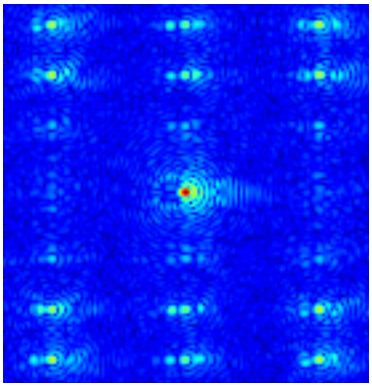
Apodizing Phase Plate Coronagraph Combined With Computer Generated Hologram Wavefront Sensor



THESIS

submitted in partial fulfillment of the
requirements for the degree of

BACHELOR OF SCIENCE
in
PHYSICS AND ASTRONOMY

Author :	Alexander G. M. Pietrow
Student ID :	1054678
Supervisor :	Christoph Keller
2nd corrector :	Martin van Exter

Leiden, The Netherlands, July 16, 2014

Apodizing Phase Plate Coronagraph Combined With Computer Generated Hologram Wavefront Sensor

Alexander G. M. Pietrow

Leiden Observatory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

July 16, 2014

Abstract

A high contrast of the order of 10^{-11} makes it challenging to directly image Earth-like exoplanets at visible wavelengths. Limited by diffraction and non-common path aberrations, current adaptive optics systems cannot reach this contrast. Exploring the combination of an apodizing phase plate and a holographic modal wavefront sensor, we show with simulations that up to six aberration modes can be suppressed in a closed loop, correcting aberrations of up to 2 rad. In the lab the same has been achieved for a single mode with a 0.5 rad amplitude. The highest achieved contrast in simulation, obtained with use of electric field conjugation, is on the order of 10^{-14}.

Contents

Chapter 1

Introduction

Over twenty two years ago the first exoplanet[1] was discovered, turning a popular science fiction topic into a new research field within astronomy. Throughout the years, this has led to the invention and improvement of many new techniques for finding exoplanets, like RV, transits and microlensing, leading to many discoveries eversince. (Seager & Deming, 2010) One of these exoplanet finding methods is direct imaging, where we try to observe these planets directly. This however turns out to be quite a challenge due to the significant contrast differences between the planets and their host star. So far, all[2] planets that have been directly imaged were observed in the near infrared, where not the reflective flux was measured, but the thermal emission of young, Jupiter-mass exoplanets orbiting relatively far from their host star. This explains the fact that less than three percent[3] of all the discovered exoplanets have ever been imaged directly. The main issue, however, is that Earth-like planets that are close enough to their host star, have a relatively small angular separation of 10 λ/D. This means that such planets will be "hidden" under the diffraction rings of their host star, adding to the difficulty of being detected.

The problem of diffraction can be addressed by a promising coronagraph method called phase apodization. Here so called 'apodizing phase plates' (APPs) are generated in such a way that a flat wavefront passing trough an APP gets destructively interfered on one side, creating a dark hole in

[1] See Wolszczan & Frail (1992)
[2] Except Fomalhaut as seen in Kalas et al. (2008)
[3] See exoplanet.eu (2014)

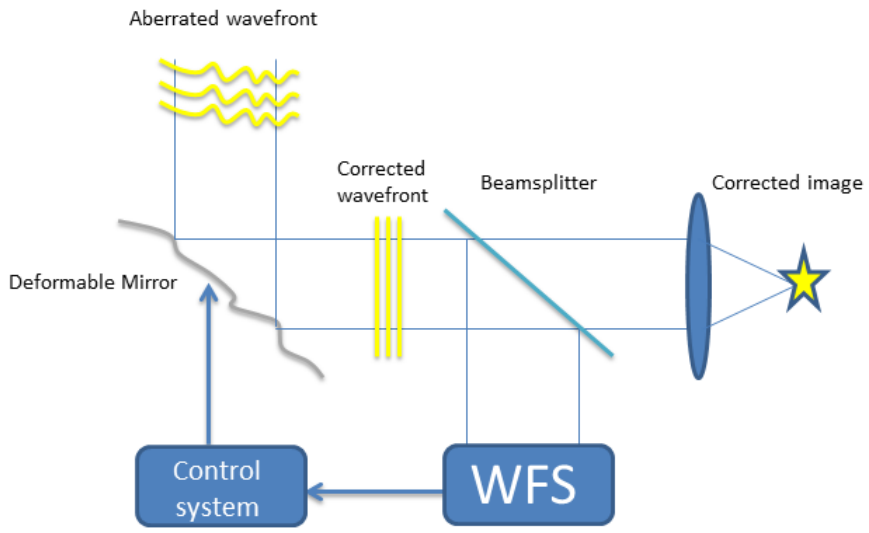


Figure 1.1: *Basic AO system setup, with a beam splitter which sends light to both the wave front sensor and the camera.*

the diffraction rings of the imaged point spread function (PSF), as can be seen in Codona et al. (2006)

To image a planet in the V-band, we require a strong reflected flux, which means that we need it to be relatively close to its host. The contrast difference, which for Earth like planets in the V-band is known to be 10^{-11}. (Keller, 2014a) This requires the aid of coronagraphy and adaptive optics (AO). With the former we mask the parent star, the diffraction caused by it, and with the latter we correct wavefronts, which were perturbed by the atmosphere. In this way we decrease the required contrast difference in the captured images, and increase our chances of obtaining a valid detection.

Currently there are many ways of making a closed-loop AO system, with wavefront sensors (WFS) that are incorporated into these loops by means of a beamsplitter. Figure 1.2 shows the standard setup of a typical AO system, where the WFS and the science camera are separate. Because the WFS cannot detect any aberrations that are introduced after the beamsplitter in the optical path to the science camera, these aberrations will not be corrected. These aberrations are called Non-Common Path Aberrations (NCPA). If we compare all the aberrations in a closed loop AO system, the NCPA are the most significant and therefore the main limiting factor

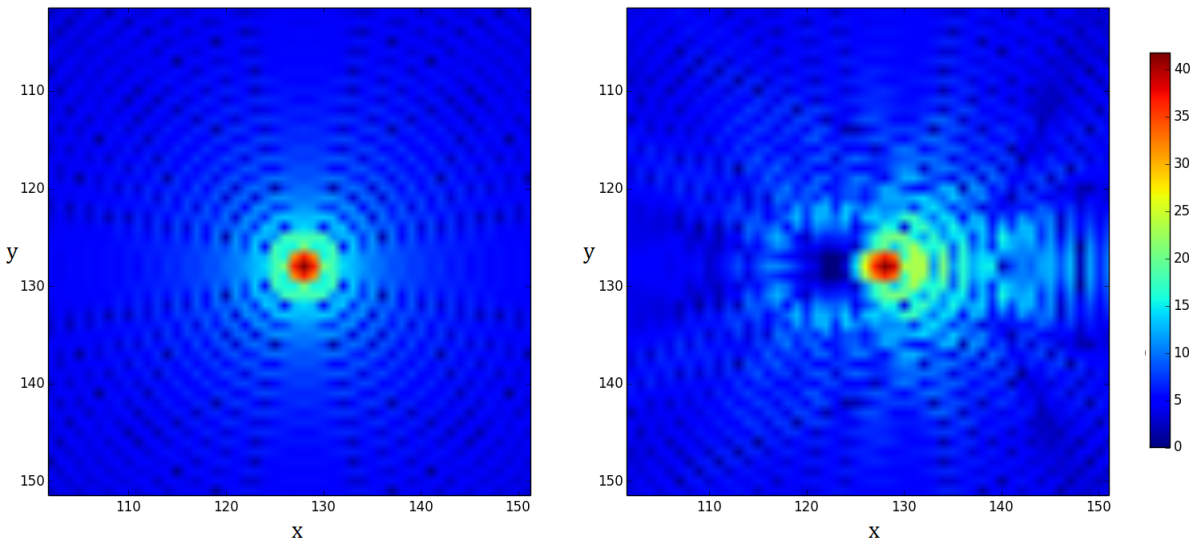


Figure 1.2: *a) A flat wavefront. b) The same wavefront but with an added APP that creates a dark hole on the left side. The counts in both images are raised to the 0.2th power to lower the dynamic range.*

in modern AO systems.[4] While methods of correcting these aberrations do exist[5], a way of circumventing them all together is much more desirable. One WFS which can do exactly that is called the holography based modal wavefront sensor (HMWFS), as described in Neil et al. (2000). This WFS works by imaging directly onto the science camera, removing the need for a beamsplitter and therefore getting rid of the NCPAs. These holograms respond to given aberration modes, which can be tailored to the hole which the APP creates. Hence, only a limited amount of modes is needed, as the main interest is on the hole and not the rest of the image.

This report describes the combination of the APP coronagraph and the HMWFS for low order aberration correction of a small dark area inside of the diffraction rings of stars, the achievable contrasts and merits of using this in an AO system.

[4]See Frazin (2014)
[5]See Sauvage et al. (2007)

Chapter 2

Theory

2.1 Apodizing Phase Plates

The challenge with imaging exoplanets that have a small enough angular separation from their host star, is that they get covered by the hosts diffraction rings. These rings, which can be many times brighter than the planet, can be reduced by with an AO system but has to be further suppressed by means of a coronagraph. There are several types of coronagraphs, like Lyot, vortex and phase mask coronagraphs. Each type having their own trade-offs in throughput and angular resolution for diffraction suppression. For this research we have chosen the Apodizing phase plate coronagraph because of its ability to create a dark hole of a chosen dimension and location. APPs are shaped similarly to a diffraction grating, optimised to cause small fractions of the main PSF to be diffracted into the surrounding halo. This plate also aberrates this light in such a way that it matches the amplitude and location of region it illuminates, only with a phase difference of π radians. This causes destructive interference and a darkening of this area.[1] However, due to symmetry this also means that the plate will have an inverted amplitude and identical phase on the location opposite to the aforementioned area, which will have the inverse effect and constructively interfere with the diffracted light. This means that one can not suppress more than half of the region around the star.[2]

[1] As explained in Method I from Codona et al. (2006)
[2] See Kenworthy et al. (2007)

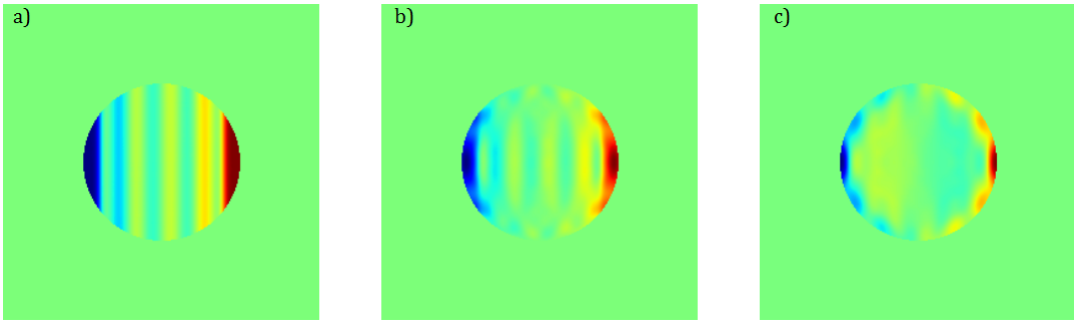


Figure 2.1: *Three APPs designed for creating small holes around stars. Figure 4.2 illustrates the effects of these plates.*

For this research three APPs, made by Christoph Keller, were used. These were intentionally chosen to work on a small area because of the nature of the CGH wavefront sensor, but also because of the trade-off between area and achievable depth.[3] In Fig. 2.1 we can see the three APPs which will be used throughout this research.

2.2 Modes

When talking about aberrated wavefronts and their correction, one of the first terms that comes to mind is the 'Zernike polynominals', these being a set of orthogonal modes describing aberrations. Combinations of these modes offer an approximation of a complex wavefront[4] and are therefore used to express a wavefront. There are, however, many other possible orthonormal bases of aberration modes that can be used. Especially if one does not wish to correct the entire image but only a small part, for example a small area created by an APP. In such a case it is better to look for modes which lead to the most intensity in this specific area. These modes can of course be described by a superposition of Zernike modes, but if sorted by their ability to affect the hole, we can avoid most light with a subset of the first few aberrations, rather than many Zernike or other modes. Every APP will have a unique set of modes equal to twice the amount of pixels in the hole. This is because these modes are related to the amplitude and phase of the electric field of every pixel that is being measured there.

[3]See (Keller, 2014b)
[4]See Noll (1976)

2.3 CGH generation

Holography based modal wavefront sensors (HMWS) are an interesting alternative to conventional wavefront sensors because, unlike most other WFSs, they do not suffer from non common path errors. This is achieved by imaging the holograms directly onto the science camera, ensuring that the entire path between the WFS and the science imaging camera is identical. The HMWFS works by assuming that every incoming wavefront can be decomposed into a set of orthogonal aberration modes (Zernike modes or other custom modes). While this can be a large number of modes, the majority of the perturbations is caused by the low-order modes. Each one of these modes can be separately described by a computer generated hologram (CGH). Those holograms then create a pair of spots on the science frame in a chosen location. Comparing the normalised intensity difference between a pair of those spots, we can find the amplitude of a given aberration (see below). The holograms, describing single aberration modes, can then be multiplexed, by simple addition, into an arbitrarily large collection. Given enough of these holograms, we can reconstruct a complex wavefront by measuring the intensity difference of the dots. Therefore the correction speed is mainly being limited by the speed of the detector.
The main problem with this method is that these holograms redistribute light from the main PSF, and therefore lower the S/N of the image, meaning that the more holograms we stack, the lower our image S/N will be. Two holograms are needed for every pixel that we wish to correct, and therefore correcting large areas with this method would not be recommended. However if we only wish to work on a small area of only a few pixels, the loss in peak intensity is limited to about 20%.

In this section we explore the theory behind creating a single CGH, following Booth (2003). Computer generated holograms work in the pupil plane of the telescope and function much like an ordinary diffraction grating. When light sent trough the CGH passes a convergent lens, it will be Fourier transformed and hence convolved with the PSF resulting in a set of point symmetrical dot pairs, as seen in Fig. 2.2, which are in fact copies of the main PSF. The top of these dots wil be called A^+ and the bottom one A^-. Such a hologram is designed in such a way that when it is illuminated by a wavefront containing a certain aberration M_k, it will give a high intensity at the A^+, while the aberration $-M_k$ leads to the opposite effect and raises the intensity of A^-.

The intensities of these two dots, can then be used to obtain a_k, the mag-

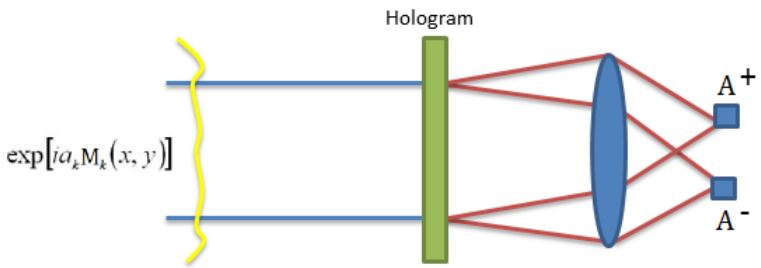


Figure 2.2: *Diagram illustrating the basic principles of the HMWFS. A wavefront containing a certain aberration M_k illuminates the CGH, which produces two dots. The normalised intensity difference between these two dots gives us the amplitude of the measured aberration.*

nitude of the incoming aberration mode. Our method of making CGHs is basically the same as the physical method for making off-axis holograms, as described in Leith & Upatnieks (1962). To make such holograms, we combine a tilted monochromatic plane wave, the reference beam (R_k) with the object beam (O_k). Figure 2.3 shows that these two waves, having the same flat wavefront as a source, take different paths along the setup. The reference beam remains unaltered, while the object wave is projected onto the hologram target. The reflected light off this object then meets with the unaltered reference beam and creates an interference pattern on the holographic film. When this film is illuminated by the same reference beam, it will be distorted into the object beam, thereby creating an image of the object that was stored in the hologram. To make a WFS we choose our reference wave not as a flat wavefront but as one perturbed by the aberration that we wish to measure. The object wave, a tilted plane wave which will not encounter an object along its path, will move trough the setup unaltered. Once combined, these waves will make a hologram that reacts to the given aberration.

The easiest way to show this is by assuming that we have a perturbed reference wave, $R_k = \exp\left(i\beta_k M_k\right)$ and a tilted object wave, $O_k = \exp\left(ic_p\left(x+y\right)\right)$. With the chosen mode M_k, the mode amplitude β_k and the positioning parameter c_p (see below). If we then take the absolute value of the sum of these two waves, we will obtain the amplitude transmittance function H_k, the CGH.

$$\begin{aligned} H_k &= |O_k + R_k|^2 \\ &= \cancel{|O_k|^2} + \cancel{|R_k|^2} + O_k R_k^* + O_k^* R_k \end{aligned}$$

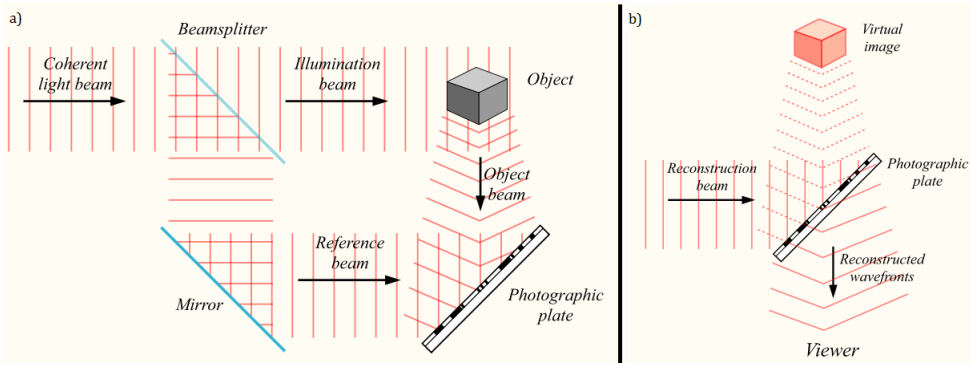


Figure 2.3: *(a) Diagram of a standard holography setup, creating a hologram of a small globe. Image credit Wikipedia (b) One can reconstruct the image of the object by illuminating the hologram with the reference wave. Image credit Wikipedia*

The two crossed off term are both equal unity and are removed because at this point we are not interested in the zeroth order of the hologram.

$$H_k = O_k R_k^* + O_k^* R_k \tag{2.1}$$

Since we are only interested in the phase part of this function, we can take the argument and obtain our CGH. This is the function that we will be using for the CGH generation.
After making our hologram, we need to understand the dot pairs that it will create. Their intensity can be described by the amplitude of the Fourier transform of the transmittance function.

$$I = \left|\mathcal{F}\left\{H_k \cdot R\right\}\right|^2, \tag{2.2}$$

where $R = \exp\left(i\alpha_k M_k\right)$, is some input wave containing the aberration, with amplitude α_k, that we are trying to measure. Expanding Eq. (2.2) will give us two terms.

$$\begin{aligned} I &= \left|\mathcal{F}\left\{O_k R_k^* R + O_K^* R_K R\right\}\right|^2, \\ &= \left|\mathcal{F}\left\{O_k R_k^* R\right\} + \mathcal{F}\left\{O_k^* R_k R\right\}\right|^2. \end{aligned}$$

Since both terms represent a convolution of the main PSF with an Airy function[5], we get two distinct dots at two different locations. Because these two dots are separated by a large enough distance, we can assume

[5]Fourier transform of O_k will give an Airy function rather than a delta function because of a finite aperture.

that they have no mutual interference and that we can therefore neglect the cross term, giving us the two intensity terms I^+ and I^-.

$$\begin{aligned} I^+ &= \iint_{A_1} |\delta_{k+} * \mathcal{F}\left\{\exp\left(i\left(\alpha_k + \beta_k\right) M_k\right)\right\}|^2 \, dA1 \\ I^- &= \iint_{A_2} |\delta_{k-} * \mathcal{F}\left\{\exp\left(i\left(\alpha_k - \beta_k\right) M_k\right)\right\}|^2 \, dA2 \end{aligned} \tag{2.3}$$

Here A_1 and A_2 denote the integration area in the image and $\delta_{k\pm}$ the Fourier transform of $O_{k\pm}$, which will result in a displaced Airy function as $O_{k\pm}$ is a tilted plane wave. This basically means that we have convolved our object wave with the reference wave, which results in a copy of the reference wave at two positions. In our derivation we have chosen the object wave to have one combined constant for both the x and y position, which results in a simpler airy function, $\delta_{k\pm}$. However, these can can be chosen differently, which allows the user to decide the location of the dots. Taking this into consideration, the object wave can be described as

$$O_k = \exp\left(i2\pi\left(f_{k_x}x + f_{k_y}y\right)\right), \tag{2.4}$$

where $f_{k_x} = \frac{x'_k}{\lambda f}$ is the spatial frequency, λ, the wavelength and f the focal length of the convergent lens and the position x_k, y_k. From this we can derive that $\delta_k = \delta\left(x' - x'_k\right)\delta\left(y' - y'_k\right)$.

These holograms can be used as a WFS. When used in combination with an APP they need to be weakened to also let trough the zeroth order, namely the original PSF, besides only showing the two first order dots. (See Fig. 2.4) This can be done by multiplying the amplitude of the phases of the holograms by a dampening factor. In this research this is chosen to be 0.6 in simulations and 0.4 in the lab.

We can see from Eq. (2.3) that β_k, the modulation amplitude of the aberration, is reversed in the two I's. This means that if we would input a reference wave $R = \exp\left(i\alpha_k M_k\right)$ with $\alpha_k = \beta_k$, then I^+ would dominate, while if we have $\alpha_k = -\beta_k$, then I^- would dominate. Everything in between is described by a linear relationship given by,

$$\alpha_k = \beta_k \frac{I^+ - I^-}{I^+ + I^-}, \tag{2.5}$$

where α_k is the amplitude of M_k. In Fig. 2.5 we can see one of these holograms applied to a flat wavefront. Besides the PSF in the center, we see the

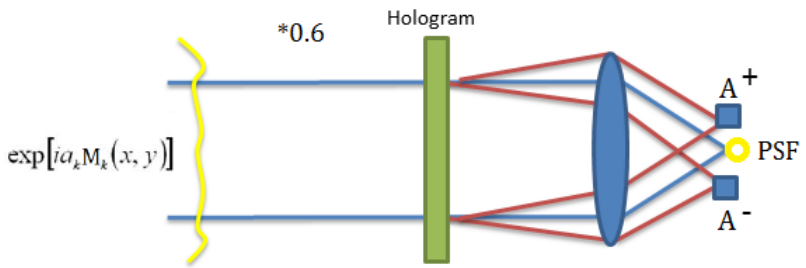


Figure 2.4: *Once the hologram is weakened, it will also show the zeroth order PSF.*

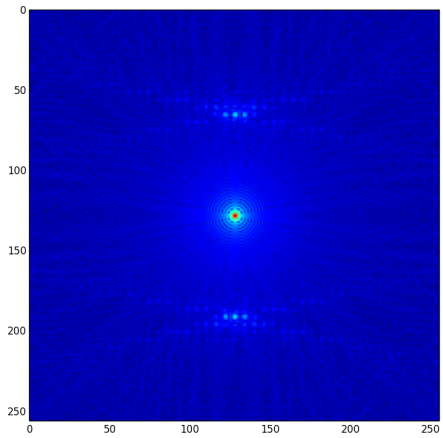


Figure 2.5: *An example of a single CGH applied to a flat wavefront. The top and bottom dot intensities are given by I^+ and I^- respectively. The image is scaled to the power of 0.2 to compress the dynamic range.*

two hologram dots, which are copies of the central PSF. The only reason for us seeing the main PSF is because of the dampening factor mentioned above. Besides these three dots we can also see several higher order copies, their intensity depends on β_k. This value should be chosen in such a way as to ensure that these higher orders do not dominate. In Booth (2003), this value is chosen to be 1.5, but here the lower values of 0.5 and 0.1 were used.

2.4 Combined APP and CGH

Knowing that our coronagraph provides us with a dark hole at the desired location and with a WFS that can detect the dominating aberrations,

we can combine these two methods into a closed-loop system: a system that will keep the hole, created by the APP, dark during observations. As mentioned before, the HMWFS is imaged in the science frame and therefore we have no NCPA, meaning that we are able to find all the aberration mode amplitudes in one iteration. Assuming that no other processes influence this method, this would give us a loop that can quickly reduce any aberrations in a chosen area.

However, before this can be done, we need to calibrate the dots to known aberrations and fit a linear relation to the response curve. This curve often diverges from the linear fit for large aberration amplitudes, but can be described well around zero. This means that if we set the feedback gain to a sufficiently low value and the derivative of the curve does not change signs, the loop should still converge. During this research project the holograms have been calibrated by iterating trough 21 amplitudes for their corresponding mode, ranging from -1 to 1 with steps of 0.1. Beforehand every mode is normalised to have a standard deviation of 1 rad to ensure that the scaling is the same for all of the modes. The line is then fitted to the middle 5 points, taking into account only the approximately linear regime near zero. After we obtain the coefficients for the linear fit, we can invert the equation to get the aberration factor as a function of the difference between the two dots, and obtain

$$\alpha_k = \frac{\Delta I - b}{a}, \tag{2.6}$$

with a and b being the two coefficients of the fitted line S= aF+b, where S is the normalised dot signal difference and F the aberration factor. (See Fig. 2.6.)

Now that we can obtain the amplitude of aberrations by merely measuring the normalised intensity difference between the two dots, we can use this as an AO system. We can then create a closed-loop system that will correct the measured aberrations after every iteration.

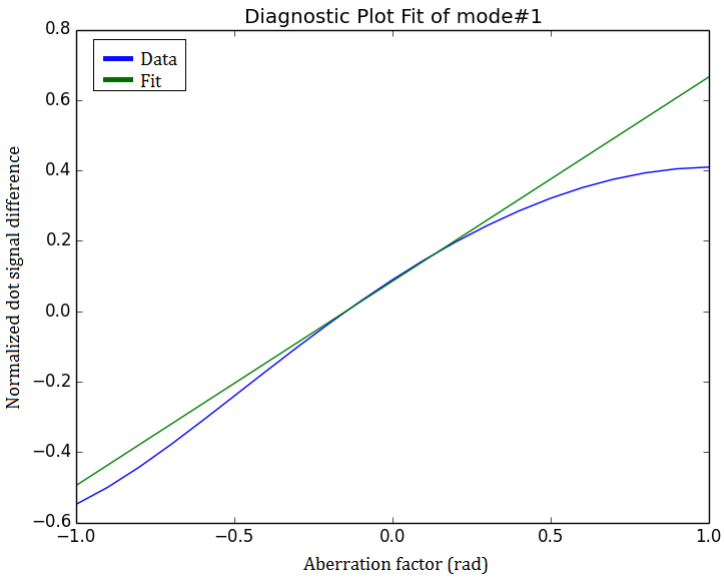


Figure 2.6: *Normalised dot intensity relation of an aberration mode with a linear fit made to the 5 central points. Note that the fit gets better once it gets closer to zero.*

Chapter 3

Setup

To test the simulated data it is desirable to check the theory with a lab setup. This can be done in many ways, but the method chosen here makes use of a reflective spatial light modulator (SLM) and a CMOS imaging camera. The SLM, model BNS P512, has a 512x512 pixel surface, with each of these pixels 15x15μm in size, being capable of creating a 2π rad phase change. The camera, a Basler piA640-210gm, has 648x488 pixels and a dynamic range of 12 bits. The light source used for this setup is a fibre-coupled laser diode, Qphotonics QFLD-660-2S, with a wavelength of 656 nm that is collimated in such a way that it illuminates an area of 245x245 pixels on the SLM. In figure 3.1 one can see that the laser light is sent trough a diaphragm in the pupil plane to limit the beam. Next it is scaled with a two-lens system to the correct size just before it hits the SLM. There is a linear polarizer just before and after the SLM, because the SLM can only properly manipulate one polarization. After that the beam is focused onto the imaging camera, and an image can be made, which is done by merging multiple exposures with different integration times to increase the dynamic range.

For obtaining a flat wavefront as the base of our system we use the 'Fast and Furious algorithm' as seen in Korkiakoski et al. (2014), which calibrates the SLM and returns a zero setting that corresponds to a flat wavefront.

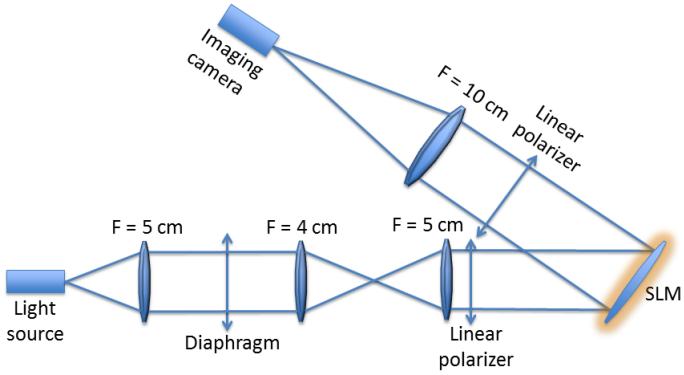


Figure 3.1: *A diagram illustrating the lab setup. All the lenses are standard 1 inch doublets. The beam diameter is 3.7 mm at the SLM. Image from Korkiakoski et al. (2014) with authors permission.*

Chapter 4

Results

During this entire project, three APPs have been used for the generation of a hole in the diffraction rings of an unperturbed PSF. The holes are located at about 3-4 λ/D from the central PSF peak and vary in size and shape. They are optimised to keep a certain region dark, which shall be expressed in pixels of a Nyquist sampled PSF. In the simulation this is a 1:1 ratio, in the lab this is different due to the reimaging onto the camera's detector chip.

As explained in section 2.2, we have twice as many aberration modes than the amount of pixels per hole in every APP, due to the fact that we have two degrees of freedom for every pixel. These modes are all tailored to their respective APP and are illustrated in Fig. 4.1. In this chapter we will cover the results obtained from simulations and the results obtained in the lab. The code used for the simulations can be found in Appendix A.

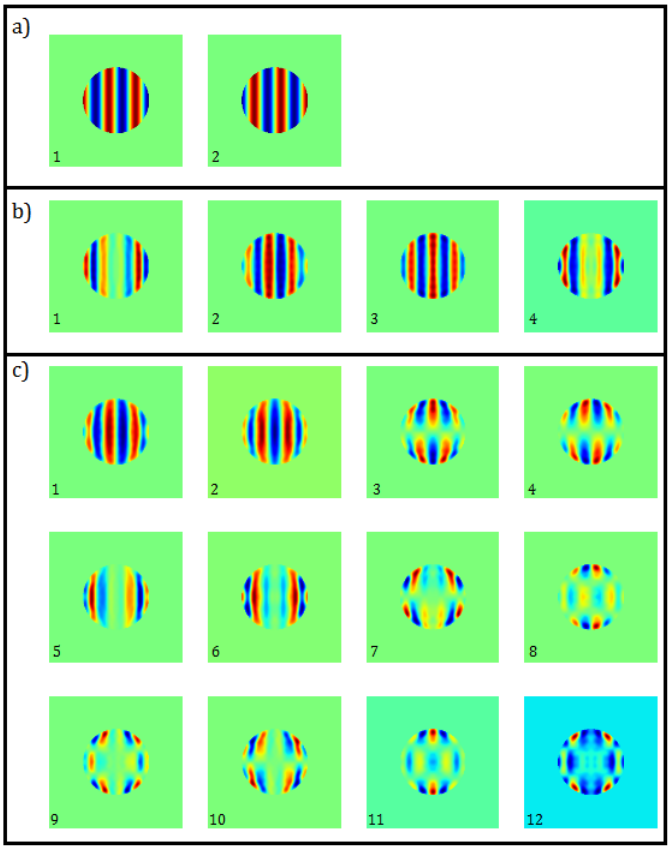


Figure 4.1: *A collection of all the modes used by the three APPs. Section a corresponding to the first, b to the second and c to the third APP. The modes are numbered from left to right and will be referred to by the letter and number.*

4.1 Simulations

Figure 4.2 shows the effect of three different APPs working on a perfectly flat, Nyquist sampled PSF. The size of the holes is 1x1 for the first, 2x1 for the second and 2x6 pixels for the third APP. After adding the associated CGHs to these PSFs, as illustrated in Fig. 4.3, we will see a difference in the depth of the holes, clearly visible to the naked eye. Something which can be quantified by taking a cut through the focal plane images. These cuts, taken along the y=125 line will be used to measure the contrast between the PSF peak and the depth of the hole.

From Fig. 4.4 we can see that all the holes are several magnitudes deeper than the surrounding area. Going down by almost sixteen orders of magnitude, these holes are basically void of any signal save for some numerical noise. However, once we add the CGH's to these images we observe a dramatic change in the depth of the holes, seeing them being filled up significantly. This effect, demonstrated in Fig. 4.5, is caused by the nature of Fourier transforms, which leave high frequency terms when taken over a finite area. These high frequencies mix and create lower beat frequencies that are invisible to the HMWFS, and thus can not be corrected by the loop. (Give'On et al., 2006) One other limiting effect is the crosstalk between the holograms, where our CGH's are affected by aberrations other than the one they are calibrated for. This effect increases with the amount of holograms in a system. Costing over ten orders of magnitude, this effect is quite significant and should be corrected, but because of a lack of time, this is only explored superficially in section 5.

After calibrating the CGH, as explained in section 2.3, we can put in a random combination of corresponding aberrations given in Fig. 4.1 into the setup and run the AO loop to remove the aberration. In Fig. 4.6 we can see how one aberration can be corrected in only a few iterations, and how the corrected image gets back to an almost aberration free state. The rate at which the hole gets darker depends on how aggressively one decides to correct for the aberrations. In part b of Fig. 4.6 we see this very clearly as a function of the parameter c, which is used as a feedback gain for the measured corrections in the AO loop. A low c ensures a steady but slow convergence rate while higher values tend to oscillate and diverge quickly. For a single hologram we can even set c to unity, as there is no crosstalk that could affect the measurements. However, for any other system this is not recommended. In the images below this is demonstrated with all three APPs and various values of c. The standard value used in this report

is c=0.2, while it is generally slow, it proves to be stable in most situations.

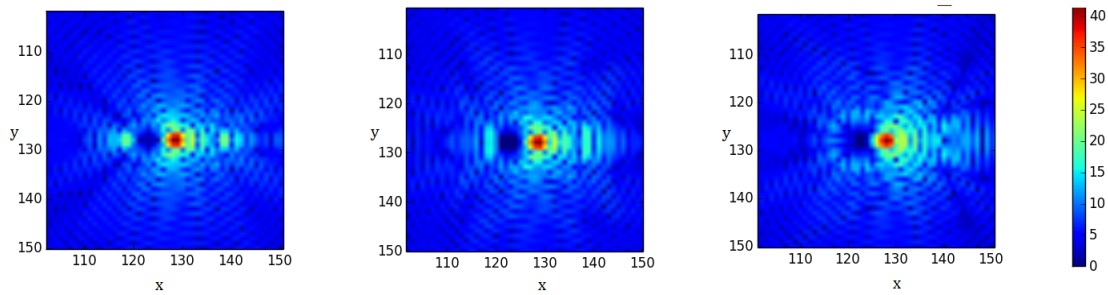


Figure 4.2: *Three PSFs formed by their respective APPs. Different APPs cause different sized and shaped holes. The images are enlarged in to show the PSF structure and the counts scaled to the power of 0.2 to compress the dynamic range.*

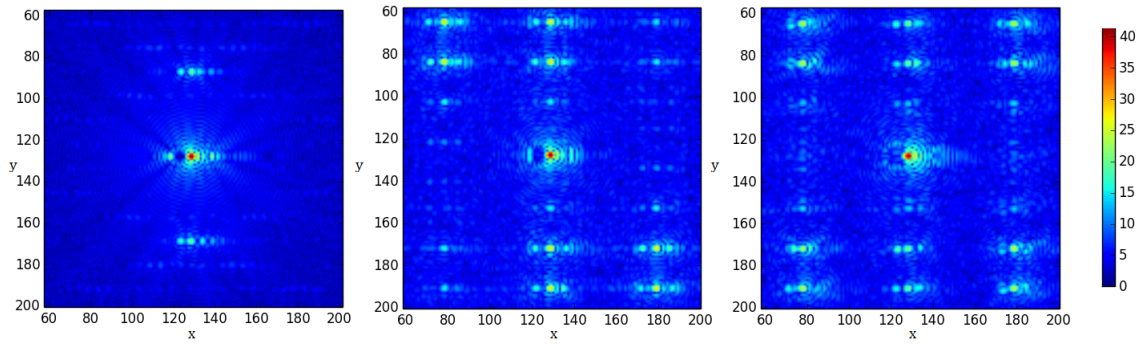


Figure 4.3: *The same APPs as in Fig. 4.2, but now with added holograms to sense the aberration modes. The holes can again be found at the x-value of 120. The counts are scaled to the power of 0.2 to compress the dynamic range.*

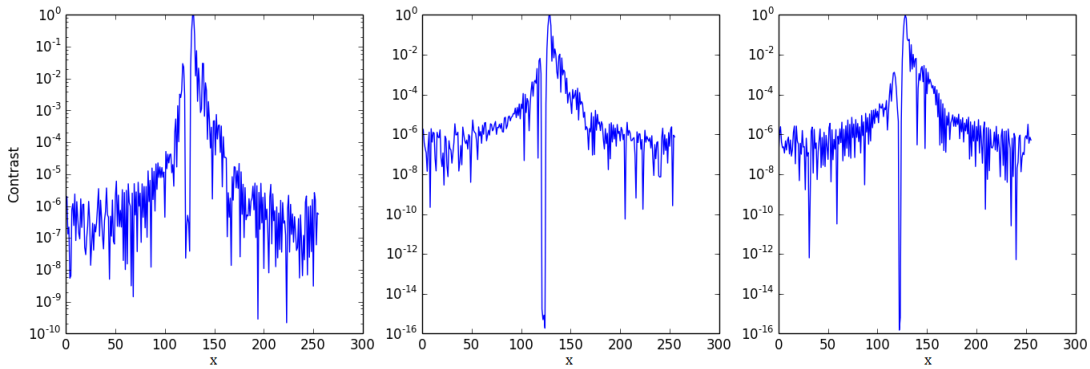


Figure 4.4: *Contrast curves of the three APP's from Fig. 4.2. The hole is located around an x-value of 120 and in all cases can clearly be discerned by eye.*

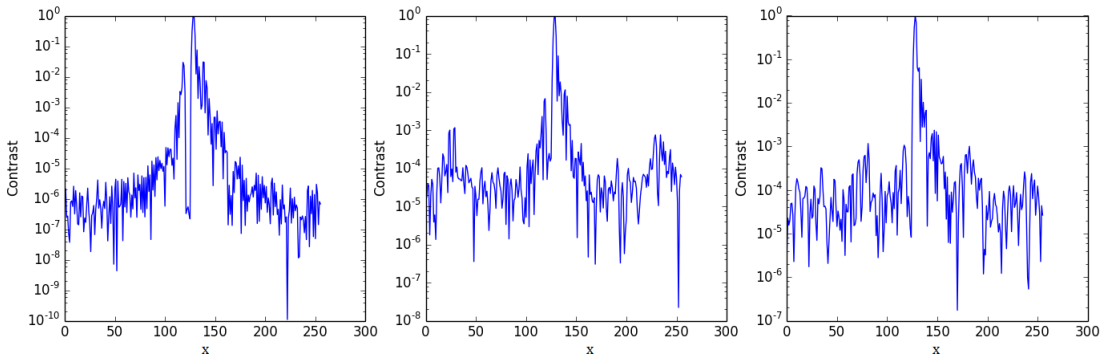


Figure 4.5: *A cut of Fig. 4.3, illustrating the filling effects caused by the holograms. The hole is located around an x-value of 120.*

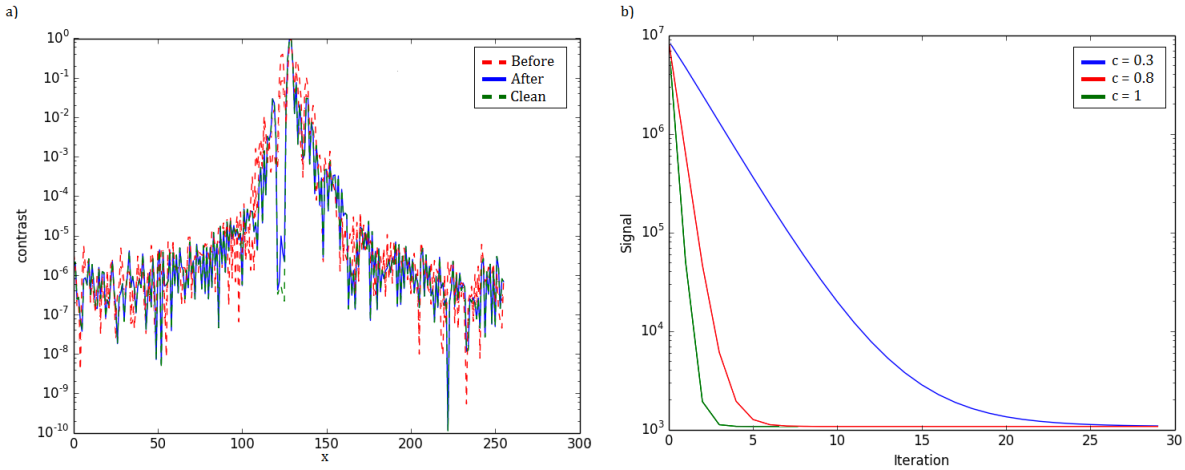


Figure 4.6: *a) The system with an added aberration which has the amplitude of mode a1, corresponding to a phase difference of about 1 radian. After 20 iterations it is almost completely corrected, gaining over six orders of magnitude. b) The rate of correction depends strongly on c.*

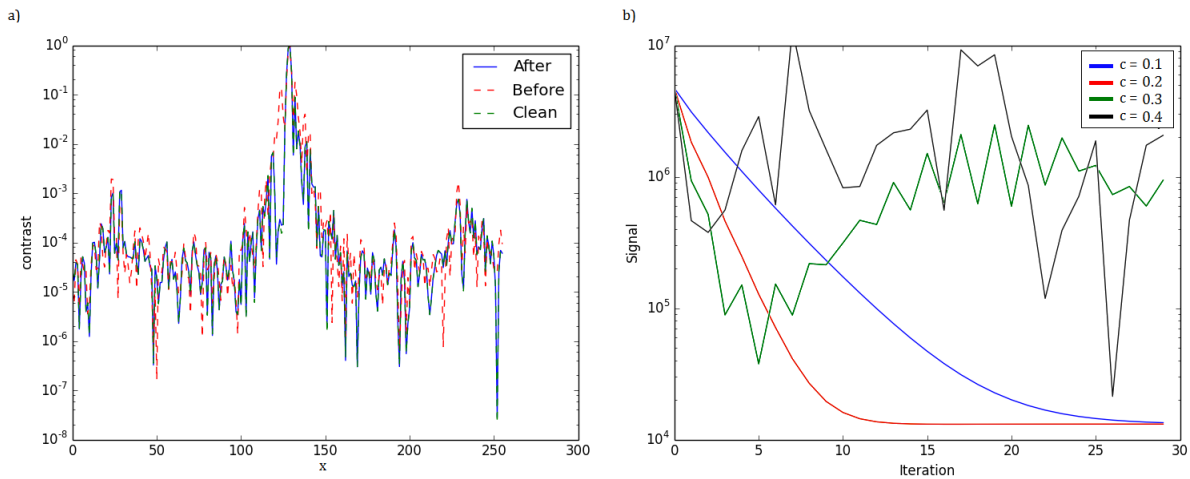


Figure 4.7: *a) The system with an added aberration which is the superposition of all 4 modes with random coefficients, with a phase difference of 2 radians. After 25 iterations it is almost completely corrected, gaining over three orders of magnitude. b) Here we clearly see that a too high value for c will cause the loop to become unstable due to crosstalk effects.*

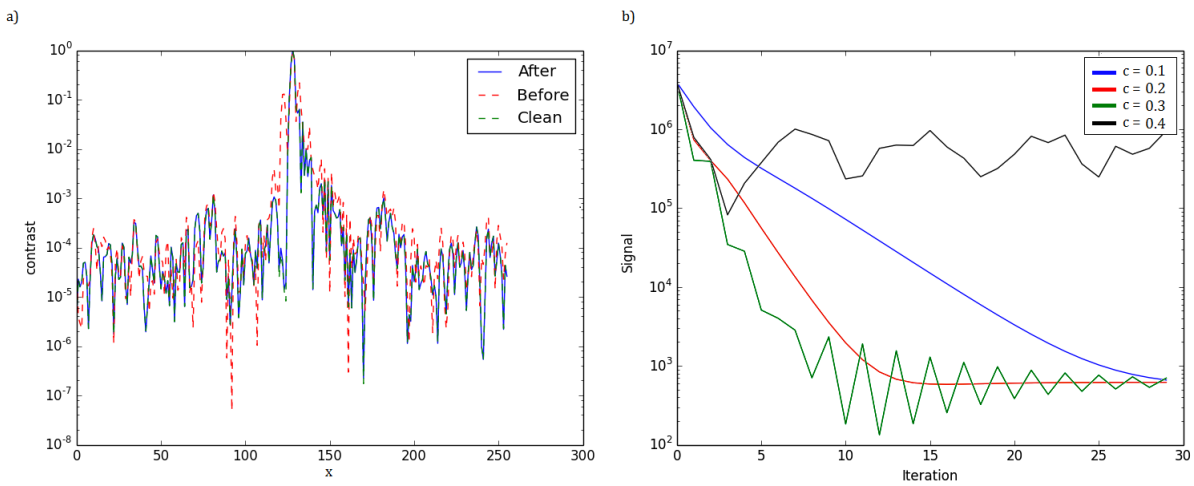


Figure 4.8: *a) The system with an added aberration which is the superposition of the first 6 modes with random coefficients, with a phase difference of 2 radians. After 30 iterations it is corrected so well that the blue and green line are nearly identical, giving us a gain of over three orders of magnitude. b) Again we see the crosstalk making the loop unstable once c is too large.*

4.2 Lab

If we plug the same APPs into the lab setup as we have used in the simulation part, we will be able to compare the results and see how well nature follows our simple simulations. Adding the CGHs, the holes again fill up. Again we could address this problem by use of electric field conjugation, as explained in Give'On et al. (2007), but due to time issues this has only been done superficially. (see section 5)

In Fig. 4.9 we can see lab measurements of the PSFs altered by the same three APPs as discussed in section 4.1. Figure 4.10 illustrates how these change once we add the corresponding CGHs. In Fig. 4.11 and Fig. 4.12 we can see the corresponding contrast curves and how they get filled up bt the holograms. Note that the filling is significantly less than in the simulated counterparts, the reason for this is unknown.
When looking at Fig. 4.13, we can see that the calibration curves are a lot more jumpy and generally not comparable to the simulated curve in Fig. 2.6. Both of these curves are of systems with multiple holograms, while the curve in Fig. 4.14 is comparable to the simulated result. This suggests that these curves are dominated by crosstalk and therefore no stable relation is achievable. Because of this, we have not been able to get a closed-loop running with more than one CGH.

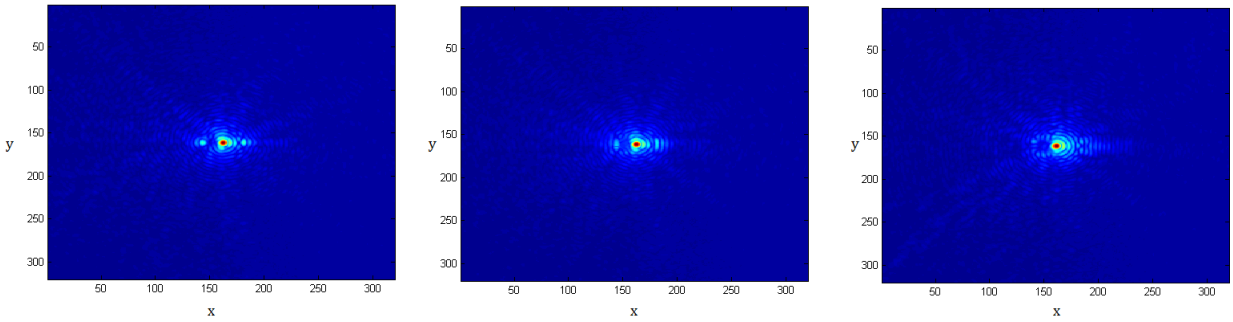


Figure 4.9: *The three APP's with their CGH. Image values are raised to the 0.2th power to compress the dynamic range.*

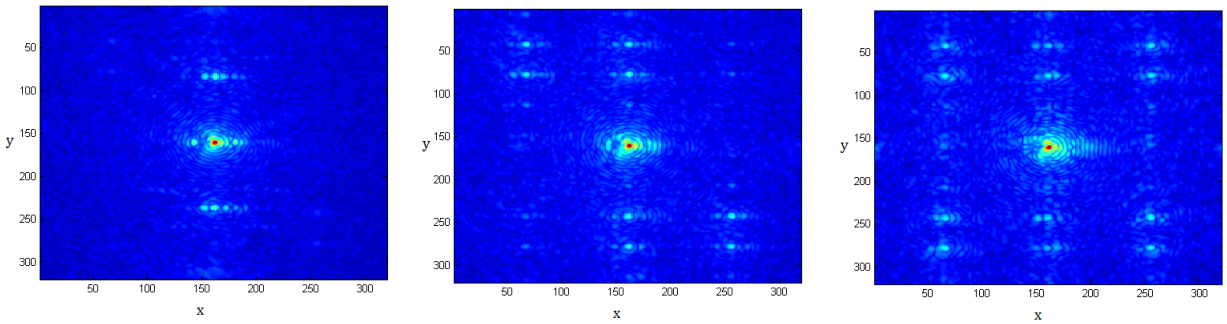


Figure 4.10: *The three APP's with their CGH. Image values are raised to the 0.2th power to compress the dynamic range.*

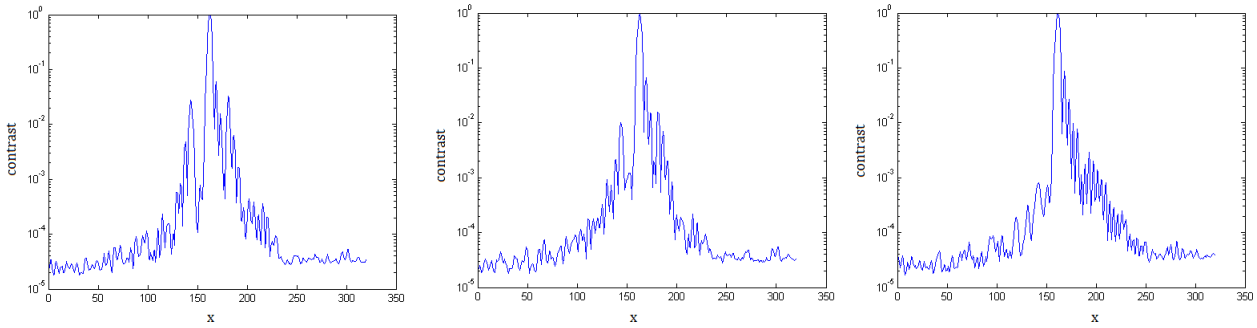


Figure 4.11: *Contrast curves of Fig. 4.9 cut at y=125.*

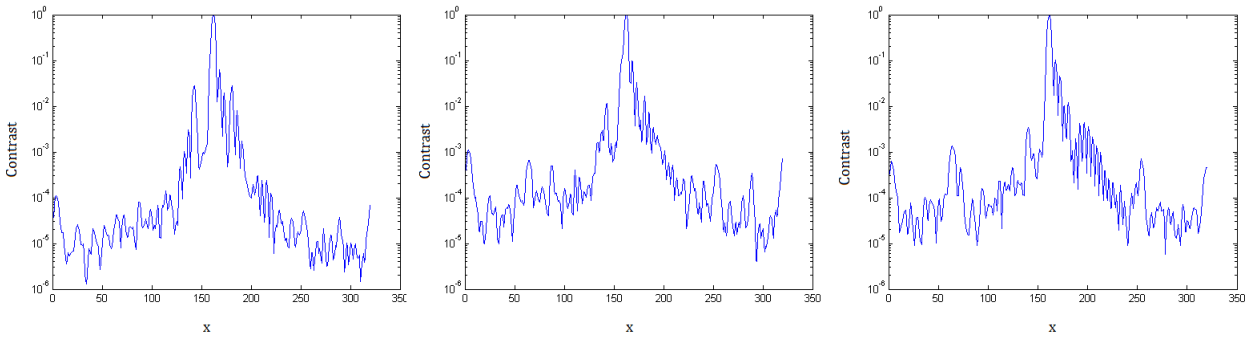


Figure 4.12: *Contrast curves of Fig. 4.10 cut at y=125.*

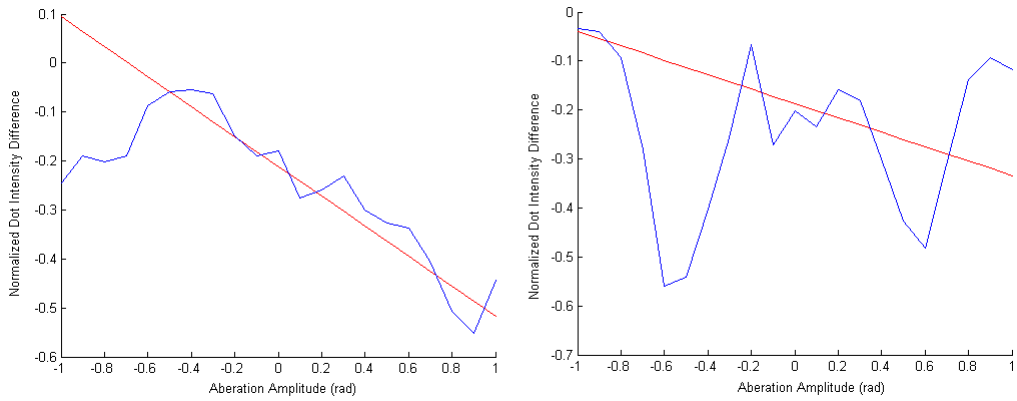


Figure 4.13: *Calibration curves for two modes of the third APP.*

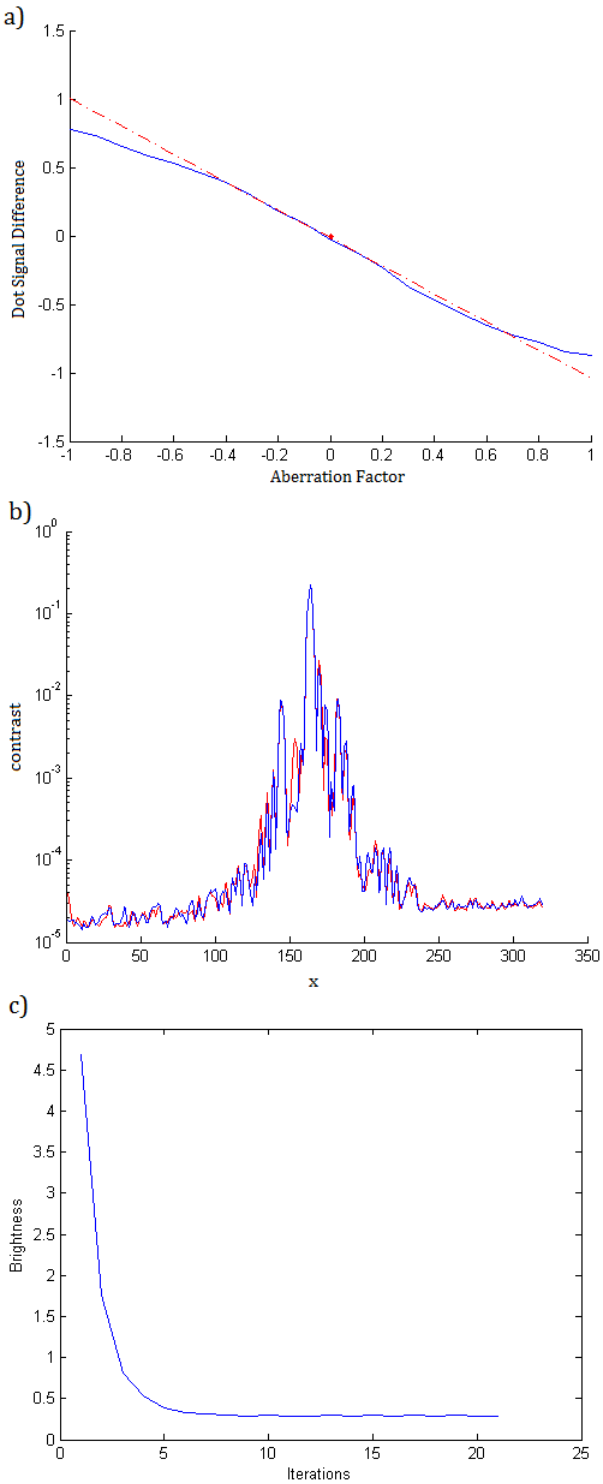


Figure 4.14: *a) The response curve for a single hologram for the first APP, it is very much linear and unaffected by crosstalk. b) An image showing the PSF before and after correction of a single mode with an 0.5 rad amplitude. Nearly one order of magnitude has been gained. c) The AO response curve, showing that after only 5 iterations the maximum amount of aberrations has been compensated.*

Chapter 5

Discussion and Conclusion

The results obtained from the simulated data are a good indication of the validity of the HMWFS as a means of keeping small areas dark with a closed loop. We have shown that for any of the three APPs it is possible to correct for aberrations with at least one radian phase difference in the form of orthogonal aberration modes. Knowing that currently the average AO system suffers from about a 50 to 100 nm RMS after the corrections[1], we see that in the worst case this translates to nearly 1 rad @ 656 nm, which should be correctable by this system. However, this only shows in the simulations, while the lab measurements have not yet been successful.

The difference between the performance of the two could be explained by a number of reasons, and we expect that if these are addressed the lab data will more closely resemble the simulations. First the crosstalk in the lab environment is much stronger then in the simulations. We can see this back in the response curves for the second and third APP in Fig. **??**, which are much less smooth than can be seen in Fig. 2.6. The response curve for the single hologram APP reacts much like our simulation. Therefore the effects can be explained as crosstalk between the modes. Fortunately, this is a well-understood process, and methods exist to counter the crosstalk, which can be read in Dong et al. (2012). Ideally one could also try to find a set of orthogonal modes that have no crosstalk by finding the eigenmodes of the response matrix of the separate modes. These modes would be statistically independent like Karhunen–Loève modes.

[1] See (Macintosh et al., 2014)

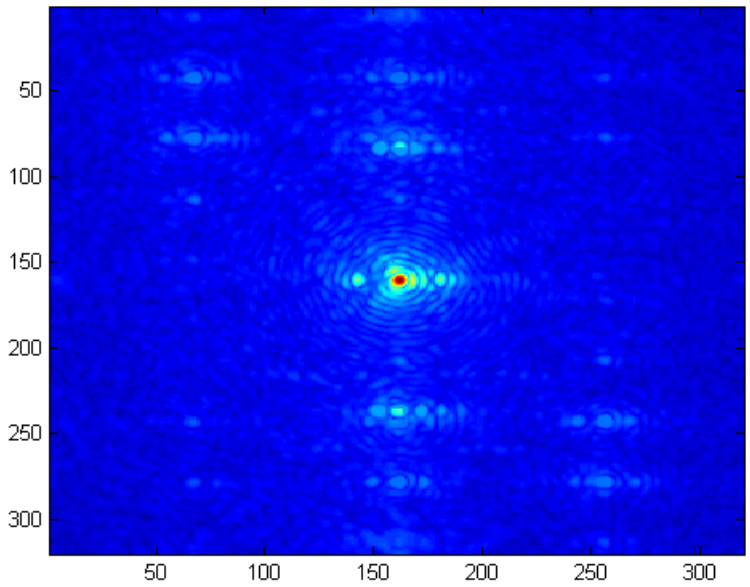


Figure 5.1: *An example of the artifacts left by the slow response time of the SLM. Here one can see a superposition of the first and second APP and their corresponding holograms.*

Secondly, the images also often display artifacts from the last exposure, introduced by the slow response time of the SLM. These effects are strongest after a sudden change in aberrations, as they cause the hologram dots to drastically change as opposed to the last image, giving an erroneous measurement. This problem is solvable by waiting longer (up to 20 seconds) between two exposures, but the source of the problem is unknown. Besides that we could improve β_k, here it has been chosen as a standard value for all CGHs, but in reality one should optimise it for every mode separately. This would ensure that all holograms are imaged as single dots, instead of the diffracting patterns that can be seen in Fig. 4.3 and all the other CGH images. These higher orders take light away from the main dot and thus lower the S/N of the measurement.

Furthermore, the setup of course contains various sources of noise that have not been taken into consideration in the simulations. For instance due to the limitations of the camera, it is impossible to get the hole darker than 10^{-5} without changing the dynamic range of the imager. This could be done by changing the exposure algorithm and make the camera satu-

rate in the peak while gaining in the darker regions. Depending on how aggressively this will be done, up to four orders of magnitude should be obtainable. However, such depths are only required for very deep holes, and as can be seen in Fig. 4.5, the CGHs fill them up considerably. As explained shortly in section 2.2, the filling is caused by high frequency Fourier signals that create low-order beat frequencies. These frequencies cannot be corrected by the modes and thus redefine the hole depth. However, by using electric field conjugation, one could remove these frequencies. This can be done under the assumption that every mode is present with a certain amplitude, and that this amplitude can be found similarly to that of a potential. This means that by measuring the depth of the hole at three points, for example with the added aberration amplitudes of $[-1, 0, 1]$ for a given mode M_k. With these three measurements, we can then fit a parabola and find its minimum. Knowing how to subtracting the mode with this amplitude should then largely remove that mode from the system. Figure 5.2 shows that after making a basic simulation we can deepen the hole by nine orders of magnitude for the third APP by looping five times through the first six modes. This deepening will then be stable and can be kept dark in a closed-loop as described in this report.

Taking all of these points into consideration, we conclude that the APP and CGH WFS has the potential of reaching the required contrast range for direct imaging Earth-like planets. However due to the small area of the hole, it could prove difficult to use this method to search for exoplanets, but could prove useful for confirming and characterising known planets. Further research is certainly recommended.

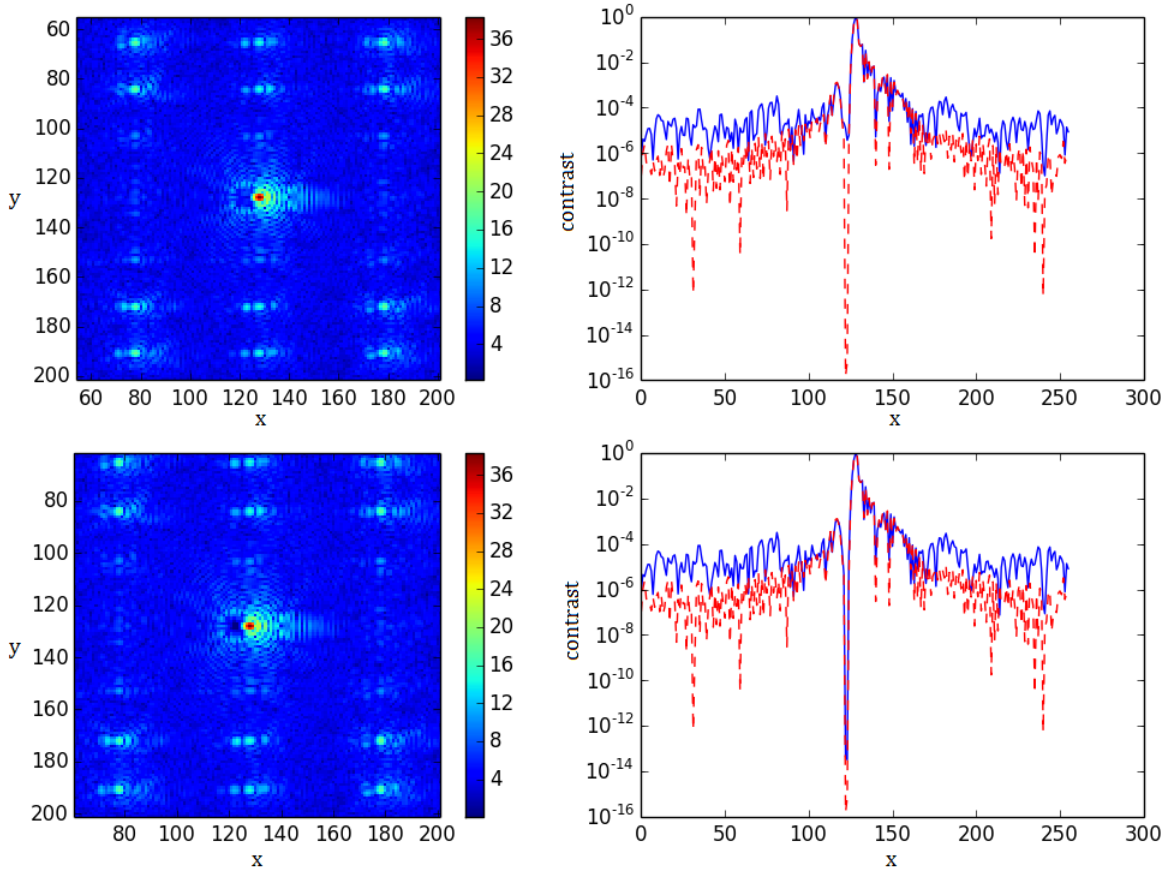


Figure 5.2: *The third APP with CGH, the intensity inside the hole is significantly decreased with electric field conjugation. The hole can again be found at x=120.*

Acknowledgements

After spending nearly half a year on this research project I can honestly say that I have leaned a great lot in the area of Fourier optics, coronagraph and wavefront sensors. I greatly enjoyed the balance between theory and experimental work and my general first taste of academic research. Because of this I would like to express my deep gratitude to my supervisors, Professor Keller and Professor Van Exter, for their patient guidance, constructive critiques and relentless pushing to achieve the best possible results.I would also like to thank Gilles Otten, Sebastiaan Haffert and Emiel Por for their advice and suggestions concerning this project. My grateful thanks are also extended to Visa Korkiakoski for letting me use his setup.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

References

Booth, M. J. 2003, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 5162, Advanced Wavefront Control: Methods, Devices, and Applications, ed. J. D. Gonglewski, M. A. Vorontsov, & M. T. Gruneisen, 79–90

Codona, J. L., Kenworthy, M. A., Hinz, P. M., Angel, J. R. P., & Woolf, N. J. 2006, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 6269, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series

Dong, S., Haist, T., Osten, W., Ruppel, T., & Sawodny, O. 2012, Applied Optics, 51, 1318

exoplanet.eu. 2014, Direct imaged exoplanets in exoplanet.eu catalogue

Frazin, R. A. 2014, ArXiv e-prints

Give'On, A., Belikov, R., Shaklan, S., & Kasdin, J. 2007, Optics Express, 15, 12338

Give'On, A., Kasdin, N. J., Vanderbei, R. J., & Avitzour, Y. 2006, Journal of the Optical Society of America A, 23, 1063

Kalas, P., Graham, J. R., Chiang, E., et al. 2008, Science, 322, 1345

Keller, C. 2014a, private communication, `http://i.imgur.com/ayyxJey.png`

Keller, C. 2014b, private communication

Kenworthy, M. A., Codona, J. L., Hinz, P. M., et al. 2007, Astrophysical Journal, 660, 762

Korkiakoski, V., Keller, C. U., Doelman, N., et al. 2014, ArXiv e-prints

Leith, E. N. & Upatnieks, J. 1962, Journal of the Optical Society of America (1917-1983), 52, 1123

Macintosh, B., Graham, J. R., Ingraham, P., et al. 2014, ArXiv e-prints

Neil, M. A. A., Booth, M. J., & Wilson, T. 2000, Journal of the Optical Society of America A, 17, 1098

Noll, R. J. 1976, Journal of the Optical Society of America (1917-1983), 66, 207

Sauvage, J.-F., Fusco, T., Rousset, G., & Petit, C. 2007, Journal of the Optical Society of America A, 24, 2334

Seager, S. & Deming, D. 2010, Annual Review of Astron and Astrophys, 48, 631

Wikipedia. 2007a, Hologram Recording Image - Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org/wiki/File:Holograph-record.svg` [Online; accessed July-2014]

Wikipedia. 2007b, Reconstructing a Hologram Image - Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org/wiki/File:Holography-reconstruct.svg` [Online; accessed July-2014]

Wolszczan, A. & Frail, D. A. 1992, Nature, 355, 145

Appendix A

Code

A.1 CGH Code

```
import pyfits as pf
import matplotlib.pyplot as plt
import numpy as np
from pylab import *

# test data
aperture = pf.getdata('app/apertv1.fits') #Aperture
modes = pf.getdata('app/modesv2.fits')
X,Y=np.mgrid[-pi:pi:2*pi/256, -pi:pi:2*pi/256]*aperture #X and Y array


fx = np.array([7, 10, 7,10])#,7,10])  #Set x position dot
fy = np.array([0, 0, 8, 8])#,-8,-8])
b=0.1 #1.5 in Dong, we make it lower because dots are too bright and splitting into 2nd orders


#pf.writeto("app/exampleaber3.fits",Z, clobber=True) #save aberations
for i in range(4):
        Z = modes[i]*aperture*200
```

```
        #Ok = np.exp(1j*(2*pi*fx*X + 2*pi*fy*Y))
        Rk = np.exp(1j*b*Z) #reference wave

        Hk = np.exp(1j*(2*pi*fx[i]*X + 2*pi*fy[i]*Y))*np.exp(-1j*b*Z) + np.exp(-1j*(2*pi*fx[i]*X + 2*pi*fy[i]*Y))*np.exp(1j*b*Z) #hologram
        if i == 0:
                cgh = Hk
        else:
                cgh = np.dstack([cgh, Hk])
        print(cgh.shape,i)

        #make hologram
        FOR = np.abs(np.fft.fftshift(np.fft.fft2(Hk*Rk)))**2
        FOR = np.abs(np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(FOR))))**2*aperture
sumcgh = np.sum(cgh, axis=2)

pf.writeto("app/alexcghapp2.fits",np.angle(sumcgh),clobber=True)

plt.subplot(1,2,1)
plt.title('Aberration Mode')
plt.imshow(Z)
plt.colorbar()
#plt.subplot(1,3,2)
#plt.title('Hologram Amplitude')
#plt.imshow(np.log(np.abs(sumcgh)))
#plt.colorbar()
plt.subplot(1,2,2)
plt.title('Computer Generated Hologram')
plt.imshow(np.angle(sumcgh))
plt.colorbar()
plt.show()
```

A.2 APP+CGH Code

```
#  Load FITS and Numeric module
import pyfits as pf
import matplotlib.pyplot as plt
import numpy as np

#  Open data file and show number of extensions
appcgh1 = pf.getdata('app/wsappv1.fits')
aperture = pf.getdata('app/apertv1.fits') #Aperture

app1 = pf.getdata('app/appv1.fits') #app1
app2 = pf.getdata('app/appv2.fits') #app2
app3 = pf.getdata('app/appv3.fits') #app3

modes3 = pf.getdata('app/modesv3.fits')*200 #modes (so
    their std ~1)
modes2 = pf.getdata('app/modesv2.fits')*180
modes1 = pf.getdata('app/abberv1.fits')
modes =  modes3[0:6] #pick first 6

cgh = pf.getdata('app/alexcghlab.fits') #modes (so
    their std ~1)
cgh2 = pf.getdata('app/alexcghapp2.fits')

aberations1 = (modes[0]*0.3+modes[1]*0.2+modes[2]*0.2+
    modes[3]*0.1)*0.5#+modes[4]*0.1+modes[5]*0.1)*0.5
aberations2 = pf.getdata('app/aberrv3a.fits')*10 #
    Aberations
aberations3 = pf.getdata('app/aberrv3b.fits')*10 #
    Aberations
aberations4 = (np.random.rand(256,256)-0.5)*aperture

allmodes = np.array([modes[0], modes[1], modes[2],
    modes[3], modes[4], modes[5]])#, modes[6], modes
    [7], modes[8], modes[9], modes[10], modes[11]])*1

numberofmodes = 6#12
appcgh = app3 + cgh*0.6

```

```
#calculate the psf and return it
def appcal(appin, aberin):
        rel = np.zeros(21)
        appcgh = appin + aberin

        fits = appcgh
        E = aperture * np.exp(1j*fits) #put into exp with amplitude =1
        E = np.fft.fftshift(E) #FFTshift because fft2 algorithm demands it

        angle= np.angle(np.fft.fftshift(np.fft.fft2(E, s=(1024,1024)))) #Angular

        #make fft
        fft = np.abs(np.fft.fft2(E))**2
        fft = np.fft.fftshift(fft) #shift is back because we shifted it before

        p=128
        hole = np.sum(np.sum(fft[127:130, 122:124]))

        return fft, hole

#Check the dots and return delta brightness
def cghhole(n, fft):
                #X and Y coordinate of top and bottom dots
                #for app1
                #xt = [126,131]
                #yt = [84,89]
                #xb = [126,166]
                #yb = [131, 171]
                hole = [123,124,128,129]
                #for app3
                xt = [126,131,126,131,75,80, 75, 80, 176, 181, 176, 181]
                yt = [81, 86, 62, 67, 81, 85, 62, 67, 81, 86, 62, 67]
                xb = [126, 131, 126, 131, 176, 181, 176, 181, 75, 80, 76, 80]
```

```
                    yb = [170, 174, 188, 193, 170, 174, 190, 192, 170, 173, 188, 193]
                    #+1 to include end coordinate
                    tophole =  np.sum(np.sum(fft[yt[n*2]:yt[n*2+1]+1, xt[n*2]:xt[n*2+1]+1]))
                    bothole = np.sum(np.sum(fft[yb[n*2]:yb[n*2+1]+1, xb[n*2]:xb[n*2+1]+1]))
                    centralhole = np.sum(np.sum(fft[hole[2]:hole[3], hole[0]:hole[1]]))
                    #print(tophole, bothole, n*2, n*2+1)
                    p = tophole/(tophole+bothole)

                    return((tophole-bothole)/(tophole+bothole), centralhole)

def linfit(mode): #linfit a line trough 3 dots around 0
        x = [-0.2,-0.1,0,0.1,0.2] #Grab a few points around 0 and fit a line
        y = mode[8:13]

        print(x,y)
        fit = np.polyfit(x,y,1)
        xfit = np.arange(-1,1.1,0.1)
        line = fit[0]*xfit + fit[1]

        plt.plot(xfit, mode)
        plt.plot(xfit, line)
        plt.title('Diagnostic Plot Fit of mode#'+str(i+1))
        plt.xlabel('Aberration Factor')
        plt.ylabel('Dot Signal Difference')
        plt.show()

        return fit

def abcor(dots):
        xlin =  np.arange(len(dots), dtype=float)
        for i in range(len(dots)):
                xlin[i] = ((dots[i]-lin[i,1])/lin[i,0])*1.0
```

```
        return xlin


# Check the slope for all dots.

# holes = np.arange(21, dtype=float)
# loophole = holes.copy()
# for i in range(numberofmodes):
        # holes = np.arange(21, dtype=float)
        # for j in range(-10,11):
                # fft, hole = appcal(appcgh, (j/10.0)*modes[i]*0.3)
                # holes[j+10], centralhole = cghhole(i,fft) #get the brightness of the ith dot for the jth iteration
        # print(i)
        # loophole = np.vstack([loophole, holes])
# loophole = np.delete(loophole, (0), axis=0)

# linhole = np.array([0,0])
# eks = np.arange(-10,11)
# for i in range(numberofmodes):
        # linfitar = linfit(loophole[i])
        # linhole = np.vstack([linhole, linfitar])
# linhole = np.delete(linhole, (0), axis=0)

# pf.writeto("lin/lin1.fits",linhole, clobber=True)
# plt.matshow(linhole)
# plt.show()

lin = pf.getdata('lin/lin1.fits') #Aberations
#try to get rid of error
aber = aberations4
aberold = aber.copy()
allmodesloop = modes.copy()
iter = 30
holedepth = np.arange(iter)
for j in range(iter):
        holes = np.arange(numberofmodes, dtype=float)

        fft, hole = appcal(appcgh, aber)
```

```
        ##################plt.matshow(fft**0.2)
        for i in range(numberofmodes):
                holes[i], centralhole = cghhole(i, fft) #get the brightness of the ith dot
        xlin = abcor(holes)
        ####Turn off all but first modes
        #xlin[4:12] = 0
        #xlin[1] = 0
        #xlin[4] = 0
        print(xlin)

        for k in range(len(allmodesloop)):
                allmodesloop[k] = modes[k]*xlin[k]
        print('modes: '+str(np.sum(np.abs(allmodesloop))))
        print('hole: '+str(centralhole))
        holedepth[j] = centralhole

        aber = aber - sum(allmodesloop)*0.1
        print('aber: '+str(np.sum(np.abs(aber))))

fft, hole = appcal(appcgh, aber)
plt.imshow(fft**0.2)
plt.show()
plt.semilogy(fft[128]/fft.max(), label='After')
fft, hole = appcal(appcgh, aberold)
plt.semilogy(fft[128]/fft.max(), 'r--', label='Before')
fft, hole = appcal(appcgh, aberold*0)
plt.semilogy(fft[128]/fft.max(), 'g--', label='Clean')
plt.ylabel('contrast')
plt.legend()
plt.show()
plt.semilogy(holedepth)
plt.ylim([10**4,10**7])
plt.xlabel('Iteration')
plt.ylabel('Signal')
plt.show()
```

A.3 Electric Field Conjugation Code

```
#  Load FITS and Numeric module
import pyfits as pf
import matplotlib.pyplot as plt
import numpy as np

#  Open data file and show number of extensions
appcgh1 = pf.getdata('app/wsappv1.fits')
aperture = pf.getdata('app/apertv1.fits') #Aperture

app1 = pf.getdata('app/appv1.fits') #app1
app2 = pf.getdata('app/appv2.fits') #app2
app3 = pf.getdata('app/appv3.fits') #app3

modes3 = pf.getdata('app/modesv3.fits')*200 #modes (so
    their std ~1)
modes2 = pf.getdata('app/modesv2.fits')*180
modes1 = pf.getdata('app/abberv1.fits')
modes =  modes3[0:6] #pick first 6

cgh = pf.getdata('app/alexcghlab.fits') #modes (so
    their std ~1)
cgh2 = pf.getdata('app/alexcghapp2.fits')

aberations1 = (modes[0]*0.3+modes[1]*0.2+modes[2]*0.2+
    modes[3]*0.1)*0.5#+modes[4]*0.1+modes[5]*0.1)*0.5
aberations2 = pf.getdata('app/aberrv3a.fits')*10 #
    Aberations
aberations3 = pf.getdata('app/aberrv3b.fits')*10 #
    Aberations
aberations4 = (np.random.rand(256,256)-0.5)*aperture

allmodes = np.array([modes[0], modes[1], modes[2],
    modes[3], modes[4], modes[5]])#, modes[6], modes
    [7], modes[8], modes[9], modes[10], modes[11]])*1

numberofmodes = 6#12
appcgh = app3 + cgh*0.6

```

```
#calculate the psf and return it
def appcal(appin, aberin):
        rel = np.zeros(21)
        appcgh = appin + aberin

        fits = appcgh
        E = aperture * np.exp(1j*fits) #put into exp with amplitude =1
        E = np.fft.fftshift(E) #FFTshift because fft2 algorithm demands it

        angle= np.angle(np.fft.fftshift(np.fft.fft2(E, s=(1024,1024)))) #Angular

        #make fft
        fft = np.abs(np.fft.fft2(E))**2
        fft = np.fft.fftshift(fft) #shift is back because we shifted it before

        p=128
        hole = np.sum(np.sum(fft[127:130, 122:124]))

        return fft, hole

#Check the dots and return delta brightness
def cghhole(n, fft):
                #X and Y coordinate of top and bottom dots
                #for app1
                #xt = [126,131]
                #yt = [84,89]
                #xb = [126,166]
                #yb = [131, 171]
                hole = [123,124,128,129]
                #for app3
                xt = [126,131,126,131,75,80, 75, 80, 176, 181, 176, 181]
                yt = [81, 86, 62, 67, 81, 85, 62, 67, 81, 86, 62, 67]
                xb = [126, 131, 126, 131, 176, 181, 176, 181, 75, 80, 76, 80]
```

```
                    yb = [170, 174, 188, 193, 170, 174, 190, 192, 170, 173, 188, 193]
                    #+1 to include end coordinate
                    tophole =  np.sum(np.sum(fft[yt[n*2]:yt[n*2+1]+1, xt[n*2]:xt[n*2+1]+1]))
                    bothole = np.sum(np.sum(fft[yb[n*2]:yb[n*2+1]+1, xb[n*2]:xb[n*2+1]+1]))
                    centralhole = np.sum(np.sum(fft[hole[2]:hole[3], hole[0]:hole[1]]))
                    #print(tophole, bothole, n*2, n*2+1)
                    p = tophole/(tophole+bothole)

                    return((tophole-bothole)/(tophole+bothole), centralhole)

def linfit(mode): #linfit a line trough 3 dots around 0
        x = [-0.2,-0.1,0,0.1,0.2] #Grab a few points around 0 and fit a line
        y = mode[8:13]

        print(x,y)
        fit = np.polyfit(x,y,1)
        xfit = np.arange(-1,1.1,0.1)
        line = fit[0]*xfit + fit[1]

        plt.plot(xfit, mode)
        plt.plot(xfit, line)
        plt.title('Diagnostic Plot Fit of mode#'+str(i+1))
        plt.xlabel('Aberration Factor')
        plt.ylabel('Dot Signal Difference')
        plt.show()

        return fit

def abcor(dots):
        xlin =  np.arange(len(dots), dtype=float)
        for i in range(len(dots)):
                xlin[i] = ((dots[i]-lin[i,1])/lin[i,0])*1.0
```

```
        return xlin


# Check the slope for all dots.

# holes = np.arange(21, dtype=float)
# loophole = holes.copy()
# for i in range(numberofmodes):
        # holes = np.arange(21, dtype=float)
        # for j in range(-10,11):
                # fft, hole = appcal(appcgh, (j/10.0)*modes[i]*0.3)
                # holes[j+10], centralhole = cghhole(i,fft) #get the brightness of the ith dot for the jth iteration
        # print(i)
        # loophole = np.vstack([loophole, holes])
# loophole = np.delete(loophole, (0), axis=0)

# linhole = np.array([0,0])
# eks = np.arange(-10,11)
# for i in range(numberofmodes):
        # linfitar = linfit(loophole[i])
        # linhole = np.vstack([linhole, linfitar])
# linhole = np.delete(linhole, (0), axis=0)

# pf.writeto("lin/lin1.fits",linhole, clobber=True)
# plt.matshow(linhole)
# plt.show()

lin = pf.getdata('lin/lin1.fits') #Aberations
#try to get rid of error
aber = aberations4
aberold = aber.copy()
allmodesloop = modes.copy()
iter = 30
holedepth = np.arange(iter)
for j in range(iter):
        holes = np.arange(numberofmodes, dtype=float)

        fft, hole = appcal(appcgh, aber)
```

```
        ##################plt.matshow(fft**0.2)
        for i in range(numberofmodes):
                holes[i], centralhole = cghhole(i, fft) #get the brightness of the ith dot
        xlin = abcor(holes)
        ####Turn off all but first modes
        #xlin[4:12] = 0
        #xlin[1] = 0
        #xlin[4] = 0
        print(xlin)

        for k in range(len(allmodesloop)):
                allmodesloop[k] = modes[k]*xlin[k]
        print('modes: '+str(np.sum(np.abs(allmodesloop))))
        print('hole: '+str(centralhole))
        holedepth[j] = centralhole

        aber = aber - sum(allmodesloop)*0.1
        print('aber: '+str(np.sum(np.abs(aber))))

fft, hole = appcal(appcgh, aber)
plt.imshow(fft**0.2)
plt.show()
plt.semilogy(fft[128]/fft.max(), label='After')
fft, hole = appcal(appcgh, aberold)
plt.semilogy(fft[128]/fft.max(), 'r--', label='Before')
fft, hole = appcal(appcgh, aberold*0)
plt.semilogy(fft[128]/fft.max(), 'g--', label='Clean')
plt.ylabel('contrast')
plt.legend()
plt.show()
plt.semilogy(holedepth)
plt.ylim([10**4,10**7])
plt.xlabel('Iteration')
plt.ylabel('Signal')
plt.show()
```