Control for High Resolution Imaging

Adaptive Optics in Python - Wavefront Generation

June 23, 2014

Lars Sybren van Leeuwaarden

Contents

1 Introduction

1.1 Overview on Wavefront Generation

The goal of the wavefront generation part of the project Adaptive Optics in Python is as the name suggests, to create a wavefront that can be processed by the following sections of the adaptive optics. For the current implementation of the wavefront generation, the wavefront generation is split into two sections. The first one is the creation of aberrations in the optical set-up deterministic Zernike modes. The second part involves the creation of aberrations based on stochastic analysis of atmospheric aberrations.

The distinction between these two sections is somewhat ambiguous, as both of these components will be added up to combine to the final wavefront. The choice to separate them is more based on implementation considerations rather then mathematical reasons.

The current implementation of the wavefront generation is static. There is no temporal dependency implemented. This can be a shortcoming in phase screen generation, as it currently cannot model a passing cloud for example. The choice for this is simply because project was bounded. This is a topic for improvement in the future. Where possible, effort has been done to make the wavefront generation code ready to be extended in the future. Simply by adding functions and methods to the classes and making existing functions and methods easy to be extended - the object oriented programming approach.

1.2 Wave Equation

The behaviour of light is, as is well known, explained by Maxwell's equations. Clever substitution of Ampère's law into Faraday's law - or vice versa - and using Gauss' law to ensure dependency on only the magnetic field or the electric field. This will yield a neat equation where the wave behaviour is naturally apparent, the wave equation. This reader is probably not the best location to bore you with the entire derivation of the wave equation. It is however important to remember the plane wave behaviour of light later on, as it will make understanding much easier [1].

In equation 1, the wave equation is given. In this equation, polarization is ignored. In the equation, $n^2 = \epsilon/\epsilon_0$ is the ratio of the dielectric constants, or the refractive index and c is the propagation speed of the wave, or the speed of light. The vector $U = (E, B)^T$ is the combined vector, containing the electric field and the magnetic field.

$$\left(\nabla^2 - \frac{n^2}{c^2}\frac{\partial^2}{\partial t^2}\right) U(\mathbf{r}, t) = 0 \tag{1}$$

In order to solve equation 1, separation of variables is the usual approach. In this case, separating it in a spatially dependant part and a time dependant, hence $U(\mathbf{r}, t) = U(\mathbf{r})u(t)$.

$$\left(\nabla^2 - \frac{\omega^2 n^2}{c^2}\right) U(\mathbf{r}) = 0 \tag{2}$$

$$\left(\frac{\partial^2}{\partial t^2} - \frac{\omega^2}{n^2}\right) u(t) = 0 \tag{3}$$

[1] For a more extensive derivation of the wave equation, please refer to the appendix.

2 Aberrations

2.1 Generalized Pupil Plane

With the symmetry in time and propagation direction, solving the spatial derivative will suffice for a temporal analysis. This variant of the well known Helmholtz equation, will give the result in 4. Here the wavenumber $k_i = {}^{2\pi}/_{\lambda_i}$ to separate the wave into it's respective spatial directions.

$$U(\mathbf{r}) = A(x,y)e^{in\mathbf{k}\cdot\mathbf{r}+\phi(x,y)} \tag{4}$$

Equation 4 can be simplified, for the goal of introducing the source of the aberrations. Here $\phi(x,y) = 2\pi W(x,y)$ is created a used in equation 5. Furthermore, the power function can be further separated in z-direction and (x,y)-directions. Hence: $P(z) = e^{ikz}$ and $P(x,y) = A(x,y)e^{ikp(x,y)}$, this makes sense if the z-direction is assumed to be the direction propagation. Since this is source of the radiation, z can be set to zero, thus the exponent reduces to one. The result of these assumptions and simplifications introduce the generalized pupil function, $\mathcal{P}(x,y)$.

$$\mathcal{P}(x,y) = P(x,y)e^{i2\pi W(x,y)} \tag{5}$$

2.2 Seidel Aberrations

The pupil function $\mathcal{P}(x,y)$ introduces the wavefront $W(x,y)$. The wavefront introduces a phase offset on the wave which is not necessarily equivalent on the pupil plane. As such, it varies corresponding to the location (x,y) on the plane. The wavefront W could be any arbitrary number. However, commonly the wavefront is constructed of distinctive modes. Distinctive in the sense that the mode is linearly independent from the other modes, or in other words orthogonal. This seems far fetched from this point, but consider optical set-ups where lenses can introduce for example a tilt, a defocus or a stigma to the image. These are distinctive or unique aberrations, from a physical point of view. That is, any one of the aberrations cannot be expressed in terms of the others.

Table 1: The first six physical modes of aberration, and their expression [1].

Physical Description	Mathematical Expression
Piston Mode	A_0
Lens tilt	$A_1 r\cos\theta + A_2 r\sin\theta$
Defocus	$A_3 r^2$
Stigma	$A_4 r^2\cos(2\theta) + A_5 r^2\sin(2\theta)$
Coma	$A_6 r^3\cos\theta + A_7 r^3\sin\theta$
Spherical Aberration	$A_8 r^4$

In table 1, a list of common aberrations is given together with a mathematical expression for the aberration. The parameters A_i are the weights of the corresponding modes. The parameters r and θ are the polar coordinates where r is the normalized distance to the (local) optical origin, on the range $[0,1]$ and the angle θ defined the range $[0,2\pi]$.

Notice here how for example a defocus in the wavefront cannot be expressed in a linear multiple of a stigma or a lens tilt. In figure 1 plots are given of the six distinctive aberrations described in the table.

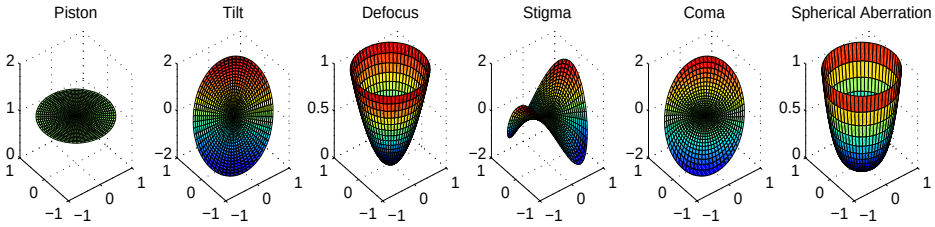

Figure 1: Seibel aberrations.

In order to create the total aberration of the wavefront, all the modes are summed together with their respective weights A_i. This will give polynomial series, the Seidel aberration, as is given in 6. In equation 6 the polynomial expansion is limited to the stigma aberration.

$$W(x,y) = A_0 + A_1 r\cos\theta + A_2 r\sin\theta + A_3 r^2 + A_4 r^2\cos(2\theta) + A_5 r^2\sin(2\theta) + \dots \tag{6}$$

This formula already gives a easy and reasonable representation of aberration. It however has limitations, which call for a better description of aberration, as will be pointed out in the section below, 3.

3 Zernike

Seibel aberrations are easy to compute, but have the downside of being limited in the mathematical sense. For the purpose of identifying weights of the various aberration modes, it is desired to have modes that are at least orthogonal to each other. This way decomposition in the different aberration modes is easy in later stages.

3.1 Zernike Polynomials

As with the Seibel aberrations, a polar coordinate system $[r,\theta]$ is used to express the location on the Zernike polynomial. This is a natural choice as most apertures are round. The polynomial is chosen such that the modes are orthogonal on the range $r \leq 1$ or $x^2 + y^2 \leq 1$. Further, the Zernike polynomals consist of a radial and an annular part with some scaling factor, like in equation 7 [2].

$$Z_n^m(r,\theta) = A_n^m R_n^m(r) G_n^m(\theta) \tag{7}$$

By definition, orthogonality of two possibly complex functions, can be found by taking the inner, or dot product of the two functions $V_i(x,y)$. With respect of the range of the function, it is possible to write the orthogonality as in equation 8.

$$\langle V_\alpha(x,y), V_\beta(x,y)\rangle = \iint\limits_{x^2+y^2\leq 1} V_\alpha^*(x,y)V_\beta(x,y)dxdy = A_{\alpha,\beta}\delta_{\alpha,\beta} \tag{8}$$

Now take $V_i(x,y) = V_i(r\cos\theta, r\sin\theta) = a_i r e^{\pm im\theta} = R_i(r)G_i(\theta)$ and it can be proven that there is one and only one set of polynomials that is orthogonal in the interior of the unit circle. From the section on Seibel aberrations, it is clear that the total aberration is built from multiple

modes. Thus validating the polynomial expansion from equations 9 and 10 [2], with the introduction of index m, indicating the angular mode. Furthermore, the index n is introduced below to indicate the radial mode of the polynomial.

$$\mathcal{V}_n^{\pm m}(x,y) = \mathcal{R}_n e^{\pm im\theta} \tag{9}$$

$$\mathcal{R}_n(r) = a_{k,0}^m r^m + a_{k,2}^m r^{m+2} + \cdots + a_{k,2k}^m r^{m+2k} \tag{10}$$

The requirement of orthogonality modes can be put to use here. The orthogonality condition of equation 8 can be expanded with the equations 9 and 10 above. Additionally, separating the orthogonality problem in the radial and angular part, as in equation 11 and will allow to solve for a *unique* set of orthogonal Zernike modes [2].

$$\begin{cases} \iint\limits_{x^2+y^2\leq 1} \mathcal{V}_n^{*m}(x,y)\mathcal{V}_n^m(x,y)dxdy & = a_n\delta_n \\ \int\limits_0^1 \mathcal{R}_n^*(r)\mathcal{R}_n(r)rdr & = a_n\delta_n \end{cases} \tag{11}$$

Without getting to into the exact derivation, which is beyond the scope of this report, for that see [2]. Equation 11 guarantees orthogonality and completeness of the Zernike polynomials. In order to get a realization for the conditions specified, an orthogonalization is done. Note that $\mathcal{V}_n^{*m}(x,y)$ can be rewritten from polar to Cartesian format to a conditional sine-cosine relation as in equation 12 [1] [3] [2].

$$\left.\begin{matrix} V_n^m(x,y) \\ V_n^{*m}(x,y) \end{matrix}\right\} = R_n^m(r)e^{\pm im\theta} = R_n^m(r) \begin{cases} \sin(m\theta) & \text{odd mode} \\ \cos(m\theta) & \text{even mode} \end{cases} \tag{12}$$

With the angular part of the equation solved, only the radial part is left. This is a somewhat less obvious derivation. Summarizing the Zernike polynomials, with respect to the format proposed in equation 7, equation 13 is obtained [1].

$$Z_n^m(r,\theta) = \begin{cases} \sqrt{2(n+1)}R_n^m(r)G^m(\theta) & m \neq 0 \\ R_n^0 & m = 0 \end{cases} \tag{13}$$

Where $R_n^m(r)$ is given by equation 14. Equation 15 defines the angular part $G^m(\theta)$, similar to equation 12, just above. Note the the orthogonality framework presented here also sets limits on the choices of n and m. Firstly, $|m| \leq n$, or even more precise $m = -n, -n+2, ..., n-2, n$, see the exponent in equation 10. Secondly, $n \geq 0$ must also hold for the index n.

$$R_n^m(r) = \sum_{s=0}^{(n-m)/2} \frac{(-1)^s(n-s)!}{s!(\frac{n+m}{2}-s)!(\frac{n-m}{2}-s)!} r^{n-2s} \tag{14}$$

$$G^m(\theta) = \begin{cases} \sin(m\theta) & m \text{ odd} \\ \cos(m\theta) & m \text{ even} \end{cases} \tag{15}$$

3.2 Noll Indices

As explained in the section above, the indices m and n are bound to certain conditions set by the orthogonality constraint. Choosing some n will give a range of possibilities for m. It is safe to say that this is not very practical and generally a mapping is chosen such that there is some integral number i that maps to a specific (m, n). The mapping $i \to (m, n)$ used here is the Noll mapping. Its system is given in table 2 for the first 28 modes.

Table 2: Noll's mapping of the first 28 modes. The number comprising the pyramid shape is the index i.

n \ m	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
0							1						
1						3		2					
2					5		4		6				
3				9		7		8		10			
4			15		13		11		12		14		
5		21		19		17		16		18		20	
6	27		25		23		22		24		26		28

When taking a closer look at the entries of the table, the values are not entirely arbitrary. It is obvious that negative values of m have an odd index i and positive values of m have a even index. For the $m = 0$, the odd and even indices are alternating, starting with a odd number at $n = 0$.

Furthermore, the minimum index value at a any given row n alternates at a set pattern: $m_n = 0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots$. This will make it easy to find starting point at for every radial mode n.

Now list the minima of the indices for every table row n: $i_n^{m=0} = [1, 2, 4, 7, 11, 16, 22, \ldots]$. Then take take the difference of every element in the list: $\Delta i_n^{m=0} = [1, 2, 3, 4, 5, 6, \ldots]$. Indeed, every entry in the list $i_n^{m=0}$ is incremented with the next radial index n. Thus, given some i of the Noll index, it is possible to find the corresponding value n by subtracting $\Delta i_n^{m=0} = [1, 2, 3, 4, 5, 6, \ldots]$ from i one by one and checking if $i - \sum \Delta i_n^{m=0} \geq 0$.

The remainder can then be used to compute m. By simply checking if i is odd or even the, m can be set positive or negative. For the actual value of m take into account the alternating series of m_n and the fact that they are defined on range of m: $m = -n, -n + 2, ..., n - 2, n$ as pointed out earlier.

This will give a single scalar that can be used to map to two scaler, making it much easier to compute wavefronts as will be pointed out in the next section. Figure **??** plots the modes with corresponding Noll's index.

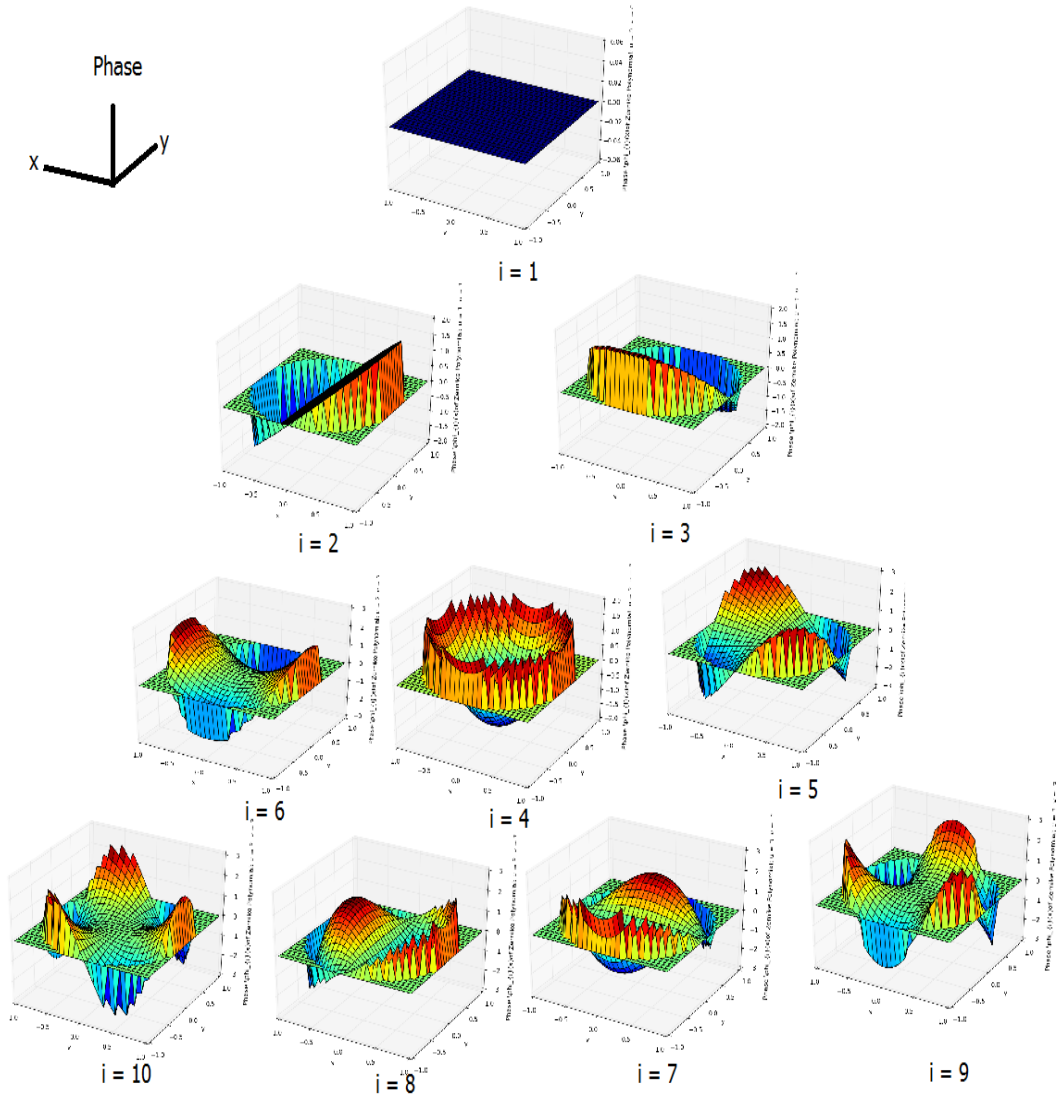

Figure 2: Noll's mapping, for the first then modes with corresponding plots of the Zernike modes in Python.

3.3 Constructing the Wavefront

Remember the section on Seidel aberrations, a practical wavefront rarely exists of only one mode of aberration. Usually, it is a linear combination of multiple modes. With the guarantee of orthogonality between the various modes by definition, it easy to compose a sum for the linear combination, as in equation 16 [1]. Also note how the newly introduced Noll indexing has replaced the former two indexes as in equation 13. Also note similarities with the field of dynamics. With dynamical systems, the time behaviour is a composition of the eigenmodes. The eigenvectors and eigenvalues determine the spatial direction and the natural frequency respectively. Since a static aberration is considered static for this coding project, only the spatial part is considered. Otherwise $W(r,\theta)$ can considered to be $W(r,\theta,t) = W(r,\theta)w(t)$, a similar decomposition a is considered in equations 2 and 3.

$$W(r,\theta) = \sum_{i=1}^{\infty} a_i Z_i(r,\theta) \tag{16}$$

Equation 16 introduces the weighting factor a_i. It has the same purpose as it had in the Seidel aberrations, to give a certain strength to a aberration mode. In figure 3 a example is plotted. Here the weightings a_i are defined as follows, corresponding to the following physical modes: tilt, defocus and a y-direction pentafoil respectively.

$$\begin{cases} a_2 = 0.5 & i = 2 \\ a_4 = 0.25 & i = 4 \\ a_{21} = -0.6 & i = 21 \\ a_i = 0 & i \neq 2, 4, 21 \end{cases} \tag{17}$$

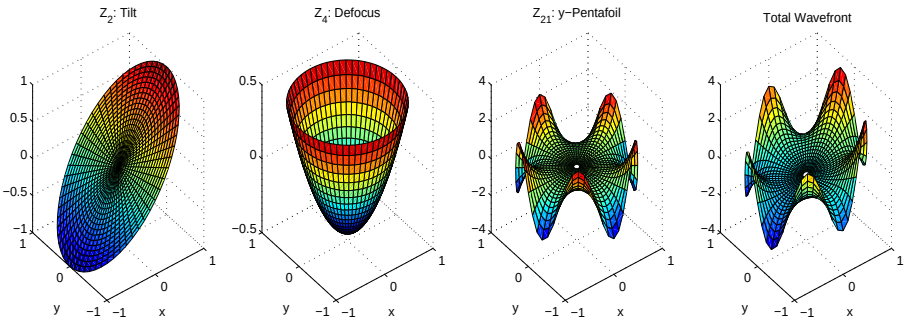

Figure 3: Zernike aberrations. From left to right: tilt Z_2, defocus Z_4, y-pentafoil Z_{21} and the composed wavefront $W(r,\theta)$.

3.4 Decomposition of the Wavefront

The framework built for constructing wavefront by adding the several desired modes, can also be used the other way round. Given some wavefront and a set of modes to be used, what weighting factors do the modes need to get in order to minimize the difference between the 'real' wavefront and the composed wavefront.

Again the definition of orthogonality for the Zernike comes in handy. That way the problem reduces to a decomposition of orthogonal projections, much like the well known Gram-Schimdt algorithm in linear algebra. It is not required to create a orthogonal basis, as the orthogonal bases comprises of the Zernike modes Z_i where $W(r, \theta)$ needs to be projected on, which are already orthogonal.

Let's explain according to linear algebra concepts. Imagine two vectors $\mathbf{a}$ and $\mathbf{b}$. Now project vector $\mathbf{a}$ onto the basis vector $\mathbf{b}$, the amount of common mode of is given with the projection formula, see equation 18 [4].

$$\text{proj}_{\mathbf{b}}(\mathbf{a}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}} \mathbf{b} \tag{18}$$

In this case, it is not necessary to find the actual basis, that is already known. Thus the weighting is the only matter of interest, or the amount of common mode with respect to the basis vector $\mathbf{b}$.

From the crude Gram-Schimdt algorithm, it is possible to develop a algorithm which is more suited to the decomposition of the wavefront in a particular set of Zernike modes. Recall that only the amount of common mode, the weighting, is desired as the basis is already orthogonal, see equation 19.

$$a_i = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}} \tag{19}$$

Rewriting the dot product into a the calculus format as in equation 20, as proposed earlier in equation 8.

$$a_i = \frac{\int_0^{2\pi} \int_0^1 W(r, \theta) Z_i(r, \theta)}{\int_0^{2\pi} \int_0^1 Z_i^2(r, \theta)} \tag{20}$$

For a numerical implementation, it is necessary to discretize the system, as is done in equation 21 [5]. Note the change in coordinate system, as the discrete grid is rectangular, for Fourrier transform purposes later on. Furthermore, p and q iterate over the grid formed by the discretization.

$$a_i = \frac{\sum_p \sum_q W(x_p, y_q) Z_i(x_p, y_q)}{\sum_p \sum_q Z_i^2(x_p, y_q)} \tag{21}$$

Equation 21 will satisfy, if only one mode requires computation. More often, it is desired to compute all modes simultaneously. On top of that, it is desired to utilize the numerical power of linear algebra. For that, consider equation 16. Rewriting it results in equation 22.

$$W(x_p, y_q) = \begin{bmatrix} Z_1 & | & Z_2 & | & \dots & | & Z_n \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix} \tag{22}$$

Rewriting $Z_i(x_p, y_q)$ to a vector in order to reduce the sets of systems to a single system of equations. Introduce $Z_i(x_p, y_q) = \bar{Z}_i(j)$ and $W(x_p, y_q) = \bar{W}(j)$, by stacking all the columns of Z_i under each other. The system reduces to equation 23.

$$W(x_p, y_q) = \mathcal{Z}A = \begin{bmatrix} \bar{Z}_1 & \bar{Z}_2 & \dots & \bar{Z}_n \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix} \tag{23}$$

Finding the vector of weights A is easy now, by taking the pseudo-inverse as is done in equation 24. Note that the normal inverse generally does not work, as $\mathcal{Z}$ not square and not necessarily non-singular.

$$A = \left(\mathcal{Z}^T \mathcal{Z}\right)^{-1} \mathcal{Z}^T \bar{W}(j) \tag{24}$$

However note here: $\mathcal{Z}$ is column orthogonal, hence it's construction in equation 23. If this is the case, $\mathcal{Z}^T\mathcal{Z}$ should reduce to a diagonal matrix [2]. Numerical inaccuracies in a computer will introduce derivations from true identity. More importantly, the wavefront is sampled with a finite number of samples. This can also cause the computation of $\mathcal{Z}^T\mathcal{Z}$ to no longer give "algebraic" diagonal matrices.

A small test was done with the code written for this project. For the small test, the modes and weights from equation 17 were imputed to the code, together with a 30-by-30 grid. The relative difference between the diagonal entries and the upper and lower triangles of the matrix - excluding the diagonal entries - are in the order of 10^{16} and larger. Considering this, it is fair to state that the matrix is diagonal.

[2]The matrix $\mathcal{Z}^T\mathcal{Z}$ is not identity, as equation 8 allowed a scalar to exist in the orthogonality condition, the system is thus not orthonormal.

3.5 Validity of the Decomposition

This method is in general only as good as the number of modes supplied to the decomposition. In general, more modes should give a better approximation of the real aberration. For validation purposes, it can be desired to compute the root mean square error - or standard deviation - that is introduced by using a finite series expansion of the Zernike modes. Equation 25 [5] introduces a algebraic framework for the RMS error for a 2D surface. In this equation $W(r,\theta)$ is the original wavefront and $\bar{W}$ the mean of the wavefront.

$$\sigma^2 = \frac{1}{\pi}\int_0^{2\pi}\int_0^1 \left(W(r,\theta) - \bar{W}\right)^2 r dr d\theta \tag{25}$$

With $\bar{W}$ behaving as a piston mode, thus as the mode $i = 1$ in Noll's indexing. It possible to introduce equation 16 again into equation 25. In equation 26], the last equation is substituted and split into a integral part and constant part.

$$\sigma^2 = \frac{1}{\pi}\sum_{i=2}^{\infty} a_i \sum_{i=2}^{\infty} a_i' \int_0^{2\pi}\int_0^1 Z_i(r,\theta) Z_i'(r,\theta) r dr d\theta \tag{26}$$

Then recall the orthogonality property from 8 and computing π as the constant for the Zernike orthogonality, then equation 27 is obtained.

$$\sigma^2 = \frac{1}{\pi}\sum_{i=2}^{\infty} a_i \sum_{i=2}^{\infty} a_i' \pi \delta_{ii'} \tag{27}$$

Reducing to equation 28, due to the Dirac delta function. Also consider it with substitution of the result of equation 24.

$$\sigma^2 = \sum_{i=2}^{\infty} a_i^2 = A^T A \tag{28}$$

A powerful result, as the variance on the wavefront is simply the sum weights squared. Again by made easy by the orthogonal properties of the Zernike modes.

4 Atmospheric Aberrations

4.1 Turbulence in the Atmosphere

If you've ever looked up to the night's clear sky and looked at a star, you've probably noticed a twinkling. Now, the visual surface of a star is by far a steady surface, so the twinkles might be down to intensity variations. But considering the huge distance between Earth and the star, it is not possible to distinguish these variations with our eyes. Instead, this 'twinkling' down to column of atmosphere that between the star and the eye. The air in the atmosphere is obviously not a vacuum and thus has a refractive index.

This refractive index is close to unity compared to the vacuum. Usually close enough to pretend as if the light was travelling trough a vacuum. However, as distance through the medium increases and as the medium is showing turbulence, it becomes a problem for the static optical set-up. From equation 1 it can be shown that changes in the refractive index will directly influence the frequency of the light wave. Thus changes in refractive index, combined with continuity of electromagnetic fields and conservation of energy, will cause changes in propagation speed of the wave and hence changes in the direction of propagation of the wave. This in turn will be observed as phase shifts to the observer.

Thus the model to introduce will need to describe variations in the refractive index in the path of a light wave. The variations in refractive index, caused by temperature differences, pressure differences and air humidity. This creates equations like the Edlen equation [6]. Empirical equations and often upgraded and amended to fit new measurements.

Unfortunately, none of the parameters, like the temperature at any height, are available for observers on the ground. This will complicate the use of (modified [3]) Fresnel propagation theory for example, the Split-Step Beam Propagation Method. Therefore, statistical models are introduced that match the statistic properties of a column of air.

4.2 Kolmogorov

Simply by the fact that an algebraic model is extremely difficult to come by with this many parameters. It will involve creating solutions for the Navier-Stokes equation, a non-linear problem for developed turbulent flows. The Kolmogorov theory of turbulence introduced the idea of matching statistical behaviour of a model with that of experimental data, based on turbulent flows [7].

From fluid dynamics, the average size of turbulence 'bubbles' or eddies, can be related to their speed via the proportionality in equation 29. Note thet for the functions to come isotropy and local homogeneity is assumed. That is, fully developed turbulence is assumed and spatial changes in the turbulences are 'quasi-static'.

$$v \propto r^{\frac{1}{3}} \tag{29}$$

[3]The term 'modified' is used in the sense that it uses the common Fresnel propagation of light in vacuum and exploits the fact that the refractive index is close to one.

From this point a structure function is used $D(r) = c^2 v^2$. The structure function is created from the statistical properties of the aberrations. In the structure function here, the constant c is dependant on the length scale, as is given in the equation set 30. Here two length scales are introduced, the inner small scale and the largest scale. The inner scale l_0 is in the range of the smallest turbulence bubble. L_0 on the other hand is in the range of the height of the atmosphere column [5].

$$D_v(r) = \begin{cases} C_v^2 l_0^{-\frac{4}{3}} r^{\frac{2}{3}} & 0 \leq r \ll l_0 \\ C_v^2 r^{\frac{2}{3}} & l_0 \ll r \ll L_0 \end{cases} \tag{30}$$

Similarly, this can be done for the relation between r and the temperature potential θ, this proportionality reads $r \propto \theta^{\frac{1}{3}}$. Again creating the structure functions as in set 31 [5].

$$D_\theta(r) = \begin{cases} C_\theta^2 l_0^{-\frac{4}{3}} r^{\frac{2}{3}} & 0 \leq r \ll l_0 \\ C_\theta^2 r^{\frac{2}{3}} & l_0 \ll r \ll L_0 \end{cases} \tag{31}$$

Now for the index of refraction n is made as a stochastic variable and is introduced as in equation 32. Note that the expected value of the refractive index is close to one as this is a property of air and the stochastic variable n_1 can be empirically found as in equation 33. Here $P(r)$ represents the pressure, $T(r)$ the ordinary temperature and λ the wavelength [5].

$$n(r) = \mu_n + n_1(r) \tag{32}$$

$$n_1(r) = 77.6 \cdot 10^{-6}(1 + 7.52 \cdot 10^{-3}\lambda^2)\frac{P(r)}{T(r)} \tag{33}$$

Taking the derivative of the refractive index, and assuming the wavelength equal to $\lambda =$ 500nm, equation 34 is obtained [5].

$$dn = 7.99 \cdot 10^{-5}\left(dP - \frac{-dT}{T^2}\right) \tag{34}$$

Now assume the pressure locally constant in the turbulence bubble, and the temperature relation to the temperature potential as $d\theta = d(T - T_0) = dT$. Thus assuming the reference temperature T_0 constant in the bubble. This allows to reduce the change in refractive index to equation 35.

$$dn = 7.99 \cdot 10^{-5}\frac{d\theta}{T^2} \tag{35}$$

Recall equation 31 and observe that the derivative of the refractive index is proportional to the derivative of the potential temperature. Thus the structure function of the potential temperature can be used to describe the structure function of the refractive index, under the assumptions described above as in equation 36 [5].

$$D_n(r) = \begin{cases} \left[7.76 \cdot 10^{-7}\left(1 + 7.52 \cdot 10^{-3}\lambda^{-2}\right)\frac{P}{T^2}\right]^2 C_\theta^2 l_0^{-\frac{4}{3}} r^{\frac{2}{3}} & 0 \leq r \ll l_0 \\ \left[7.76 \cdot 10^{-7}\left(1 + 7.52 \cdot 10^{-3}\lambda^{-2}\right)\frac{P}{T^2}\right]^2 C_\theta^2 r^{\frac{2}{3}} & l_0 \ll r \ll L_0 \end{cases} \tag{36}$$

Using the structure function found above for the index of refraction, it is now possible to calculate a power spectral density of the refractive index. Power spectral density allows transforms the structure functions to their corresponding spatial frequency domain [5].

$$\Phi_n(f) = \frac{1}{4\pi^2 f^2} \int_0^\infty \frac{\sin(fr)}{fr} \frac{d}{dr}\left[r^2 \frac{d}{dr} D_n(r)\right] dr \tag{37}$$

Here the spatial frequency f is defined as $f = 2\pi(f_x + f_y)$ The spatial frequency components in x and y direction of the phase plane. The integral computes to equation 38, the Kolmogorov power spectral density function. The spatial frequency term is introduced here as $f^2 = f_x^2 + f_y^2$ [5] [3] [4].

$$\Phi_n(f) = 0.033 C_n^2 f^{\frac{-11}{3}} \quad \text{for} \quad \tfrac{1}{L_0} \leq f \leq \tfrac{1}{l_0} \tag{38}$$

What remains, is the term C_n to be computed. Using the result from equation 5, it is possible to derive a mean value for the optical field, as in equation 39 and the covariance of the optical field in equation 40, both calculated from Rytov theory.

$$\langle U(\mathbf{r})\rangle = U_0(\mathbf{r})\langle e^{\psi(\mathbf{r})}\rangle \tag{39}$$

$$B(\mathbf{r}, \mathbf{r}^{'}) = \langle U(\mathbf{r}) U^*(\mathbf{r}^{'}) = e^{-\frac{1}{2} D_\phi(\mathbf{r})} \tag{40}$$

The covariance function can be used to compute the coherence factor of the wave given in equation 41. Also note that the spatial position on the wavefront $\mathbf{r}$ can be replaced by a distance between the points $\mathbf{r}$ and $\mathbf{r}^{'}$ as the the system is assumed isotropic and homogeneous [5].

$$\mu(\mathbf{r}, \mathbf{r}^{'}, z) = \frac{B(\mathbf{r}, \mathbf{r}^{'}, z) B(\mathbf{r}, \mathbf{r}^{'}, z)}{B(\mathbf{r}, \mathbf{r}, z) B(\mathbf{r}^{'}, \mathbf{r}^{'}, z)} = \mu(|\Delta\mathbf{r}|, z) \tag{41}$$

Using the same principle of the structure functions as was done previously with the refraction index, temperature and turbulence bubble speed, a structure function for the wavefront is introduced as in equation 42. Under the assumption that it behaves as plane wave [3].

$$D(\mathbf{r}, \mathbf{r}^{'}) = -2 \ln \mu(\mathbf{r}, \mathbf{r}^{'}) = D_\chi(\mathbf{r}, \mathbf{r}^{'}) + D_\phi(\mathbf{r}, \mathbf{r}^{'}) \tag{42}$$

The last equation combined with the PSD function from equation 37, but only with D_n and Φ_n replaced by D_ϕ and Φ_ϕ respectively. Without going into the extensive derivation [8]: the results for μ are obtained in equation 43 for a plane wave [2].

$$\mu(|\Delta\mathbf{r}|, z) = \exp\left\{-4\pi^2 k^2 \int_0^{\Delta z} \int_0^\infty \Phi_n(f, z)(1 - J_0(f|\Delta\mathbf{r}|)) df dz\right\} \tag{43}$$

The actual evaluation of μ will depend on the assumptions taken. Since Kolmogorov is the reference frame for now, the spatial frequency f is limited to the range $\frac{1}{L_0} \leq f \leq \frac{1}{l_0}$. Now C_n will need to be replaced by quantities from experimental data, the quantity μ.

As such, μ can be expressed as in equation 44 under Kolmogorov assumptions. The constant constant is determined in from the Strehl ratio for various aperture diameters [9].

$$\mu^{Kol}(|\Delta\mathbf{r}|, z) = \exp\left\{-1.46 k^2 |\Delta\mathbf{r}|^{\frac{5}{3}} \int_0^{\Delta z} C_n^2(z) dz\right\} \tag{44}$$

[4] In the original article proposed by A.N. Kolmogorov in 1939, he actually proposed a proportionality to a length r [7], rather then equalities. Only later sources such as [5] describe an actual scalar.

[5] In other words the travelled path of the light is assumed not to be of interest, in is considered not to be a path integral. As if the distortions are 'white', to see it's time domain equivalence.

Now, introducing the Fried parameter r_0, the typical scale length of the turbulence, as is done for a plane wave in equation 45 [3] [2].

$$r_0 = 0.423k^2|r_0|^{\frac{5}{3}} \int_0^{\Delta z} c_n^2(h)dz \tag{45}$$

Also D_ϕ in equation 46 and $B(\Delta\mathbf{r})$ in equation 47 can be obtained and expressed in the Fried parameter [2].

$$D_\phi(\mathbf{r}, \mathbf{r}^{'}) = 6.88\left(\frac{|\mathbf{r}|}{r_0}\right)^{\frac{5}{3}} \tag{46}$$

$$B(\mathbf{r}) = e^{-3.44\left(\frac{|\mathbf{r}|}{r_0}\right)^{\frac{5}{3}}} \tag{47}$$

All things considered, C_n can now be replaced from the Kolmogorv PSD equation as suggested in 48. Here C_n is replaced with the Fried parameter, the characteristic scale length of the turbulence.

$$\Phi_n(f) = 0.023r_0^{-\frac{5}{3}} f^{\frac{-11}{3}} \quad \text{for} \quad \tfrac{1}{L_0} \leq f \leq \tfrac{1}{l_0} \tag{48}$$

4.3 Power Spectral Densities

As indicated in equations the validity of the Kolmogorov power spectral density is limited to a range of $\frac{1}{L_0} \leq f \leq \frac{1}{l_0}$. These bounds can be limiting in some cases. The solution is partly in evaluating the equation in the paragraphs above again, but now with other constraints set on the spatial frequency.

A other PSD for practical implementation is the von Kármán PSD and the modified von Kármán. Their respective PSD's are given in the equations below in equation 49 and 50 [2]. Note the strong similarities between the PSD, only element are added that become increasingly dominant on certain ranges of spatial frequency.

$$\Phi_n^{vKar}(f) = \frac{0.023 r_0^{\frac{5}{3}}}{\left(f^2+f_0^2\right)^{\frac{11}{6}}} \quad \text{for} \quad 0 \leq f \leq \frac{1}{l_0} \tag{49}$$

$$\Phi_n^{MvKar}(f) = \frac{0.023 r_0^{-\frac{5}{3}} \exp\left\{\frac{f^2}{f_m^2}\right\}}{(f^2+f_0^2)^{\frac{11}{6}}} \quad \text{for} \quad 0 \leq f < \infty \tag{50}$$

Von Kármán removes the lower boundary and modified Von Kármán also removes the upper boundary. Other PSD, like Tararskii, remove the upper boumd compared to Kolmogorov PSD. in figure 4, Kolmogorov and Von Kármán power spectral densities are plotted. The way it is presented shows strong similarities with bode plots. The spatial frequency is then replaced by time frequency, but still the y-axis can be considered a gain at a certain frequency.

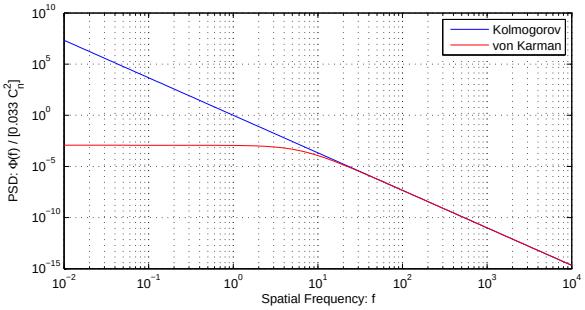

Figure 4: Kolmogorov and von Kármán power spectral density functions, L_0 set to one, normalized in the common constant of both PSD functions.

4.4 Discrete Phase Screens

So far the phase screen was considered to be continues of nature. Both in the plane of the phase, but also in the direction of propagation. Let's first consider discretization the direction of propagation. Consider the light wave to be travelling through a series of Phase Screen that distort the phase of the incoming wavefront. Equation 51 [2] introduces the series of phase screens. Consider it to have have different independent stages of travelling of the wave trough different aberrations.

$$\sum_{i=0}^{n} C_n^2 z_i^m \Delta z_i \tag{51}$$

Now the phase plane will need to be dicretized as well for an implementation in code. Consider the Fourier of the phase screen in discrete form in equation 52.

$$\phi(x,y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} c_{m,n} e^{2\pi j(f_{xn}x + f_{ym}y)} \tag{52}$$

The term $c_{m,n}$ represents the Fourier weights. As the atmosphere was modelled as stochastic process, these weights will need to satisfy statistics of the atmosphere. For that the realization of the Fourier coefficients is drawn from a normal or Gaussian distribution. The draws are spatially independent from each other. Similar how white noise in time domain is drawn independent with respect to other time instance but itself. Since $c_{m,n}$ is in general a complex constant, both the real and complex part will need to be drawn from a normal distribution. In equation 53 a random phase screen is a every spatial frequency point f. It is then weighted with the PSD simply by multiplication. This feature can be exploited as the numbers are drawn from the normal distribution $N(0,1)$ in Fourier domain, rather then doing a convolution.

$$\mathcal{F}\{\phi(x,y)\} = (N(0,1) + iN(0,1))\,(f)\sqrt{\Phi_\phi(\mathbf{r})}\Delta f \tag{53}$$

The transformation back to the real spatial domain can be done efficiently by a Fast Fourier Transform. However practical problems arise when this method is used in a coding environment. Since every sample is taken at complete random in Fourier domain, discrepancies arise in low spatial frequency due to spectral leakage [10]. Using the equation of the phase structure function of equation 46: in low frequency, that is in high $\frac{|\Delta \mathbf{r}}{r_0}|$, it will be lower in simulation with respect to the theory proposed in high frequency.

Many solutions have been proposed to make up for the error arising in the low frequency domain. Some have tried to introduce random draws from Zernike modes as a compensation. This will obviously run into trouble when decomposing in the wavefront in Zernike modes, as one will find the "random" mode directly back during decompression. An alternative method is the "subharmonics" method [2].

The subharmonics method used in the implementation was introduced by Lane *et al.* [10]. It divides a grid point in the spatial frequency domain in consecutively smaller and smaller 3-by-3 grids, like the example of 5

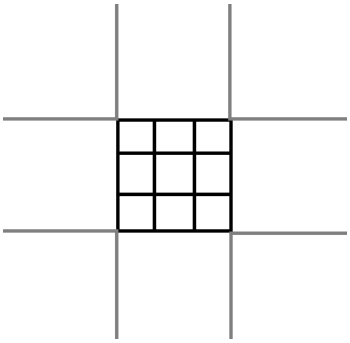

Figure 5: The subharmonics division grid, here only 2 divisions are shown.

For every subdivision, the same method is repeated as was outlined in equation 53. In order to make sure that the power of the signal in all the subdivisions remains in the statics set the the aberration, weights are introduced. The weights are easy to compute, as they must be equal to the area in spatial domain covered. The final phase screen is simply the addition of all the subdivision and the "high frequency" phase screen from equation 53. Equation 54 proposes the used format.

$$\mathcal{F}\{\phi(x,y)\} = \sum_{p=1}^{N_p} \sum_{f_x=-\infty}^{\infty} \sum_{f_y=-\infty}^{\infty} c_p \left(N(0,1) + iN(0,1)\right) \left(f_{x,p}, f_{y,p}\right) \sqrt{\Phi_\phi(\mathbf{r})} \Delta f_p \tag{54}$$

For practical reasons, p, f_x and f_y are finite. In the code p is limited to three and the sample grid is picked by the user. Note that there is no reason to take lower frequencies into account then the sample length of the data set. On the other side, there is no reason to take into account frequencies above the Nyquist criterion. Furthermore, the choice of r_0, for von Kármán also l_0 and L_0, must be outside of the sample length and Nyquist constraints. In figure 6 a realization of a Kolmogorov phase aberration.

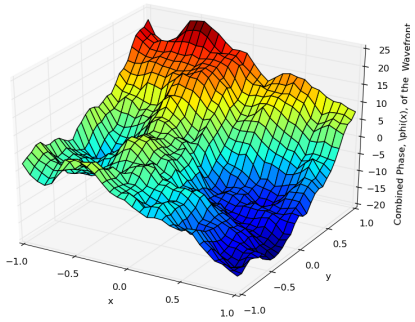

Figure 6: A realization of a Kolmogorov phase aberration, here with $r_0 = 0.1$, $p = 3$ and $N_x, N_y = 30$.

To conclude, Kolmogorov, von Kármán and Zernike are created in a similar manner. The final wavefront is simply a addition of both aberrations. This will not be discussed further.

5 WFG in Python

This is a manual is intended as reference guide to show around the wavefront generation code of Adaptive Optics in Python. This manual will comprise of a description of the classes together with their methods and functions. A new page is made for every function, for readability, due to the many indentations, consider this when printing!

As of current, the WFG part includes two classes. One class on the Zernike aberrations. The second class is on the phase screens of the the wavefront, as for example Kolmogorov behaviour.

5.1 Zernike: Coding Example

Coding Example: This code creates a wavefront consisting of three Zernike modes 2,4 and 21, with the corresponding weights 0.5, 0.25 and -0.6. The wavefornt is plotted and then decomposed. The array (vector) `a`, should return 0.5, 0.25 and -0.6! The computation `numpy.dot(numpy.transpose(z),z)` should return near identity as was described in the text earlier.

```
zw = ZernikeWave()
zw.addMode([2,4,21], [.5,.25,-.6])
W = zw.createWavefront(40,40)
zw.plotWavefront(40,40)
z,a = zw.decomposeWavefront(W)
```

5.2 Zernike functions

`Z = zernike(rho, theta,u,v = None)`

Arguments

`rho`
: Meshgrid array or scalar containing the distance from the optical origin r.

`theta`
: Meshgrid array or scalar containing the cylindrical radial angle θ.

`u`
: u mode index of the Zernike mapping, if v is omitted, this is i in Noll's mapping.

`v`
: v mode index of the Zernike mapping, this can be omitted if Noll's mapping is desired.

Returns

`Z`
: The (u, v) or i Zernike mode, on the grid points defined in `rho` and `theta`.

`u,v = zernikeIndex(i)`

Arguments

`i`
: Noll's mapping index

Returns

`u`
: u of the Zernike mapping

`v`
: v of the Zernike mapping

5.3 ZernikeWave Class: Methods and Functions

`zw = ZernikeWave()`

Arguments

`None`

Returns

`zw` An instance of the ZernikeWave class

`addMode(i, alpha_i = 1)`

Arguments

`i`
: Array or scalar containing the Zernike modes in Noll's mapping to add

`alpha_i`
: Array or scalar containing the weight of the modes specified, if omitted, a weight of one will be assumed. Ideal for decomposition, when the weighting is not important.

Returns

`None`

`changeModeWeight(iModeToChange,alpha_i = 1)`

Arguments

`i`
: Array or scalar containing the Zernike modes in Noll's mapping to add

`alpha_i`
: Array or scalar containing the weight of the modes specified, if omitted, a weight of one will be assumed. Ideal for decomposition, when the weighting is not important.

Returns

`None`

`removeMode(iToRemove)`

Arguments

`iToRemove`
: Array or scalar containing the modes to remove.

Returns

`None`

`exists = modeExists(i)`

Arguments

`i` Scalar of Noll's index, to check.

Returns

`exists` Returns 1 (True) if mode is existent, else 0 (False).

```
i = getModes()
```

Arguments

None

Returns

`i` List or scalar, with the modes specified in this instance.

`alpha_i = getWeights()`

Arguments

`None`

Returns

`alpha_i` List or scalar, with the weights of the modes specified in this instance.

`W = createWavefront(nX,nY)`

Arguments

`nX`
: Number of elements in x-direction of the grid

`nY`
: Number of elements in y-direction of the grid

Returns

`W` The created wavefront, containing the zernike weights specified in the instance, on the grid elements specified, the spatial domain is in the unity circle.

```
Z,A = decomposeWavefront(W)
```

Arguments

`W`
: The wavefront to decompose

Returns

`Z` The constructed basis matrix $\mathcal{Z}$, can be used to check identity of $\mathcal{Z}^T\mathcal{Z}$ to determine the validity of the decomposition.

`A` The computed weights, according to the weights sepified in the instance of the ZernikeWave class.

`plotMode(iToPlot,nX,nY)`

Arguments

`iToPlot` The mode in Noll's index to plot, can be used for checking intermediate results of the construction.

`nX`
Number of elements in x-direction of the grid

`nY`
Number of elements in y-direction of the grid

Returns

`None`

`plotWavefront(nX,nY)`

Arguments

`nX`
: Number of elements in x-direction of the grid

`nY`
: Number of elements in y-direction of the grid

Returns

`None`

5.4 PhaseScreen Class: Example

Coding Example: Create a Kolmogorov phase screen with $r_0 = 0.1$. The other parameters l_0 and L_0 are specified but not used. They can be deleted. Also consider trying to replace 'Kolmogorov' by 'vonKarman' for a von Kármán pahse screen. The phase screen is then created and plotted. `ps = PhaseScreen()` `ps.setType('Kolmogorov')` `ps.setParams({'r0' : .1, 'l0' : .01, 'L0' : 100})` `W = ps.createWavefront(40,40)` `ps.plotWavefront(40,40)`

5.5 PhaseScreen Class: Methods and Functions

`ps = PhaseScreen()`

Arguments

`None`

Returns

`ps` An instance of the PhaseScreen class

`setType(phaseScreenType = 'Kolmogorov'`

Arguments

`phaseScreenType` A string containing the name of the phase screen to use, as of current, 'Kolmogorov' or 'vonKarman' can be used. If omitted, a Kolmogorov phase screen is assumed.

Returns

`None`

`screenType = getType()`

Arguments

`None`

Returns

`screenType` A string containing the name of the used phase screen.

`setParams(parameters)`

Arguments

`screenType` A 'dict' data type containing the the parameters of the phase screen. For Kolmogorov, 'r0' is used, for 'von Karman', 'r0', 'l0' and 'L0' is used. Use this format, as an example: `'r0': 0.1, 'l0':0.1, 'L0' : 100`. If more parameters are specified then necessary, the unused are omitted. If to few are specified the code throws an error.

Returns

`None`

```
params = getParameters()
```

Arguments

`None`

Returns

`params` A 'dict' datatype containing the parameters used by the specific type set-up. The returned format is similar to `setParameters(...)` and dependant on the set type: 'Kolmogorov' or 'von Karman'.

`W = createWavefront(nX,nY)`

Arguments

`nX`
: Number of elements in x-direction of the grid

`nY`
: Number of elements in y-direction of the grid

Returns

`W` The created wavefront, containing the set type screen specified in the instance, on the grid elements specified, the spatial domain is in the unity rectangle.

```
phi = kolmogorov(N, r0, D = 2.)
```

Arguments

`N`
: Number of elements in x, y-direction of the grid.

`r0`
: The Fried parameter r_0, the caracteristic eddy or turbulent bubble size.

`D`
: The range on which to create the wavefront. By default, it is plotted on teh unity rechtangle, so the rectangle edge length D is 2, if anything else is desired, consider replacement.

Returns

`W` The created wavefront, constructed via the subharmonics method, containing the set type screen specified in the instance, on the grid elements specified, the spatial domain is in the unity rectangle.

```
phi = vonkarman(N, r0, l0, L0, D = 2)
```

Arguments

`N`
: Number of elements in x, y-direction of the grid.

`r0`
: The Fried parameter r_0, the caracteristic eddy or turbulent bubble size.

`l0`
: The von Karman small length scale.

`L0`
: The von Karman large length scale.

`D`
: The range on which to create the wavefront. By default, it is plotted on teh unity rechtangle, so the rectangle edge length D is 2, if anything else is desired, consider replacement.

Returns

`W` The created wavefront, constructed via the subharmonics method, containing the set type screen specified in the instance, on the grid elements specified, the spatial domain is in the unity rectangle.

`phi = hfkolmogorov(N, r0, D = 2.)`

Arguments

`N`
: Number of elements in x, y-direction of the grid.

`r0`
: The Fried parameter r_0, the caracteristic eddy or turbulent bubble size.

`D`
: The range on which to create the wavefront. By default, it is plotted on teh unity rechtangle, so the rectangle edge length D is 2, if anything else is desired, consider replacement.

Returns

`W` The created wavefront, containing the set type screen specified in the instance, on the grid elements specified, the spatial domain is in the unity rectangle.

`phi = hfvonkarman(N, r0, l0, L0, D = 2)`

Arguments

`N`
: Number of elements in x, y-direction of the grid.

`r0`
: The Fried parameter r_0, the caracteristic eddy or turbulent bubble size.

`l0`
: The von Karman small length scale.

`L0`
: The von Karman large length scale.

`D`
: The range on which to create the wavefront. By default, it is plotted on teh unity rechtangle, so the rectangle edge length D is 2, if anything else is desired, consider replacement.

Returns

`W` The created wavefront, containing the set type screen specified in the instance, on the grid elements specified, the spatial domain is in the unity rectangle.

`W = plotWavefront(nX,nY)`

Arguments

`nX`
: Number of elements in x-direction of the grid

`nY`
: Number of elements in y-direction of the grid

Returns

`None`

5.6 Support Functions

The following functions are used inside the phase screen and zernike classes to provide some services that show up more often.

`f = kroneckerDelta(x)`

Arguments

`x` The input argument.

Returns

`f` Dirac delta function, ∞ on $x = 0$, zero otherwise.

```
f = gamma2(n)
```

Arguments

`x` The input argument.

Returns

`f` Factorial or gamma function, ∞ for $x < 0$, else it returns the factorial. Recall: Gamma(n+1) = $n!$, currently a lazy implementation, without the need to add 1 to the argument. The function does NOT solve $int_0^{inf} = t^{n-1} * e^{-t} dt$. It only extends validity beyond $n < 0$. The result is only guaranteed for integer arguments.

`R,Theta = cart2pol(X,Y)`

Arguments

`X` The position on the x axis in meshgrid format.

`X` The position on the y axis in meshgrid format.

Returns

`R` The position on the r axis in meshgrid format.

`Theta` The position on the θ axis in meshgrid format.

`X,Y = pol2cart(R,Theta`

Arguments

`R` The position on the r axis in meshgrid format.

`Theta` The position on the θ axis in meshgrid format.

Returns

`X` The position on the x axis in meshgrid format.

`X` The position on the y axis in meshgrid format.

`X,Y = createGrid(nX, nY,rangeX=[-1,1],rangeY=[-1,1])`

Arguments

`nX` Distance between grid point on the x-axis.

`nY` Distance between grid point on the y-axis.

`rangeX` The x range where the grid is defined, assumes a unity rectangle by default.

`rangeY` The y range where the grid is defined, assumes a unity rectangle by default.

Returns

`f` Returns a meshgrid, with a specified number of elements in both x an y direction. Additionally, the range can be specified, if the ranges are not specified, it is assumed to be in [-1,1]

```
inCircle = circ(X,Y,radius = 1.,datatype = 'bool')
```

Arguments

`X` The position on the x axis in meshgrid format.

`X` The position on the y axis in meshgrid format.

`radius` The radius to define the aperture radius with. Set to one by default.

`datatype` The datatype to return. Set to boolean by default.

Returns

`f` Returns a circular aperture based on the grid X and Y in the specified datatype. The radius is assumed at one, if it is not specified. Whether to include the boundary of the circle can be a topic of discussion.

6 Appendix: Maxwell Equations

6.1 Obtaining the Wave-Equation

Maxwell equations provide insight of the behaviour of light a continues wave, that propagates through a medium. In the case of a homogeneous, non-dispersive medium in absence of charges, Maxwell's equations reduce to a neat wave equation.

To get to the wave description of the Maxwell equations, start with the orthogonality between the electric field and the magnetic field, as decribed by Faraday's law. First take the curl of Faraday's law, by multiplying it with the gradient operator in 55 on both sides. The electric field is described by the vector E, the magnetic field is represented by B. Both μ and ϵ are material dependant constants.

$$\nabla \times (\nabla \times E) = -\frac{\partial}{\partial t}(\nabla \times B) \tag{55}$$

Now consider Ampère's law below in equation 56.

$$\nabla \times B = \mu \left(J + \epsilon \frac{\partial E}{\partial t} \right) \tag{56}$$

Then substitute Ampère's law in Faraday's law. Further, since no charge was assumed, the charge density J can be set to zero, as below in 57.

$$\nabla \times (\nabla \times E) = -\mu\epsilon \frac{\partial^2}{\partial t^2} E \tag{57}$$

Now consider the vector identity from equation 58.

$$\nabla \times (\nabla \times E) = \nabla(\nabla \cdot E) - \nabla^2 E \tag{58}$$

The reason for introducing the identity in equation 58 becomes obvious when considering Gauss' law below in equation 59. Now note that the gradient of the scalar $\nabla \cdot E$ is zero.

$$\nabla \cdot E = \frac{\rho}{\epsilon} \tag{59}$$

Thus reducing to equation 60 below. It is now apparent that the equation is second order time dependant differential equation.

$$\nabla^2 E - \mu\epsilon \frac{\partial^2}{\partial t^2} E = 0 \tag{60}$$

Similar manipulations can also be done for the magnetic field, start here with taking the curl of Ampère's law and substituting Faraday's law into it. The wave equation for the magnetic field is described in equation 61.

$$\nabla^2 B - \mu\epsilon \frac{\partial^2}{\partial t^2} B = 0 \tag{61}$$

For Cartesian coordinates, the Laplacians $\nabla^2 E$ and $\nabla^2 B$ are i independent with respect to the coordinates. Giving rise to equation with the intensity vector $U(r,t)$ in equation 62

$$\left(\nabla^2 - \mu\epsilon \frac{\partial^2}{\partial t^2} \right) U(r,t) = 0 \tag{62}$$

The common way to solve these second order time dependant wave equations, is by separating the it into a spatial part and a time dependant part. Giving the result as is in equation 63. Note now the constant $\mu\epsilon$ can the repalced by the wave number $k = 2\pi/\lambda$ an the wave propagation speed $c = 1/\sqrt{\mu_0\epsilon_0}$.

(63)

References

[1] *Numerical Simulation of Optical Wave Propagation with Examples in MATLAB.* Society of Photo-Optical Instrumentation Engineers (SPIE), 2010, ch. 5.

[2] A. Bhatia and E. Wolf, "On the circle polynomials of zernike and related orthogonal sets," in *Proc. Cambridge Philos. Soc*, vol. 50, no. 1. Cambridge Univ Press, 1954, pp. 40–48.

[3] *Control for High Resolution Imaging.* Delft University of Technology, 2014, ch. 2.

[4] *Linear Algebra And It's Applications.* Pearson Education Inc., 2003.

[5] *Numerical Simulation of Optical Wave Propagation with Examples in MATLAB.* Society of Photo-Optical Instrumentation Engineers (SPIE), 2010.

[6] B. Edlén, "The refractive index of air," *Metrologia*, vol. 2, no. 2, p. 71, 1966.

[7] A. N. Kolmogorov, "The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers," in *Dokl. Akad. Nauk SSSR*, vol. 30, no. 4, 1941, pp. 299–303.

[8] R. J. Sasiela, *Electromagnetic wave propagation in turbulence: evaluation and application of Mellin transforms*, 2007.

[9] R. N. Tubbs, "Lucky exposures: Diffraction limited astronomical imaging through the atmosphere," *arXiv preprint astro-ph/0311481*, 2003.

[10] R. Lane, A. Glindemann, J. Dainty *et al.*, "Simulation of a kolmogorov phase screen," *Waves in random media*, vol. 2, no. 3, pp. 209–224, 1992.