

# Efficient Cartesian representation of Zernike polynomials in computer memory

Hedser van Brug

Delft University of Technology, Dept. of Applied Physics  
Optics Research Group, Lorentzweg 1, NL-2628 CJ Delft, the Netherlands

## ABSTRACT

The use of Zernike polynomials, in their cartesian form, as a basis for wave front fitting in shearography, has been enabled by an efficient representation in computer memory. This method will be presented and applications of a portable computer program employing this method will be discussed. The value of this program for educational purposes will be indicated.

## 1. INTRODUCTION

Despite the fact that the Zernike polynomials are already used in many areas, their use in wave front fitting for interferometry is restricted to non shearing interferometers. This is due to the fact that their Cartesian description is somewhat awkward, see Sec.2.

This article will describe a new method to represent Zernike polynomials in computer memory, as well as the implementation of this method into a C++ class. The main goal of the creation of this class was to ease the use of Zernike polynomials within a computer program. The class contains several useful methods to handle the polynomials; e.g. creating Zernike polynomials, adding several polynomials together and the calculation of one or several points within a unit pupil given by the sum of Zernike polynomials under investigation. A command driven computer program, based on this implementation will be discussed. Examples of the use of this program for educational purposes will be given. The here presented efficient method to calculate a large number of Zernike polynomials simultaneously allows the use of these polynomials in lateral shearing interferometry.<sup>1</sup>

An alternative method for the use of Zernike polynomials is a data base describing all function, in combination with a parser that interprets the ascii-strings and that casts them into a computer processor understandable format. Since it is impossible in this method to add the polynomials efficiently prior to calculation of e.g. phase distributions, it is much slower than the here presented method.

Owing to the simplicity of the presented method, it is fast enough to make programs based upon it suitable for educational use, such as for instance MTF calculations as a function of the aberrations in the wave front, and the expansion of the Seidel and higher order aberration in terms of Zernike polynomials.

## 2. ZERNIKE POLYNOMIALS

The Zernike polynomials, or short Zernikes, are often used to describe the wave front in a circular pupil. This is mainly because the Zernikes form a complete set of functions that are orthogonal, such that each wave front can uniquely be described by the set of Zernikes. Another strong property of Zernikes is that the effect of aberrations added together, e.g. spherical aberration and defocus, such that the negative effect of one of the aberrations on the image formation is balanced by the other, comes out to be equal to a single Zernike polynomial. Each Seidel aberration on its own is in general a combination of several Zernike polynomials.<sup>2</sup>

The Zernike polynomials are identified by two integers,<sup>3</sup> often  $n$  and  $m$ , where  $n \geq 0$  and  $0 \leq m \leq n$ . A single integer number is also widely used. The different numbering schemes will be elaborated upon.

**Table 1.** Zernike polynomials, the different numbering schemes, their representation in both Cartesian coordinates  $(x, y)$  and polar coordinates  $(\rho = \sqrt{x^2 + y^2}, \theta = \arctan(y/x))$ , plus an example of how they are represented in computer memory within the C++ class CZernike.

$i$	$n$	$m$	$l$ $(n - 2m)$	$Z_{n,m}(\rho, \theta)$ polar	$Z_{n,m}(x, y)$ cartesian	fMat
1	0	0	0	1	1	(1)
2	1	1	-1	$\rho \cos(\theta)$	$x$	$\begin{pmatrix} 0 & 0 \\ 1 & - \end{pmatrix}$
3	1	0	1	$\rho \sin(\theta)$	$y$	$\begin{pmatrix} 0 & 1 \\ 0 & - \end{pmatrix}$
4	2	2	-2	$\rho^2 \cos(2\theta)$	$x^2 - y^2$	$\begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & - \\ 1 & - & - \end{pmatrix}$
5	2	1	0	$2\rho^2 - 1$	$-1 + 2x^2 + 2y^2$	$\begin{pmatrix} -1 & 0 & 2 \\ 0 & 0 & - \\ 2 & - & - \end{pmatrix}$
6	2	0	2	$\rho^2 \sin(2\theta)$	$2xy$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & - \\ 0 & - & - \end{pmatrix}$
7	3	3	-3	$\rho^3 \cos(3\theta)$	$x^3 - 3xy^2$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & - \\ 0 & 0 & - & - \\ 1 & - & - & - \end{pmatrix}$
8	3	2	-1	$(3\rho^3 - 2\rho) \cos(\theta)$	$3x^3 + 3xy^2 - 2x$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ -2 & 0 & 3 & - \\ 0 & 0 & - & - \\ 3 & - & - & - \end{pmatrix}$
9	3	1	1	$(3\rho^3 - 2\rho) \sin(\theta)$	$3y^3 + 3x^2y - 2y$	$\begin{pmatrix} 0 & -2 & 0 & 3 \\ 0 & 0 & 0 & - \\ 0 & 3 & - & - \\ 0 & - & - & - \end{pmatrix}$
10	3	0	3	$\rho^3 \sin(3\theta)$	$-y^3 + 3x^2y$	$\begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & - \\ 0 & 3 & - & - \\ 0 & - & - & - \end{pmatrix}$

In this section the definition of the Zernikes will be described, both in polar and cartesian representation. Interferograms are often recorded by means of CCD cameras, that have rectangular pixels, such that the use of Cartesian coordinates is preferred.

## 2.1. Zernike Definitions

A wave front, if expanded up to the power  $k$  into Zernike polynomials, using polar coordinates  $(\rho, \theta)$ , can be given by

$$\mathcal{W}(\rho, \theta) = \sum_{n=0}^k \sum_{m=0}^n A_{nm} U_{nm}(\rho, \theta) = \sum_{n=0}^k \sum_{m=0}^n A_{nm} R_n^l(\rho) \begin{cases} \sin(l\theta) \\ \cos(l\theta) \end{cases}, \quad (1)$$

with

$$l = n - 2m, \quad (2)$$

where the sine function is used for  $l > 0$ , and the cosine function is used for  $l \leq 0$ . The factor  $A_{nm}$  is the coefficient for the Zernike polynomial  $U_{nm}$  in the expansion of the wave front. The integer value of  $l$  is therefore limited to values between  $-n$  and  $+n$  where  $n - l$  has to be even valued.

For a given combination of  $n$  and  $m$ , the function  $R_n^l(\rho)$  is given by

$$R_n^l(\rho) = R_n^{-l}(\rho) = R_n^{|l|}(\rho) = \sum_{s=0}^m (-1)^s \frac{(n-s)!}{s!(m-s)!(n-m-s)!} \rho^{n-2s}, \quad (3)$$

where  $l$  is found from Eq.(2), after which  $m$  has to be calculated from

$$m = \frac{n - |l|}{2}, \quad (4)$$

since the sign of  $l$  might be changed due to taking the absolute value of it. In Column 5 of Table 1, the Zernike polynomials are given in polar coordinates.

The wave front function  $\mathcal{W}$  is often written in terms of monomials, that is, powers of  $x$  and  $y$ , the cartesian representation. In Column 6 of Table 1 the Zernike polynomials are given in cartesian coordinates. For the cartesian representation we use the definitions

$$\begin{aligned} x &= \rho \cos(\theta) \\ y &= \rho \sin(\theta). \end{aligned} \quad (5)$$

Rewriting the function  $U_{nm}$  in terms of powers of  $x$  and  $y$  yields

$$\begin{aligned} U_{nm} &= R_n^{|l|} \begin{cases} \sin(l\theta) \\ \cos(l\theta) \end{cases} \\ &= \sum_{i=0}^q \sum_{j=0}^m (-1)^{i+j} \sum_{k=0}^{m-j} \binom{|l|}{2i+p} \binom{m-j}{k} \\ &\quad \times \frac{(n-j)!}{j!(m-j)!(n-m-j)!} x^{n-2(i+j+k)-p} y^{2(i+k)+p}, \end{aligned} \quad (6)$$

where  $m$  is again obtained from Eq.(4),  $p = 1$  for  $l > 0$  and zero otherwise, and

$$q = \begin{cases} |l|/2 - 1 & \text{for } n \text{ even and } l > 0 \\ |l|/2 & \text{for } n \text{ even and } l \leq 0 \\ (|l| - 1)/2 & \text{for } n \text{ odd.} \end{cases} \quad (7)$$

The Zernikes can be calculated with the C++ program as given in Listing.1. The format of the matrix `fMat` is such that the coefficients  $a_{i,j}$  for each monomial  $x^i y^j$  are placed in matrix element `fMat[i][j]`. For example, in the case that  $n = 5$ , the matrix is filled like

	1	$y$	$y^2$	$y^3$	$y^4$	$y^5$
1	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$x$	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	0
$x^2$	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	0	0
$x^3$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	0	0	0
$x^4$	$a_{4,0}$	$a_{4,1}$	0	0	0	0
$x^5$	$a_{5,0}$	0	0	0	0	0

(8)

where the rows  $i$  and the columns  $j$  are labelled with the corresponding  $x^i$  and  $y^j$  components of the monomials, raised to the proper power. All coefficients, for which the sum of  $i$  and  $j$  exceeds  $n$ , are by definition zero.

---

**Listing 1** Excerpt from CZernike.cp, a C++ class to calculate Zernike polynomials

---

```
void Zernike::CalcZern( int n, int m, float scaling) {
int p, q, l= n-2*m, xpow, ypow;
float factor;

if ( l <= 0 ) {
    p = 0;
    q = ( n%2==0 ? -1/2 : (-l-1)/2 );
} else {
    p = 1;
    q = ( n%2==0 ? 1/2 - 1: (l-1)/2 );
}
l = (l<0 ? -1 : 1 );    // now l is positive
m = (n-1)/2;          // m might have changed
for ( int i=0; i<=q; i++ ) {
    for ( int j=0; j<=m; j++ ) {
        for ( int k=0; k<=(m-j); k++ ) {
            factor = ( (i+j)%2==0 ? 1 : -1 );
            factor *= Binom( l, 2*i+p );
            factor *= Binom( m-j, k );
            factor *= Fact(n-j)/( Fact(j) * Fact(m-j) * Fact(n-m-j) );
            ypow = 2 * (i+k) + p;
            xpow = n - 2 * (i+j+k) - p;
            fMat[xpow][ypow] += factor*scaling;
        }
    }
}
}
```

---

The `fMat` matrices, as created in the class `CZernike.cp`, are given in Column 7 of Table 1. The fact that the sum of the powers of  $x$  and  $y$  should remain below or equal to  $n$  is here indicated by placing the symbol ‘-’ at those coefficient positions that can not occur. During the initialisation of the matrix `fMat` all locations are set to zero since that is required to enable the addition of Zernikes.

The numbering scheme where only a single integer is used to indicate the Zernike polynomial is directly linked to the here adopted  $n$  and  $m$  scheme. From the integer value  $i$ , where  $i > 0$ ,  $n$  can be obtained by using

$$n = (\text{int})\text{ceil}(-1.5 + 0.5 * \text{sqrt}(1 + i * 8)), \quad (9)$$

where the ‘ceil’ function takes the nearest integer value higher than the value returned by its argument, and  $m$  by using

$$m = n - i + (n + 1) * n / 2. \quad (10)$$

The different numbers,  $\{i, n, m, l\}$ , are include in Table 1.

### 3. CZERNIKE

Based upon the here presented method, to store the Zernikes efficiently in computer memory, a C++ class was created to handle Zernikes in a simple manner. This class contains methods for the creation of a new Zernike, to add other Zernikes to it, to calculate the value for a given position  $(x, y)$  within the unit pupil, and others.

The figures presented in Table 2 were all created by using this C++ class. In this table the phase distributions are presented, in 8 bit gray-scale, over a unit pupil, where the phase is given by the Zernike polynomials ranging from zeroth to fifth order. Underneath the phase distribution the corresponding function in monomial presentation is included. For all of these figures the amplitude was set to unity.

In order to be able to use the CZernike class, first CZernike.h has to be included, by

```
#include ``CZernike.h``
```

Once this file has been included, a Zernike polynomial has to be declared. This can be done using

```
Zernike myZern;
```

At this time myZern is only a placeholder for a Zernike polynomial. To actually initialise and display a polynomial the following has to be executed

```
myZern.NewZern(3,2,1.3);
```

for the creation of the Zernike polynomial, and then

```
cout << myZern;
```

will output the polynomial to the console. As a result of this short piece of program, the Zernike polynomial with  $n = 3$  and  $m = 2$  is created, having an amplitude of 1.3. The output following `cout << myZern;` was:  $-2.6 + 3.9xy^2 + 3.9x^3$ . In the following all methods defined in the CZernike class will be described.

### 3.1. Description of the methods

`NewZern(n, m, scaling)` This method creates a new Zernike polynomial in the form of the matrix 'fMat'.

If an older polynomial already exists it will first be deleted. The input variables are the  $n$  and  $m$  indices that define the Zernike polynomial, and the amplitude that is used to scale the polynomial.

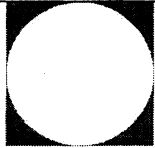
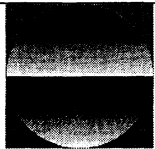
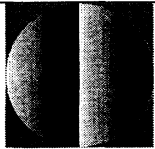
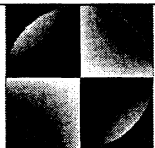
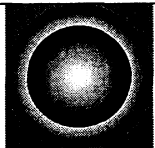

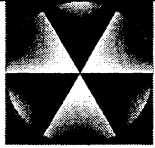
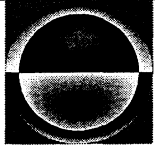
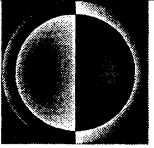
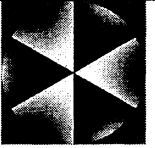
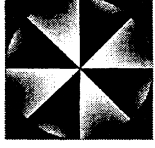
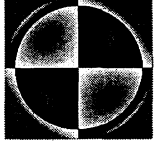
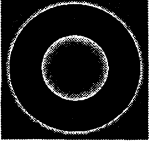
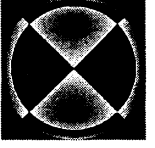
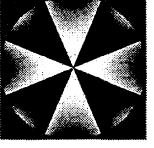
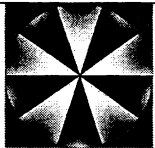
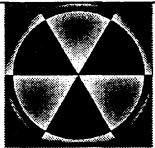
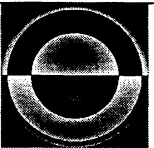
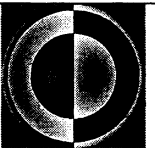
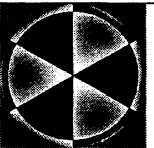
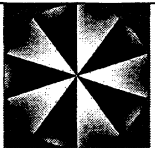
`AddZern(n, m, scaling)` Basically identical to the 'NewZernike' method, except that now the components in the polynomial are added to the polynomial already contained in the 'fMat' pertaining to myZern. In the case that the polynomial to be added is of a higher order than the polynomials already present in the matrix fMat, the dimensions of fMat are adjusted.

`NewZern(i, scaling)` If the 'NewZernike' method is called with only two arguments, the first argument  $i$  is used to calculate the variables  $n$  and  $m$  by subsequently Eqs.(9) and (10). Then the method with three arguments is called.

`AddZern(i, scaling)` Identical to the 'NewZernike' method with two arguments, but now to add a polynomial to the existing one.

`NewMonomial(xpow, ypow, scaling)` This method, combined with the following method, can be used to fill an fMat matrix manually as can be useful for the fitting of unknown phase fronts with Zernike polynomials. The size of the fMat matrix is set to  $(fOrder + 1) \times (fOrder + 1)$  where fOrder is equal to the sum of the powers of  $x$  and  $y$  in the monomial, that are given in the input variables xpow and ypow. After a call to this method a single monomial is defined (`fMat[xpow][ypow]=scaling`). Calls to the AddMonomial method are required to add other monomials to the matrix fMat.

Table 2. Zernike polynomials up to the 5<sup>th</sup> order

$n$	$m = 0$	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$
0	 1					
1	 $y$	 $x$				
2	 $2xy$	 $-1 + 2x^2 + 2y^2$	 $x^2 - y^2$			
3	 $-y^3 + 3x^2y$	 $3y^3 + 3x^2y - 2y$	 $3x^3 + 3xy^2 - 2x$	 $x^3 - 3xy^2$		
4	 $4x^3y - 4xy^3$	 $8x^3y + 8xy^3 - 6xy$	 $6x^4 + 6y^4 - 6x^2 - 6y^2 + 12x^2y^2 + 1$	 $4x^4 - 4y^4 - 3x^2 + 3y^2$	 $x^4 + y^4 - 6x^2y^2$	
5	 $y^5 - 10x^2y^3 + 5x^4y$	 $15x^4y + 10x^2y^3 - 12x^2y - 5y^5 + 4y^3$	 $10x^4y + 20x^2y^3 - 12x^2y + 10y^5 - 12y^3 + 3y$	 $10x^5 + 20x^3y^2 - 12x^3 + 10xy^4 - 12xy^2 + 3x$	 $5x^5 - 10x^3y^2 - 4x^3 - 15xy^4 + 12xy^2$	 $x^5 - 10x^3y^2 + 5xy^4$

`AddMonomial(xpow,ypow,scaling)` This will add a monomial to the existing `fMat`. If the size has to be changed, i.e. when the sum of `xpow` and `ypow` exceeds `fOrder`, the matrix `fMat` is adjusted to this new size, after which the new monomial is added. See also the description of `NewZern`.

`MTF(NPTS, filename)` This method calculates the modulation transfer function as would be pertaining to a system having the given wave front distribution `fMat` in its exit pupil. The integer value 'NPTS' gives the number of points wanted in the output, i.e. it determines the step size between the zero frequency and the highest possible frequency. If the `char* filename` is zero, the output will be written to the console, otherwise it will be directed to an output file named `filename`.

`DrawZernike(NPTS, int-phase)` The execution of this method creates an unsigned `char's` array of  $(NPTS) \times (NPTS)$  points. This data type can contain 8 bit values and is therefore well suited to store the gray-scale images. The character variable `int-phase` determines whether the intensity distribution (`int-phase = 'i'`) or the phase distribution (`int-phase = 'p'`) will be calculated. The intensity distribution represents the case where the given phase distribution interferes with a planar wave front. The resulting array of unsigned `char's` can for instance be saved to file using the freeware package `libtif`, or shown on the monitor

`Calc(x,y)` This method calculates the phase at a position  $(x,y)$  inside the unit pupil.

`Calc(xp,yp,NPTS)` This method is identical to the previous one, but it now calculates the phase in `NPTS`  $(x,y)$  points. The points are given in arrays `xp` and `yp`. The output will also be an array, having the length of `NPTS`.

`GetZernike()` This method, requiring no input, returns the matrix containing the coefficients of all the monomials present in the Zernike polynomials.

`GetOrder()` This method, requiring no input, returns the order of the Zernike polynomial.

For a command driven program, based upon the `CZernike` class, one can contact the author for a listing. This program assumes the presence of 'libtif' and the 'numerical recipes' library in C.<sup>4</sup> Apart from functions to use the above given methods, this program contains functions to output either the phase or the intensity distribution to a TIF file, to fit a given input function in terms of Zernike polynomials, and to load a command file, while other functions will be implemented in the near future. Among these are functions to investigate the properties of phase unwrapping algorithms in phase stepped interferometry.

### 3.2. Educational use

As an example of the use of this method we will show how the standard third order Seidel aberrations can influence the Modulation Transfer Function (MTF). The wave front aberrations under consideration are

$$\text{Spherical aberration} : (x^2 + y^2)^2, \quad (11)$$

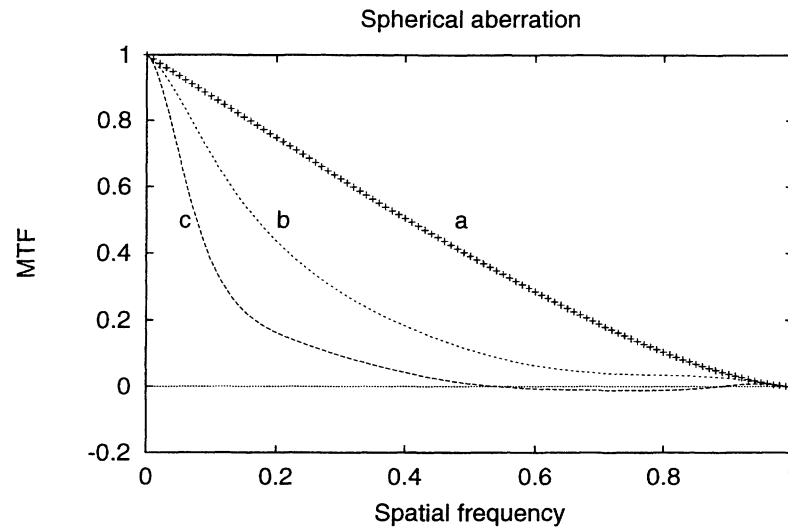
$$\text{Coma} : y(x^2 + y^2), \quad (12)$$

$$\text{Defocus} : x^2 + y^2, \quad (13)$$

$$\text{Astigmatism} : x^2 + 3y^2. \quad (14)$$

These aberrations were inserted into the matrix `fMat` using the methods `NewMonomial` and `AddMonomial`. Each of these aberrations was fitted with Zernike polynomials, and the following expansions were obtained

$$\text{Spherical aberration} : \frac{1}{3}Z_1 + \frac{1}{2}Z_5 + \frac{1}{6}Z_{13}, \quad (15)$$



**Figure 1.** Modulation Transfer Function as a function of the (normalised) spatial frequencies. The lines denote a) diffraction limit, b) spherical aberration scaled by 0.5, and c) spherical aberration with unity scaling.

$$\text{Coma} : \frac{2}{3}Z_3 + \frac{1}{3}Z_9, \quad (16)$$

$$\text{Defocus} : \frac{1}{2}Z_1 + \frac{1}{2}Z_5, \quad (17)$$

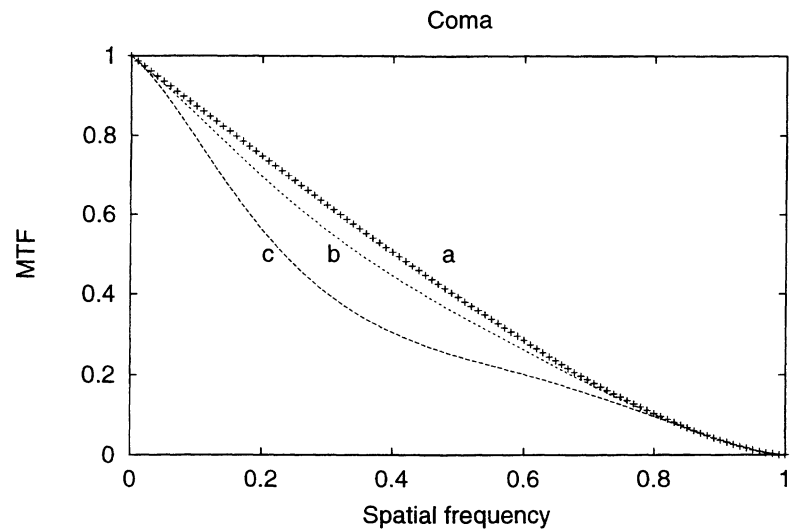
$$\text{Astigmatism} : Z_1 - Z_4 + Z_5, \quad (18)$$

where  $Z_i$  denotes the  $i^{\text{th}}$  Zernike polynomial.

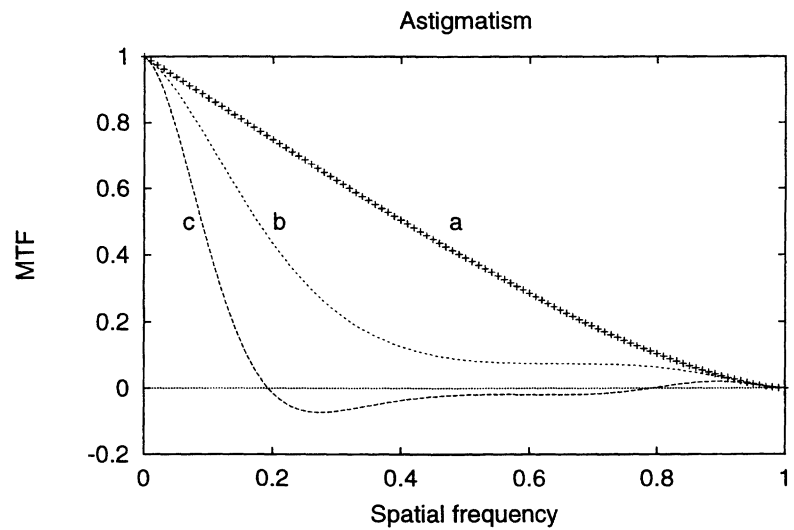
Inspection of the expansions of the Seidel aberrations into Zernike polynomials shows that if defocus is subtracted from spherical aberration, that then only a piston term  $Z_1$  and a, small amplitude, higher order term remains  $\frac{1}{6}Z_{13}$ . Since the piston term does not perturb the wave front, it can be stated that a system suffering from spherical aberration only can be improved by shifting the image plane, i.e. by introducing some defocus. Other examples on aberration balancing can be found in the book of Mahajan.<sup>2</sup> Figures 1–3 show examples of modulation transfer functions for the case that either spherical aberration, coma or astigmatism is present. The MTF for spherical aberration balanced by defocus is presented in Fig.4. The spatial frequency was normalised since the MTF shown here is not restricted to one value of the numerical aperture NA and the wavelength  $\lambda$ . The spatial frequency of '1' corresponds to  $2NA/\lambda$ . The MTF for defocus is not shown since for the MTF along the  $x$ -direction, that MTF equals the MTF of astigmatism. To obtain the MTF along the  $y$ -direction the  $x$  and  $y$  components in the wave front function have to be interchanged, since the present version of the program does not contain a method for MTF-calculations other than along the  $x$ -direction.

The final example for the use of the presented method and program is error analysis in phase stepped interferometry. For a four bucket system the phase steps should be  $\pi/2$  and can be added to a set of Zernikes, stored in a single `fMat` using the `AddZernike` method with the first Zernike (piston) having amplitude  $0.25 + \epsilon$ , where  $\epsilon$  represents the error in the size of the phase step. For each phase step the resulting interferogram can be calculated very rapidly since the description of the phase distribution is calculated only once and inserted into the matrix `fMat`, after which it can be used for all data points in the interferogram. The effect of the size of  $\epsilon$  can be found by using the four simulated images to calculate the phase, and to compare the known phase distribution with the phase distribution fitted to the calculated phase distribution.

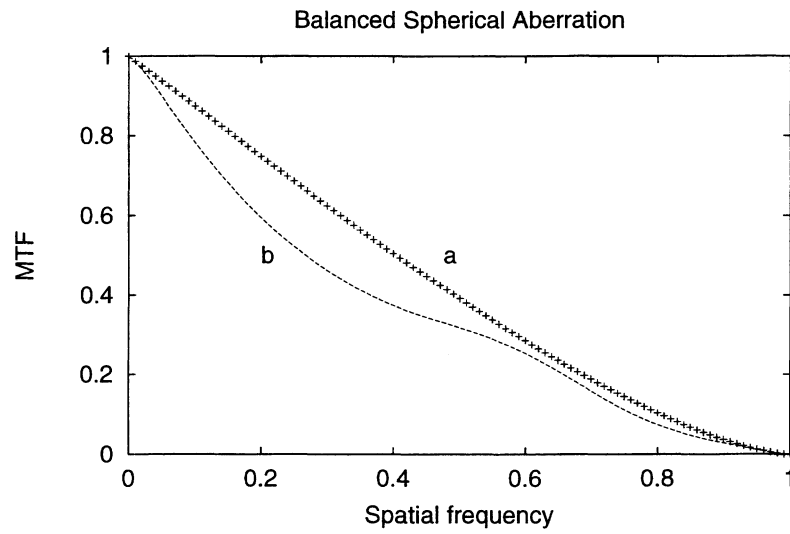




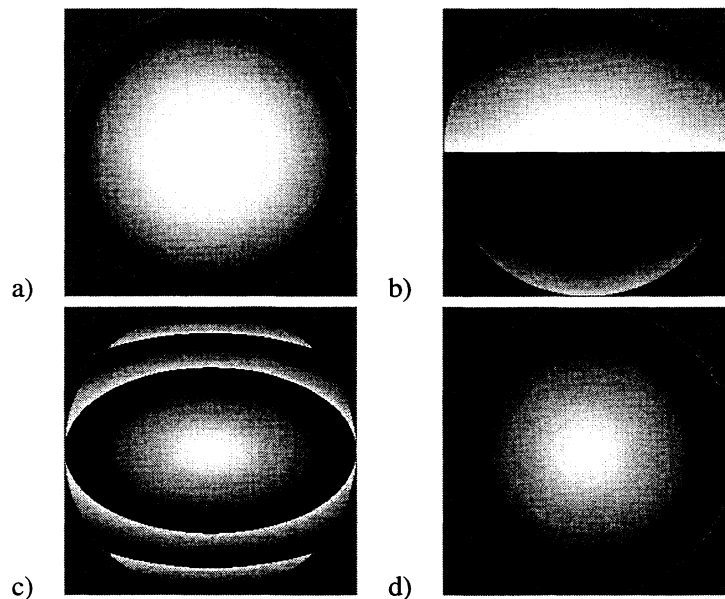
**Figure 2.** Modulation Transfer Function as a function of the (normalised) spatial frequencies. The lines denote a) diffraction limit, b) coma scaled by 0.5, and c) coma with unity scaling.



**Figure 3.** Modulation Transfer Function as a function of the (normalised) spatial frequencies. The lines denote a) diffraction limit, b) astigmatism scaled by 0.5, and c) astigmatism with unity scaling.



**Figure 4.** Modulation Transfer Function as a function of the (normalised) spatial frequencies for a) diffraction limited system and b) spherical aberration balanced by defocus. The amplitude for spherical aberration was '1', and the amplitude for defocus was '-1'.



**Figure 5.** Phase distributions for a) spherical aberration, b) coma, c) astigmatism, and d) defocus. The amplitude for all of these figures was set to unity.

#### 4. CONCLUSIONS

An efficient method of handling Zernike polynomials in computer memory has been presented. The possibilities to use this method within a computer program have been shown with several examples. The first was the fitting of an unknown phase distributions with Zernike polynomials, where for the 'unknown' phase distributions some Seidel aberrations were taken. In the second example the influence of the Seidel aberrations on the modulation transfer function was shown. Finally it was pointed out how the program might be used for error analysis purposes in phase stepped interferometry.

#### REFERENCES

1. H. v. Brug, "Zernike polynomials as basis for wavefront fitting in lateral shearing interferometry," *Applied Optics* **36**, pp. 2788 – 2790, May 1997.
2. V. N. Mahajan, *Aberration Theory Made Simple*, Tutorial texts in optical engineering: v. TT 6, SPIE Optical Engineering Press, 1991.
3. D. Malacara and S. L. DeVore, "Phase Shifting Interferometry", Ch. 13 of "*Optical Shop Testing*", Ed. Daniel Malacara. Wiley series in pure and applied optics, John Wiley & Sons, Inc., New York, 2nd ed., 1992.
4. W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery, *Numerical Recipes in C - the Art of Scientific Computing*, Cambridge University Press, second ed., 1995.