3-2006

# Polar Phase Screens: A Comparison with Other Methods of Random Phase Screen Generation

Rebecca J. Eckert


# Polar Phase Screens: A Comparison with Other Methods of Random Phase Screen Generation

THESIS

Rebecca J. Eckert, 2nd Lieutenant, USAF

AFIT/GE/ENG/06-18

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

The views express in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/06-18

# Polar Phase Screens: A Comparison with Other Methods of Random Phase Screen Generation

THESIS

Presented to the Faculty

Department of Electrical Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Rebecca J. Eckert, B.S.E.E.

2nd Lieutenant, USAF

March 2006

AFIT/GE/ENG/06-18

# Polar Phase Screens: A Comparison with Other Methods of Random Phase Screen Generation

Rebecca J. Eckert, B.S.E.E.
2nd Lieutenant, USAF

Approved:

| | |
|---|---|
| /signed/ | 23 Mar 2006 |
| Lt. Col. M.E. Goda, PhD (Chairman) | date |
| /signed/ | 23 Mar 2006 |
| Dr. S.C. Cain (Member) | date |
| /signed/ | 23 Mar 2006 |
| Dr. M.A. Marciniak (Member) | date |

# *Abstract*

Random phase screens are essential elements of simulating light propagation through turbulent media. In order to be effective, they must accurately reflect theory and be implementable by the user. This document explains and evaluates three methods of generating random phase screens: using a Fourier series upon a polar frequency grid with logarithmic spacing; using the fast Fourier transform, with its cartesian frequency grid; and using Zernike polynomials. It provides a comparison of the polar Fourier series technique with the two more common techniques (fast Fourier transform and Zernike), with the end result of giving the users enough information to choose which method best fits their needs. The evaluation criteria used are generation time (usability) and phase structure function (accuracy).

*Acknowledgements*

First and foremost, I should like to thank my advisor, Col. Goda, for his incite and patience in reviewing my work, answering my questions, and keeping me focused on my goals. I thank my fellow students, who were always willing to help me overcome 'technical difficulties' from spelling errors to learning new programs. I thank my friends and family, especially Lt. Wibisono, who did countless little things that left me free to focus on my research, and Ginger Chezem, for, among other things, being a wonderful proof audience for my defense presentation. Finally, I should like to thank the members of my committee for taking time out of their already full schedules to ensure that my thesis is a quality product.

Rebecca J. Eckert

*Table of Contents*

Page

# *List of Figures*

# *List of Tables*

# *List of Symbols*

| Symbol | | Page |
|---|---|---|
| $l_0$ | Lower Bound of Inertial Subrange | 4 |
| $L_0$ | Upper Bound of Inertial Subrange | 4 |
| $n_0$ | Mean Index of Refraction of Air | 5 |
| $\lambda$ | Opical Wavelength (microns) | 5 |
| $P_{atm}$ | Pressure (millibars) | 5 |
| $T$ | Temperature (Kelvin) | 5 |
| $m_X$ | Mean of X | 7 |
| $R_X$ | Autocorrelation of X | 7 |
| $B_X$ | Autocovariance of X | 7 |
| $\Phi_X$ | Power Spectral Density of X | 7 |
| $P_{ave}$ | Average power of X | 8 |
| $D_n$ | Atmospheric Structure Function | 9 |
| $\Phi_n$ | Atmospheric Power Spectrum | 9 |
| $C_n^2$ | Atmospheric Index of Refraction Structure Constant | 12 |
| $D_\phi(R)$ | Phase Structure Function | 12 |
| $\rho_0$ | Atmospheric Coherence Radius | 12 |
| $r_0$ | Fried Radius | 12 |
| $a_i$ | $i^{th}$ Zernike coefficient | 15 |
| $Z_i$ | $i^{th}$ Zernike polynomial | 15 |
| $W(\vec{r})$ | Windowing function | 15 |
| $\phi(\vec{r})$ | Phase screen | 15 |
| $N_\kappa$ | Number of Frequencies | 27 |
| $Q$ | Polar frequency spacing parameter | 27 |
| $N_\theta$ | Number of arcs in polar grid | 28 |
| $A$ | Area represented by phase screen | 29 |

| Symbol | | Page |
|---|---|---|
| $T_i$ | PS initialization time | 30 |
| $T_s$ | Per screen PS generation time | 30 |
| $N_S$ | Number of Phase Screens | 30 |
| $T_{i-Z}$ | Zernike PS initialization time | 31 |
| $T_{i-FFT}$ | FFT PS initialization time | 31 |
| $T_{i-FS}$ | FS PS initialization time | 31 |
| $S_K$ | Kernel Size | 34 |
| $N_Z$ | Number of Zernike Polynomials | 38 |

# *List of Abbreviations*

| Abbreviation | | Page |
|---|---|---|
| IOR | Index of Refraction | 1 |
| pdf | probability distribution function | 7 |
| PSD | Power Spectral Density | 7 |
| PSD | Power Spectrum (or Spectral) Density | 9 |
| FFT | Fast Fourier Transform | 14 |
| FS | Fourier Series | 14 |
| SHFE | Sub-Harmonic Frequency Expansion | 23 |
| $M$ | Memory Size | 36 |
| KB | Kilobyte | 36 |

# Polar Phase Screens: A Comparison with Other Methods of Random Phase Screen Generation

## I. Overview

Random phase screens are key elements of simulating optical atmospheric turbulence. There several methods of generating random phase screens, three of which (Zernike, Fast Fourier Transform, and Polar Fourier Series) are compared herein. This is the first organized analysis of the benefits and drawbacks of each method using the structure function as a measure of accuracy to theory. The phase screen generation time is used as a measure of usability. Together, the structure function and the generation time allow one to choose the most effective method for a given application.

### *1.1 Why Random Phase Screens?*

Light travelling through the atmosphere (or any turbulent medium) is affected by the random fluctuations of the index of refraction (IOR). Coherent masses of air, called eddies, are characterized by indices of refraction determined by their particular temperature, pressure and humidity. The effect of these eddies upon the light incident upon them is observable as distortion, or even loss, of the signal incident upon the detector. If the optical path were non-random, the system could be modelled and each distortion compensated for, allowing the perfect reconstruction of the source. However, as the optical path is random, complex algorithms are required to statistically model the random data.

Random phase screens are used to model the random IOR of a turbulent medium. They do this by adding a correlated random phase to the propagating

field, resulting in a 'detected' image similar to that which would be the result of actual propagation. More than one screen can be used to simulate a complete optical path; and, because each phase screen compresses the effects of propagation through a three-dimensional volume into a two-dimensional 'screen', they can be placed at any location along the propagation path.

There are several reasons for using more than one phase screen to simulate a propagation. For instance, consider a telescope viewing an object in space. Using only a single phase screen and placing it in the lens of the receiving aperture would entirely negate the effect of scintillation. Scintillation, or fluctuation of signal power, is effected by a phase screens located away from the aperture. Also, typical Fourier phase screen generation techniques have difficulty representing low frequency abberations accurately. However, placing the screen near the source or even equidistant between the source and detector, fails to capture the fact that most of the turbulence is in the first few 100 meters of above an earth based telescope. These difficulties can be overcome by using multiple screens placed at strategic locations along the propagation path. Because the low frequency tip and tilt abberations account for approximately 80% of the system's aberrant power, artificial limitations upon them can greatly decrease the efficacy of the model.

The use of multiple phase screens has some drawbacks. Each screen takes a certain amount of time to generate, depending upon its size and the technique used. Some applications call for the generation of extremely large phase screens (e.g.,the implementation of Taylor's *frozen turbulence* hypothesis to simulate time dependent fluctuations [1, 59]). Using current techniques, the generation of multiple large phase screens can take a prohibitive amount of time.

As phase screens are only a tool used to test the efficacy of what are often cumbersome algorithms, it is important that they be as efficient as possible. Minimizing the generation time and accuracy of phase screens maximizes their usefulness to the researcher.

## *1.2 Thesis Overview*

Chapter two describes the theoretical and mathematical basis for phase screens and their analysis.

Chapter three describes particular methods used to generate and evaluate phase screens, including MATLAB® commands and mathematical formulae.

Chapter four gives the results of the comparative analysis performed using the method of generation and evaluation outlined in chapter three.

Chapter five presents the conclusions expands to possible applications and recommendations.

## *1.3 Contributions*

This document gives an technical overview and evaluation of three methods of generating random phase screens: Zernike, Fast Fourier Transform, and Polar Fourier Series. It provides information for someone using random phase screens to decide which method best fits their needs.

# II. Atmospheric, Mathematical and Simulation Theory

Phase screens are valuable tools in simulating turbulent media (such as the earth's atmosphere) and their effects upon the propagation of light. This chapter will highlight their significance after presenting necessary background material. Section 2.1 gives a rough overview of atmospheric turbulence and how it affects the propagation of light. Section 2.2 discusses random variables and some of their statistics. Random variable are used extensively in optical atmospheric modelling and 2.2 gives and explanation of some of the basic tools used. Section 2.3 expands upon those basic tools by development of the structure function. Section 2.4 covers the most common models of the atmospheric power spectrum, including Kolmogorov's. Section 2.5 uses the structure function and power spectrum to define key atmospheric descriptors, including the Fried radius. Section 2.6 gives an overview of the types of phase screens focused upon in this thesis. Section 2.7 defines Zernike polynomials, their covariance and useful related details. Section 2.8 covers the Fourier transform as it is used in creating phase screens. It also covers the distinctions between the three methods.

## *2.1 Atmospheric Turbulence*

Kolmogorov modelled the atmosphere as a viscous fluid, subject to conditions of turbulent and laminar flow [1, Ch. 3]. The nature of the flow can be characterized by a turbulent intensity using the Reynolds Number [9, p. 58], which increases with the degree of turbulence. Turbulence in the atmosphere is caused by local unstable air masses resulting from convection and wind shear in the atmosphere. These air masses, called *eddies* range in size from small (on the order of millimeters or centimeters) to large (on the order of 100 meters). The smallest an eddy can be and still retain its own distinct refraction characteristics is called the inner scale, denoted $l_0$. Thus air masses smaller than $l_0$ are seen only as elements of larger eddies. The largest and eddy can be and still maintain viscosity, seen as a correlated index of refraction (IOR), is called the outer scale, denoted $L_0$. The range of eddy sizes $l_0$ to $L_0$ is called the

*inertial subrange*, within which there is a continuous transfer of energy as large eddies break down into smaller ones and small eddies eventually dissipate as heat.

The defining characteristic of an eddy is that, within itself, each eddy is homogeneous and isotropic. The *inertial subrange* is the complete range of eddy sizes of interest. Air masses smaller than $l_0$ are assumed not to exist and air masses larger than $L_0$ are no longer assumed homogeneous or isotropic. It should be noted that the inner scale is inversely proportional to turbulence strength, becoming even smaller in strong turbulence, and the outer scale is directly proportional to turbulence strength, becoming larger in strong turbulence.

It is the random formation and dissipation of these eddies that is responsible for the stochastic nature of the atmospheric IOR. While the mean value of the IOR of air $n_0$ is approximately 1, it varies slightly with temperature, pressure, and humidity. These conditions are slightly different for each eddy. By suppressing the time variations, the index of refraction can be expressed as

$$n(\mathbf{R}) = n_0 + n_1(\mathbf{R}) \tag{2.1}$$

where $n_1(\mathbf{R})$ is the deviation of the IOR from the mean at some location $\mathbf{R}$. Setting $n_0 = 1$, and writing $n(\mathbf{R})$ in terms of temperature and pressure dependence yields

$$\begin{aligned} n(\mathbf{R}) &= 1 + 77.6 \times 10^{-6}(1 + 7.52 \times 10^{-3}\lambda^{-2})\frac{P_{atm}(\mathbf{R})}{T(\mathbf{R})} && (2.2) \\ &\approx 1 + 79 \times 10^{-6}\frac{P_{atm}(\mathbf{R})}{T(\mathbf{R})} && (2.3) \end{aligned}$$

where $\lambda$ is the optical wavelength in microns, $P_{atm}$ is the pressure in millibars and $T$ is the temperature in Kelvin. The approximation in the second line assumes visible wavelengths.

Statistically, the atmospheric IOR deviates from $n_0$ as a Gaussian (or normal) distribution with a slowly varying mean. Though not strictly stationary, the atmosphere does possess *stationary increments* which make it well suited to description by

structure function $D_n(\mathbf{R})$, discussed in more detail later. Given that the ensemble average of $n(\mathbf{R})$, denoted $\langle n(\mathbf{R})\rangle$, is such that $\langle n(\mathbf{R})\rangle = m(\mathbf{R})$, the spatial auto-covariance function can be expressed as

$$B_n(\mathbf{R}) = \langle [n(\mathbf{R}_1) - m(\mathbf{R}_1)][n^*(\mathbf{R}_2) - m^*(\mathbf{R}_2)]\rangle. \tag{2.4}$$

Given the further assumption of *statistical homogeneity*, and substituting eqn. (2.1) while noting that $\langle n_1(\mathbf{R})\rangle = 0$, the covariance expression simplifies to

$$B_n(\mathbf{R}) = \langle n_1(\mathbf{R}_1) n_1(\mathbf{R}_1 + \mathbf{R})\rangle \tag{2.5}$$

where $\mathbf{R} = |\mathbf{R}_2 - \mathbf{R}_1|$. [1, Ch.2]

### *2.2 Correlated Random Variables*

Random variable are correlated if they are non-orthogonal, that is, if $E[XY] \neq 0$ for any random variables $X$ and $Y$. The covariance [5, sec. 4.7] of any two random variables X and Y is

$$\begin{aligned} COV(X,Y) &= E[(X - E[X])(Y - E[Y])] && (2.6)\\ &= E[XY - XE[Y] - YE[X] + E[X]E[Y]] && (2.7)\\ &= E[XY] - 2E[X]E[Y] + E[X]E[Y] && (2.8)\\ &= E[XY] - E[X]E[Y]. && (2.9) \end{aligned}$$

However, if $X$ and $Y$ are independent (and thus uncorrelated), then $E[XY] = E[X]E[Y]$ and so $COV(X,Y) = 0$.

Random processes, also known as a *stochastic processes* have a detailed definition, comprehensively covered in [5, Ch. 6]. They are closely related to random variables and can be described using the same language of statistics.

The statistics of a random process can be functions of time, location, or other non-random variables. Let $t$ be an independent, non-random variable (possibly a vector), and $X(t)$ be a random process.

The *mean* of $X(t)$, denoted by $m_X$, is

$$m_X(t) = E[X(t)] = \int_{-\infty}^{\infty} x f_{X(t)}(x) dx \tag{2.10}$$

where $f_{X(t)}(x)$ is the probability distribution function (pdf) of $X(t)$.

The *autocorrelation* of $X(t)$, denoted $R_X$, is

$$R_X(t_1, t_2) = E[X(t_1)X(t_2)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xy f_{X(t_1),X(t_2)}(x, y) dxdy \tag{2.11}$$

where $f_{X(t_1),X(t_2)}(x, y)$ is the second order pdf of $X(t)$.

The *autocovariance* of of $X(t)$, denoted $B_X$, can be expressed in terms of the mean, as

$$B_X(t_1, t_2) = E[\{X(t_1) - m_X(t_1)\}\{X(t_2) - m_X(t_2)\}] \tag{2.12}$$

and also in terms of the autocorrelation, as

$$B_X(t_1, t_2) = R_X(t_1, t_2) - m_X(t_1)m_X(t_2). \tag{2.13}$$

Note that $B_X(t_1, t_2) = R_X(t_1, t_2)$ for zero mean processes.

For a *stationary* random process, or one that can be described in stationary independent increments, the autocorrelation can be expressed as $R_X(\tau)$ in terms of the offset $\tau = t_2 - t_1$. The Power Spectral Density (PSD) of a random process X(t) is defined as the Fourier Transform [4, 5] of the autocorrelation $R_X(\tau)$. The PSD, denoted $\Phi_X$, then becomes

$$\Phi_X(f) = \mathcal{F}\{R_X(\tau)\} = \int_{-\infty}^{\infty} R_X(\tau) e^{-j2\pi f\tau} d\tau \tag{2.14}$$

Through the use of Parseval's Theorem, $P_{ave}$ the average power of X(t) can be expressed as

$$\begin{aligned} P_{ave} &= E[X^2(t)] && (2.15)\\ &= R_X(0) && (2.16)\\ &= \int_{-\infty}^{\infty} \Phi_X(f)df && (2.17) \end{aligned}$$

### *2.3 Structure Function*

The *structure function* is the expected variance (or, equivalently, mean square difference) of the random process with respect to the separation distance. Mathematically, the structure function is a difference function

$$D_n(\mathbf{R}_1, \mathbf{R}_2) = E[(n(\mathbf{R}_1) - n(\mathbf{R}_1 + \mathbf{R}))^2] \tag{2.18}$$

where $\mathbf{R} = \mathbf{R}_2 - \mathbf{R}_1$. In a statistically homogeneous medium, the structure function is related to the covariance function as follows:

$$D_n(\mathbf{R}) = 2[B_n(\mathbf{0}) - B_n(\mathbf{R})] \tag{2.19}$$

The additional assumption of isotropy of the propagation medium renders the statistics rotationally invariant (translational invariance is given by homogeneity) and allows the structure function to be expressed purely as a function of the magnitude of the offset,

$$D_n(R) = 2[B_n(0) - B_n(R)] \tag{2.20}$$

where $R = |\mathbf{R}|$.

There are advantages inherent in the structure function that make it preferable to the covariance for this research. First, due to the spectrum discussed later, the covariance has a singularity at $R = 0$ which the structure function avoids. Second, the structure function is a mean difference metric, so $D_n(0)$ will always be zero, and

$D_n(R \neq 0)$ will be directly related to the similarity, or correlation, of the values of $n(R_1)$ to those of $n(R_1 + R)$, making it an intuitive error metric. Third, the structure function exists for non-stationary random process that have stationary increments, whereas the covariance function is not well defined for such.

The atmospheric IOR structure function using the Kolmogorov power spectrum (see section 2.4) is given in terms of $C_n^2$ as

$$D_n(R) = \begin{cases} C_n^2 R^{2/3}, & l_0 \ll R \ll L_0 \\ C_n^2 l_0^{-4/3} R^2, & R \ll l_0 \end{cases} \tag{2.21}$$

where, as above, $l_0$ is the inner scale of the inertial subrange. From observation, $D_n$ should increase exponentially with offset distance, $D_n \propto R^{2/3}$, within the inertial subrange.

### *2.4 Power Spectra*

The *power spectrum* of a process can give enormous insight into the nature of the process, as it provides information about the frequency content. This has a direct application in evaluating phase screen efficacy because different techniques are suited to different ends of the spectrum. For instance, Zernike polynomials efficiently describe the low frequency portion of the spectrum but can become unwieldy when the upper frequencies need to be accurately modelled; and it is just the opposite for the FFT technique. Because optical systems must perform across the entire spectrum, phase screens must be designed to include the entire spectrum. Therefore, it is convenient to express the descriptive statistics of both phase screens and the atmosphere which they simulate in terms of the power spectrum density (PSD) $\Phi_n$.

First, the power spectral density is itself a Fourier transform of the autocorrelation (or autocovariance if the process is zero mean) function of a random process [1,5]

$$\Phi_n(\kappa) = \frac{1}{(2\pi)^3} \int \int \int_{-\infty}^{\infty} B_n(R) exp(-j\mathbf{K} \cdot \mathbf{R}) d^3R \tag{2.22}$$

where $\kappa = |\mathbf{K}|$ is the scalar wave number. Assuming that the autocorrelation is homogeneous and isotropic the above equation becomes

$$\Phi_n(\kappa) = \frac{1}{2\pi^2\kappa} \int_0^\infty B_n(R) sin(\kappa R) R dR \tag{2.23}$$

Using the inverse Fourier Transform to express the autocovariance in terms of the power spectrum yields

$$B_n(R) = \frac{4\pi}{R} \int_0^\infty \kappa \Phi_n(\kappa) sin(\kappa R) d\kappa. \tag{2.24}$$

Using the relationship between the structure function and the autocovariance, $D_n$ can also be expressed in terms of the PSD

$$D_n(R) = 8\pi \int_0^\infty \kappa \Phi_n(\kappa) \left(1 - \frac{sin(\kappa R)}{\kappa R}\right) d\kappa. \tag{2.25}$$

There are several power spectrum models, all of which have been derived empirically from observations of the atmosphere and/or unit analysis of relevant quantities [1].

*2.4.1 Kolmogorov.* Kolmogorov's PSD is the simplest, but only accurate within a bandwidth proportional to the inertial subrange.

$$\Phi_n(\kappa) = 0.033 C_n^2 \kappa^{-11/3}, \quad 1/L_0 \ll \kappa \ll 1/l_0 \tag{2.26}$$

*2.4.2 Tatarski.* Tatarski's model extended the accuracy of the model to high frequencies, but is still inaccurate for $\kappa$ less than $1/L_0$.

$$\Phi_n(\kappa) = 0.033 C_n^2 \kappa^{-11/3} exp(-\kappa^2/\kappa_m^2), \quad \kappa \gg 1/L_0 \tag{2.27}$$

Note that $\kappa_m = 5.92/l_0$.

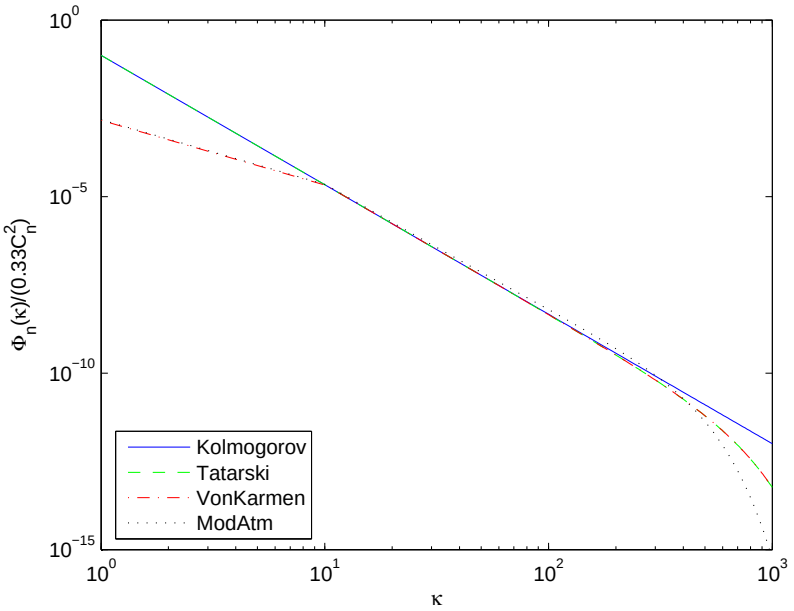


Figure 2.1: Atmospheric power spectra models based upon empirical observations. Parameters: $l_0 = 0.01m$, $L_0 = 10m$

*2.4.3 Von Karmen.* The Von Karmen PSD model is of use across the entire spectrum

$$\Phi_n(\kappa) = 0.033C_n^2\kappa^{-11/3}\frac{exp(-\kappa^2/\kappa_m^2)}{(\kappa^2+\kappa_0^2)^{11/16}}, \quad 0 \leq \kappa < \infty \tag{2.28}$$

and $\kappa_m = 5.92/l_0$ and $\kappa_0 = 1/L_0$ or $\kappa_0 = 2\pi/L_0$.

*2.4.4 Modified Atmospheric.* The modified Atmospheric spectrum is the most complicated of the PSD models discussed herein. It largely follows the Von Karmen PSD, but allows for the 'bump' observed around $1/l_0$, as follows

$$\Phi_n(\kappa) = 0.033C_n^2[1 + 1.802(\kappa/\kappa_l) - 0.254(\kappa/\kappa_l)^{7/6}]\frac{exp(-\kappa^2/\kappa_l^2)}{(\kappa^2+\kappa_0^2)^{11/16}}, \quad 0 \leq \kappa < \infty \tag{2.29}$$

where $\kappa_l = 3.3/l_0$ and $\kappa_0 = 1/L_0$ or $\kappa_0 = 2\pi/L_0$.

### *2.5 Atmospheric Turbulence Descriptors*

Optical atmospheric models are the subject of ongoing research. Most models include the concept of the atmospheric index of refraction structure constant $C_n^2$ denoting the intensity of atmospheric turbulence. The integrated $C_n^2$ profile is defined as

$$C_n^2 = \int C_n^2(z)dz \tag{2.30}$$

where $z$ is the optical path. $C_n^2$ can change drastically with time of day, location and weather conditions. However, for relatively clear conditions, an reasonable $C_n^2$ value is on the order of $10^{-16}$.

For a plane wave, still using the Kolmogorov power spectrum, the phase structure function $D_\phi(R)$ is related to the integrated $C_n^2$ as

$$D_\phi(R) = 2.91k^2R^{5/3}C_n^2 \tag{2.31}$$

where $k$ is the wave number. The atmospheric coherence radius $\rho_0$ is defined as the point when the $D_\phi(R) = 2$. This corresponds to the separation at which the autocorrelation is $1/e$. The Fried radius $r_0$ is related to $\rho_0$ as

$$r_0 = 2.1\rho_0 \tag{2.32}$$

The Fried radius is used together with the diameter of a system optic to provide a single widely used system descriptor, $D/r_0$.

In terms of $C_n^2$, $r_0$ can be derived by noting the relationship between the phase and atmospheric structure functions, in a homogeneous, isotropic medium, is

$$D_\phi(R) = 2.91k^2RD_n(R) \tag{2.33}$$

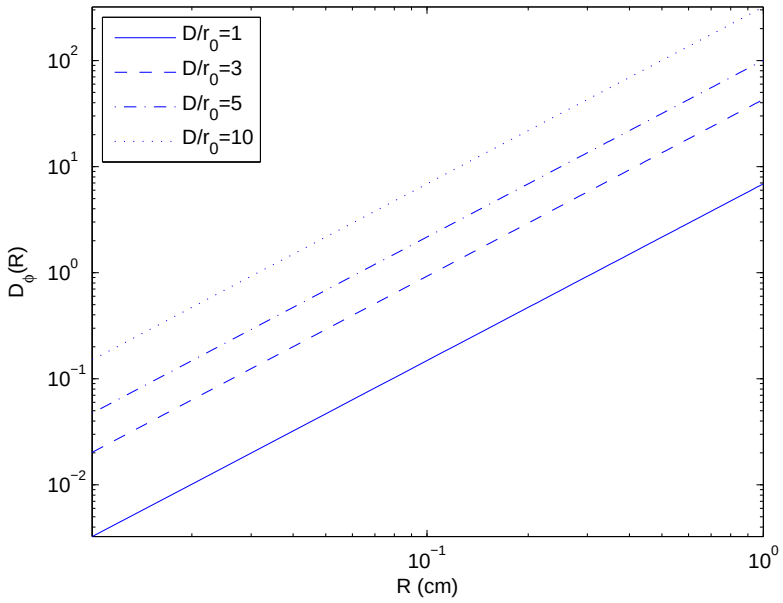


Figure 2.2: Theoretical phase structure functions for various $D/r_0$ values.

Using the equation above along with equation 2.21, $r_0$ of a plane wave can be expressed as

$$r_0 = 0.185 \left[ \frac{4\pi^2}{k^2 C_n^2} \right]^{3/5} \tag{2.34}$$

This in turn allows the $D_\phi$ to be expressed as

$$D_\phi(R) = 6.88(R/r_0)^{5/3} \tag{2.35}$$

where the proportionality $D_\phi$ to $R^{5/3}$ is particularly evident.

A full derivation of the phase and atmospheric IOR structure functions can be found in Roggemann and Welsh's *Imaging Through Turbulence* [9, sec. 3.3,3.4].

## *2.6 Types of Phase Screens*

Phase screens are used in wave optics models to simulate the correlated random phase change that occurs during propagation of light through turbulent media, such as the atmosphere. There are three methods commonly used to generate phase

Table 2.1: Zernike Polynomials.

| n | m | i | Polynomial | Name |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | Piston (Ignored herein) |
| 1 | 1 | 2 | $2r\cos\theta$ | Tip |
| 1 | 1 | 3 | $2r\sin\theta$ | Tilt |
| 2 | 0 | 4 | $3.464r^2 - 1.732$ | Defocus |
| 2 | 2 | 5 | $2.449r^2\sin 2\theta$ | Astigmatism 1 |
| 2 | 2 | 6 | $2.449r^2\cos 2\theta$ | Astigmatism 2 |
| 3 | 1 | 7 | $(8.485r^3 - 5.657r)\sin\theta$ | Coma 1 |
| 3 | 1 | 8 | $(8.485r^3 - 5.657r)\cos\theta$ | Coma 2 |
| 3 | 3 | 9 | $2.828r^3\sin 3\theta$ | |
| 3 | 3 | 10 | $2.828r^3\cos 3\theta$ | |
| 4 | 0 | 11 | $3.416r^4 - 13.416r^2 + 2.236$ | Spherical Abberation |

screens, Zernike polynomials, Fast Fourier Transforms (FFT), and Fourier Series (FS). There is also a modification of the FFT technique called Sub-Harmonic Frequency Expansion (SHFE) which is worth mentioning, but is not covered in depth due to its similarity to both the FFT and the FS methods. Each of these phase screen generation methods relies upon random magnitudes applied to sets of basis functions. For the Zernike method, the functions are a set of indexed polynomials, while the FFT and FS methods rely upon sinusoids.

### *2.7 Zernike Polynomials*

Zernike phase screens are generated as a weighted set of polynomials [9]. Each Zernike polynomial is continuous and describes a unique pattern of abberation as may be found in a lens of unit diameter. The set Zernike polynomials form a normal, orthogonal basis set thus, for any Zernike polynomials $Z_i$ and $Z_j$

$$\int_0^{2\pi}\int_0^1 Z_i(r,\theta)Z_j(r,\theta)drd\theta = \begin{cases} 0 & \text{if} \quad i \neq j \\ 1 & \text{if} \quad i = j \end{cases} \tag{2.36}$$

The equations in table 2.1 are derived from the more general equations

$$Z_{i,even}(r,\theta) = \sqrt{2(n+1)}R_n^m(r)\cos(m\theta), m \neq 0 \quad (2.37)$$

$$Z_{i,odd}(r,\theta) = \sqrt{2(n+1)}R_n^m(r)\sin(m\theta), m \neq 0 \quad (2.38)$$

$$Z_i(r,\theta) = R_n^0, m = 0 \quad (2.39)$$

where $m$ and $n$ are the indices of azimuthal order and radial order, respectively, and $r \leq 1$. If a screen of other than unit radius ($r > 1$) is required, the following transform is used: For a screen of desired radius $R$, with absolute position as $r$, and normalized position $\rho = r/R$, the phase can be expressed as

$$\phi(R\rho,\theta) = \sum_{i=1}^{\infty} a_i Z_i(\rho,\theta) \quad (2.40)$$

$$\phi(r,\theta) = \sum_{i=1}^{\infty} a_i Z_i\left(\frac{r}{R},\theta\right) \quad (2.41)$$

Note that table 2.1 also uses the index $i$, which is used in applying the Zernike coefficient $a_i$. The subscript $i$ is unique to an individual polynomial $Z_i$ (as $n$ and $m$ are not) and allows one to specify the power of each separate abberation within the system once the polynomials have been generated.

The Zernike coefficients $a_i$ relating to to Zernike polynomials $Z_i$ can be recovered from a given phase screen due to their orthonormality, using the following equation

$$a_i = \frac{\int W(\vec{r})\phi(\vec{r})Z_i(\vec{r})d\vec{r}}{\int W(\vec{r})|Z_i(\vec{r})|^2 d\vec{r}} \quad (2.42)$$

where $W(\vec{r})$ is a windowing function that defines the extent of the Zernike polynomials, $\phi(\vec{r})$ is the zero mean input phase map and

$$\int W(\vec{r})d\vec{r} = 1$$

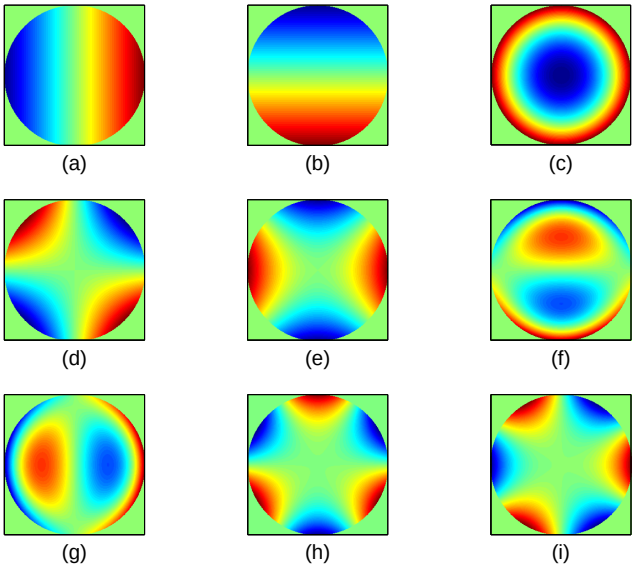


Figure 2.3: Zernike Polynomials 2 through 10. Note that the Zenike basis set assumes a unit circle and must be scaled if that is not the case.

The assumption that $\phi(\vec{r})$ is zero mean implies that piston error ($Z_1$) is ignored, which is practical as it corresponds to a non-measurable constant time delay from the source to the detector. If the screen is not zero mean, it must be normalized before equation 2.42 is applied. $a_1$ can be recovered by noting that it is equal to the mean of the phase before normalization.

The total aberrant power (i.e. the average mean square wavefront error) of a Zernike phase screen can be expressed by the sum of its squared Zernike coefficients, excluding $a_1$ for the reasons given above.

$$\epsilon^2 = \int W(\vec{r})\phi^2(\vec{r})d\vec{r} \tag{2.43}$$

$$= \sum_{2}^{\infty} a_i^2 \tag{2.44}$$

Table 2.2: Zernike Covariance Matrix (Empty squares represent zeros).

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 0.448 | | | | | | -0.0141 | | |
| 3 | | 0.448 | | | | -0.0141 | | | |
| 4 | | | 0.0232 | | | | | | |
| 5 | | | | 0.0232 | | | | | |
| 6 | | | | | 0.0232 | | | | |
| 7 | | -0.0141 | | | | 0.00618 | | | |
| 8 | -0.0141 | | | | | | 0.00618 | | |
| 9 | | | | | | | | 0.00618 | |
| 10 | | | | | | | | | 0.00618 |

Table 2.2 shows the Zernike covariance matrix for a system $D/r_0 = 1$, denoted $\mathbf{A}_{i,j}$, using the Komogorov spectrum. The covariance matrix for any given $D/r_0$ value is

$$\mathbf{B}_{Z_i Z_j} = \mathbf{A}_{i,j}(D/r_0)^{5/3} \tag{2.45}$$

Scaling the covariance matrix and applying it to the set of polynomials are key steps in construction a Zernike phase screen, as is further discussed in chapter 3.

The Zernike method builds phase screens from low frequency to high, so the first few polynomial describe the abberations of greatest power. As noted by Roggeman and Welsh [9] nearly 80% of the aberrative power of an optical system is contained within the tip and tilt ($Z_2$ and $Z_3$) error alone. This is advantageous for 'ball-park' simulations, when an adequate phase screens may be constructed with only a few

polynomials. The disadvantage arises when attempting more precise high-frequency models, as they can require a large number of polynomials to be generated.

## *2.8 Fourier Phase Screens*

The discrete Fourier transform is the representation of a signal as the sum of its frequency content [4, 6, 8]. The result is a complex valued function of the frequency variable $\vec{k} = (k_x, k_y)$:

$$\begin{aligned} F(k_x, k_y) &= \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-j\frac{2\pi}{N}(xk_x + yk_y)} && (2.46) \\ &= F_R(k_x, k_y) + jF_I(k_x, k_y) && (2.47) \\ &= |F(k_x, k_y)| e^{j\angle F(k_x, k_y)} && (2.48) \end{aligned}$$

Both the FFT and the FS methods use this transform, as does SHFE. However, the difference between the methods is readily discussed using the inverse transform. The two-dimensional inverse transform represents a signal $f(x, y)$ as the sum of its frequency content $F(k_x, k_y)$

$$f(x, y) = \sum_{k_x} \sum_{k_y} F(k_x, k_y) e^{-j2\pi(xk_x + yk_y)} \tag{2.49}$$

The FFT method uses frequencies $k_x$ and $k_y$ at regular intervals on a cartesian grid, such that $dk_x = dk_y$ for all $k_x$ and $k_y$ (see figure 2.4). The FS method presented here uses frequencies at logarithmic intervals on a polar grid (figure 2.5). The SHFE method uses the same base frequency grid as the FFT, but iteratively expands the lower frequencies (figure 2.6) to build more low frequency content.

*2.8.1 Fast Fourier Transform.* By evenly spacing the component frequencies on a cartesian grid and using code that has been extensively optimized, the FFT method requires only $N \log N$ operations, reducing the number of operations required

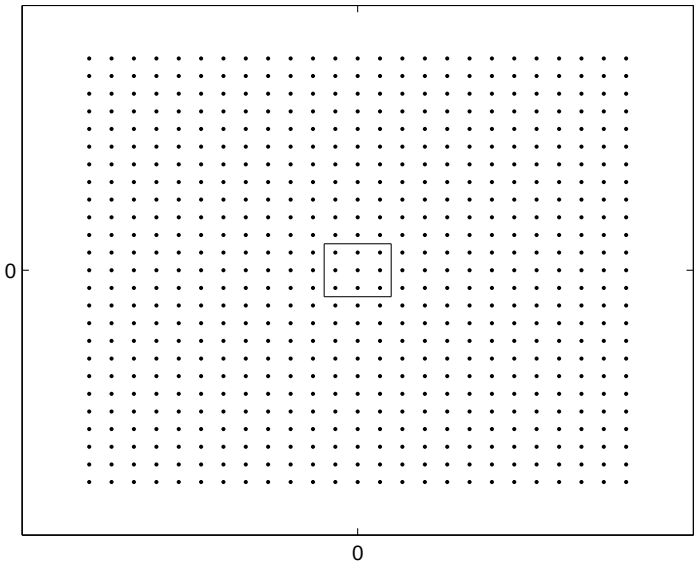


Figure 2.4: The base frequency grid of an FFT phase screen.

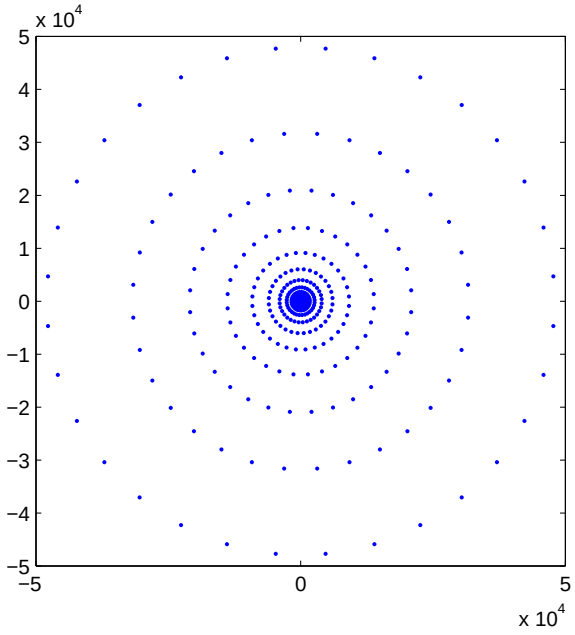


Figure 2.5: The base frequency grid of an FS phase screen.

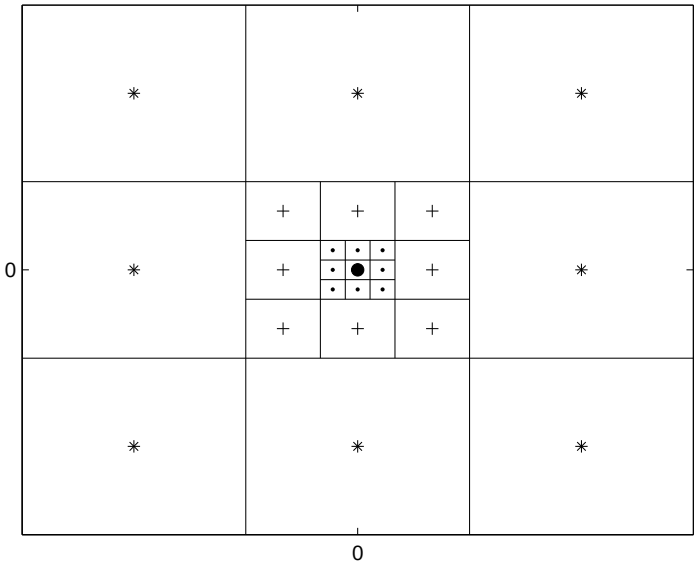


Figure 2.6: The zero frequency pixel of an FFT phase screen after three iterations of Sub-Harmonic Frequency Expansion.

by an order of $N/logN$. The atmospheric phase is modelled via a power spectrum $\Phi(\vec{\kappa})$ used as a filter. The filter is applied to Gaussian random variables to create the weighting function for each frequency component, and the Fourier transform is taken.

$$S_{FFT}(\vec{r}) = \mathcal{F}^{-1}\{G(\vec{\kappa})\sqrt{\Phi(\vec{\kappa})}\} \tag{2.50}$$

where $G(\vec{\kappa})$ is a matrix of unit variance Gaussian random variables and $\mathcal{F}^{-1}$ is the inverse Fourier transform operator. If $G(\vec{\kappa})$ is complex Gaussian,

$$S_{FFT}(\vec{r}) = S_R(\vec{r}) + jS_I(\vec{r})$$

where $S_R$ and $S_I$ are independent. This yields two distinct phase screens $\phi_1(\vec{r}) = S_R(\vec{r})$ and $\phi_2(\vec{r}) = S_I(\vec{r})$.

There are two negative characteristics associated with the FFT frequency grid. First, all frequencies lower than the sampling frequency $f_s$ are lumped into the DC component (see figure 2.4. Due to the fact that atmospheric turbulence power is

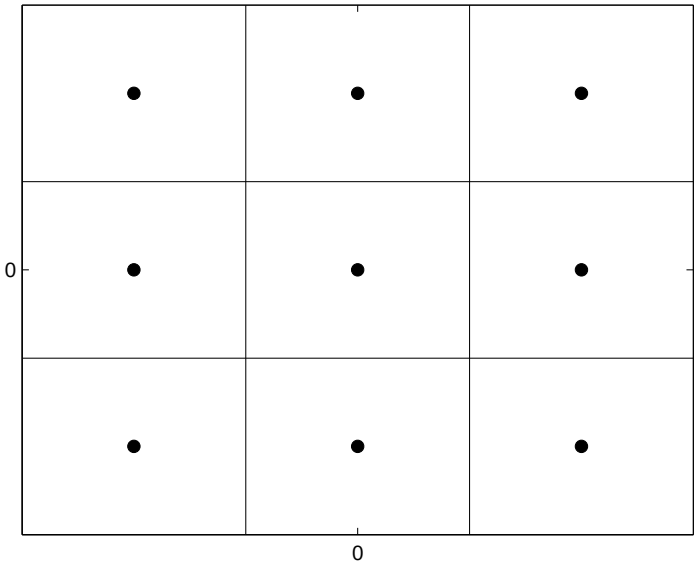


Figure 2.7: The nine lowest frequencies of an FFT phase screen. The diagonal frequency points are farther from zero than the on-axis points by a factor of $\sqrt{2}$.

highest for low frequencies, this may lower fidelity. Second, the frequency steps are larger on the diagonal than on either of the axes by a factor of $\sqrt{2}$ (see figures 2.7 and 2.8).

*2.8.2 Fourier Series.* The FS phase screen generated on a logarithmically scaled polar grid, rather than an equidistant rectangular grid. The logarithmic scaling allows for a higher density of frequencies near zero while not requiring the same density at the higher frequencies (figure 2.9). Figure 2.10 is the magnified center of figure 2.5, showing the low frequency components. No that the low frequency grid is divided into multiple low frequencies. The polar distributions ensures that the frequency step will be the same regardless of direction; eliminating the $\sqrt{2}$ difference seen on the FFT diagonals.

*2.8.3 Sub-Harmonic Frequency Expansion.* In order to avoid generating excess high frequency data while retaining the speed of the FFT, the Sub-Harmonic

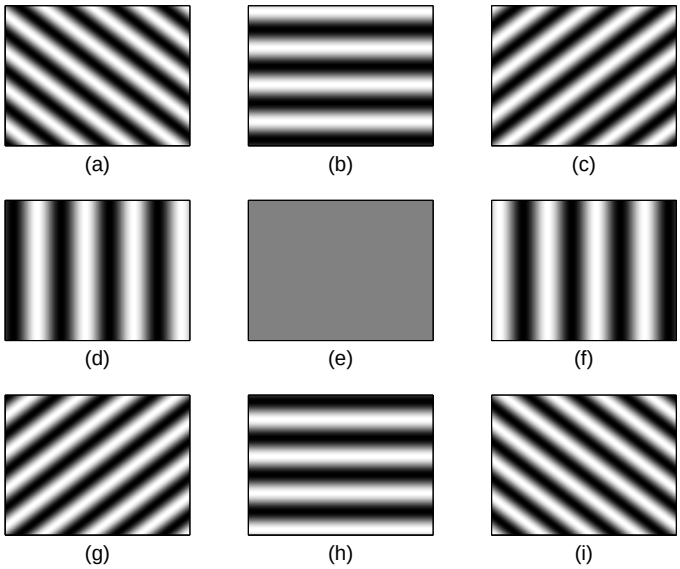


Figure 2.8: Example of the nine lowest frequencies within an FFT phase screen, corresponding to figure 2.7. The effect of the $\sqrt{2}$ factor in the frequency step of the diagonal elements can be seen in the closer spacing of the lines in (a),(c),(g) and (f).

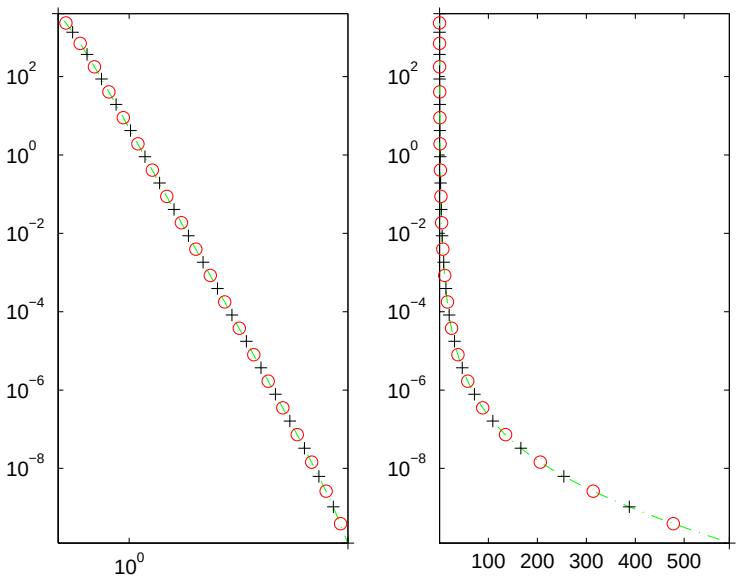


Figure 2.9: Frequency domain locations of the sinusoids contained within the FS phase screen on log-log and semi-log axes.

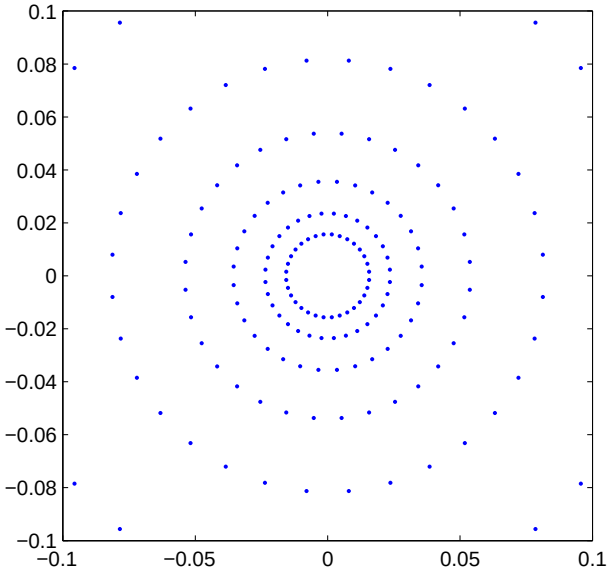


Figure 2.10: Frequency domain locations of the sinusoids contained within the FS phase screen.

Frequency Expansion (SHFE) can be used to expand only the low frequency terms of an FFT phase screen. This is done by dividing the zero frequency pixel of the FFT screen into increasingly finer grids (2.11). Each new frequency is used to generate a two-dimensional sinusoid which is then weighted with a random added to the original FFT phase screen. Due to the decrease in frequency, the phase screen will be to small to contain an entire sinusoidal period, causing the screen to gain a non-zero mean. Therefore, the phase screen may need to be re-normalized after SHFE.

The SHFE method combines the optimized code of the FFT method with some of the individual frequency specification of the FS method. It is more cumbersome than the pure FFT method and still retains some of the flaws, such as the $\sqrt{2}$ difference between axial and diagonal frequency steps (figure 2.12). The SHFE frequency spread is not as efficient as that of the FS method, even though it does increase frequency density near zero. The SHFE method can be viewed as a hybrid combination of FFT and FS methods. Therefore, though worth mentioning for completeness, SHFE will not be evaluated individually herein.

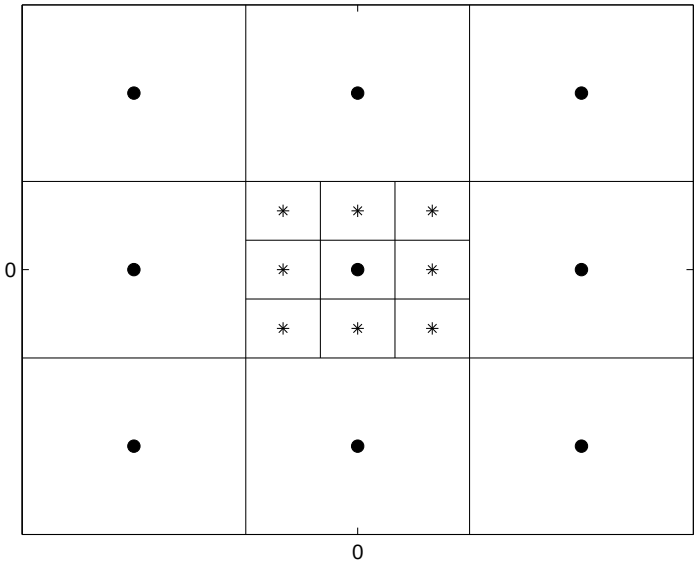


Figure 2.11: The low pixels of an FFT frequency grid after one iteration of SHFE. The expanded frequencies are denoted with an asterisk.

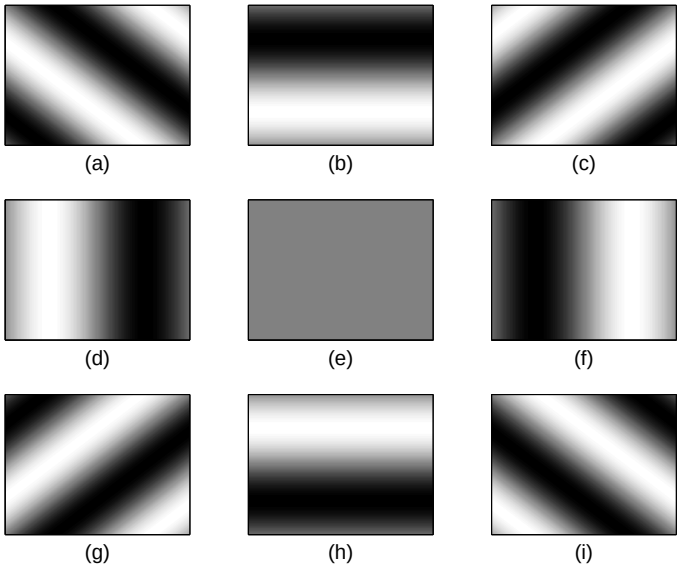


Figure 2.12: Phase effects examples corresponding to the expanded frequencies from figure 2.11. Note that they are lower frequency than those of 2.8 but still contain the $\sqrt{2}$ difference on the diagonals.

# III. Methods of Comparison

Three types of phase screens are generated - Zernike, FFT, and FS - in various sizes. Each of which must then evaluated for time efficiency and correspondence to theory. This chapter outlines the evaluation methods used herein. Section 3.1 covers the methods used to generate all three, in the above order. Section 3.2 covers what needs to be considered when deciding the size of phase screen to generate. Section 3.3 discusses how the generation time of each method is analyzed. Section 3.4 contains information about the frequency content of the Zernike polynomials. Section 3.5 is a brief summary of the use of the structure function.

## *3.1 Generating Phase Screens*

*3.1.1 Zernike.* A Zernike phase screen is generated by applying weighted coefficients to the set of Zernike polynomials. The coefficient values are determined by the Zernike covariance matrix for the given $D/r_0$. The random samples are calculated by using the *cholescky factorization* via the MATLAB® 'chol' command [6]

$$\mathbf{A} = CHOL(\mathbf{K}_a)' * \mathbf{G} \tag{3.1}$$

where $\mathbf{K}_a$ is the Zernike covariance matrix, $\mathbf{G}$ is a vector of unit variance, zero mean Gaussian random variables and $\mathbf{A}$ is the resulting vector of random, correlated Zernike coefficients. Note that the Zernike covariance matrix for any $D/r_0$ can be generated by multiplying the covariance matrix for $D/r_0 = 1$ by $(D/r_0)^{(5/3)}$. This allows one Zernike covariance matrix to be loaded from memory and used with only a multiplicative change.

The MATLAB® routine *MakeZernikePhzScrn.m* generates an $N \times N$ pixel, zero mean phase screen with a covariance corresponding to a given $D/r_0$. It takes as input four parameters: the system $D/r_0$ value, the size (in pixels) of the screen, the square Zernike covariance matrix for $D/r_0 = 1$ ($\mathbf{K}_{a0}$), and the number of Zernike polynomials to be used. The process of generating a Zernike phase screen can be summarized as follows:

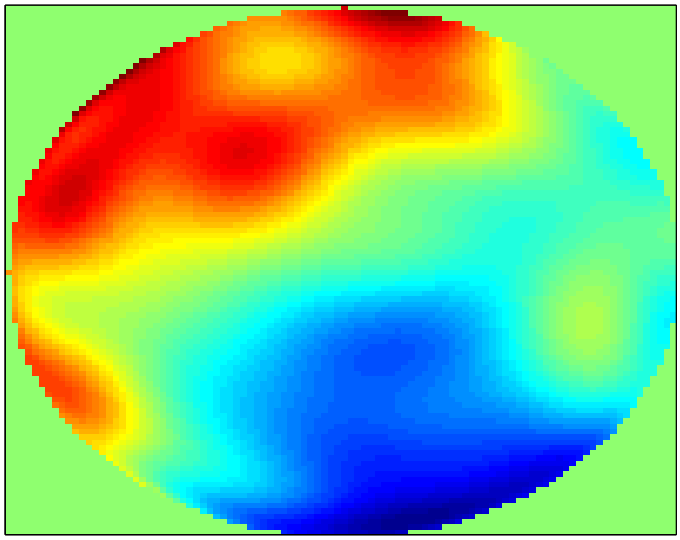

Figure 3.1: A Zernike Phase Screen constructed using Zernike polynomials 2 through 81. $N = 100$ pixels per side.

- Calculate given Zernike polynomials $Z_i$ on a unit circle.
- Load Zernike covariance matrix $\mathbf{K}_a$ and normalize it to $D/r_0$
- Calculate random weights $a_i$ and apply them to $Z_i$.
- Sum $Z_i$'s, after any necessary scaling.

There are three items of particular note when generating a Zernike phase screen using *MakeZernikePhzScrn.m*. First, the dimensions of $\mathbf{K}_{a0}$ must be at least as large as the number of Zernike polynomials to be used, as it does in fact describe their covariance. Second, the first degree Zernike polynomial (describing time delay, or 'piston' error) is not present in the covariance matrix and not used in constructing the phase screen. The user can compensate for this by adding a constant to the screen, making it a non-zero mean matrix. Third, as can be seen in figure 3.1, Zernike phase screens are circular. Therefore the usable screens size is only $\pi N^2/4$ rather than $N^2$ which is the size of the matrix generated.

*3.1.2 FFT.* The MATLAB® code *Make_FFTPhzScrn.m* generates a phase screen corresponding to a particular atmospheric spectral model $\Phi$ at a given $C_n^2$. The input spectra $\Phi$ must be the same dimensions as the desired phase screen. A peculiarity of the FFT method is the continuity across edges. This leads to higher than theory correlation of values on opposite sides of the screen, causing non-physical edge effects (Figure 3.2). This is an inherent result of the Fourier transform method and can be compensated for by generating a larger than necessary screen and discarding the edges. In summary, to build an FFT phase screen

- Build a two-dimensional cartesian frequency grid (shown here using Von Karmen PSD)

$$\kappa = \left(k_x^2 + k_y^2 + \left(\frac{D}{L_0}\right)^2\right)^{-11/12} exp\left[-0.56(l_0/D)^2(k_x^2 + k_y^2)\right]$$

- Generate random weights $\mathbf{G}$ as a set of zero mean, unit variance, complex Gaussian random numbers.
- Take the Fourier transform of the weighted frequency grid $\kappa G$ and multiply by $0.1517\left(\frac{D}{r_0}\right)^{(5/6)}$.
- Take the real and imaginary parts of the result to form two distinct phase screens.

*3.1.3 FS.* An FS phase screen is generated by first constructing a kernel of two-dimensional sinusoids with frequencies relating to logarithmically spaced points on a polar grid. Using the multiplicative spacing factor $Q$, the number of frequencies $N_\kappa$ is inversely proportional to the value of $Q$, such that

$$N_\kappa = N_\theta \left\lceil \frac{\log\left(\frac{14.2L_0}{\pi l_0}\right)}{\log Q} \right\rceil \tag{3.2}$$

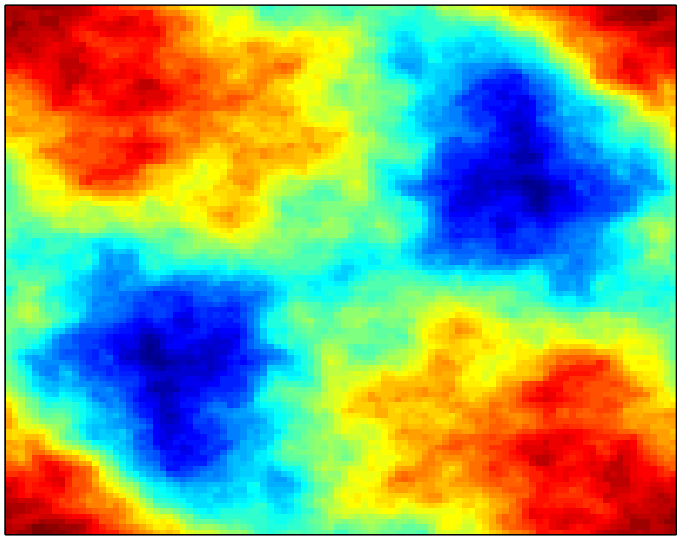

Figure 3.2: An FFT Phase Screen constructed using Von Karmen frequency spectrum.

where $N_\theta$ is the number of equal angles in each polar grid. Using a triply nested *for* loop, and the mean PSD, the value of each sinusoid at each pixel is calculated, yielding a three-dimensional kernel size of $S_K = N_x \times N_y \times N_\kappa$. Generating square phase screens, as is done here, $N = N_x = N_y$, so the kernel size is

$$S_K = N^2 \times N_\kappa \tag{3.3}$$

Randomness is applied to the kernel by multiplying by a zero mean, unit variance, complex Gaussian random number. The real and imaginary parts are taken to form two sets of data, then each set is summed and the mean is subtracted, forming two distinct random phase screens. In summary, to form an FS phase screen

- Generate a kernel sinusoids with frequencies relating to a logarithmic polar grid, using the mean PSD.
- Weight each sinusoid with a zero mean, unit variance, complex Gaussian random number.

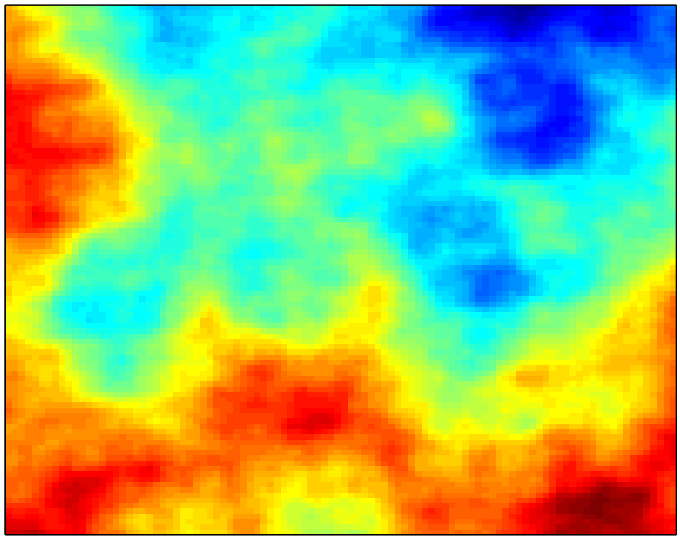

Figure 3.3: An FS Phase Screen constructed using a logarithmic kernel.

- Use the real and imaginary parts of the result to for two distinct phase screens.

Note that the itemized steps for FS greatly resemble those of the FFT. The main difference is that, due to the frequency grid used, the FS routine cannot make use of the optimized *fft2.m* code in MATLAB® and must instead construct each sinusoid a pixel at a time. The result (figure 3.3) contains a greater spectrum representation of low frequency content than in typical FFT techniques. It also avoids the continuity across edges that causes some non-physical edge effects.

## *3.2 Sizing Considerations*

When generating phase screens as part of a model of physical phenomena the physical area represented is important. The physical size is specified by the sample period $\Delta x$ and the number of pixels across $N$, yielding a total area $A$ of

$$A = N\Delta x \times N\Delta x = N^2\Delta x^2 \tag{3.4}$$

As mentioned above, both Zernike and FFT phase screens have a useable size somewhat less than the $N \times N$ pixel grid generated. For FFT phase screens, this is because the method itself causes the opposite edges of the generated phase screen to be highly correlated, which is contrary to observation of the atmosphere. This is a problem that can be overcome by generating a phase screen large than what is needed and using only the center portion for simulation.

In the case of Zernike phases screens, the size discrepancy is the result of the definition of Zernike polynomials. Each Zernike polynomial is defined only on the unit circle, and must be scaled to represent a larger area using the following transform

$$r = R_a \rho \tag{3.5}$$

where $r$ is the absolute position, $R_a$ is the constant radius of aperture and $\rho$ is the normalized position. However, even once scaled, the corners of the square screen will still be zero, leaving a represented area of

$$A_Z = \frac{\pi}{4} N^2 \Delta x^2 \tag{3.6}$$

FS screens avoid both limitations, being defined as sinusoids upon an infinite grid and having a distribution of frequency components that does not artificially correlate the outer edges of the finite screen.

### *3.3 Generation Time Analysis*

Each phase screen has some parameter(s) that may be generated once at the beginning of given scenario and used to make any number of screens. This initial generation time $T_i$ can be separated from the individual screen generation time $T_s$, allowing the total time used to generate $N_S$ screens ($T_t$) to be expressed as

$$T_t = T_i + N_S T_s. \tag{3.7}$$

Optimization of $T_t$ must consider the optimal number of phase screens needed to effectively model a given scenario. As the number of screens needed goes up, the effect of $T_i$ becomes less important. Also, it may be feasible to complete all initial calculations prior to the implementation the simulation, again lessening the emphasis upon initialization time. However, if one must change conditions drastically during the simulation, $T_i$ can have a great impact upon the efficiency of the model.

*3.3.1 Initialization Time.* In MATLAB®, execution time can be measured using the *tic* and toc commands. In the case of the Zernike PS, *tic* is used before loading the covariance matrix and Zernike polynomial index; *toc* is used after they are loaded. The Zernike PS initialization time $T_{i-Z}$ is then set equal to toc.

Similarly, tic and toc can be used to record the FFT PS initialization time $T_{i-FFT}$, and the FS PS initialization time $T_{i-FS}$. $T_{i-FFT}$ encompasses calculating the analytical spectra ($\Phi$). $T_{i-FS}$ encompasses defining a frequency grid spacing and generating a polar kernel of sinusoids, one for each frequency grid point.

*3.3.2 Per Screen Time.* The per screen generation time $T_s$ is calculated by using *tic* and *toc* to record the time it takes to construct $N$ phase screens after the initial constructs have been formed ($T_i$), then dividing by $N$:

$$T_s = (T_t - T_i)/N$$

However, in practice $T_i$ and $T_s$ are recorded independently within the code.

## *3.4 Zernike Frequency Analysis*

Taking the FFT's of the Zernike equations (2.37) both individually and as a screen made of of equally weighted polynomials allows one to compare the frequency content of the Zernike phase screens with that of the FFT and FS phase screens. This is done by generating an set of Zernike polynomials, $Z_i$ through $Z_j$, using non-random

coefficients

$$a_i = a_j \text{ for all } i, j$$

The polynomials are then Fourier transformed, and their power spectrum evaluated, both in one dimension and two. For the purpose of cross-sectional comparisons, a selection of polynomials with increasing radial order $n$ is used. Results are presented in chapter 4.

### *3.5 Structure Function Analysis*

Being a mean square difference function, the structure function of a phase screen can be found as follows: For any phase screen $\phi$, let $\phi(R)$ be the original phase screen $\phi(0)$ shifted in some direction by $R$ pixels, then

$$D_\phi(R) = E[(\phi(0) - \phi(R))^2] \tag{3.8}$$

Implementing this for a phase screen with values distributed on a discrete cartesian grid means that $D_\phi(R)$ is affected by the direction of the frequency shift. If the shift is along either of the axes, each step $\Delta R$ will be a single unit of separation. However, if it is off axis, such as along the diagonal, each step will be greater than a single unit

$$\Delta R = \sqrt{(\Delta x + \Delta y)^2} \tag{3.9}$$

and $D_\phi(R)$ must be scaled accordingly when plotted.

# IV. Results

This chapter contains the results from the methods described in chapter 3. Section 4.1 covers the generation times for all three methods, as well as how the the Zernike method and FS method times are affected by varying $N_Z$ and $Q$, respectively. Section 4.2 addresses how the frequency content of Zernike phase screens compares to that of FFT and FS phase screens. Section 4.3 covers how the structure functions of the various screens relate to the theoretical atmospheric phase structure function. Finally, section 4.3 discusses quality/usability comparisons between FFT and FS phase screens.

## *4.1 Time Results and Analysis*

This section is broken down into four subsections: the first two, initialization time and per screen time, give the usability of each method in terms of how long each one takes. The second two, Zernike time explanation and FS time explanation, are included because there is much optimization that the routines lack. The FFT code has already been extensively optimized because it is used so ubiquitously within signal processing, and so there is no particular need to focus upon it. The results presented here are based upon the performance of a particular computer, so the times will change given different hardware. However, they should be accurate as a relative measure of performance.

*4.1.1 Initialization time.* The initialization time $T_i$ includes different elements for each method. Initialization of an $N \times N$ phase screen requires the following: Using FFT's, one must initially calculate the $N \times N$ atmospheric spectra using a model such as those discussed in section 2.4. Using FS, one must generate the polar kernel, choosing each included frequency on a two-dimensional grid. Each frequency point corresponds to an $N \times N$ sinusoid in the time domain. Using Zernikes, one must load both a covariance matrix and a polynomial index, then each indexed polynomial must be generated in a $N \times N$ matrix.

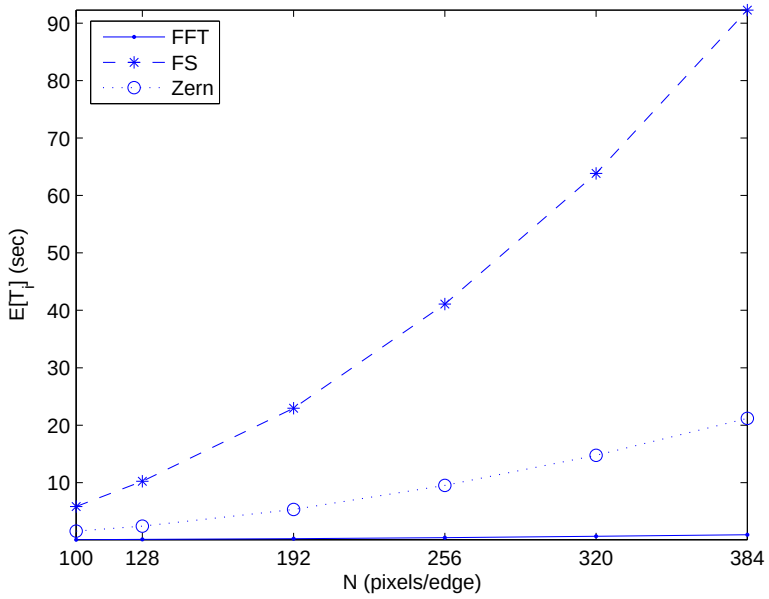


Figure 4.1: Initialization times for all three types of phase screens with respect to the number of pixels per screen edge $N$.

Figure 4.1 shows the initialization times for all three methods. Times were calculated using 84 Zernike polynomials and a $Q$ value of 2.0. Recall that $T_{i-FFT}$, $T_{i-FS}$, $T_{i-Z}$ are the initialization times for the FFT, FS, and Zernike methods, respectively, and $T_{s-FFT}$, $T_{s-FS}$, $T_{s-Z}$ are their respective per-screen times. $T_{i-FFT}$ is by far the smallest and is nearly linear in the range given. This is due to the small amount of preparation needed for the FFT method, most of which makes use of optimized MATLAB® vector operations. $T_{i-Z}$ and $T_{i-FS}$ are more interesting. Both are quadratically proportional to the number of pixels per screen edge $N$, making them linearly proportionate to the number of pixels per screen, $N^2$.

$T_{i-FS}$ is the highest due to the kernel size and the nested *for* loop used in generating it. The FS kernel size $S_K$ is proportional to $N^2$ and the *for* loop execution time is directly proportional to the kernel size. The FS initialization encompasses the generation of 416 $N \times N$ sinusoids. By comparison, $T_{i-Z}$ is lower because it

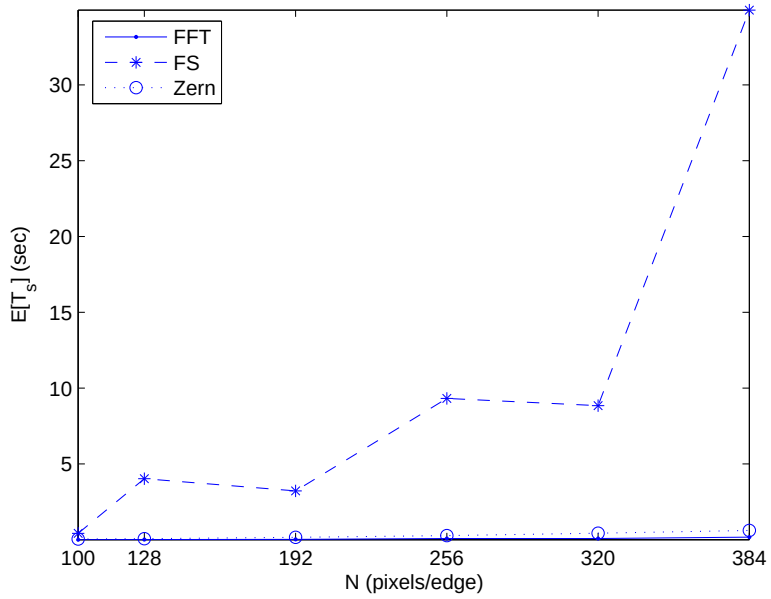


Figure 4.2: (a) The time to generate each phase screen after the initial calculations have been made, with respect to the square root of the number of pixels in the screen.

encompasses the calculation of only 84 $N \times N$ polynomials. $T_{i-FFT}$ is lowest of all because it encompasses the calculation of only 4 $N \times N$ power spectra.

Judged purely upon initialization time, the FFT method is by far the most efficient. The Zernike method comes second and the FS method is the least efficient.

*4.1.2 Per-Screen Time.* The per-screen time $T_s$ (figure 4.2) is taken from those calculations that cannot be made in advance for a given set of phase screens. For each method, this involves the generation of complex random weights. In the case of the FFT, it also involves taking the two-dimensional fast Fourier transform of the $N \times N$ data matrix and separating the result into its real and imaginary parts. In MATLAB®, this process has been extensively optimized using vector operations, with excellent results.

As can be seen in figure 4.2, $T_{s-FFT}$ is the lowest of the three per-screen generation times shown. Figure 4.3 shows this more clearly by ignoring the far larger

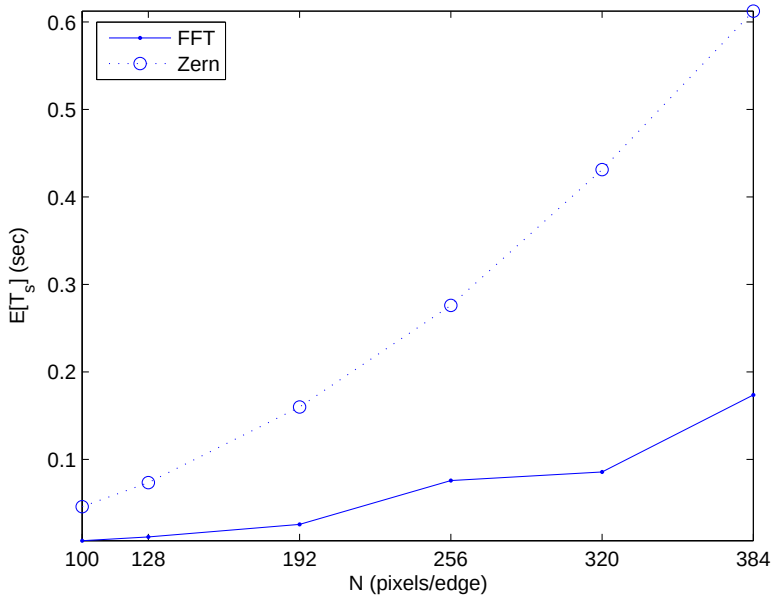


Figure 4.3: (a) The time to generate each Zernike and FFT phase screen after the initial calculations have been made, with respect to the square root of the number of pixels in the screen.

$T_{s-FS}$ and showing only $T_{s-FFT}$ and $T_{s-Z}$. Though not entirely linear and increasing monotonically, $T_{s-FFT}$ follows the expected upward trend proportional to the number of operations required, $N \log N$. It does not rise above a tenth of a second until after $N = 320$. Figure 4.3 also shows that $T_{s-Z}$ passes the tenth of a second mark shortly after $N = 128$, and figure 4.2 shows that $T_{s-FS}$ rises above 0.01 s for $N = 100$.

The irregular progression of $T_{s-FS}$ in figure 4.2 is particularly noteworthy. Although it possesses a general quadratically increasing trend, it is not monotonic. Decreases are evident from $N = 128$ to $N = 192$ and again from $N = 256$ to $N = 320$. These are most likely a result of the manner in which MATLAB® allocates memory: one segment at a time.

By default, MATLAB® uses 64-bit (8-byte) double precision floating point numbers. The memory requirements $M$ for a kernel can therefore be calculated in kilobytes (KB)

$$M = 8S_K/1024 = 8N^2N_\kappa/1024 \tag{4.1}$$

using the binary definition of 1 KB = 1024 bytes and 1 MB = 1024 KB. For $N = 100$ and $N_\kappa = 416$ (i.e.,$Q = 2.0$), the memory required is $M = 32,500$ KB $\approx 33$ MB. For $N = 384$ at the same $Q$ value, $M = 479232$ KB = 468 MB.

Complex numbers are treated as two separate floating point numbers, the real part ant the imaginary part. Thus one complex number actually takes 128 bits (or 16 bytes) of memory. Therefore, when the random weights are applied to the kernel to generate each new screen, two new matrices the same size are generated. Where the before the memory was allocated for one object of size $S_K$, it must now be allocated for three such objects. If $N = 100$, this means 96 MB are now needed.

Memory allocations are done by segment, so when the need exceeds the memory available, another entire segment is allocated. The following operations use the newly allocated memory until it to is overfilled, whereupon, the process of request and allocation will be repeated until there is no more memory to allocate. However, the request and allocation take time, increasing the execution time of routines that must pause to do so. It is this increase that is reflected in the peaks in figure 4.2.

As far as per-screen generation time, $T_{s-FS}$ both starts out higher and rises more rapidly than both $T_{s-Z}$ and $T_{s-FFT}$, making the FS method the least per-screen time efficient. $T_{s-Z}$ is more comparable to $T_{s-FFT}$, but the FFT method is the most per-screen time efficient.

*4.1.3 FS Time Enhancement.* The FS method can be made to execute more rapidly by decreasing the number of frequency components, controlled by the multiplicative frequency spacing parameter $Q$. However, there a tradeoff between execution time and number of frequencies, which becomes lower as $Q$ increases. For instance, increasing $Q$ from 1.5 to 2.0 retains 76% of the frequencies while using only 56% of the time: a 24% loss for a 44% gain. But when you increase $Q$ from 2.0 to 2.5 there is a 20% loss for a 20% gain, and that is about as good as it gets from there on out.

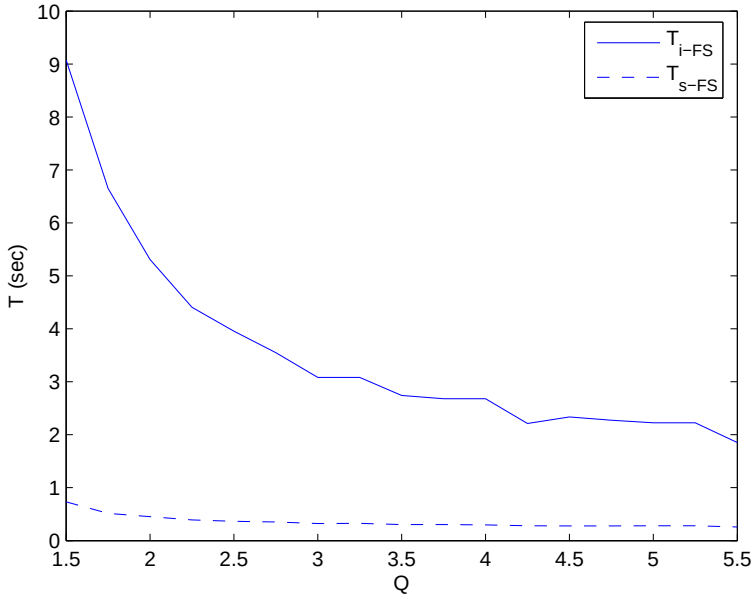


Figure 4.4: Mean Time taken to generate a $100 \times 100$ pixel FS phase screen, with respect to the $Q$ value. $Q$ is inversely proportional to the number of frequency components.

The frequency grid remains polar and logarithmic, so both low and high frequencies are retained. They are simply more sparse as $Q$ increases. However, eventually the frequencies will become so sparse that one loses the advantage gained by the polar frequency grid. There are other possible options for optimizing the FS method that should be explored as well. They will be discussed in chapter 5.

*4.1.4 Zernike Time Enhancement.* The Zernike method will also execute faster if you decrease the number of Zernike polynomials used $N_Z$. The difference being that what is lost with each polynomial is not a specific frequency component, but a particularly shaped abberation. Zernike polynomials can be selected to include those abberations most important to the user. This may allow an effective model to be developed using a very few polynomials that would be extremely time efficient.

Figures 4.5 and 4.6 show the initialization and per-screen generation times, respectively, for various numbers of Zernike polynomials. It can be seen that $T_{i-Z}$ increases quadratically as a function of $N_Z$, while $T_{s-Z}$ increases linearly. The total

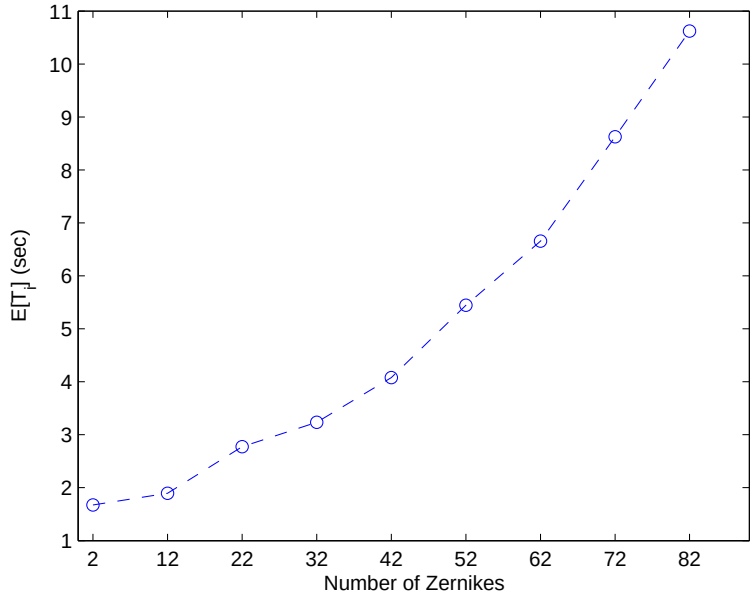


Figure 4.5: Time spent initializing $100 \times 100$ pixel Zernike phase screens with respect to the number of Zernike polynomials used.

time is second order function of $N_Z$ as

$$T_{t-Z} = T_{i-Z} + N_{scrns} T_{s-Z} \propto N_Z^2 + N_Z$$

As the number of phase screens $N_{scrns}$ generated increases, the quadratic initialization time factor becomes less dominant.

An advantage that the Zernike method has over the Fourier methods is that it does not require the generation of new polynomials for changing atmospheric conditions. Such variations are included in the random weighting factors correlated by the $D/r_0$ proportionate Zernike covariance matrix. Therefore, regardless of how many or how different the phase screens needed, a large time investment in the beginning can yield efficiency later on.

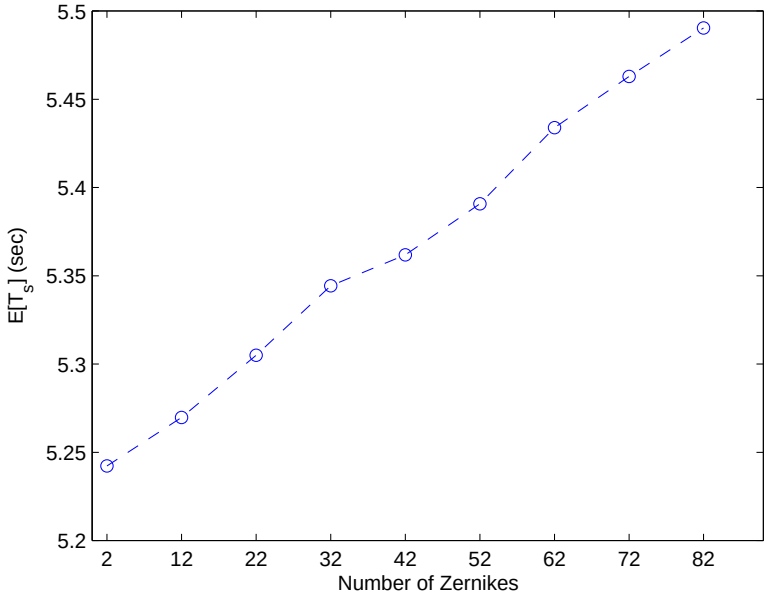


Figure 4.6: Time after $T_i$ spent per-screen generating $100 \times 100$ pixel Zernike phase screens with respect to the number of Zernike polynomials used. Does not include $T_i$.

## *4.2 Zernike vs Fourier Frequency Content*

The Fourier The Zernike polynomials have frequency content that tends to become higher as the index increases, but there is no direct correspondence to the frequency content of the FFT and FS sinusoids. Figure 4.7 shows a set of Zernike polynomials selected for their increasing radial orders $n$ and figure 4.8 shows their Fourier transforms, i.e. their frequency content. Instead of each polynomial being represented by a discrete spike in the frequency domain, as are the sinusoids from the FFT and FS methods, they form a continuous spread, showing that each one is composed of multiple frequency components. However, there is increasing power contained in the higher frequencies as the Zernike index increases.

## *4.3 Structure Function*

The structure function of an atmospheric random phase screen, being a mean square difference function, should increase with separation distance, indicating that

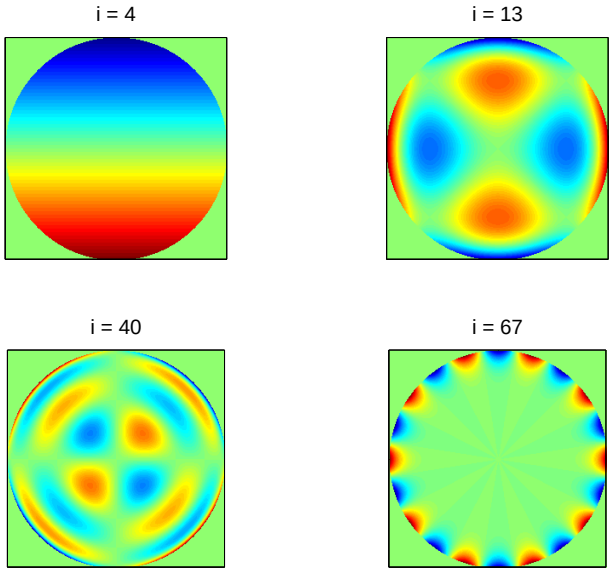


Figure 4.7: Various Zernike polynomials from $Z_2$ to $Z_{74}$

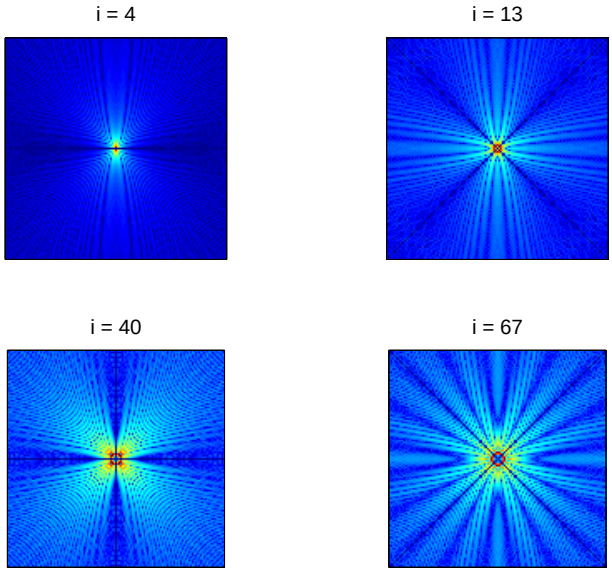


Figure 4.8: The Fourier transforms of the zernike polynomials from $Z_2$ to $Z_{74}$ shown in figure 4.7, raised to the 1/3 power.

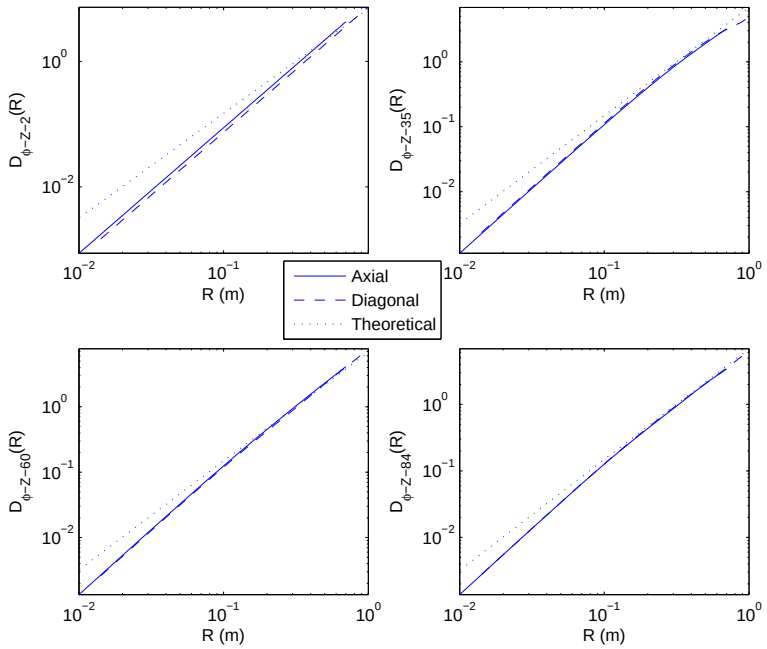


Figure 4.9: Zernike structure functions taken of screens constructed using Zernike polynomials that form complete sets of each radial order $n$. From left to right, and up to down, $N_Z = 2$, 35, 60, and 84, respectively. $D/r0 = 1$ and $N = 100$.

correlation decreases with separation distance $R$. The slope of a theoretical structure function on a loglog plot is 5/3 because $D_n \propto (D/r_0)^{5/3}$, where larger $R$ corresponds to smaller spatial frequencies $f$. Also, the structure functions taken along the diagonals of a phase map should match those taken along the axes. This shows the radial symmetry that results from the assumption of homogeneity and isotropy.

*4.3.1 Zernike Structure Function.* The structure function of Zernike phase screens is a not only a function of separation distance $R$, but also of the number and radial order of the Zernike polynomials used to form the phase screen. Figures 4.9 and 4.10 both show the structure functions of four different phase screens constructed for $D/r0 = 1$ using increasing numbers of Zernike polynomials.

In figure 4.9, all of the screens were constructed using complete sets of radial order Zernike polynomials. The 2 polynomials used in the upper left plot are tip

and tilt, the complete set of first radial order ($n = 1$) Zernike polynomials; the 35 polynomials used in the upper right plot form the complete sets of Zernike polynomials for radial orders $n = 1$ through $n = 7$, etc. This ensures that each phase screen will be radially symmetric, as fits the assumption of homogeneity and isotropy.

The radial symmetry of a phase screen is reflected in the similarity of the axial structure function to the the diagonal structure function. Using Zernike polynomials, this symmetry is seen when $n \geq 4$, as, even given complete radial order sets, a certain number are required to ensure this symmetry. Figure 4.9 gives an example: on the upper left only the set of $n = 1$ is used, on the upper right sets $n = 1$ through $n = 7$ are used. One can see that the lines are almost entirely overlapping in the latter plot. The lower plots include even greater numbers of radial order sets. In the bottom left plot especially, it is almost impossible to tell the axial structure function from the diagonal structure function, indicating an almost exact radial symmetry.

However, if the Zernike polynomials are not taken in radial order sets, radial symmetry can be lost, regardless of the number of Zernike polynomials used. Figure 4.10 illustrates this nicely. In the upper left is the structure function of a screen built with only 36 Zernike polynomials. On the lower left is the structure function of one built with 60 Zernike polynomials. Both screens were built using complete $n$ sets, and both have axial and diagonal structure functions that almost entirely overlap. On the right are structure functions from screens that each include a partial $n$ set. The screens were built with 59 and 61 polynomials, for the upper and lower right plots, respectively, but they are less radially symmetric than the screen on the upper left built with only 36 polynomials.

Increasing the number of Zernike polynomials used, however, does improve the performance at higher frequencies. One can see in figure 4.9 that as $i_{max}$ increases, high-frequency correspondence of the Zernike structure function to the theoretical structure function. Hypothetically, by including an infinite number of Zernike polynomials, the Zernike phase screen structure function would exactly match the theo-

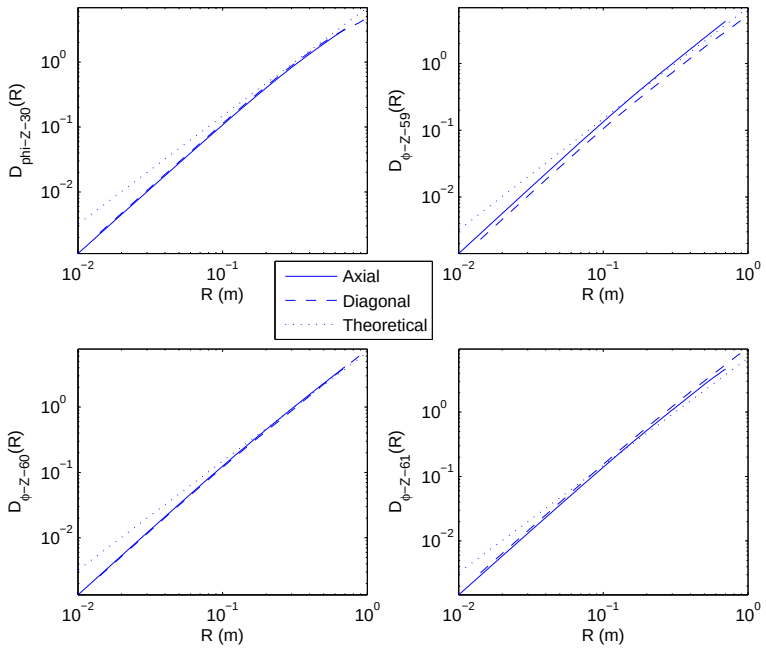


Figure 4.10: Zernike structure functions taken of screens constructed using Zernike polynomials that do (on the left) and do not (on the right) form complete sets of each radial order $n$. From left to right, and up to down, $N_Z = 35$, 59, 60, and 61, respectively. $D/r0 = 1$ and $N = 100$.

retical. It can come close for finite numbers, but at the cost of increased generation time proportional to $N_Z^2$.

*4.3.2 FFT Structure Function.* As can be seen in the upper left plot of figure 4.11, the structure function of an FFT phase screen corresponds well with theory for small $R$ (high frequencies), but falls of at large $R$ (low frequencies). The degradation of $D_{n-FFT}$ at large $R$ is due the high correlation of opposing screen edges. This correlation is caused by the lack of low frequency content in the component sinusoids The lowest non-zero frequency component has a period of exactly the screen width.

The more rapid degradation of the structure function along the diagonals (seen best in figure 4.13) is explained by the larger (by $\sqrt{2}$) diagonal frequency intervals than axial frequency intervals. The decrease in size of the longest sinusoidal period increases the rapidity of noticeable period correlation. This phenomenon is related to aliasing, and can be compensated for. As is shown in figure 4.11, by generating larger than needed phase screens and discarding all but the center portion needed for simulation, the low frequency content of an FFT phase screen can be increased.

If the generated screen size is twice the size of the outer scale, then the lower frequencies will be adequately populated and the structure function will parallel theory. This can be related to Nyquist sampling in signal processing. Given the inverse relationship of frequency to distance, the longest separations of a phase screen correspond to the lowest frequencies. In order to sample the lowest theoretical frequency, one must have a screen size of twice the theoretical limits of correlation.

For example, given a necessary screen size of $D = 1$ m with a sample size of $\Delta x = 0.01$ m, and a generated screen size of $N = D/dx = 100$, the lowest frequency included in the screen will be

$$f_L = \frac{1}{N\Delta x} = 1 \text{ m}^{-1}.$$

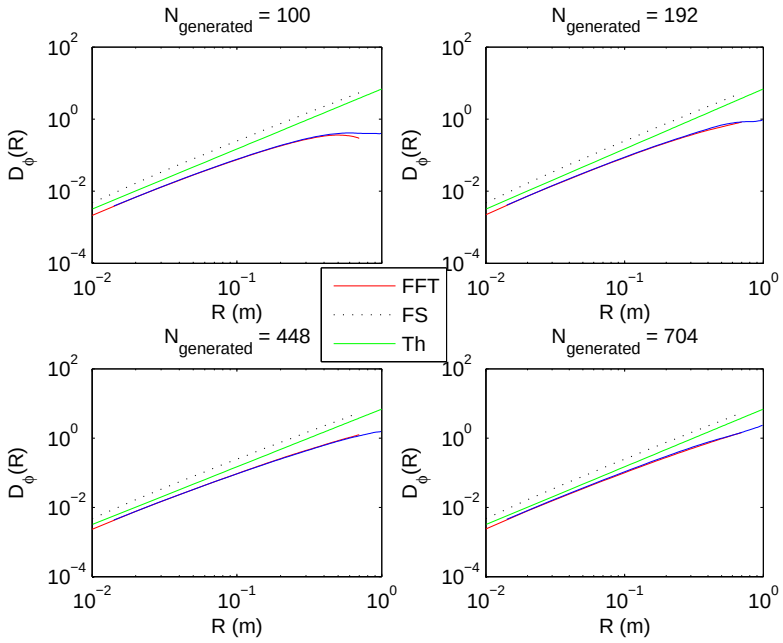


Figure 4.11: FFT structure functions for the center $100 \times 100$ pixels of screens generated using $N = 100, 193, 448$ and 704 pixels. Generating screens of excess size includes more low frequency content.

If the number of pixels (i.e. samples) $N$ is increased so that $N\Delta x = 2L_0$ where $L_0$ is the upper bound of the inertial subrange, and thus the outer limit of correlated IOR, then the lowest frequency represented becomes

$$f_L = \frac{1}{2L_0} \text{ m}^{-1}.$$

As any frequencies lower than $\frac{1}{2L_0}$ represent separations larger than are theoretically correlated in their IOR, they should not be present in the phase screen.

The center $100 \times 100$ pixels of a $2L_0 \times 2L_0$ FFT phase screen can be taken to form the desired phase screen of $D = 1$ m, which will have a structure function exactly matching theory. Unfortunately, this increase in quality requires a corresponding increase in computational inefficiency on the order of $N \log N$, which presents as an increase in generation time.

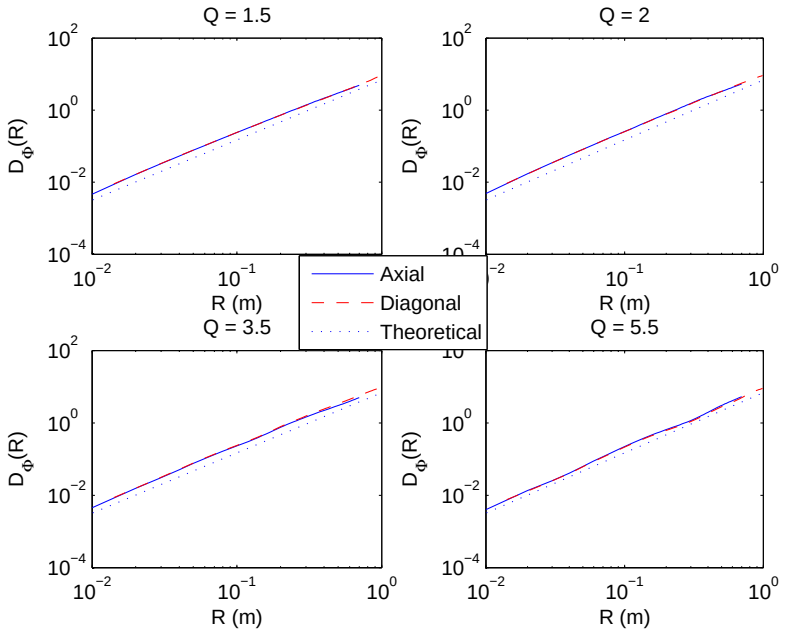


Figure 4.12: FS structure functions taken of screens constructed using various $Q$ values. There is a graceful degradation of quality as $Q$ increases, due to the increasingly sparse frequency grids.

*4.3.3 FS Structure Function.* The FS structure function parallels theory across the spectrum, though it does degrade slowly with the number of frequencies. Figure 4.12 shows the structure functions of screens constructed using various $Q$ values, corresponding to numbers of frequencies $N_K$ from 640 on the upper right ($Q = 1.5$) to 120 on the lower left ($Q = 5.5$). Note that, unlike either the FFT or the Zernke structure functions, the FFT structure function does not fall off for any extremity of frequency.

The variance from theory, presenting as 'waves' in the structure function, for higher $Q$'s is due to the sparse frequency content. There are simply not enough sinusoids to give an accurate representation of random phases. Frequencies that are not well represented show up as peaks in the structure function difference from theory. However, at no frequency band is this degradation catastrophic; all frequencies are somewhat represented, even using only 120 sinusoids, some just more so than others. This is a benefit of the logarithmic spacing of the frequency grid.

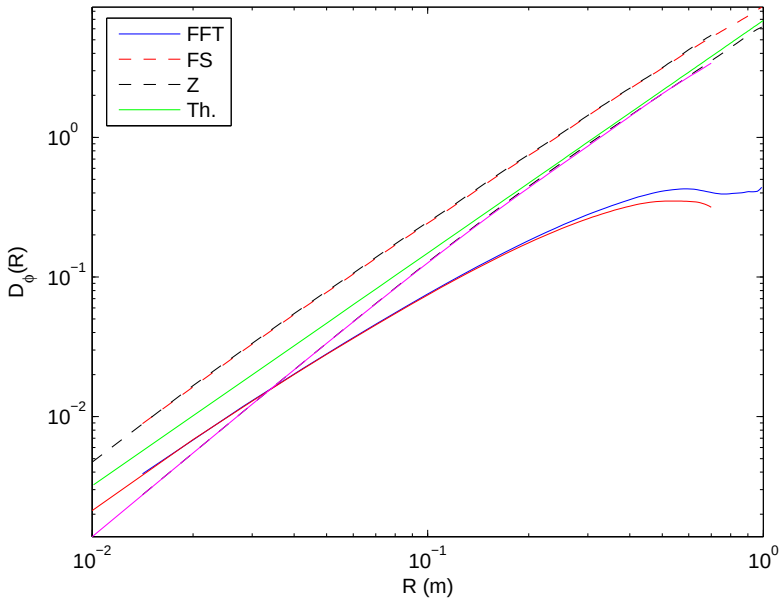


Figure 4.13: The average structure functions for FFT, FS and Zernike phase screens, calculated along the axes (on the left) and along the diagonals (on the right); and the theoretical structure function (denoted Th) for comparison. Parameters: $D/r_0 = 1$, $Q = 2.0$, $N_Z = 84$, $N = 100$.

The FS axial and diagonal structure functions almost exactly overlap, showing a high degree of radial symmetry that is only slightly affected by increasing $Q$. This is a benefit of the polar frequency grid. The radial symmetry and graceful degradation make the FS structure function useful even at high $Q$ values. However, for lower $Q$ values the FS structure function almost exactly matches theory, differing only by a constant that can be easily accounted for in code. This makes it highly applicable to all types of simulations.

## *4.4 Discussion*

Figure 4.13 shows the structure functions of the three types of phase screens discussed herein calculated along the axes and the diagonals, respectively. It also shows the theoretical structure function for comparison.

The irregularities (presenting as wavering near $R = 1$) of the diagonal of the FFT and FS structure functions are present because the sample set for the mean difference drops below acceptable statistical parameters. In fact, the largest three separations take the mean square difference of two matrices composed of only 9, 4, and 1 pixels (for $R_{max} - 3$, 2, and 1, respectively), because it is only the far corners of the screen that are overlapped. This does not happen on the axial calculation because, even at the farthest separation, there is still a set of $N$ differences to be averaged over.

*4.4.1 Maximum Phase Screen Size.* It should be noted that any phase screen generated at a size in excess of $L_0$ should have a structure function that plateaus at $R > L_0$ due to the uncorrelated random behavior of the atmospheric IOR beyond that point. I.e. the phases at separations greater than $L_0$ are completely independent, so their mean squared expected difference will reach a constant maximum.

*4.4.2 FFT vs FS Comparison.* It is interesting to note that the FS method achieves better results than the FFT method while using far fewer frequencies. In figure 4.13, the FS method uses only 416 frequencies and has a structure function parallel to theory, while the FFT method uses $N^2 = 10,000$ frequencies and still falls off at higher $R$, failing to model low frequencies. However, more time is required using the FS method.

For a 1 m ($N = 100$) screen with a conservative upper boundary $L_0 = 10$, a sampling width of $\Delta x = 0.01$ m, the FFT method would have to generate a $N_{gen} \times N_{gen}$ pixel screen where

$$N_{gen} = \frac{2L_0}{\Delta x} = 2,000$$

and $N_{gen}^2 = 4 \times 10^6$ is the total number of pixels per screen. This would require a number of operations on the order of $N_{gen} \log(N_{gen}) \approx 6,600$ and a corresponding generation time. Extrapolating from figures 4.1 and 4.3, this leads to a total genera-

tion time for two phase screens on the order of 3 to 5 seconds, the majority of which is the per-screen generation time $T_s$.

On the other hand, the FS method, generating a screen of similar quality for the same specifications, would require only $N_K = 416$ frequencies ($Q = 2.0$). The FS screen generated would be exactly the size needed so the number of operations would be on the order of $N^2 N_K = 4.16 \times 10^6$ due to the lack of optimization within the code. From figures 4.1 and 4.2, it takes 6 to 7 seconds to generate two equivalent $100 \times 100$ phase screens, most of which constitutes the initialization time $T_i$.

Because most of the time for the FS screen is spent building the kernel, subsequent phase screens can be generated at the rate of 0.5 second per pair. But because most of the FFT generation time is spent generating the screen itself, additional screens will take 3 seconds per pair. Therefore, in a situation where quality is important and more than four phase screens are required, it is more time efficient to use the FS method than the FFT method.

# V. Conclusions and Recommendations

These conclusions and recommendations stem directly from the results covered in chapter 4. They include suggested applications of the various methods as well as possible optimizations that might be made.

## *5.1 Zernike Method*

The Zernike is highly effective at capturing low frequency abberations, such as tip and tilt. It is also radially symmetric for complete radial order sets of Zernike polynomials beyond $n \geq 4$. This makes it highly effective for simulating turbulence when low frequency accuracy is required, such as for use with tracking and targeting algorithms. The main fault with the Zernike method is that it falls off at high frequencies for limited numbers of polynomials; however, including the high numbers (on the order of 500 to 1000) of polynomials needed to accurately model the high frequencies can become cumbersome as far as generation time is concerned.

## *5.2 FFT Method*

The FFT method is by far the fastest for generating phase screens, but it diverges from theory at low frequencies for screen sizes of less than $2L_0$, and becomes cumbersome at sizes near $2L_0$. This makes it good for simulating high frequency turbulence for adaptive optics algorithms, where low frequency abberations are assumed to be already removed.

## *5.3 FS Method*

The FS method takes the longest initialization time, but it has the best results across the spectrum. It does not fail catastrophically at any frequency range, instead it degrades gracefully with few frequencies modelled. FS phase screens are applicable to any scenario, but are computationally cumbersome as implemented here. The FS method is therefore well worth optimizing. One obvious area for improvement is to find a way other than a triply-nested *for* loop to construct the kernel.

Given an optimized implementation, the FS method would be universally applicable, generating quality results across the spectrum. However, even as is, the FS method has a comparable implementation time with either the Zernike or the FFT methods for the same quality of result.

Note that filenames have had underscores removed and capitals inserted for LaTex display calls. The files must be renamed to match the code before implementation.

Listing A.1: Top level code to generate all three types of phase screens. (appendix1/GeneratePhzScrnsDn.m)

```
% GeneratePhzScrnsDn.m
% Rebecca Eckert
% 15 Feb 06
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...

% GENERATEPHZSCRNSDN.M Generates Zernike, FFT  phase screens and ...
    calculates
% their structure functions given a particualr D/r0.  It also ...
    records the
% time taken to generate each screen.
%   - Iterate D/r0 = [1 3 5 10]
%   - Iterate 30 phase screens/Sturcture Functions per D/r0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
clear;clc;matlabpath(pathdef)
% INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...

% Screen Descriptors
N = 100;                        % The size in pixels of phase screens
NumScreens = 60;                % The number of phase screens to be ...
    generated
NumZernikeModes = 84;           % The number of polynomials/mode to be...
     used in
                                % generating the Zernike Phase Screen
% Atmospheric Descriptors
D = 1;                        % optic Diameter
r0 = [D]% D/3 D/5 D/10];% Fried radius
AtmSpectrum = 'VonKarmen';  % 'Kolomogorov','Tatarski','VonKarmen...
    ','ModAtm'
                                % => Used in making the FFT phase ...
                                    screen
% Large inertial subrange
lo = 0.01;                      % Inertial Subrange Inner Scale (m)
Lo = 10;                       % Inertial Subrange Outer Scale (m)
% Small Inertial subrange
% lo = 0.01;
% Lo = 10;

lambda = 0.5e-6;                % Wavelength (m)
WaveNumber = 2*pi/lambda;       %


% Reference Coordinates - Spatial Domain
```

```
R = linspace(0,D);
dx = D/N;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...

for kk = 1:length(r0)
D_r0 = D/r0(kk);
PhzFileName = ['PhzScrns_Dr0_' num2str(D_r0)]

% Ideal structure function...
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Dphi_Ideal = 6.88*(R/r0(kk)).^(5/3);

% FFT PHASE SCREEN...
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
PhzScrnFFT = zeros(N,N,NumScreens);
ind = 1;
for ii = 1:NumScreens/2
    [P1,P2] = MakeFFT_PhzScrn(D,r0(kk),Lo,lo,N);
    PhzScrn_FFT(:,:,ind) = P1;
    PhzScrn_FFT(:,:,ind+1) = P2;
    ind = ind + 2;
end
toc
% ZERNIKE PHASE SCREEN...
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note that the My_get_zernike_mode.m function only contains the ...
   first 85
% Zernike polynomial modes, starting with Piston [1].   HOWEVER, ...
   the
% Zernike_Cov matrix contains 2999x2999 covariance terms, starting...
    with tip
% and tilt [1,2] and ignoring piston.  Therefore, only 84 ...
   polynomials/modes
% are currently available, limiting the higher frequency ...
   capabilities.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...

% Zernike Coevariance Matrix
load z_cov.mat
zernike = My_get_zernike_mode(NumZernikeModes + 1,N);
zernike = zernike(2:NumZernikeModes+1,:,:);
load zernike_index
ZernikeCov = z_cov;

tic
PhzScrn_Zern = zeros(N,N,NumScreens);
for ii = 1:NumScreens
   PhzScrn_Zern(:,:,ii) = ...
       MakeZernikePhzScrn(D_r0,NumZernikeModes,ZernikeCov,N,...
          zernike);
end
```

```
toc


% Housekeepint to optimize FS routine
save(PhzFileName,'PhzScrn*','dx','Dphi_Ideal','R') % Keep phase ...
    scrns
save temp1 D r0 N kk lo Lo D_r0 NumScreens PhzFileName
clear PhzScrn*
save temp2
clear all
load temp1
% RADIAL FOURIER SERIES PHASE SCREENS...
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generates a screen similar to the FFT screen, but with ...
    logrithmic
% frequency components generated on a polar grid.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...


tic
x_p = linspace(0,.5,N);
y_p = x_p;
Q = 2.0
kern = My_make_polar_kern(x_p,y_p,r0(kk),lo,Lo,Q);
toc
tic
ind = 1;
for ii=1:NumScreens/2
    [phi1,phi2] = make_polar_screen(kern);
    PhzScrn_FS(:,:,ind) = phi1-mean(phi1(:));
    PhzScrn_FS(:,:,ind + 1) = phi2-mean(phi2(:));
    ind = ind + 2;
end
toc

save(PhzFileName,'-append','PhzScrn_FS','x_p','Q')
clear PhzScrn_FS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...

load temp2
end
```

Listing A.2: Generate a Zernike phase screen. (appendix1/MakeZernikePhzScrn.m)

```
function [ PhzScrn ] = MakeZernikePhzScrn(D_r0,numModes,z_cov,...
    numPix,zernike)
% MAKEZERNIKEPHZSCRN Creates a random phase screen using the ...
    Zernike
```

```
% technique. The phase screen is generated in cartesian ...
    coordinates on a
% rectangular grid.
% NOTE that the Zernike will be a circle inscribed on the ...
    rectangular
% frequency grid.
% NOTE that the My_get_zernike_mode.m function only contains the ...
    first 85
% Zernike polynomial modes, starting with Piston [1].

% Generate Zernike Phase screen
if (isempty(numPix) == 1)
    error('MakePhzScrn of type = 'ZERNIKE' requires input SIZE in ...
        pixels')
elseif (isempty(D_r0) == 1)
    error('MakePhzScrn of type = 'ZERNIKE' requires input ...
        atmospheric D/r0')
else npix = numPix;
end
if (mod(npix,2) == 1)
[x y] = meshgrid(-floor(npix/2):floor(npix/2));
else [x y] = meshgrid(npix/2:npix/2-1);
end
r = sqrt(x.^2 + y.^2);

% Zernike Coevariance Matrix
% load z_cov.mat
% zernike = My_get_zernike_mode(numModes + 1,npix);
% zernike = zernike(2:numModes+1,:,:);

a = chol(z_cov)'*randn(length(z_cov),1);
a_ps = a(1:numModes)*D_r0^(5/3);
Aps = (repmat(a_ps,[1 npix npix]));

PhzScrn = squeeze(sum(Aps.*zernike,1));
```

Listing A.3: Generate FFT phase screen.(Using Von Karmen spectral model.) (appendix1/MakeFFTPhzScrn.m)

```
function [ PhzScrn1 PhzScrn2 ] = MakeFFT_PhzScrn(D,r0,Lo,lo,N)
% MAKEFFT_PHZSCRN Creates a random phase screen using  the FFT ...
    technique
% and the Von Karmen spectral model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...

Dr0 =D/r0;

[p q] = meshgrid((-N/2:N/2-1));
r1 = randn(N) + j*randn(N);
```

```
G = (p.^2 +q.^2 + (D/Lo)^2).^(-11/12).*exp(-0.563*(lo/D)^2*(p.^2+q...
    .^2));
G(N/2+1,N/2+1)=0;
F = fftshift(fft2(fftshift(G.*r1)));
F = F*0.1517*Dr0^(5/6);

PhzScrn1 = real(F);
PhzScrn2 = imag(F);


% isc(PhzScrn1);colorbar


% OLD CODE
% if (isempty(PHI) == 1)
%     error('MakePhzScrn of type = 'FFT' ...
% requires input PHI = atmospheric spectrum (eg, Von Karmen)')
% elseif (isempty(numPix) == 1)|(numPix < length(PHI))
%     npix = length(PHI);
% else
npix = length(PHI);
% end
% S = sqrt(PHI);
% Generate Gaussian random phase screen using FFT technique
g = randn(npix);
PhzFFT = g.*PHI;
G = fftshift(ifft2(fftshift(PhzFFT)));
PhzScrn1 = real(G);
PhzScrn2 = imag(G);
```

Listing A.4: Generate FS phase screen.(appendix1/MakePolarScreen.m)

```
%=========================================================================...

% Creates a pair of phase screens with subharmonics sampled over a...
    pseudo-
% polar grid.
%=========================================================================...

function [phi1,phi2] = make_polar_screen(kern)

[Ny,Nx,Ns] = size(kern);
comp = randn(1,1,Ns) + j*randn(1,1,Ns);
phi = zeros(Ny,Nx);
for i1 = 1:Nx
    for i2 = 1:Ny
        phi(i2,i1) = phi(i2,i1) + sum(comp.*kern(i2,i1,:),3);
    end
end
% phi_s = sum(kern.*shiftdim(repmat(comp,[1,Ny, Nx]),1),3);

phi1 = real(phi);
```

```
phi1 = phi1-mean(mean(phi1));
phi2 = imag(phi);
phi2 = phi2-mean(mean(phi2));

return
```

Listing A.5: Generate FS kernel.(appendix1/MyMakePolarKern.m)

```
function kern = make_polar_ker(x,y,r0,l0,L0,Q)

% % Q = 2.0;   % Factor to set number of points
next_inc = 10;  %used to display progress
prog_step = 10; % used to display progress
max_iter = 1000;    % num of iterations for midpoint selection
tol = 1e-12;    % tolerance for midpoint selection



dx = x(2) - x(1);
dy = y(2) - y(1);
Nx = length(x);
Ny = length(y);

kappa0 = 1/L0;
% Set max freq also based on sampling
kappam1 = 5.92/l0;
kappam2 = 2*pi/(2*dx);
kappam = min(kappam1,kappam2);

k_max = kappam;
k_min = 2*pi/5/L0; % half the lowest frequency demanded by the ...
    outer scale

disp('Forming polar grid kernel');

log_k_min = log10(k_min);
log_k_max = log10(k_max);
delta_log = log_k_max - log_k_min;
num_freq_pts = ceil(delta_log/log10(Q));
log_k_pts = linspace(log_k_min,log_k_max,num_freq_pts);
k_pts = 10.^log_k_pts;   % These are the radial frequency values ...
    used


k_low  = k_pts(1:end - 1);
k_low2  = (k_pts(1:end - 1)).^2;
k_high = k_pts(2:end);
k_high2 = (k_pts(2:end)).^2;
delta_k = diff(k_pts);

% need to find the 2D power in each annular segment from PSD
d_phi2 = zeros(1,length(delta_k));
for i1 = 1:length(delta_k)
```

```
        d_phi2(i1) = int_vK2D2(r0,l0,L0,k_pts(i1),k_pts(i1 + 1));
end


%%%%%%%%%% loop to pick point to use for sinusoids
% define differential area
k_area = pi*(k_high2 - k_low2);
% % provide a guess at the desired midpoint and then iterate to ...
    choose best
% % midpoint as defined by place where power in annular segment is...
    equal to
% % product of area and PSD at that midpoint
k_mid = 0.5*delta_k + k_low;
% cnt = 0;
% max_iter = 1000;
% tol = 1e-12;
% for index = 1:length(k_mid)
%     % seed binary searc with endpoints and midpoint
%     k_lower = k_pts(index);
%     k_upperu = k_pts(index + 1);
%     B = k_mid(index);
%     phi_mid = vK_phi(B^2,r0,l0,L0);
%     ip = phi_mid*k_area(index);
%     while (cnt <= max_iter) && (abs(ip - d_phi2(index)) > tol)
%         if ip < d_phi2(index)
%             k_upper = B;
%             B = k_lower + .5*(B - k_lower);
%         else
%             k_lower = B;
%             B = k_lower + .5*(k_upper - B);
%         end
%         phi_mid = vK_phi(B^2,r0,l0,L0);
%         ip = phi_mid*k_area(index);
%         cnt = cnt + 1;
%         if cnt == max_iter
%             disp('Warning maximum iterations reached in kappa ...
    calc');
%             disp(['        tol: ' num2str(abs(ip - d_phi2(index)...
    ))])
%         end
%     end
%     k_mid(index) = B;
%     cnt = 0;
% end

% this can be changed later to include a progression of theta ...
    slices
num_theta = 32;  % number of slices in theta
theta = linspace(pi/num_theta,2*pi - pi/num_theta,num_theta);
theta = repmat(theta,[1,length(k_mid)]);

k = [];
```

```
    d_area = [];
    for i1 = 1:length(k_mid)
        k = [k repmat(k_mid(i1),[1,num_theta])];
        d_area = [d_area repmat(k_area(i1)/num_theta,[1,num_theta])];
 95 end

    kx = k.*cos(theta); % cartesian points for given theta value
    ky = k.*sin(theta);

100 sqrt_phi = sqrt(vK_phi(k.^2,r0,l0,L0));
    sqrt_area = sqrt(2*pi*d_area);

    tot_iter = Nx*Ny*length(k);
    kern = zeros(Nx,Ny,length(k));
105 for i1 = 1:Nx
        for i2 = 1:Ny
            for i3 = 1:length(k)
                    kern(i2,i1,i3) = sqrt_phi(i3)*sqrt_area(i3)*...
                        exp(j*(x(i1)*kx(i3) + y(i2)*ky(i3)));
110         end
            if i1*i2*i3*100/tot_iter > next_inc
                disp(num2str(next_inc));drawnow
                next_inc = next_inc + prog_step;
            end
115     end
    end
    disp('done!');
    % test code
    % output the resulting set of kappa values for analysis
120 ktst2 = logspace(log10(k_pts(1)),log10(k_pts(end)),1000);
    ptst = vK_phi(ktst2.^2,r0,l0,L0);
    % % figure(2); clf;
    % % subplot(1,2,1);
    % % loglog(ktst2,ptst,'g-.');
125 % % hold on;
    % % loglog(k_pts,vK_phi(k_pts.^2,r0,l0,L0),'k+ ');
    % % loglog(k_mid,vK_phi(k_mid.^2,r0,l0,L0),'ro ');
    % % axis tight;
    % % subplot(1,2,2);
130 % % semilogy(ktst2,ptst,'g-.');
    % % hold on;
    % % semilogy(k_pts,vK_phi(k_pts.^2,r0,l0,L0),'k+ ');
    % % semilogy(k_mid,vK_phi(k_mid.^2,r0,l0,L0),'ro ');
    % % axis tight;
135 % % hold off;

    kmtstsq = sqrt(k_mid);
    ptstsq = sqrt(vK_phi(k_mid.^2,r0,l0,L0));

140 % end test code
    return
```

Listing A.6: Calculate Von Karmen spectral model.(appendix1/vKphi.m)

```
% calculate the von Karman phase spectrum for a given kappa, r0, ...
    l0, and L0
function PHI = vK_phi(kappa2,r0,l0,L0)
% kappa_max = 5.92/l0;
% kappa_max2 = kappa_max^2;
% kappa_min = 1/L0;
% kappa_min2 = kappa_min^2;
% PHI = 0.4916693*r0^(-5/3)*exp(-kappa2./kappa_max2).*...
%     (kappa2 + kappa_min2).^(-11/6);

PHI = 0.4916693*r0^(-5/3)*exp(-kappa2./(5.92/l0).^2).*...
    (kappa2 + (1/L0).^2).^(-11/6);
return
```

# *Bibliography*


1. Andrews, Larry C. and Ronald L. Phillips. *Laser Beam Propagation through Random Media*. SPIE Optical Engineering Press, Bellingham, Washington, 1st edition, 1998.

2. Buffington, Jonanthan C. "Wavefront Curvature Sensing from Image Projections", September 2006. Air Force Institute of Technology Doctoral Prospectus.

3. Goda, Matthew. "EENG 716 Imaging Through Turbulence", Summer 2005. Air Force Institute of Technology Class Notes.

4. Goodman, Joseph W. *Introduction to Fourier Optics*. Roberts and Company Publishers, 4950 S. Yosemite Street, F2 no.197, Greenwoood Village, CO 80111, 3rd edition, 2005.

5. Leon-Garcia, Alberto. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley Longman, Reading, MA, 2nd edition, May 1994.

6. Mathworks, Inc. "MATLAB Help File". Electronic, May 2004. MATLAB version 7.0.0.19920 (R14).

7. Noll, Robert J. "Zernike Polynomials and Atmospheric Turbulence". *Journal of the Optical Society of America*, 66(3):207–211, March 1976.

8. Oppenheim, Alan V., Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing*. Prentice Hall, Inc., Upper Saddle River, New Jersey 07458, 2nd edition, 1999.

9. Roggemann, Michael C. and Byron M. Welsh. *Imaging Through Turbulence*. Laser and Optical Science and Technology. CRC Press, Boca Raton, FL, 1st edition, 1996.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704–0188*



| 1. REPORT DATE *(DD–MM–YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From — To)* |
|---|---|---|
| 23–03–2006 | Master's Thesis | Sept 2004 — Mar 2006 |

**4. TITLE AND SUBTITLE**

Polar Phase Screens:
A Comparison with Other Methods of
Random Phase Screen Generation

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Eckert, Rebecca J., 2d Lt, USAF

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way
WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT/GE/ENG/06-18

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Dr. Troy Rhoadarmer
AFRL/DES
3550 Aberdeen Ave SE
Kirtland Air Force Base, NM 87117 - 5776 DSN: 246-5022

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution unlimited

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

This research provides the first organized comparison of random phase screen generation methods, including logarithmic polar Fourier series, using structure functions. Random phase screens are essential elements of simulating light propagation through turbulent media. In order to be effective, they must accurately reflect theory and be practical to implement. This research explains and evaluates three methods of generating random phase screens: using a Fourier series upon a polar frequency grid with logarithmic spacing; using the fast Fourier transform, with its Cartesian frequency grid; and using Zernike polynomials. It provides a comparison of the Polar Fourier Series technique with the two more common techniques (Fast Fourier Transform and Zernike); with the end result of giving the users enough information to choose which method best fits their needs. The evaluation criteria used are generation time (usability) and phase structure function (accuracy).

**15. SUBJECT TERMS**

Random phase screen; polar Fourier series; fast Fourier transform; Zernike polynomial; structure function; atmospheric simulation; optical turbulence.

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Matthew Goda , Lt. Col., USAF (ENG) |
| U | U | U | UU | 78 | 19b. TELEPHONE NUMBER *(include area code)* (937) 255-2024 |