```

```
58      tic % Start timer
59      for i=1:length(t);
60          [z(i,:)]=measzer_mirror(v(i,:)); % Measure zernikes
61          while toc<dt*i
62              % wait till the time of the next sample is reached
63          end
64      end
65
66      save(strcat('testdata',date,'act',num2str(k),'.mat'),...
67      't','v','z') % Save identification data
68      % Set zero voltage to actuator
69      [~,~]=measzer_mirror(0*v(i,:),N_dis_spots,N_zer);
70  end
71  %% Set zero voltage at actuator and close connection
72  % please change when using another device
73  edac40('write',1,1.8*zeros(N_act,1));
74  edac40('close');
```

C-6 fit_ dyn_ PI

```
1   function [p,r,invp,invr,inffun,e,offset,Hest,Hinv]=...
2            fit_dyn_PI(N_op,order_act,wn,k,datstr,n)
3   % This function calculates loads identification data and
4   % identifies a model and an inverse model, as well
5   % as the influence function
6   % Inputs
7   %   N_op = amount of operators
8   %   n = toggle 0 for static model, toggle 1 for dynamic model
9   %   wn = frequency at which an extra pole is added to make
10  %   the inverse transfer function causal
11  %   k = actuator number
12  %   datstr = date of the used identification dataset
13  %   order_act = order of identified model
14  % Outputs:
15  %   p = weights of model
16  %   r = thresholds of model
17  %   invp = weights of inverse model
18  %   invr = thresholds of inverse model
19  %   inffun = influence function of the actuator
20  %   e = residual of the cost function
21  %   Hest = transfer function of model (if n>0)
22  %   Hinv = transfer function of inverse model (if n>0)
23
24  %load identificationdata
25  load(strcat('testdata',datstr,'act',num2str(k),'.mat'))
26
27  %%
28  z0=mean(z(1:35,:)); %Initial offset
29  % Zernike values at max and min voltage
30  vsub=[mean(v(120:150,k));mean(v(170:200,k))];
31  zsub=[mean(z(121:151,:)-ones(length(z(121:151,1)),1)*z0);...
32  mean(z(171:201,:)-ones(length(z(171:201,1)),1)*z0)];
33
34  % Determine influence function
35  for i=1:length(zsub(1,:));
```

```
36      inffun(1,i)=lsqlin(ones(length(zsub(:,i)),1),zsub(:,i)./vsub);
37  end
38
39  % Calculate magnitude of the influence function (output of actuator)
40  for i=1:length(v(:,k));
41      Minffun(i,1)=lsqlin(inffun(1,:)',[z(i,:)-z0]');
42  end
43
44  % Identify models
45  if n==0
46      [invp,invr,p,r,e,offset]=createinvmodel(v(:,k),...
47      Minffun(:),t(2)-t(1),N_op,n,wn,order_act);
48  else
49      [invp,invr,p,r,e,offset,Hinv,Hest]=createinvmodel(v(:,k),...
50      Minffun(:),t(2)-t(1),N_op,n,wn,order_act);
51  end
52  end
```

C-7 fitgrid

```
1   function e=fitgrid(theta,pitch,centrs,n,m)
2   % This function calculates the error of the fit of
3   % the grid with the measured spots
4   % Inputs
5   %   theta = coordinates of the middle position of the
6   %           grid in the image
7   %   pitch = distance between spots
8   %   centrs = measured positions of the spots
9   %   n = amount of rows of the grid
10  %   m = amount of columns of the grid
11  % Ouput
12  %   e = error between measured spots and fitted grid
13
14  centrsref=makegrid(theta,pitch,centrs,n,m); % Create grid
15  e=sum((centrs(:,1)-centrsref(:,1)).^2+(centrs(:,2)...
16  -centrsref(:,2)).^2); % Calculate error of fitted grid
```

C-8 invPI

```
1   function [invp,invr]=invPI(p,r)
2   % This function creates the inverse of a PI model
3   % Based on Al Janaideh, Mohammad, et al. "Generalized
4   % Prandtl–Ishlinskii hysteresis model: Hysteresis modeling and
5   % its inverse for compensation in smart actuators." Decision and
6   % Control, 2008. CDC 2008. 47th IEEE Conference on. IEEE, 2008.
7   % Inputs
8   %   p = vector with weights of PI model
9   %   r = vector with threshold values of PI model
10  % Outputs
11  %   invp = vector with weights of inverse PI model
12  %   invr = vector with threshold values of inverse PI model
13
```

```
14 if abs(r(1))>0
15     error('r(1) should be zero')
16 end
17
18 % First term is linear and its threshold does not have to be inverted
19 q=p(1);
20 p=p(2:end);
21 r=r(2:end);
22
23 % Initialize matrices
24 invr=zeros(size(r));
25 invp=zeros(size(p));
26
27 % Calculate parameters of inverse model
28 invr(1)=q*r(1);
29 invp(1)=-p(1)/((q+p(1))*(q));
30
31 for j=2:length(p);
32     s1=0;
33     for i=1:j-1
34         s1=s1+p(i)*(r(j)-r(i));
35     end
36     invr(j)=q*r(j)+s1;
37     invp(j)=-p(j)/((q+sum(p(1:j)))*(q+sum(p(1:j-1))));
38 end
39
40 invr=[0,invr];
41 invp=[1/q,invp];
```

C-9 LocReg

```
1  function Theta=LocReg(X,Y,Q,R,l)
2  % This function solves the least squares problem min_y
3  % ((ref_z-z(y)))Q((ref_z-z(y)))^T+Ry^1 with the
4  % constraint that the min/max value of
5  % an element in y is l
6  % Inputs:
7  %   X = matrix with influence functions
8  %   Y = vector with desired zernikes
9  %   R = cost of control
10 %   Q = cost of tracking error
11 %   l = min/max value of an element of y
12 % Outputs:
13 %   Theta = vector y that gives the
14 %           solution of the minimization problem
15 warning('off','all')
16 Theta=lsqlin([diag(sqrt(Q))*X;diag(sqrt(R))],[diag(sqrt(Q))*Y;zeros(...
17 length(X(1,:)),1)],[diag(ones(length(X(1,:)),1));-diag(ones(length(...
18 X(1,:)),1))],l*ones(2*length(X(1,:)),1));
```

C-10 makegrid

```
function centrsref=makegrid(theta,pitch,centrs,n,m)
% This function creates a grid with equally spaced points fitted
% to the measured measured position of the spots
% Inputs
%   theta = coordinates of the middle position
%           of the grid in the image
%   pitch = distance between spots
%   centrs = measured positions of the spots
%   n = amount of rows of the grid
%   m = amount of columns of the grid
% Outputs
%   centrsref = grid with reference position of the spots

% Create rectangular grid with parameters
centrsgrid=[combvec(-pitch*((n-1)/2)+theta(1):pitch:pitch*...
((n-1)/2)+theta(1),-pitch*((m-1)/2)+theta(2):pitch:pitch*...
((m-1)/2)+theta(2))]';

% Initialize vector with used points in grid
% (since the used grid is not rectangular)
centrsref=zeros(length(centrs),2);
for i=1:length(centrs)
    % Calculate squared error between
    % every point of fitted grid and measured point
    e=(centrs(i,1)-centrsgrid(:,1)).^2+...
    (centrs(i,2)-centrsgrid(:,2)).^2;
    [~,I]=min(e);
    % Fill vector with nearest point in grid to measured points
    centrsref(i,:)=centrsgrid(I,:);
end
```

C-11 measzer_ mirror

```
function [z]=measzer_mirror(v)
% This function calculates the zernike coefficients
% and puts a new voltage on the mirror
% Inputs:
%   v = voltage to put onto the mirror
% Outputs
%   z = measured zernike values

I_min_centroid=20;
r_centroid=15; % Radius of region which is taken into account
% for calculating the centroid of the spot
r_centroid_min=35; % Radius of region of interest

%% Characteristics SH
f = 18.6e-3;           % focal length lenslets SH from specsheet [m]
pitch_camera=5.2*1e-6; % from specsheet [m]

%% Take image
% please change when using another device
image = ueye('capture', 1);
%% Determine position of centroids from image
load('cali.mat') % Import calibration positions of spots
```

```
23
24 % Filter image
25 mask=image>I_min_centroid;
26 imagesub=double(image).*mask;
27
28 % Determine maximum in ROI
29 for i=1:length(centrsref);
30     image_ROI=imagesub(round(centrsref(i,2))-r_centroid_min:...
31     round(centrsref(i,2))+r_centroid_min,round(centrsref(i,1))...
32     -r_centroid_min:round(centrsref(i,1))+r_centroid_min);
33     [maximagesub,imax]=max(image_ROI(R+1:end-R-1,:));
34     [maximage,imax2]=max(maximagesub(R+1:end-R-1));
35      imax=imax+R;
36      imax2=imax2+R;
37     if maximage==0 % Check if spot is there
38         error('Spot has dissappeared')
39     else
40         [centrsx,centrsy]=calccentroid(image_ROI(imax(imax2)-R...
41         :imax(imax2)+R,imax2-R:imax2+R),imax2-R:imax2+R...
42         ,imax(imax2)-R:imax(imax2)+R);
43         centrs(i,1:3)=[round(centrsref(i,1))-r_centroid_min+...
44         centrsx,round(centrsref(i,2))-r_centroid_min+centrsy,maximage];
45     end
46 end
47
48 %% Calculate zernikes
49 xcoordinate_wfs=centrs(:,1);
50 ycoordinate_wfs=centrs(:,2);
51 xcoordinate_ref=centrsref(:,1);
52 ycoordinate_ref=centrsref(:,2);
53
54 %% Calculate the displacements in the X and Y directions
55
56 disp_x = (xcoordinate_wfs - xcoordinate_ref); %[pix]
57 disp_y = (ycoordinate_wfs - ycoordinate_ref); %[pix]
58
59 %% Construct the S with the wavefront gradients
60 S = (pitch_camera)*[disp_x ; disp_y]/f;
61
62 %% Reconstruct the zernikes in [m]
63 z=(B*S);
64
65 %% Apply voltages
66 N_act=20; % Amount of actuators
67 v=min([ones(1,N_act)],max(-[ones(1,N_act)],v));
68 edac40('write',1,1.8*v+1.8); % please change when using another device
```

C-12 mols

```
1 function [a,b,p]=mols(u,y,n)
2 % This function calculates the weights p and coefficients
3 % a and b of the transfer function with as input the
4 % operator values and the output
5 % Inputs:
6 %   u = matrix of values of operators
```

```
%   y = vector with output
%   n = order of transfer function to be identified
% Outputs:
%   a = denominator coefficients of fitted model
%   b = numerator coefficients of fitted model
%   p = weights of fitted model

[k,N]=size(u);
U=zeros(k,n*(N+1));

for i=n+1:k
    for j=1:n
        for m=1:N
            U(i,1+n*(m-1):n*m)=u(i-1:-1:i-n,m);
        end
    end
    U(i,N*n+1:(N+1)*n)=y(i-1:-1:i-n);
end
U=U(n+1:end,:);
y=y(n+1:end);

%% Calculate least squares solution
Theta=lsqlin(U,y);

%% Fill matrix Psi and compute its SVD
H=zeros(n,N);
for i=1:N
    H(:,i)=Theta(1+n*(i-1):n*i);
end

[svd1,svd2,svd3]=svd(H);

%% Calculate parametervalues
p=svd3(:,1).*svd2(1,1);
b=svd1(:,1);
a=[1;-Theta(n*N+1:end)];

%% Make first weight positive
if p(1)<0
    p=-p;
    b=-b;
end
```

C-13 simPI

```
function y=simPI(u,r,p,x0)
% This function simulates a PI model
% Inputs:
%   u = input
%   r = threshold values
%   p = weights
%   x0 = previous value
% Outputs
%   y = output

```

```
N_op=length(p);

% Calculate operator values
x(:,1)=u;
x(1,2:N_op)=max(u(1)-r(2:end),min(u(1)+r(2:end),x0(2:N_op)));
for k=2:length(u);
    x(k,2:N_op)=max(u(k)-r(2:end),min(u(k)+r(2:end),x(k-1,2:N_op)));
end

% Calculate output
y=x*p';
```

C-14 simPI_ asym

```
function y=simPI_asym(u,r,p,offset,x0)
% This function simulates an assymetric PI model
% Inputs:
%   u = input
%   r = threshold values
%   p = weights
%   offset = polynomial values of model
%   x0 = previous value
% Outputs
%   y = output

N_op=length(p);

% Calculate operator values
xest(:,1)=polyval(offset(1:3),u);
xest(1,2:N_op)=max(polyval(offset(1:3),u(1))-r(2:end),...
min(polyval(offset(1:3),u(1))+r(2:end),x0(2:end)+offset(3)));
for k=2:length(u);
    xest(k,2:end)=max(polyval(offset(1:3),u(k))-r(2:end),...
    min(polyval(offset(1:3),u(k))+r(2:end),xest(k-1,(2:end))));
end

% Calculate output
y=xest*p'+offset(end); % add g_4
```

C-15 simPI_ asymi

```
function y=simPI_asymi(u,r,p,offset,x0)
% This function simulates an assymetric PI model
% Inputs:
%   u = input
%   r = threshold values
%   p = weights
%   offset = polynomial values of model
%   x0 = previous value
% Outputs
%   y = output

```

```
N_op=length(p);

% Calculate operator values
x(:,1)=u-offset(end);
x(1,2:N_op)=max(u(1)-r(2:end)-offset(end),min(u(1)+r(2:end)-...
offset(end),x0(2:N_op)-offset(end)));
for k=2:length(u);
    x(k,2:N_op)=max(u(k)-r(2:end)-offset(end),min(u(k)+...
    r(2:end)-offset(end),x(k-1,2:N_op)));
end

% Calculate output
y=(-offset(2)+sqrt(offset(2)^2-4*offset(1)*...
(offset(3)-(x*p'))))/(2*offset(1));
% since if ax^2+bx+c-y=0 --> y=(-b+-sqrt(b^2-4a(c-y)))/(2a)
```

C-16 Testcomp_ mirror

```
% This function tests the compensation of the inverse model
% Output file is: 'comp',[date of identification dataset],'act',
% [actuator number],'N_op',[amount of operators],'n',[0 for
% static model, 1 for dynamic model]
clear all
close all
warning('off','all')
N_zer=6; % Maximum order of measured zernike
N_act=20; % Amount of actuators
idedactuators=[2:20]; % Used actuators
for N_op=[1:10]; % Amount of operators
    for n=[0,1]; % toggle 1 for dynamic model, 0 for static
        identifynew=1; % Toggle 0 to use previously identified
        % models
        datstr='09-Oct-2016'; % Choose date of identification
        if identifynew==0;
            try
                load(strcat('fittedmodelsN',num2str(N_op),'n',...
                num2str(n),datstr)) % Check if there is a
                % previously identified model
            catch
                identifynew=1; % If not found, identify new model
            end
        end
        if identifynew==1
            % Order of the identified model for each actuator
            % Chosen on basis of visual inspection of fitted tfs
            % to step responses
            order_act=[0,2,2,2,2,2,2,2,2,2,2,1,2,2,2,1,2,2,1,2];

%% Initialize sizes of matrices of model parameters
            inffun=zeros(N_act,sum(2:N_zer+1));
            p=zeros(N_act,N_op);
            r=zeros(N_act,N_op);
            invp=zeros(N_act,N_op);
            invr=zeros(N_act,N_op);
            offset=zeros(N_act,4);
```

```
38
39 %% Fit model
40             wn=2000; % Frequency at which poles are added to
41             % ensure causality of the inverted transfer function
42             if not(n==0) % Actuators to be identified
43                 parfor k=idedactuators;
44                     [p(k,:),r(k,:),invp(k,:),invr(k,:),...
45                     inffun(k,:),e(k),offset(k,:),Hest(k),...
46                     Hinv(k)]=fit_dyn_PI(N_op,order_act(k),...
47                     wn,k,datstr,n);
48                 end
49             else
50                 parfor k=idedactuators;
51                     [p(k,:),r(k,:),invp(k,:),invr(k,:),...
52                     inffun(k,:),e(k),offset(k,:)]=...
53                     fit_dyn_PI(N_op,order_act(k),wn,k,datstr,n);
54                 end
55             end
56
57 %% Save model
58             disp(strcat('fittedmodelsN',num2str(N_op),'n',...
59             num2str(n),datstr,' fitted'))
60             if n==0;
61                 save(strcat('fittedmodelsN',num2str(N_op),'n',...
62                 num2str(n),datstr),'N_op','order_act','n',...
63                 'inffun','p','r','invp','invr','offset','wn','e')
64             else
65                 save(strcat('fittedmodelsN',num2str(N_op),'n',...
66                 num2str(n),datstr),'N_op','order_act','n',...
67                 'inffun','p','r','invp','invr','offset',...
68                 'wn','Hest','Hinv','e')
69             end
70         end
71
72         edac40('open',1,'169.254.159.198'); % Connect to mirror,
73         % please change when using another device
74
75         for rngseed=[1]; % Use another number to generate
76             % different references for MIMO datasets
77             testedactuators=[2:20]; % Actuator number to be
78             % tested, toggle N_act+1 to test MIMO
79 %% Test compensators
80             for k=testedactuators
81                 ueye('open',1, 1);
82                 ueye('configure', 1, 0, [0,0,1280,1024], -1);
83                 ueye('settiming', 1, 0.4, 30, 35);
84
85                 calimirror(50,1); % Calibrate lens
86
87                 % set same seed for random numbers
88                 % for reproducecibility
89                 clear rng
90                 rng(1)
91
92                 dt=0.1; % Sample time
93                 t_end=40; % Duration of experiment
94                 t=0:dt:t_end;
```

```
95                  v=zeros(length(t),N_act);
96                  yd_ref=zeros(length(t),N_act);
97
98  %% Calculate input signal
99                  % Create Butterworth filter
100                 [b_filt,a_filt]=butter(6,0.1);
101                 x0=zeros(1,N_op);
102                 if not(k==N_act+1)
103 %% Calculate input for SISO
104                     ref=zeros(length(t),N_act);
105                     ref(:,k)=zeros(length(t),1);
106                     rng(1)
107                     ref(1:end,k)=2.5*(-0.5+rand(length(t),1));
108                     rng(19)
109                     ref(:,k)=[zeros(11,1);filter(b_filt,a_filt,...
110                     [ref(1:end-200,k);ones(189,1)*(rand(1,1)-0.5)])];
111                     % scale reference such that max value is
112                     % 0.9 of influence function
113                     ref(:,k)=0.9*ref(:,k)/max(abs(ref(:,k)));
114                     if n==0 % If n=1, apply inverse transfer function
115                         % Reference is added at the end to make
116                         % the voltage that is applied after the
117                         % last measurement existent
118                         yd_ref(:,k)=[ref(2:end,k);ref(end,k)];
119                     else
120                         % Reference is added at the end to make
121                         % the voltage that is applied after the
122                         % last measurement existent
123                         yd_ref(:,k)=lsim(Hinv(k),[ref(2:end,k)...
124                         ;ref(end,k)],t);
125                     end
126                     if N_op>1 % Simulate GPI if operators is
127                         % larger than 1, otherwise
128                         % just use the static gain
129                         v(:,k)=simPI_asymi(yd_ref(:,k),...
130                         invr(k,1:N_op),invp(k,1:N_op),...
131                         offset(k,1:4),x0);
132                     else
133                         v(:,k)=invp(k,1)*yd_ref(:,k);
134                     end
135
136                 else
137     %% Calculate input for MIMO
138                     rng(rngseed)
139                     refz=zeros(length(t),sum(2:N_zer+1));
140                     % sum of all influence functions, used to
141                     % determine range is which a reference is given
142                     Q=1./sum(abs(inffun));
143                     % Only give a reference to the second and
144                     % third order zernikes and
145                     % the spherical abberation
146                     Q=Q.*[ones(1,9),0,0,1,zeros(1,sum(2:N_zer+1)-12)];
147                     for k2=[1:9,12];
148                         refz(:,k2)=2.5*(-0.5+rand(length(t),1));
149                         refz(:,k2)=[zeros(11,1);filter(b_filt,...
150                         a_filt,[refz(1:end-200,k2);ones(189,1)...
151                         *(rand(1,1)-0.5)])];
```

```
152                         refz(:,k2)=12e-2*refz(:,k2)/...
153                         max(abs(refz(:,k2)))/Q(k2);
154                     end
155                     ref_y=zeros(size(v));
156                     ref_y(1,2:N_act)=LocReg(inffun(2:N_act,:)'...
157                     ,refz(1,:)',Q,1e-9*ones(1,19),0.8);
158                     for i3=2:length(refz(:,1))
159                         ref_y(i3,2:N_act)=LocReg(inffun(2:N_act,:)'...
160                         ,refz(i3,:)',Q,1e-9*ones(1,19),0.8);
161                     end
162
163 %%  Uncomment to plot predicted reference
164 %   tracking performance with given cost function
165 %                       for i3=1:length(ref_y(:,1))
166 %                           y(i3,1:sum(2:N_zer+1))=...
167 %                           inffun'*ref_y(i3,:)';
168 %                       end
169 %
170 %                       for i3=1:N_act
171 %                           figure(1)
172 %                           subplot(5,4,i3)
173 %                           plot(refz(:,i3))
174 %                           hold on
175 %                           plot(y(:,i3))
176 %                           for k4=1:18
177 %                               figure(1+k4)
178 %                               subplot(5,4,i3)
179 %                               plot(ref_y(:,k4)*inffun(k4,i3))
180 %                           end
181 %                       end
182
183 %% Calculate input voltage
184                     for k2=idedactuators % run for all actuators
185                         % If n=1, apply inverse transfer function
186                         if n==0
187                             % Reference is added at the end to make
188                             % the voltage that is applied after the
189                             % last measurement existent
190                             yd_ref(:,k2)=[ref_y(2:end,k2);ref_y(end,k2)];
191                         else
192                             % Reference is added at the end to make
193                             % the voltage that is applied after the
194                             % last measurement existent
195                             yd_ref(:,k2)=lsim(Hinv(k2),...
196                                 [ref_y(2:end,k2);ref_y(end,k2)],t);
197                         end
198                         if N_op>1 % Simulate GPI if operators is
199                             % larger than 1, otherwise just
200                             % use the static gain
201                             v(:,k2)=simPI_asymi(yd_ref(:,k2),...
202                             invr(k2,1:N_op),invp(k2,1:N_op)...
203                             ,offset (k2,1:4),x0);
204                         else
205                             v(:,k2)=invp(k2,1)*yd_ref(:,k2);
206                         end
207                     end
208                 end
```

```
209
210                 z=zeros(length(t),sum(2:N_zer+1));
211
212                 % Stop if maxvoltage is exceeded or NaNs are present
213                 if max(abs(v))>1||isnan(ref_y)==1
214                     error('max voltage exceeded or NaN present')
215                 end
216
217 %% Run measurement
218                 tic % start timer
219                 for i=1:length(t);
220                     % Measure zernikes
221                     [z(i,:)]=measzer_mirror(v(i,:));
222                     while toc<dt*i % wait till the time of
223                         % the next sample is reached
224                     end
225                 end
226                 elapsedtime=toc % show elapsed time.
227                 % If this is larger than t_end,
228                 % the sample time is too short to handle
229
230                 disp(strcat(num2str(N_dis_spots),' have disappeared'))
231 %% Save data
232                 if n==0
233                     if k==N_act+1
234                         save(strcat('comp',datstr,'act',...
235                         num2str(k),'N_op',num2str(N_op),'n',...
236                         num2str(n),'rngseed',num2str(rngseed))...
237                         ,'t','v','z','elapsedtime','ref_y'...
238                         ,'refz','N_op','order_act','n','inffun'...
239                         ,'p','r','invp','invr','offset','wn')
240                     else
241                         save(strcat('comp',datstr,'act',...
242                         num2str(k),'N_op',num2str(N_op),'n',...
243                         num2str(n),'rngseed',num2str(rngseed))...
244                         ,'t','v','z','elapsedtime','ref','N_op'...
245                         ,'order_act','n','inffun','p','r'...
246                         ,'invp','invr','offset','wn','k')
247                     end
248                 else
249                     if k==N_act+1
250                         save(strcat('comp',datstr,'act',...
251                         num2str(k),'N_op',num2str(N_op),'n',...
252                         num2str(n),'rngseed',num2str(rngseed))...
253                         ,'t','v','z','elapsedtime','ref_y'...
254                         ,'refz','N_op','order_act','n','inffun'...
255                         ,'p','r','invp','invr','offset','wn'...
256                         ,'Hest','Hinv')
257                     else
258                         save(strcat('comp',datstr,'act',...
259                         num2str(k),'N_op',num2str(N_op),'n',...
260                         num2str(n),'rngseed',num2str(rngseed))...
261                         ,'t','v','z','elapsedtime','ref','N_op'...
262                         ,'order_act','n','inffun','p','r'...
263                         ,'invp','invr','offset','wn'...
264                         ,'Hest','Hinv','k')
265                     end
```

```
266                 end
267
268 %% Put zero voltage on mirror
269                 [~,~]=measzer_mirror(0*v(i,:),N_dis_spots);
270             end
271         end
272     end
273 end
274 %% Set zero voltage at actuator and close connection
275 % please change when using another device
276 edac40('write',1,1.8*zeros(N_act,1));
277 edac40('close');
```

Appendix D

Glossary

List of Acronyms

AO	Adaptive Optics
FB	Feedback
FF	Feedforward
GPI	Generalized Prandtl-Ishlinskii
MIMO	Multiple-Input and Multiple-Output
PDM	Piezoelectric Deformable Mirror
PEA	Piezoelectric Actuator
PI	Prandtl-Ishlinskii
RMS	Root Mean Square
SH	Shack-Hartmann
SISO	Single-Input and Single-Output
VAF	Variance Accounted For

List of Symbols

δ	Displacement
ϵ	Permittivity

η	Added poles to make transfer function proper
$\hat{p}$	Weights of the inverse model
$\hat{r}$	Thresholds of the inverse model
Φ	Matrix with regressors for multiple time instances
ϕ	Vector with regressors
Ψ	Matrix with parameters
ρ	Radius
ρ_{sa}	Radius of subaperture
Σ	Singular values of SVD
σ	Stress
$\tau(M)$	Vector with random numbers between -1 and 1 of length M
Θ	Vector containing the model parameters
θ	Angle
Ξ	Vector with noise
ξ	Noise
ζ	Influence function
A	Spot number
a	Denominator coefficients of a transfer function
B	Matrix relating the displacement of the centroids of the spots to the Zernike modes
b	Numerator coefficients of a transfer function
D	Dielectric displacement
d	Piezoelectric constant
E	Electric field strength
e_{NRMS}	Normalized root mean square error used as performance measure
F	Focal length
f	Function
F_r	Operator in the PI model
f_{butter}	Butterworth filter
G	Dynamic model
g	Polynomial in GPI model
H	Hysteresis model
h	Hysteresis percentage
I	Intensity
I_x	X-coordinate of centroid
I_y	Y-coordinate of centroid
$I_{min_{centroid}}$	Minimum intensity value to be recognized as contributing to the location of the centroid of the spot
$I_{min_{spot}}$	Minimum intensity value to be recognized as a spot
J	Cost function
K	Gain

l	Maximum value of the input
M	Amount of samples
N	Amount of operators in (G)PI model
n	Order of dynamic model
P	Poles
p	Weight
Q	Output weight
q	Discrete transfer function variable
R	Control weight
r	Threshold
R_u^v	Radial term in Zernike polynomial
$r_{centroid_{min}}$	Minimum distance between detected spots in either x- or y- direction
$r_{centroid}$	Maximum distance in either x- or y- direction from the peak intensity in order to be taken into account for the calculation of the centroid
ref	Reference
S	Matrix with measured wavefront slopes
s	Compliance
t	Time instant
U	Left-singular vectors of SVD
u	Input
$u_{j_{max}}$	Maximum voltage during identification
$u_{j_{min}}$	Minimum voltage during identification
V	Right-singular vectors of SVD
v	Voltage
w	Value of an operator of the PI model
x	X-coordinate
$x_{A_{k_{ref}}}$	X-coordinate of reference spot
Y	Vector with outputs
y	Either the output of the system or y coordinate, depending on the context
y_a	Output on the ascending interval $\dot{u} > 0$
y_d	Output on the descending interval $\dot{u} < 0$
$y_{A_{k_{ref}}}$	Y-coordinate of reference spot
y_{meas}	Measured output
y_{sim}	Simulated output
z	Vector with Zernikes
Z_s	Zeros inside the unit circle
Z_u^v	Zernike mode
z_{meas}	Measured Zernike
Z_{ns}	Zeros outside the unit circle
Z_{xy}	Matrix with derivates of Zernikes to x and y