Simulating Speckle with *Mathematica*®

Library of Congress Cataloging-in-Publication Data

Names: Goodman, Joseph W., author.
Title: Simulating speckle with Mathematica / Joseph W. Goodman, Stanford University.
Description: Bellingham, Washington : SPIE, [2022] | Includes bibliographical references.
Identifiers: LCCN 2022030726 | ISBN 9781510656543 (paperback) | ISBN 9781510656550 (pdf)
Subjects: LCSH: Speckle--Computer simulation. | Speckle--Mathematical models. | Mathematica (Computer file)
Classification: LCC QC427.8.S64 G659 2022 | DDC 535/.470113--dc23/eng20220919
LC record available at https://lccn.loc.gov/2022030726

Published by
SPIE
P.O. Box 10
Bellingham, Washington 98227-0010 USA
Phone: +1 360.676.3290
Fax: +1 360.647.1445
Email: books@spie.org
Web: www.spie.org

The cover image was simulated from the painting "The Audience" by Gene Walch.

Printed in the United States of America.
First printing 2022.
For updates to this book, visit http://spie.org and type "PM355" in the search field.

Simulating Speckle with *Mathematica*®

Joseph W. Goodman

SPIE PRESS
Bellingham, Washington USA

Simulating Speckle with *Mathematica*®

Joseph W. Goodman
Stanford University

Table of Contents

1. **Introduction**
    1.1. *Mathematica* Background
    1.2. Speckle Background
    1.3. Methods for Simulating Speckle
2. **First-Order Statistics of Speckle Amplitude**
    2.1. Speckle as a Sum of Independent Random Complex Phasors
    2.2. Amplitude Statistics of the Sum of Many Random Phasors with Unit Lengths and Random Phases
    2.3. Amplitude Statistics of the Sum of a Large Number of Unit-Length Random Phasors and One Large Phasor
    2.4. Amplitude Statistics of the Sum of a Small Number of Unit-Length Random Phasors
3. **First-Order Statistics of Speckle Intensity**
    3.1. Intensity Statistics of the Sum of Many Random Phasors with Unit Lengths and Random Phases
    3.2. Intensity Statistics of the Sum of a Large Number of Unit-Length Random Phasors and One Large Phasor
    3.3. Intensity Statistics of the Sum of a Small Number of Unit-Length Random Phasors
    3.4. Intensity Statistics for Sums of Independent Speckle Patterns
    3.5. Intensity Statistics of Partially Developed Speckle
4. **Simulation of Speckle in Optical Imaging**
    4.1. Generation of a Discrete Diffuser Array with Correlated Phases
    4.2. Speckle in an Imaging Geometry
    4.3. The Autocorrelation Function of Speckle Intensity

4.4. The Power Spectrum of Speckle Intensity
4.5. Effect of Speckle on Resolution
4.6. Speckle in Color Images

5. **Simulation of Speckle in Free-Space Propagation**
5.1. The Diffuser
5.2. The Fresnel Transfer Function Approach
5.3. The Fresnel Transform Approach

6. **Speckle at Low Light Levels**
6.1. Photocount Image of a Uniform Intensity
6.2. The Negative-Binomial Distribution
6.3. Photocount Image for a Speckle Pattern with Uniform Statistics
6.4. Photocount Image for a Diffuse Structured Object

7. **Speckle Phase Vortices**
7.1. Generating the Speckle Pattern
7.2. Finding the Zeros of Intensity
7.3. Phase Behavior in the Vicinity of the Zeros of Intensity

8. **Polarization Speckle**
8.1. The Polarization Ellipse and the Degree of Polarization
8.2. Generating the Two Polarization Components
8.3. Generating a Filtered Speckle Pattern
8.4. Generating the Polarization Ellipses
8.5. Visualizing Polarization Speckle

9. **Speckle Simulation for Metrology**
9.1. Measurement of In-Plane Displacement
9.2. Electronic Speckle Pattern Interferometry
9.3. Phase-Shifting Speckle Interferometry

Appendix A – Some Subtleties in Speckle Simulation With the 4*f* System
A.1. Effects on the Speckle Contrast
A.2. Simulation With a Smoothed Phase
A.3. Simulation With an Unsmoothed Phase

Appendix B – Some Subtleties in Dealing With *Mathematica* Images
B.1. Dimensions of Data Arrays and Images
B.2. Images When the Data Range Exceeds (0, 1)

B.3. Effect of Using `ImageAdjust[]` on an Image
B.4. Arrays With Bipolar Values
B.5. A Problem Encountered When Starting With an Image

Acknowledgements

References

1. Introduction

The speckle phenomenon is ubiquitous in many fields of science and technology. Speckle phenomena can be seen in many different imaging modalities, including acoustical imaging (e.g., medical ultrasound) and microwave imaging (e.g., synthetic-aperture radar imaging). This book focuses on simulating optical speckle with Mathematica®, but the same methods used can in many cases be applied to other imaging modalities.

The reader may wonder why *Mathematica* has been chosen as the software package for this book. There are several reasons for this choice. First, and most important, *Mathematica* allows the interspersing of both continuous and discrete calculations under one umbrella. Second, using *Mathematica*, text, code, and illustrations can be included in the same document. Third, using *Mathematica* we can create dynamic figures, parameters of which the user can change at will. However, such manipulation cannot be performed in the printed version of the book, so we have avoided manipulable figures in what follows and replaced them by arrays of static figures. Lastly, this author loves *Mathematica* for its flexibility and comprehensiveness. It seems there are almost an infinite set of capabilities of the program, many of which lie hidden for the novice user but which gradually are revealed as the use of the program increases. This book has been written entirely in *Mathematica*. It can be read with the full program *Mathematica* or with the free program *Wolfram Player* available for download from the Wolfram site. The *Mathematica* files for all chapters can be found at the following URL: http://spie.org/Samples/Pressbook_Supplemental/PM355_sup.zip This book is meant as a companion to the book *Speckle Phenomena in Optics: Theory and Applications*, *2nd Edition*, published by SPIE Press (Ref. [1]). An extensive list of references can be found in that book.

1.1 *Mathematica* Background

For some background that will be helpful to the reader new to *Mathematica*, see Ref. [2]. There are many books that describe the capabilities of *Mathematica*. Ref. [3] is especially comprehensive. Our goal here is to not only present methods for simulating speckle in various situations and applications, but also to introduce the reader to the capabilities of *Mathematica*. In what follows we present some salient features of *Mathematica* that will help the novice get started.

1. Built-in functions of *Mathematica* always begin with upper-case letters. If the command consists of a concatenation of two separate words, both words must begin in upper-case letters.
2. In order to avoid confusion with built-in commands, user-defined functions should usually begin with a lower-case letter. However, it is permissible to begin commands with an upper-case letter in a font that is different than `Source Code Pro`, the usual font for *Mathematica* input. As an example, $\mathcal{N}$ is not the same as `N`.
3. *Mathematica* reduces an input that is a rational number to the equivalent simplest rational number possible. The ratio `32/6` yields `16/3`, but does not produce an approximate decimal result. To obtain a decimal result, place a decimal point at the end of either the numerator or the denominator. Thus, `32./6` yields the result `5.33333`.
4. The arguments of functions must be enclosed in square brackets `[ ]`. Curly brackets `{ }` are reserved for containing lists, which can represent vectors and matrices, and are also used for variable ranges in, for example, plot commands. Parentheses `( )` are reserved for grouping mathematics.
5. When a function is first defined, each independent variable on the left of the definition must be followed by an underbar `_`. Underbars are not used on the right of the definition, or when the function is called by later code.
6. The symbol `:=` is a delayed equality, for which the operation of equality is made only when the symbol on the left is called in later programming.
7. The symbol `%`, when used as the argument of a function, represents the last previous output, which is now used as the input to this new function. `%%` represents the second-to-last output, etc.
8. Often certain symbols may be used more than once in a program, with their meaning having changed between uses. If `h` is the symbol, then the command `Clear[h]` erases the definition of the symbol `h` and allows a new definition to be made. To clear more than one symbol, for example, `h` and `g`, use `Clear[h,g]`. To clear all symbols and functions that have been previously defined, use the command `Clear["Global`*"]`.
9. The symbol `;` at the end of an expression to be evaluated indicates that the evaluation should take place, but the result of the evaluation should be suppressed until it is needed later in the program.
10. The symbol `I` (or i) is used in *Mathematica* for the imaginary constant $\sqrt{-1}$. The symbol `D[]` indicates a derivative operation.
11. To execute a command in *Mathematica*, place the cursor in the same cell (cells are indicated by vertical lines on the right of the notebook) as the command, and press the shift key and the return key simultaneously.
12. The symbols `&&` mean a logical *and*; the symbols `||` mean a logical *or*.

the sum is the squared magnitude of the amplitude. When referring to the complex-valued optical field, we use the term complex amplitude.

In optics, speckle arises when light is reflected from a rough surface or is transmitted through a diffuser that jumbles the phase at each object point by an unpredictable amount. The contributions from various scattering regions on the object then generate a multitude of complex wavelets that interfere to produce speckle. When the reflected or transmitted light propagates to an observation plane some distance away, complicated fluctuations of amplitude, phase, and intensity occur in that plane due to random interference. These fluctuations are what we refer to as speckle. If the phase perturbations introduced by the object equal or exceed 2π radians, we say that the speckle is *fully developed*. If on the other hand the phase fluctuations introduced by the object are less than 2π radians, the resulting speckle is called *partially developed*. In some cases, one scattered wavelet may be much larger than the others, in which case the speckle is neither fully developed nor partially developed, but rather requires a special development to understand the statistics of the observed light amplitude or intensity.

It is important to remember that when we speak of the statistics of speckle, we are speaking of fluctuations over an ensemble of macroscopically similar but microscopically different rough surfaces or diffusers. The resulting perturbed wavefront is unchanging for any one surface or diffuser, but changes as different rough surfaces or different diffusers are introduced. Since we do not know the fine-scale structure of the surface fluctuations, the best we can do is specify statistics over an ensemble of possible surfaces. To experimentally discover statistical properties of the speckle, either many different microscopically different reflecting or transmitting structures must be introduced sequentially, or, in the case of speckle that is spatially ergodic (i.e., statistically similar over a wide region of the speckle pattern), spatial averages should yield the same results as an ensemble average.

In Chapter 2, we explore the first-order statistical properties of the amplitudes of sums of random phasors. In Chapter 3, we explore the first-order statistical properties of the intensity of such sums (i.e., the squared magnitude of the resultant phasor). Later chapters explore the properties of speckle in images, speckle in free-space propagation, speckle at low light levels, phase vortices in speckle, polarization speckle, and speckle in certain metrology techniques.

Note that the theoretical results for the probability density functions of amplitude or intensity, as found in Ref. [1], are based on the assumption of an infinite number of random phasor contributions. Obviously we can not simulate an infinite number of random phasors on the computer, but we can choose a large finite number. Our results, then, will yield information on how well the theoretical predictions of the statistics of amplitude and intensity match the results based on a large but finite number of phasors.

1.3 Methods for Simulating Speckle

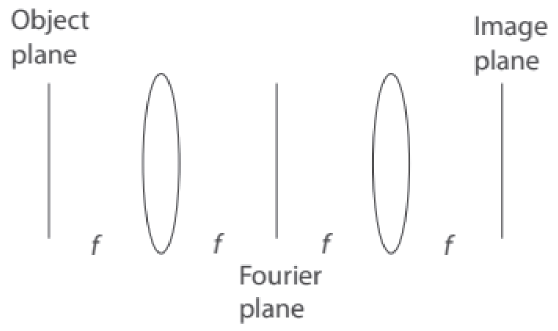
In Chapters 2 and 3, we simulate the first-order statistics (i.e., the speckle at a single point in space or time) by summing a large number of complex phasors. Assumptions are made in various sections about the statistics of the phase of the phasors or about the number of phasors. Histograms of the various results are computed and compared with the theoretical results valid for an infinite number of phasors. The ideas behind these simulations are quite straightforward. We simply sum a finite number of complex phasors and examine the statistics of amplitude (Chapter 2) or intensity (Chapter 3) by calculating histograms of the results of a large number of independent trials.

Calculating a 2D speckled image is a more complex issue, since the image speckles generally must remain correlated in intensity over a number of adjacent pixels, thus generating speckles of finite width. We can identify two different methods for generating a field of intensity speckles. One we call the *physics-based method* and the second we call the *statistics-based method*. I thank Prof. James Fienup of the University of Rochester for suggesting that I add this section to this introduction. The two different approaches will now be discussed.

Physics-Based Methods

In the physics approach we generate speckle by simulating the optical system through which light passes from input to output, or in some cases the physical laws that govern the propagation of light from one plane to another.

The majority of the physics-based speckle simulation approaches in the chapters that follow are based on what is known as a $4f$ optical system, as shown below.



One can think of the first lens as the objective of a microscope and the second lens as a tube lens, although we have artificially held the magnification to unity by virtue of the two equal focal lengths. Thus, the $4f$ system is perhaps more representative of imaging systems than might be thought at first glance. For a more complex optical system, the exit pupil of the system plays the same role as a Fourier-plane stop in the $4f$ system. The beauty of the $4f$ system from the simulation point-of-view is that the complex amplitude distribution of the light in the focal plane is simply the scaled Fourier transform of the complex field distribution leaving the object plane, and we can use discrete Fourier trans-

2. First-Order Statistics of Speckle Amplitude

By *first-order* statistics of speckle we mean the statistics observed at a point in space or a point in time, with the statistics being over an ensemble of rough surfaces or rough diffusers. First we consider the statistics of speckle complex amplitude, relevant when using ultrasound or microwave illumination of surfaces that are rough on the scale of their individual wavelengths. For such imaging modalities, it is possible to measure both the magnitude and the phase of the wavefields. In Chapter 3 we turn to the statistics of speckle intensity, which is the most relevant quantity for the optical region of the spectrum, where a detector can measure only intensity unless some form of interferometry is used.

2.1 Speckle as a Sum of Independent Random Complex Phasors

The statistics of speckle at a point are the same as the statistics of a sum of complex phasors with independent amplitudes and phases. Let the symbol c_k represent the k^{th} element in an array of $\mathcal{N}$ different complex phasors of the form

$$c_k = a_k \exp(i\, \phi_k), \qquad k = 1, \cdots, \mathcal{N},$$

where a_k is a non-negative amplitude and ϕ_k is a phase. Each phasor represents an independent contribution to the complex amplitude of the field at a point in space or time. We assume that a_k and ϕ_k are random variables drawn from a statistical ensemble, and that they are statistically independent of each other and statistically independent of all other random variables occurring in the array of phasors. We then form the normalized sum

$$s = \frac{1}{\sqrt{\mathcal{N}}} \sum_{k=1}^{\mathcal{N}} c_k = \frac{1}{\sqrt{\mathcal{N}}} \sum_{k=1}^{\mathcal{N}} a_k \exp(i\, \phi_k) = \frac{1}{\sqrt{\mathcal{N}}} \sum_{k=1}^{\mathcal{N}} a_k \cos\phi_k + i \frac{1}{\sqrt{\mathcal{N}}} \sum_{k=1}^{\mathcal{N}} a_k \sin\phi_k,$$

where the normalization by $1/\sqrt{\mathcal{N}}$ is introduced to preserve a finite second moment of the sum. We next explore the *Mathematica* representation of random phasor sums.

2.2 Amplitude Statistics of the Sum of Many Random Phasors with Unit Lengths and Random Phases

From Ref. [1] we have many analytic results regarding the statistics of random-phasor sums. Here we will develop several discrete *Mathematica* calculations that are approximations to those analytical results. As mentioned previously, the approximations stem primarily from the fact that in simulations we can only include a finite number of random phasors, while the analytical results strictly hold only for an infinite number of such phasors.

The first case considered will be one for which the lengths of all phasors in the sum are unity (aside from the $1/\sqrt{\mathcal{N}}$ normalization) and the individual phases are uniformly distributed on the interval $(-\pi, \pi)$. The appropriate sections of Ref. [1] for this case are 2.1 and 2.2. Our goal is to calculate discrete approximations to the probability density functions of the real part, the imaginary part, and the magnitude of s. Note that the magnitude of s is the same as the length $\mathcal{A}$ of the resultant phasor. In this section we consider only a large number ($\mathcal{N} \gg 10$) of phasors in the sum.

Two parameters need to be chosen at the start: the first is $\mathcal{N}$, the number of independent phasors in the sum defining s. The second is $\mathcal{M}$, which is the number of times the simulation is run with independent phases for the individual phasors contributing to the sum. This multitude of runs collects the statistical data we desire. The value of $\mathcal{M}$ = 10,000 is usually used throughout the entire notebook. If you wish to obtain more accurate histograms with less error, increase the value of $\mathcal{M}$, but you do so at the cost of longer computation times. For example, if you set $\mathcal{M}$ = 100,000, the histograms are much improved, but you may need to evaluate this notebook with an overnight run. Sometimes we let the number of phasors $\mathcal{N}$ equal the number of trials $\mathcal{M}$ for simplicity.

We must construct a loop that will calculate the real and imaginary parts of the random phasor sum s over $\mathcal{M}$ trials, each with a different realization of the random phase sequence. We first construct two tables into which the results of the calculation can be placed, one for the real part of s and the other for the imaginary part of s. Initially, the entries in the two tables are all zeros.

We construct a loop that will calculate a sequence of $\mathcal{M}$ values of the sum s, each with a different realization of the random phase sequence. The code is explained as follows:

Line 1- This clears the values of variables that will be used.

Line 2 - This specifies the value for $\mathcal{N}$ (outside the loop). We choose $\mathcal{N}$ = 100 to assure that the results will be valid for a large number of phasors. Also, the larger $\mathcal{M}$, the more accurate our estimates of the statistics will become. The computation time required for $\mathcal{M}$ = 10,000 trials

2.4 Amplitude Statistics of the Sum of a Small Number of Unit-Length Random Phasors

As a matter of curiosity, we consider results similar to those above, but for small numbers of phasors in the sum. We will compare the results obtained to those in Fig. 2.7 of Ref. [1].

The loop we constructed in Section 2.1 is perfectly capable of finding results when the number of phasors is small—it is simply a matter of choosing a small value for $\mathcal{N}$.

Here we wish to calculate histograms of the length of the phasor $\mathcal{A}$ for several different small values of $\mathcal{N}$. For that reason we replace the outer `For[]` loop with one that cycles the calculation through several values of the index $\mathcal{N}$ representing the number of phasors added. The value of $\mathcal{N}$ starts with $\mathcal{N} = 1$ and terminates after $\mathcal{N} = \mathcal{N}0$, the largest value of $\mathcal{N}$ of interest. The value of $\mathcal{M}$ (the number of trials) remains 10,000. Three tables of zeros are defined, again for the arrays `real` and `imag` but also for a new array `hist`, which is $\mathcal{N}0$ elements long ($\mathcal{N}0 = 6$ in this example) and will hold the $\mathcal{N}0$ different histogram lists generated by the outer `For[]` loop. When the histogram list for the $\mathcal{N}$th phasor is generated, it is stored in the $\mathcal{N}$th element of `hist`, i.e., `hist[[𝒩]]`:

```
In[ ]:= Clear[𝒩, ℳ, 𝒩0 , real, imag, hist, phi, t]
        𝒩0 = 6; ℳ = 10000;
        real = Parallelize[Table[0, {k, 1, ℳ}]];
        imag = Parallelize[Table[0, {k, 1, ℳ}]];
        hist = Parallelize[Table[0, {k, 1, 𝒩0}]];
        For[𝒩 = 1, 𝒩 ≤ 𝒩0, 𝒩++,
         For[i = 1, i ≤ ℳ, i++,
          phi = Parallelize[Table[RandomReal[{-π, π}], {k, 1, 𝒩}]];
          t = (1 / Sqrt[𝒩]) * Exp[I * phi];
          real⟦i⟧ = Total[Re[t]];
          imag⟦i⟧ = Total[Im[t]]];
         hist⟦𝒩⟧ = Histogram[Sqrt[real^2 + imag^2], 100, "PDF"];]
```

Next we wish to compare these histograms with more exact results calculated by numerical integration. As shown in Ref. [1], Section 2.5, the probability density function of the amplitude $\mathcal{A}$ of the sum of $\mathcal{N}$ unit-amplitude, random-phase phasors is given by the following equation:

$$p(\mathcal{A}) = 4\pi^2 \mathcal{A} \int_0^\infty r J_0\left(\frac{2\pi r}{\sqrt{\mathcal{N}}}\right)^{\mathcal{N}} J_0(2\pi \mathcal{A} r)\, dr,$$

where J_0 is the Bessel function of the first kind, order 0. We now calculate (using numerical integration) and plot $p(\mathcal{A})$ for $\mathcal{N} = 1$ through 6 and superimpose those plots as red curves on top of the corresponding histograms. Note that for $\mathcal{N} = 1$, the amplitude will be a constant equal to unity and the probability density function will be a delta function at $\mathcal{A} = 1$. *Mathematica* does its best to represent that delta function. The *Mathematica* code representing the expression for $p(\mathcal{A})$ for an arbitrary value of $\mathcal{N}$ follows:

```
In[ ]:= Clear[𝒩, k]
        k[𝒜_, 𝒩_] := Assuming[𝒜 ∈ Reals && 𝒜 > 0,
          4 * π^2 * 𝒜 * NIntegrate[r * (BesselJ[0, 2 * π * r / Sqrt[𝒩]]) ^ 𝒩 * BesselJ[0, 2 * π * 𝒜 * r],
            {r, 0, Infinity}, MaxRecursion → 20, PrecisionGoal → 100]]
```

The command `MaxRecursion->20` instructs the numerical integration procedure to recursively subdivide the integration interval no more than 20 times. `PrecisionGoal->100` instructs the procedure to aim for a relative error of 100 digits. That is, the error in calculating a quantity g should aim for error $\leqslant |g| \times 10^{-100}$. These commands are included to promote accuracy in the numerical integration.

We can now calculate (and suppress) the plots obtained by numerical integration for $\mathcal{N} = 1$ to 6. As $\mathcal{N}$ increases, the probability density function approaches a Rayleigh density, which is the result for $\mathcal{N} = \infty$. A `Quiet[]` command around the calculations is included to suppress certain warnings from *Mathematica*. The 6 plots are included as individual elements of the list `q`. These calculations take a substantial amount of time. Each numerical integration has been placed in a separate cell so that if readers execute the code, they can keep track of the progress of the computations by examining which cells have completed their work.

```
In[ ]:= q = Table[0, {i, 1, 6}];
        Quiet[
         q⟦1⟧ = Plot[k[𝒜, 1], {𝒜, 0.6, 1.4}, PlotRange → {0, 100}, Filling → None, PlotStyle → Red];]
```

3. First-Order Statistics of Speckle Intensity

In the optical region of the spectrum, detectors are unable to follow the ultra-fast cycles of the optical field amplitude, but rather, they respond to incident power or intensity, averaged over some response time of the detector and some finite area of the detector element. For that reason, in studying speckle in the optical region of the spectrum, the intensity statistics of random phasor sums are of much greater interest than amplitude statistics. We now turn our attention to simulating the intensity statistics of random phasor sums.

The *Mathematica* code used in the previous chapter for various amplitude cases can be reused with small changes. Instead of calculating the length of the random phasor sums, we must calculate the squared length of those sums, for the squared length corresponds to intensity. Accordingly, we modify the previous code to calculate intensity statistics.

3.1 Intensity Statistics of the Sum of Many Random Phasors with Unit Lengths and Random Phases

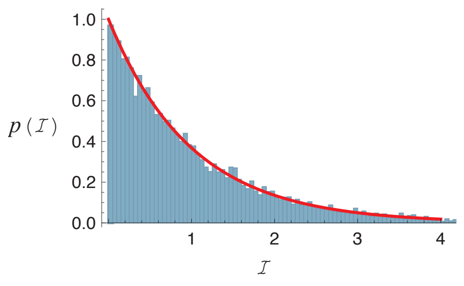
The modified code for calculating an approximation to the intensity statistics, as drawn from Section 2.1 above, is as follows:

```
In[ ]:= Clear[𝒩, ℳ, real, imag, phi, t, p1, p2]
        𝒩 = 100; ℳ = 10 000;
        real = Parallelize[Table[0, {k, 1, ℳ}]];
        imag = Parallelize[Table[0, {k, 1, ℳ}]];
        For[i = 1, i ≤ ℳ, i++,
        phi = Parallelize[Table[RandomReal[{-π, π}], {k, 1, 𝒩}]];
        t = (1 / Sqrt[𝒩]) * Exp[I * phi];
        real⟦i⟧ = Total[Re[t]];
        imag⟦i⟧ = Total[Im[t]];]

In[ ]:= p1 = Histogram[real^2 + imag^2, 200, "PDF"];
```

One change has been made in the last line of the code. The square root needed to calculate amplitude has been removed, with the result that we are calculating a histogram of intensity.

We wish to compare this histogram with the theoretical result for the negative-exponential probability density function (PDF) of intensity (with mean unity), valid when an infinite number of normalized phasors compose the sum:

```
In[ ]:= p2 = Plot[PDF[ExponentialDistribution[1], x],
            {x, 0, 4.0}, Filling → None, PlotStyle → Red, ImageSize → 200];
        Labeled[Show[{p1, p2}, ImageSize → 200], {"p(ℐ)", "ℐ"}, {Left, Bottom}]

Out[ ]=
```

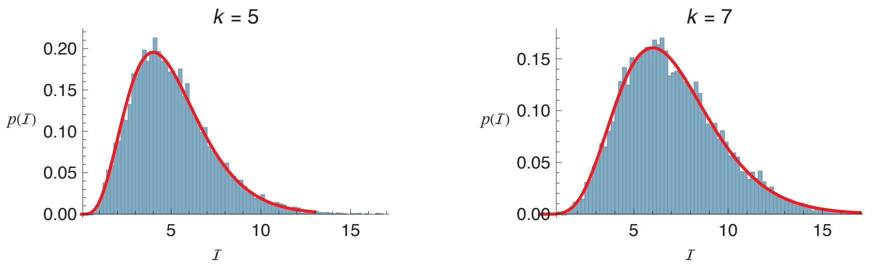


As can be seen, the match is quite good, and would be even better if we used a larger number of trials (larger $\mathcal{M}$).

It is of interest to know the mean and the standard deviation of the resulting approximate PDF represented by the histogram. *Mathematica* has commands that allow us to directly calculate these quantities:

In[∘]:=
```
GraphicsRow[{a5, a7}, ImageSize → 500]
```

Out[∘]=



Again, the agreement is reasonably good, and would be improved with a larger value of $\mathcal{M}$. As k increases, the histogram is approaching a Gaussian probability density with a mean equal to k, i.e., the number of speckle patterns added, since each pattern had a mean of unity.

3.5 Intensity Statistics of Partially Developed Speckle

Attention is now turned to the case of partially developed speckle. Such speckle occurs in practice when light is reflected from a surface that has a roughness that is less than a half wavelength of the light, or when light is passed through a diffuser that randomizes the phase of the light by less than 2π radians. It may also be observed when an adaptive-optics system in a telescope does not fully compensate for the wavefront perturbations introduced by the turbulent atmosphere. The parameter $\mathcal{S}0$ is the total number of different phase standard deviations that are included in the computation, while again the parameter $\mathcal{M}$ is the total number of trials in the calculation and $\mathcal{N}$ is the number of phasors added on each trial. Each phasor has a length $1/\sqrt{\mathcal{N}}$. We assume that the phase of the scattered light is a Gaussian (or normal) random variable with zero mean and variable standard deviation equal to $s \times \pi/8$, where s ranges from 1 to 8 in increments of 1. Thus, the phase standard deviation ranges from $\pi/8$ to π in increments of $\pi/8$.

During each step through the i index, a magnitude-squared averaged over $\mathcal{M}$ trials of the random phasor sum is calculated. Each step of the s index increases the phase standard deviation in increments of $\pi/8$, with s ranging from 1 to $\mathcal{S}0 = 8$. For each value of s, histograms of the real part of the complex amplitude, the imaginary part, and the squared magnitude are calculated. If we choose $\mathcal{M} = 10{,}000$ to obtain accurate histograms, the calculation takes a long time.

First we establish tables into which various results can be placed. `histr` holds the histograms of the real part of the complex amplitude for different values of s, `histi` holds histograms of the imaginary part of the complex amplitude, and `hist` holds histograms of the intensity of the speckle pattern.

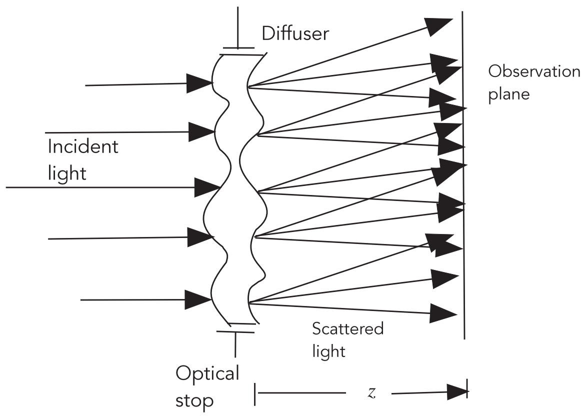
In[∘]:=
```
Clear[S, M, N]
S = 8; M = 10 000; N = 100;
real = Parallelize[Table[0, {k, 1, M}]];
imag = Parallelize[Table[0, {k, 1, M}]];
magsq = Parallelize[Table[0, {k, 1, M}]];
mag = Parallelize[Table[0, {s, 1, S}]];
histr = Parallelize[Table[0, {s, 1, S}]];
histi = Parallelize[Table[0, {s, 1, S}]];
hist = Parallelize[Table[0, {s, 1, S}]];
```

Next we run 10,000 trials, adding 100 phasors on each trial, and increase the standard deviation of the Gaussian phase by increments of $\pi/8$ on each of the 8 outer iterations. Histograms of the real and imaginary parts of the complex amplitude are calculated for each phase standard deviation, as is also a histogram of the intensity.

5. Simulation of Speckle in Free-Space Propagation

Calculating the diffraction patterns of clear apertures of various shapes is a common task. However, similar calculations for apertures with a complex internal structure are less common. In the case studied here, the internal structure of a rectangular aperture is a random pattern of phase representing a diffuser. To perform the simulation of propagation, choices must be made for the number of samples within the aperture (the parameter $\mathcal{M}$ in what follows) and the total number of samples in the simulation (the parameter $\mathcal{N}$ in what follows), as well as the method of calculation. Guidelines for choices of these parameters and methods for clear apertures do not apply in this more complex case.

The geometry assumed for these simulations is shown in the figure below:



A normally incident plane wave from a highly coherent source, indicated by the arrows on the left, is incident on a diffuser, and light is scattered in many directions. The incident uniform light spot is bounded by an aperture. The goal is to find the speckle pattern in a plane at a normal distance z from the average diffuser surface, indicated by the vertical line labeled *observation plane*, and to observe the changes in that pattern as z is changed.

Because of the long computation times required for 2D simulations, the simulations in this chapter will be 1D. Generalizations to two dimensions will be fairly obvious, and as computer speeds continue their upward march, 2D simulations may become less time consuming in the near future.

5.1 The Diffuser

We will now construct a 1D correlated-phase diffuser similar to the 2D version constructed in the previous chapter. A difference from the previous chapter, besides the smaller dimensionality, is that this 1D diffuser will be imbedded in a surrounding padding of zeros. The first simulation will have $\mathcal{N}$ pixels, $\mathcal{M}$ of which are diffuser and $\mathcal{N} - \mathcal{M}$ of which are zeros. We occasionally use the symbol $\mathcal{Q}$ to represent the ratio $\mathcal{N}/\mathcal{M}$, which is the ratio of the total number of samples to the number of those samples lying in the diffuser aperture. The reason for padding with zeros is that the more the diffuser is padded with zeros, the finer the sampling will be in the spectrum of the diffuser and in its diffraction pattern. The choices of $\mathcal{N}$ and $\mathcal{M}$ must be tailored to the problem at hand. We elaborate on this issue in the two sections to follow.

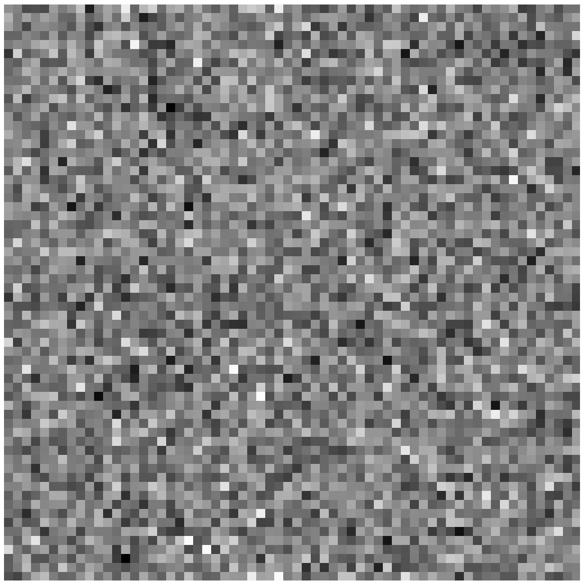
Initially, we let $\mathcal{N} = 4096$ and $\mathcal{M} = 512$. Note that in the *Mathematica* code, we are using script $\mathcal{N}$, $\mathcal{M}$, and $\mathcal{Q}$ to avoid any confusion with *Mathematica* commands, which always begin with non-script upper-case letters. The second approach will use a different value for $\mathcal{M}$.

We wish to create a diffuser Fourier spectrum that is tapered to small values at the window edges to avoid significant aliasing. Constructing such a diffuser requires the introduction of correlations between phase samples. This is accomplished by first generating a length-$\mathcal{M}$ array of uncorrelated Gaussian-distributed phases and then smoothing that array with a finite window via convolution. For the kernel in this case we choose a sinc-function ($\text{sinc}[x]=\frac{\sin[\pi x]}{\pi x}$) smoothing window (i.e., a convolution kernel) with a main-lobe half-width of $\mathcal{S} = 8$ pixels.

```
In[∘]:= poissondata =
          Parallelize[Table[RandomVariate[PoissonDistribution[integratedintens[[i, j]]]],
            {i, 1, N / Q}, {j, 1, N / Q}]];
```

Finally, we visualize the super-pixel photocounts as an image for a uniform incident intensity:

```
In[∘]:= ImageAdjust[Image[poissondata]]
```

Out[∘]=


We see that the uniform intensity incident on the detector array generates a significant variation of the detected signal due to the Poisson statistical variations of the photoevents associated with detection.

6.2 The Negative-Binomial Distribution

If instead of a constant intensity falling on a super-pixel, there is a speckle pattern, the photocount statistics are no longer Poisson-distributed. Instead the proper distribution to use is a negative-binomial distribution for the number of photocounts (Ref. [1], p. 424). The extra statistical fluctuations of the speckle pattern induce extra fluctuations of the photocounts from a super-pixel, and these extra fluctuations lead to the negative-binomial distribution. Note that the fluctuations of the incident speckle pattern are partially reduced by integration of those fluctuations over each super-pixel when the speckle is finer than the size of the super-pixel, and this is why the negative-binomial distribution is appropriate, rather than the Bose–Einstein distribution, which would apply if there were a single speckle incident on each detector element.

Mathematica has knowledge of the negative-binomial distribution, but not exactly in the form we need. From the reference given above, the probability of k photocounts occurring in a super-pixel is given by

$$P(k) = \frac{\Gamma(k+m)}{\Gamma(k+1)\,\Gamma(m)}\left(1+\frac{m}{\overline{k}}\right)^{-k}\left(1+\frac{\overline{k}}{m}\right)^{-m},$$

where k is the number of counts, $\overline{k}$ is the average number of counts, and m is the average number of speckles per super-pixel. On the other hand, the *Mathematica* form of the negative-binomial distribution is

$$P(k) = \binom{m+k-1}{m-1} p^m (1-p)^k,$$

where $\binom{m+k-1}{m-1}$ is a binomial coefficient and is equivalent to $\frac{\Gamma(m+k)}{\Gamma(k+1)\,\Gamma(m)}$. The mean of the first form of the distribution is $\overline{k}$, while the mean of the second form of the distribution is $\frac{m(1-p)}{p}$. Equating the two expressions for the mean, we obtain the equivalences

$$p = \left(1+\frac{\overline{k}}{m}\right)^{-1}$$
$$1-p = \left(1+\frac{m}{\overline{k}}\right)^{-1}.$$

9. Speckle Simulation for Metrology

In most imaging applications, speckle is a nuisance, and various methods are used to attempt to suppress it. However, in the field of metrology, speckle can be a friend rather than a foe, and many methods for using speckle in measurement have been devised. Here we will simulate only a few of such methods; the interested reader is referred to Ref. [1], (Chapter 9), for a more comprehensive discussion of this topic, together with references to the pertinent original publications.

9.1. Measurement of In-Plane Displacement

An early application of speckle to metrology was for measurement of lateral in-plane displacement. Assume that we are imaging a finite region on a rough object and we observe speckle in the image. The object then moves laterally and we wish to use a second exposure to determine the amount of lateral movement. As the object translates, the speckle in its image translates as well, but with some change as new scatterers move into the fixed illuminated region and previous scatterers move out of the illuminated region.

We assume that the optical imaging system is a $4f$ system of the same type we have used in other chapters. The rough object lies in the front focal plane of a positive lens with focal length f. In the rear focal plane of that lens there is an aperture, centered on the optical axis, that restricts the area through which light can pass. At one further focal length a second positive lens, again with focal length f, captures the light passed by the aperture and passes it on to its rear focal plane where a filtered image of the rough surface is found.

First we generate a diffuser representing the rough object that will be laterally translated. Again, we smooth the diffuser so that it has a finite phase correlation length. Next we define a finite window corresponding to the finite illumination spot on the diffuser. This window remains fixed while the diffuser moves under it. The number of pixels in the simulation is $\mathcal{N} \times \mathcal{N}$, $\mathcal{P}$ represents the width of the rectangular smoothing functions that introduce correlations in the phase function of the diffuser, $\mathcal{W} \times \mathcal{W}$ represents the area of the finite illumination window on the diffuser ($\mathcal{W} < \mathcal{N}$), and $\mathcal{D} \times \mathcal{D}$ represents the area of the square focal-plane aperture.

First we establish the values of the parameters in the simulation.

```
In[∘]:= Clear[𝒩, 𝒟, 𝒲, 𝒫]
        𝒩 = 512; 𝒫 = 8; 𝒲 = 256; 𝒟 = 𝒩 / 4;
```

The diffuser is illuminated by a finite spot of light of dimension $\mathcal{W} \times \mathcal{W}$ and the diffuser translates through this spot, moving new diffuser pixels into the illuminated spot and losing some old diffuser pixels at the same time. We have chosen $\mathcal{W}$ to be 256, half the size of the full diffuser in this simulation. We can then define the square window function on the diffuser:

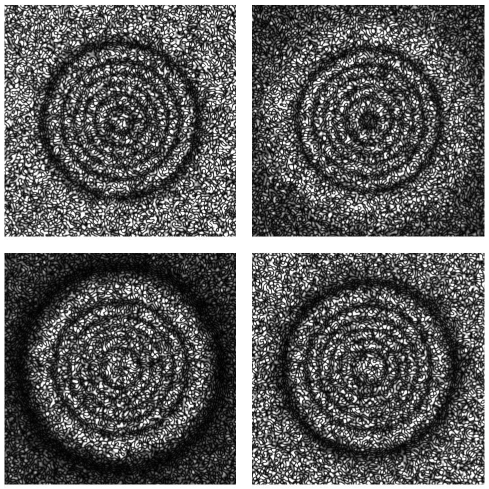
```
In[∘]:= window = Parallelize[Table[rect[(i - 𝒩 / 2) / 𝒲] * rect[(j - 𝒩 / 2) / 𝒲], {i, 1, 𝒩}, {j, 1, 𝒩}]];
```

Next we generate the windowed diffuser with smoothed phase and calculate its complex Fourier spectrum, centering the spectrum at indices ($\mathcal{N}/2$, $\mathcal{N}/2$). In order to generate a diffuser spectrum that tapers down at high frequencies, we find by trial and error that a choice of standard deviation of 32π in `uncorrelated` and a choice of 8 for $\mathcal{P}$ yields the result shown below. Other solutions are also possible, for example, a smaller value of $\mathcal{P}$ and a smaller value of standard deviation.

```
In[∘]:= uncorrelated =
          Parallelize[Table[RandomVariate[NormalDistribution[0, 32 * π]], {i, 1, 𝒩}, {j, 1, 𝒩}]];

In[∘]:= kernel = Parallelize[Table[rect[i / 𝒫] * rect[j / 𝒫], {i, -𝒫, 𝒫}, {j, -𝒫, 𝒫}]];

In[∘]:= smooth = (1 / (2 * 𝒫 + 1)) ^ 2 * ListConvolve[kernel, uncorrelated, 1];

In[∘]:= diffuser1 = Exp[I * smooth] * window;

In[∘]:= diffuserspectrum1 = RotateRight[dft[diffuser1], {𝒩 / 2, 𝒩 / 2}];
```

As used in previous chapters, the `dft` is a user-defined command that performs a discrete Fourier transform using the FFT algorithm.

```
In[ ]:= Clear[𝒜, ℬ, 𝒞, 𝒟]
        𝒜 = Abs[imageintens1 - imageintens2a];
        ℬ = Abs[imageintens1 - imageintens2b];
        𝒞 = Abs[imageintens1 - imageintens2c];
        𝒟 = Abs[imageintens1 - imageintens2d];

In[ ]:= GraphicsGrid[{{Image[𝒜], Image[ℬ]}, {Image[𝒞], Image[𝒟]}}, ImageSize → 250]

Out[ ]=
```



To recover the Gaussian phase bump on `diffuser2`, one method is to form the following calculation:

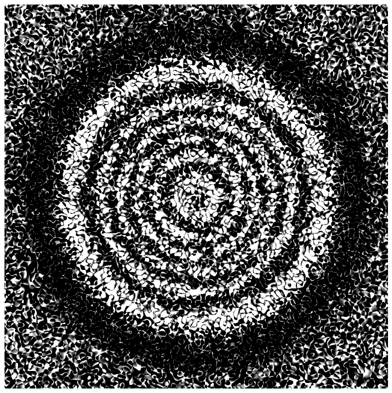
$$\Delta\phi = \arctan\left(\frac{\mathcal{C} + \mathcal{D} - \mathcal{A} - \mathcal{B}}{\mathcal{A} + \mathcal{D} - \mathcal{B} - \mathcal{C}}\right).$$

Define `num` as the numerator of the fraction and `denom` as the denominator:

```
In[ ]:= num = 𝒞 + 𝒟 - 𝒜 - ℬ;
        denom = 𝒜 + 𝒟 - ℬ - 𝒞;
```

Now form an image of the arctan, which should yield $\Delta\phi$.

```
In[ ]:= Image[ArcTan[denom, num], ImageSize → 200]

Out[ ]=
```



There are two important questions to ask about this result. First, why are there rings in this image of $\Delta\phi$? The answer is again that the method recovers $\Delta\phi$ only modulo 2π; and the rings are jumps between π and $-\pi$. The plot below shows the original continuous Gaussian function modulo 2π, and the rings are seen in this image, too, although they are much more narrow than in the result of the speckle calculation.

```
In[ ]:= f = Table[12 * Pi * gaus[Sqrt[((i - 𝒩 / 2) ^2 + (j - 𝒩 / 2) ^2)] / 400], {i, 1, 𝒩}, {j, 1, 𝒩}];
        g = Mod[f, 2 * Pi];
```

Appendix A – Some Subtleties in Speckle Simulation With the 4*f* Imaging System

A.1 Effects on the Speckle Contrast

As pointed out in Section 1.3, there are some subtleties associated with the generation of simulated speckle using the 4*f* imaging system that arise from the finite size of the arrays used in the simulation. The phenomenon involves a complicated relationship between the size of the arrays used and the diameter of the focal-plane stop. It is possible to choose these parameters in such a way that the contrast of the speckle (i.e., the standard deviation of intensity normalized by the mean intensity) never achieves its ideal value of unity. Two effects can be identified.

To understand these effects, it is helpful to consider how much of the diffuser contributes to the value of intensity at any single point in the image plane. With a finite focal-plane aperture diameter, there is a weighting function on the diffuser that averages over a finite region, adding complex-valued sample points that contribute to the image amplitude and intensity at a point. The smaller the focal-plane aperture, the broader that weighting function on the diffuser becomes; and the larger the diameter of the aperture, the narrower that weighting function becomes.

The sum of the complex phasors within that averaging region generates a new complex phasor, one that has length and phase determined by interference between the complex diffuser pixels lying within the averaging region. If the focal-plane aperture is 1 pixel in diameter, a single spectral sample passes the focal plane and the intensity in the image plane is constant, with contrast equal to zero for any single diffuser. If the focal-plane aperture has a diameter of size $\sqrt{2}\,\mathcal{N}$ or larger (this size circle covers the corners of the rectangular sample array), the extent of the weighting function on the diffuser will be one pixel, covering only one phase cell of the diffuser. The result is a pure phase image with contrast zero.

Thus, we see that in the limits of small or large focal-plane apertures, the speckle contrast in the image can be reduced, either due to the small number of phasors contributing to image intensity at an image point in the former case, or due to partial resolution of the pure phase diffuser in the latter case. See the figure just above Section 3.4 in Chapter 3 for a discussion of speckle contrast for small numbers of equal-strength phasors.

A.2. Simulation With a Smoothed Phase

In this section we investigate the contrast of image speckle as a function of Fourier-plane aperture size, starting with a phase that has been smoothed to reduce aliasing. We start with an $\mathcal{N} \times \mathcal{N} = 1024 \times 1024$ array of statistically independent phase samples. Each phase sample is a Gaussian random variable with zero mean and standard deviation $\sigma = 25\pi$. The choice of σ is made in conjunction with the diameter of the smoothing function to produce a diffuser Fourier spectrum that is tapered towards the edges, thus decreasing aliasing. The smoothing function is a discrete approximation to a uniform circle with a diameter of $\mathcal{K} = 20$ pixels. The standard deviation of the smoothed phase is calculated to make sure it's reasonable. The diffuser is then calculated as a complex exponential with argument i times the smoothed phase, and the diffuser spectrum is calculated.

```
In[ ]:= 𝒩 = 1024;
        𝒦 = 20;
        Clear[uncorrelated, kernel, smooth, diffuser, diffuserspectrum, 𝒟];
        uncorrelated = Parallelize[Table[RandomVariate[NormalDistribution[0, 25 * π]], {𝒩}, {𝒩}]];
        kernel = (1 / (π * (𝒦 / 2) ^ 2)) *
            Table[circ[Sqrt[(i - 𝒦 / 2) ^ 2 + (j - 𝒦 / 2) ^ 2] / (𝒦 / 2)], {i, 1, 𝒦}, {j, 1, 𝒦}];
        smooth = ListConvolve[kernel, uncorrelated, 1];
        diffuser = Exp[I * smooth];
        diffuserspectrum = RotateRight[Fourier[diffuser], {𝒩 / 2, 𝒩 / 2}];
```

The standard deviation of the smoothed phase is a reasonable number, given that we would like to have some excursions that exceed 2π.

Appendix B – Some Subtleties in Dealing with *Mathematica* Images

In the chapters of this book, we have often had occasion to use the *Mathematica* commands `Image[]` and `ImageData[]` when moving between numerical data and images. Given an array of real numbers, an image can be obtained by using the command `Image[`*array*`]`, while to go in the opposite direction, from an image to real numbers, the command `ImageData[`*image*`]` is used. There are some peculiarities of these commands that we wish to point out here.

B.1 Dimensions of Data Arrays and Images

To start, we define an array of random real numbers that lie between 0 and 1, intentionally making the lengths of the two dimensions of the array unequal:

```
In[73]:=  data = Table[RandomReal[], {i, 1, 4}, {j, 1, 3}]

Out[73]=  {{0.196136, 0.518268, 0.233036}, {0.756317, 0.0983559, 0.335059},
           {0.478396, 0.604937, 0.662963}, {0.894975, 0.707075, 0.916227}}
```

The array can be shown in the form of a matrix:

```
In[74]:=  MatrixForm[data]

Out[74]//MatrixForm=
          0.196136  0.518268  0.233036
          0.756317 0.0983559  0.335059
          0.478396  0.604937  0.662963
          0.894975  0.707075  0.916227
```

The dimensions of the array are:

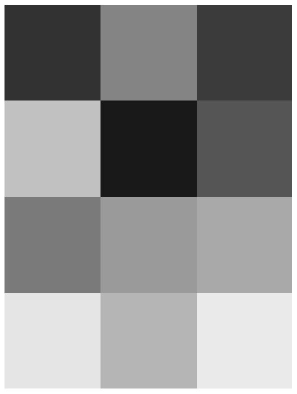
```
In[75]:=  Dimensions[data]

Out[75]=  {4, 3}
```

Note that the first number that `Dimensions[]` returns is the length of the columns of the matrix (the number of rows), while the the second number is the length of the rows (the number of columns).

We now create an image of the data:

```
In[76]:=  myimage = Image[data, ImageSize → 150]
```

Out[76]=



Acknowledgement

I owe major thanks to two persons for their help in bringing this book to reality in its current form. First is the Senior Editor from the SPIE Press, Dara Burrows, whose 600+ suggestions greatly improved this book. Second is an anonymous reviewer, who read the manuscript extremely carefully and pointed out many improvements that could be made. I adopted nearly all of the suggestions made by both of these individuals.

References

[1]. J.W. Goodman, *Speckle Phenomena in Optics, Theory, and Applications*, 2nd Edition, SPIE Press, Bellingham, Washington (2020) [doi: 10.1117/3.2548484].

[2]. B.F. Torrence and E.A. Torrence, *The Student's Introduction to Mathematica and the Wolfram Language*, 3rd Edition, Cambridge University Press, Cambridge, UK (2019).

[3]. H. Ruskeepää, *Mathematica Navigator: Mathematics, Statistics and Graphics*, 3rd Edition, Elsevier Academic Press, Burlington, Massachusetts (2009).

[4]. J.W. Goodman, *Statistical Optics*, 2nd Edition, John Wiley & Sons, Hoboken, New Jersey (2015).

[5]. I. Freund, "Optical vortices in Gaussian random wave fields: statistical probability densities," *J. Soc. Am. A* **11**, 1644–1652 (1994).

[6]. W. Wang, S.G. Hanson, and M. Takeda, "Statistics of polarization speckle: theory versus experiment," *Proc. SPIE* **7388**, 738803 (2009) [doi: 10.1117/12.855761].

[7]. R.A. Chipman, W-S.T. Lam, and G. Young, *Polarized Light and Optical Systems*, CRC Press, Boca Raton, Florida (2019).

[8]. J.N. Butters, "Speckle pattern interferometry using video techniques," *Opt. Eng.* **10**, 5–9 (1971) [doi: 10.1117/12.7971587].

[9]. K. Creath, "Phase-shifting speckle interferometry," *Appl. Opt.* **24**, 3053–3058 (1985).

[10]. S. Nakadate and H. Saito, "Fringe scanning speckle-pattern interferometry," *Appl. Opt.* **24**, 2172–2180 (1985).

[11]. J.C. Wyant, Phase-Shifting Interferometry, https://wp.optics.arizona.edu/jcwyant/wp-content/uploads/sites/13/2016/08/Phase-Shifting-Interferometry.nb_.pdf

[12]. D.C. Ghiglia and M.D. Pritt, *Two-Dimensional Phase Unwrapping: Theory, Algorithms and Software*, Wiley Interscience, New York (1998).