# Phase retrieval with unknown sampling factors via the two-dimensional chirp $z$-transform

**Alden S. Jurling and James R. Fienup***

*Institute of Optics, University of Rochester, Rochester, New York 14627, USA*
**Corresponding author: fienup@optics.rochester.edu*

Received May 7, 2014; accepted June 29, 2014;
posted July 14, 2014 (Doc. ID 211626); published August 5, 2014

We derive the analytic gradient of a phase retrieval error metric with respect to the sampling factor or the $f$-number that produced the measured point-spread function. This allows us to efficiently optimize over the sampling factor, thereby improving the accuracy of the phase estimate. Computer simulation results show its effectiveness. © 2014 Optical Society of America

*OCIS codes:* (010.7350) Wave-front sensing; (110.6770) Telescopes; (100.5070) Phase retrieval.
http://dx.doi.org/10.1364/JOSAA.31.001904

## 1. INTRODUCTION

It is useful to measure the wavefront aberrations of an optical system. In a laboratory environment this may be done using interferometry or wavefront sensing hardware, such as a Shack–Hartmann wavefront sensor. However, it is often desirable to measure the wavefront of a system using only the hardware already present for its imaging function. This is particularly relevant for space-based imaging systems where mass and volume are at a premium and external test hardware such as interferometers are not feasible. Alternately, a wavefront sensing approach using built-in imaging hardware may be required simply because a system was not designed originally to support wavefront sensing and no such hardware is available; the classic case of the Hubble Space Telescope spherical aberration problem combines both of these scenarios [1].

Phase retrieval algorithms [2,3] have been used for image-based wavefront sensing in a number of applications, and are planned for use in fine phasing of the James Webb Space Telescope (JWST) [4]. These algorithms work by modeling the system and its aberrations and fitting a point-spread function (PSF) computed from this model against the measured PSF data, using an iterative-transform or nonlinear optimization approach. In this paper we will focus on the nonlinear optimization phase retrieval [2,3] family of algorithms for wavefront sensing. These algorithms typically take some provided information about the system as a given, such as the operating wavelength or a known pupil constraint. Uncertainty in these given parameters can limit the accuracy of the algorithms, so there has been a trend of extending phase retrieval algorithms to treat formerly fixed parameters as additional variables. This has led to algorithms able to retrieve point-by-point or parametric amplitude models, spectral weights, and defocus distances, for example. For nonlinear optimization algorithms, it is highly desirable to obtain an analytic form of the gradient of the error metric with respect to the unknown parameters. One parameter in particular has until now resisted such analysis: the sampling parameter $Q = \lambda F/p$, where $\lambda$ is the wavelength, $F$ is the $f$-number, and $p$ is the detector pixel pitch [5]. Uncertainty in $Q$ can be equivalently seen as uncertainty in the wavelength, effective focal length, $f$-number, pixel pitch, or plate scale. Previously, $Q$ optimization has been done using grid searching [6] and finite difference gradients [7]. In this paper, we demonstrate a new method for $Q$ retrieval using a chirp $z$-transform and a reverse-mode algorithmic differentiation [8,9] process to obtain efficient scaling and analytic gradients.

In Section 2, we review the $Q$-optimization problem and the chirp $z$-transform algorithm. In Section 3 we define some notation conventions for the mathematical sections of the paper. In Section 4 we review the mathematical model for a simple nonlinear optimization phase retrieval algorithm. In Section 5 we develop the analytic gradient for the chirp $z$-transform in one dimension, which we extend to two dimensions in Section 6. In Section 7 we incorporate the chirp $z$-transform model into the phase retrieval algorithm from Section 4. In Section 8 we present the results of a computer simulation study. Finally in Section 9 we draw conclusions. Some of this material was previously presented in the conference paper [10].

## 2. $Q$ OPTIMIZATION

The difficulty in obtaining analytic gradients for $Q$ arises from two aspects. First, the sampling in a numerical propagation is typically controlled by using zero padding in a fast Fourier transform (FFT), which induces a fundamental quantization on $Q$. In this model, $Q$ is not a continuous quantity, and we cannot differentiate with respect to $Q$. This aspect of the problem can be addressed [6] by using a matrix-multiply discrete Fourier transform (DFT). This, however, introduces an additional problem: by using a matrix-multiply DFT to compute the propagation, we have given up the favorable asymptotic scaling of the FFT [11], so the cost of computing propagations for large array sizes grows faster than for typical FFT-based phase retrieval algorithms. Second, because the sampling parameter appears in the kernel of the transform rather than its argument, the analytic gradient does not follow the typical

 

pattern of phase retrieval gradients seen in [3]. For this reason, previous work [7] has relied on a finite-difference gradient with respect to $Q$. One of the issues addressed by [7] is fixed FFT sampling relations: if the phase retrieval model uses FFTs, the finite integer array size places limits on the step size in the gradient so that it may not be possible to achieve an adequate finite difference approximation. This problem is addressed in [7] by combining the finite difference gradient approximation with a forward model using the matrix multiplication two-dimensional (2D) DFT, which allows continuous sampling control at the expense of potentially increased computational costs. Although a finite-difference gradient is not excessively computationally onerous for a single parameter, it introduces several potential problems. First, the user is required to choose an appropriate step size for the finite difference approximation. Second, it degrades the numerical accuracy of that component of the gradient, due to taking differences of two similar numbers.

In this paper, we address the requirement to continuously control the sampling parameter by adopting a chirp $z$-transform [12–15] propagation model. The chirp $z$-transform is a generalization of Bluestein's FFT algorithm [13] for computing FFTs with prime and nonhighly composite transform lengths, which can also be used to compute transforms with arbitrary sampling. It can compute any transform that could be computed using a matrix-multiply DFT (uniformly sampled transforms with arbitrary spacing in both domains) and maintains the same asymptotic scaling behavior as the FFT (with a modest constant-factor penalty). See [16] for an application to digital holography. Depending on the array sizes and implementation details, in some cases there can be a performance benefit to using the matrix-multiply DFT over the chirp $z$-transform, particularly for smaller arrays. Either algorithm could be used as the basis for a $Q$ optimization algorithm; in this paper we limit our consideration to chirp-$z$-transform-based solutions. Finally, we employed manual algorithmic differentiation [8,9] to obtain the analytic gradients with respect to $Q$. This provides a formal step-by-step process for transforming the algorithm used to compute the chirp $z$-transform in the forward direction into an algorithm for computing the gradient with respect to $Q$, which was not previously achieved.

## 3. NOTATIONAL CONVENTIONS

In this paper, boldface symbols denote vector or array quantities, italics but unbolded symbols denote scalar quantities, and the overbar denotes differentiation of the final error metric with respect to the quantity denoted by the symbol under the bar. The overbar notation is adapted from [8] and developed in detail in [9]. Following the convention we adopted in [9], the small circle operator $\circ$ will denote the element-wise scalar product of two arrays. Also, a superscript applied to a bold variable, such as $\boldsymbol{D}^2$, indicates element-wise raising to a power.

## 4. PHASE RETRIEVAL REVIEW

In order to place $Q$ retrieval within the context of a larger phase retrieval algorithm, we will present here a monochromatic single-image phase retrieval algorithm that nevertheless exhibits many of the qualities of more sophisticated algorithms that might be used in practice. We begin by parameterizing the wavefront $\boldsymbol{W}$ that we wish to retrieve in terms of a set of Zernike polynomials $\boldsymbol{Z}$,

$$\boldsymbol{W} = \sum_n a_n \boldsymbol{Z}_n, \tag{1}$$

where the $\boldsymbol{a}$ vector denotes the Zernike coefficients of the wavefront. Next, using that wavefront we form a complex field in the pupil:

$$\boldsymbol{g} = \boldsymbol{A} \circ \exp\left(i\frac{2\pi}{\lambda}\boldsymbol{W}\right), \tag{2}$$

where $\boldsymbol{A}$ is a discrete representation of the amplitude transmittance of the pupil of the system and $\lambda$ is the wavelength of the light. In this (fixed sampling) model, $Q$ is controlled by constructing $\boldsymbol{A}$ so that the clear aperture is embedded in a larger array of zeros. We then propagate this field to the image plane using an FFT (in order to do $Q$ optimization, this will later be replaced with the chirp $z$-transform, and the requirement to embed the clear aperture in zeros will be lifted):

$$\boldsymbol{G} = \mathrm{FFT}\{\boldsymbol{g}\}. \tag{3}$$

We form a model of the intensity of the image:

$$\boldsymbol{I} = |\boldsymbol{G}|^2 = \boldsymbol{G} \circ \boldsymbol{G}^*. \tag{4}$$

Finally, using this modeled intensity we can form an error metric:

$$E = \sum_m (I_m - D_m)^2, \tag{5}$$

where $\boldsymbol{D}$ is the measured PSF data (an image of a point source) and $m$ indexes the pixels of the image. This formulation of the error metric relies on the energy in the pupil amplitude in Eq. (2) being selected such that after the FFT propagation in Eq. (3), the overall intensity in Eq. (4) is normalized appropriately compared with the data, so that the error metric in Eq. (5) will be approximately zero when the model Zernike coefficients match those that produced the data PSF. In practice, we prefer to employ the bias and gain invariant metric developed in [17], but for explanatory purposes will retain the simple sum of squared differences metric. See the discussion at the end of Section 8 for a caveat relating to this formulation. Once we have defined the error metric, we must find the vector of Zernike coefficients that minimizes it, formally:

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}}[E(\boldsymbol{a})]. \tag{6}$$

Finding the global minimum of the error surface defined by Eq. (5) is not trivial. Even limiting our consideration to low order Zernikes (perhaps the first 15), the search spaces are of too high a dimensionality to permit exhaustive brute force searches. Instead, we employ gradient search algorithms such as nonlinear conjugate gradient or the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [18]. In order to keep the computational cost of the algorithm from growing proportionally to the number of parameters, we use analytic gradient techniques to form expressions for the gradients of the error

metric as shown in [2,3]. Here we use the reverse mode gradient formalism developed in [9] to derive the needed gradients. For our example forward model, the gradient is given by

$$\begin{aligned}\bar{\boldsymbol{I}} &= 2(\boldsymbol{I}-\boldsymbol{D}),\\ \overline{\boldsymbol{G}} &= 2\boldsymbol{G}\circ\bar{\boldsymbol{I}},\\ \bar{\boldsymbol{g}} &= \mathrm{IFFT}\{\bar{\boldsymbol{G}}\},\\ \overline{\boldsymbol{W}} &= \frac{2\pi}{\lambda}\Im\{\boldsymbol{g}^*\circ\bar{\boldsymbol{g}}\},\\ \bar{a}_n &= \sum_p(\overline{\boldsymbol{W}}\circ\boldsymbol{Z}_n)_p,\end{aligned} \tag{7}$$

where $\Im\{\cdot\}$ denotes taking the imaginary part and $p$ indexes over the wavefront pixels. With these gradients in hand, we proceed to minimize the error metric and find a local minimum of the error surface from Eq. (6). The success of the technique is then predicated on three requirements: (1) A global minimum exists, (2) that global minimum is near to the point corresponding to the true wavefront coefficients (if there is noise in the data it will not be at the exact truth), and (3) our starting guess for phase retrieval is close enough to the global minimum that the nonlinear optimizer will locate the global minimum rather than another local minimum. The first two requirements are typically satisfied by phase retrieval models, with the exception of trivial degeneracy like the twin image problem in single-plane phase retrieval (where there are two solutions, and one is a reflection and complex conjugation of the other). The third requirement is referred to as the "capture range problem," and whether it is met in practice depends on the magnitude of the wavefront aberrations and quality of initial wavefront guesses. Although there is at present no general solution to the capture range problem, for many practical problems the capture range is sufficient to use phase retrieval effectively and we will not consider it further here.

The phase retrieval model shown above exhibits fixed FFT sampling relationships enforced by Eq. (3) and optimizes only over wavefront parameters, not sampling parameters. Over the next two sections we will replace the FFT propagation model with the more flexible chirp $z$-transform and provide the required analytic gradients for the sampling parameters.

## 5. ONE-DIMENSIONAL TRANSFORM

While our goal is to develop a 2D chirp $z$-transform, we will first consider the simpler 1D DFT case. Instead of a fixed FFT length $L$, the scaling of the transform will be described by the quantity $\alpha$, which is inversely proportional to $Q$:

$$\alpha = \frac{1}{QN_c}, \tag{8}$$

where $N_c$ is the number of pixels across the nominal clear aperture of the system. For a padded FFT, $\alpha$ is one over the FFT length. Ultimately, we wish to obtain the gradient with respect to $\alpha$. For a general uniformly sampled transform, we have

$$X_m = \sum_{n=0}^{N-1} x_n e^{-2\pi i(m-\Delta m)(n-\Delta n)\alpha}. \tag{9}$$

We have added the terms $\Delta m$ and $\Delta n$ to allow for translations in the input and output domains and to account for moving the origin to the center of the array if desired. We are interested in particular in an input vector $\boldsymbol{x}$ of length $N$ and an output vector $\boldsymbol{X}$ of length $M$. The chirp $z$-transform is based on the factoring identity

$$nm = \frac{n^2+m^2-(n-m)^2}{2}. \tag{10}$$

Including the linear shift factor, this can be written

$$\begin{aligned}(n-\Delta n)(m-\Delta m) = \frac{1}{2}[&(n-\Delta n)^2+(m-\Delta m)^2\\ &-(m-n-\Delta m+\Delta n)^2].\end{aligned} \tag{11}$$

Inserted into Eq. (9), this gives rise to three factors; the first, a pre-factor, depends on the input coordinates:

$$\begin{aligned}\beta_n &= \exp[-i\pi(n-\Delta n)^2\alpha],\\ \boldsymbol{\beta} &= \exp(-i\pi\hat{\boldsymbol{n}}^2\alpha),\end{aligned} \tag{12}$$

where $\hat{\boldsymbol{n}}$ is a vector of indices running from $-\Delta$n to $N-\Delta n-1$. The second, a post-factor, depends on the output coordinates:

$$\begin{aligned}\gamma_m &= \exp[-i\pi(m-\Delta m)^2\alpha],\\ \boldsymbol{\gamma} &= \exp(-i\pi\hat{\boldsymbol{m}}^2\alpha),\end{aligned} \tag{13}$$

where $\hat{\boldsymbol{m}}$ is a vector of indices running from $-\Delta m$ to $M-\Delta m-1$. The third depends on the difference of coordinates:

$$\begin{aligned}z_{m-n} &= \exp[i\pi(m-n-\Delta m+\Delta n)^2\alpha],\\ z_p &= \exp[i\pi(p-\Delta m+\Delta n)^2\alpha],\end{aligned} \tag{14}$$

where the $p=m-n$ allows us to define $z$ as an array independently from its indexing by $m$ and $n$. So we can write the whole DFT in Eq. (9) as

$$X_m = \gamma_m\sum_{n=0}^{N-1} x_n\beta_n z_{m-n}. \tag{15}$$

We can recognize the above as a discrete convolution. Letting $\otimes$ denote convolution and $\circ$ denote element-wise product, this can be written in vector form as

$$\boldsymbol{X} = \boldsymbol{\gamma}\circ[(\boldsymbol{x}\circ\boldsymbol{\beta})\otimes z]. \tag{16}$$

Essentially, the chirp $z$-transform algorithm consists of embedding this discrete convolution in a circular discrete convolution and then computing the convolution as a product in the Fourier domain using FFTs. This process is well described in [12], but we will sketch it here as well. First, we must select a transform length,

$$L \geq N+M-1, \tag{17}$$

for the FFT used to compute the circular convolution; it can be chosen arbitrarily to maximize FFT performance if desired, provided that Eq. (17) is satisfied. Next we define extension and truncation operators. The operator

$$\hat{\boldsymbol{w}} = \text{extend}(\boldsymbol{w}, L) \tag{18}$$

takes the vector $\boldsymbol{w}$ and adds sufficient zeros to the end so that the total length of $\hat{\boldsymbol{w}}$ is $L$. The truncation operator

$$\boldsymbol{w} = \text{truncate}(\hat{\boldsymbol{w}}, M) \tag{19}$$

takes the first $M$ elements of $\hat{\boldsymbol{w}}$ to yield $\boldsymbol{w}$ with length $M$. Finally, we define the circular extension of $z$:

$$\hat{z}_p = \begin{cases} \exp\{i\pi[p - \Delta m + \Delta n]^2\alpha\}, & 0 \le p \le M-1 \\ \exp\{i\pi[p - L - \Delta m + \Delta n]^2\alpha\}, & L-N+1 \le p \le L-1 \\ \text{arbitrary}, & \text{otherwise} \end{cases}. \tag{20}$$

The $z$ array can (depending on choice of $L$) contain arbitrary values in the central region because we are computing a circular convolution but are interested only in $M$ outputs and $N$ inputs, so some parts of the convolution kernel are either multiplied with zeros in the input or discarded from the output. The arbitrary region and the construction of $z$ can be seen more clearly in Fig. 4 of [12]. If we define $\boldsymbol{p}$ as the index vector of length $M$ from 0 to $M-1$ and $\boldsymbol{q}$ as the index vector from $L-N+1$ up to $L-1$, we could write this as

$$\begin{aligned} \hat{z}_{\boldsymbol{p}} &= \exp\{i\pi[\boldsymbol{p} - \Delta m + \Delta n]^2\alpha\}, \\ \hat{z}_{\boldsymbol{q}} &= \exp\{i\pi[\boldsymbol{q} - L - \Delta m + \Delta n]^2\alpha\}. \end{aligned} \tag{21}$$

Using these definitions, we can write the chirp $z$-transform algorithm (adapted from [12]) as

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{\beta} \circ \boldsymbol{x}, \\ \hat{\boldsymbol{y}} &= \text{extend}(\boldsymbol{y}, L), \\ \boldsymbol{Y} &= \text{FFT}(\hat{\boldsymbol{y}}), \\ \boldsymbol{Z} &= \text{FFT}(\hat{\boldsymbol{z}}), \\ \boldsymbol{H} &= \boldsymbol{Y} \circ \boldsymbol{Z}, \\ \hat{\boldsymbol{h}} &= \text{IFFT}(\boldsymbol{H}), \\ \boldsymbol{h} &= \text{truncate}(\hat{\boldsymbol{h}}, M), \\ \boldsymbol{X} &= \boldsymbol{h} \circ \boldsymbol{\gamma}. \end{aligned} \tag{22}$$

The reverse mode gradient for the input vector using the relations from [9] is

$$\begin{aligned} \bar{\boldsymbol{h}} &= \bar{\boldsymbol{X}} \circ \boldsymbol{\gamma}^*, \\ \hat{\bar{\boldsymbol{h}}} &= \text{extend}(\bar{\boldsymbol{h}}, L), \\ \bar{\boldsymbol{H}} &= \text{GIFFT}(\hat{\bar{\boldsymbol{h}}}), \\ \bar{\boldsymbol{Y}} &= \bar{\boldsymbol{H}} \circ \hat{\boldsymbol{Z}}^*, \\ \hat{\bar{\boldsymbol{y}}} &= \text{GFFT}(\bar{\boldsymbol{Y}}), \\ \bar{\boldsymbol{y}} &= \text{truncate}(\hat{\bar{\boldsymbol{y}}}, N), \\ \bar{\boldsymbol{x}} &= \bar{\boldsymbol{y}} \circ \boldsymbol{\beta}^*. \end{aligned} \tag{23}$$

The operations GFFT and GIFFT denote the reverse mode gradients for FFT and inverse FFT (IFFT), respectively. GFFT is proportional to IFFT and GIFFT is proportional to FFT, but both may have scaling factors depending on how the particular FFT and IFFT implementations used are defined. If the forward and inverse FFT have the form (in one dimension)

$$\begin{aligned} \text{FFT}\{\boldsymbol{x}\}_m &= a\sum_{n=0}^{N-1} x_n \exp\left(\frac{-i2\pi nm}{N}\right), \\ \text{IFFT}\{\boldsymbol{x}\}_m &= b\sum_{n=0}^{N-1} x_n \exp\left(\frac{i2\pi nm}{N}\right), \end{aligned} \tag{24}$$

then

$$\begin{aligned} \text{GFFT}\{\bar{\boldsymbol{x}}\} &= \frac{a}{b}\text{IFFT}\{\bar{\boldsymbol{x}}\}, \\ \text{GIFFT}\{\bar{\boldsymbol{x}}\} &= \frac{b}{a}\text{FFT}\{\bar{\boldsymbol{x}}\}. \end{aligned} \tag{25}$$

See [9] for a more complete discussion.

This derivation allows one to differentiate with respect to the input variables of the transform itself. But what if we want to differentiate with respect to parameters of the transform kernel? In particular, we are interested in $\alpha$. Since the $\alpha$ dependence is located in $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\boldsymbol{z}$, we require their gradients first. Calculating the derivatives for $\alpha$ involves differentiating a number of expressions of the form

$$\boldsymbol{y} = \exp(\boldsymbol{a}x), \tag{26}$$

where $\boldsymbol{a}$ is a complex constant vector. This has the gradient rule [9]

$$\bar{x} = \sum_n (\boldsymbol{a}^* \circ \boldsymbol{y}^* \circ \bar{\boldsymbol{y}})_n. \tag{27}$$

For the post factor we can apply the gradient rules to get

$$\bar{\boldsymbol{\gamma}} = \bar{\boldsymbol{X}} \circ \boldsymbol{h}^*, \quad \bar{\alpha}_\gamma = \sum_m [(i\pi\hat{\boldsymbol{m}}^2) \circ \gamma^* \circ \bar{\gamma}]_m, \tag{28}$$

where $\bar{\alpha}_\gamma$ denotes the contribution to the derivative with respect to $\alpha$ through the dependence of $\gamma$ on $\alpha$. For the pre-factor, we apply them to get

$$\bar{\boldsymbol{\beta}} = \bar{\boldsymbol{y}} \circ \boldsymbol{x}^*, \quad \bar{\alpha}_\beta = \sum_n [(i\pi\hat{\boldsymbol{n}}^2) \circ \boldsymbol{\beta}^* \circ \bar{\boldsymbol{\beta}}]_n. \tag{29}$$

Like $\bar{\alpha}_\gamma$, $\bar{\alpha}_\beta$ denotes the contribution to the gradient with respect to $\alpha$ via $\beta$'s dependence on $\alpha$. For the kernel factor, we apply the gradient rules and get two gradient contributions:

$$\begin{aligned} \bar{\boldsymbol{Z}} &= \bar{\boldsymbol{H}} \circ \boldsymbol{Y}^*, \\ \hat{\bar{\boldsymbol{z}}} &= \text{GFFT}(\bar{\boldsymbol{Z}}), \\ \bar{\alpha}_{\text{p}} &= \sum_{p=0}^{M-1} \{-i\pi[p - \Delta m + \Delta n]^2\} \circ \hat{z}_p^* \circ \hat{\bar{z}}_p, \\ \bar{\alpha}_{\text{q}} &= \sum_{q=L-N}^{L-1} \{-i\pi[q - L - \Delta m + \Delta n]^2\} \circ \hat{z}_q^* \circ \hat{\bar{z}}_q, \end{aligned} \tag{30}$$

where $\bar{\alpha}_{\text{p}}$ and $\bar{\alpha}_{\text{q}}$ represent the contribution of the two parts of $z$ defined by Eq. (21) to the gradient with respect

to $\alpha$. The full gradient with respect to $\alpha$ is obtained by combining the four gradient terms to get

$$\bar{\alpha} = \bar{\alpha}_\beta + \bar{\alpha}_\gamma + \bar{\alpha}_\mathrm{p} + \bar{\alpha}_\mathrm{q}. \tag{31}$$

## 6. TWO-DIMENSIONAL EXTENSION

We now extend the discussion in the previous section to arbitrary (uniformly) sampled 2D DFTs, given by

$$X_{m_\mathrm{r},m_\mathrm{c}} = \sum_{n_\mathrm{r}=0}^{N_\mathrm{r}-1,} \sum_{n_\mathrm{c}=0}^{N_\mathrm{c}-1,} x_{n_\mathrm{r},n_\mathrm{c}} e^{-2\pi i[(m_\mathrm{r}-\Delta m_\mathrm{r})(n_\mathrm{r}-\Delta n_\mathrm{r})\alpha_\mathrm{r}+(m_\mathrm{c}-\Delta m_\mathrm{c})(n_\mathrm{c}-\Delta n_\mathrm{c})\alpha_\mathrm{c}]}, \tag{32}$$

where the Roman subscripts "r" and "c" denote row and column, respectively. Note that the Fourier kernel can be directly factored into row and column components, so that the factorization in Eqs. (10) and (11) can be applied separately along the two dimensions. The form in Eq. (16) applies equally to the 2D case, provided we interpret the convolution as 2D and extend $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ into two dimensions. Because of the complete factoring of the kernel, rather than constructing full 2D extensions of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, we can instead construct 1D versions for each dimension and combine them. Analogously to Eq. (15), we have

$$X_{m_\mathrm{r},m_\mathrm{c}} = \gamma_{m_\mathrm{r}} \gamma_{m_\mathrm{c}} \sum_{n_\mathrm{r}=0}^{N_\mathrm{r}-1,} \sum_{n_\mathrm{c}=0}^{N_\mathrm{c}-1,} x_{n_\mathrm{r},n_\mathrm{c}} \beta_{n_\mathrm{r}} \beta_{n_\mathrm{c}} z_{m_\mathrm{r}-n_\mathrm{r}} z_{m_\mathrm{c}-n_\mathrm{c}}. \tag{33}$$

This allows us to define the 1D vector quantities $\boldsymbol{\beta}_\mathrm{r}$, $\boldsymbol{\beta}_\mathrm{c}$, $\boldsymbol{\gamma}_\mathrm{r}$, $\boldsymbol{\gamma}_\mathrm{c}$, $\mathbf{z}_\mathrm{r}$, and $z_\mathrm{c}$ in the same way we did previously, with $N_\mathrm{r}$ and $N_\mathrm{c}$ taking the role of $N$, $M_\mathrm{r}$ and $M_\mathrm{c}$ taking the role of $M$, and $L_\mathrm{r}$ and $L_\mathrm{c}$ taking the role of $L$. Note that we adopt a subscript convention where (for example) $N_\mathrm{r}$ denotes the length of a row (the number of columns) so that $N_\mathrm{r}$ is the length of $\boldsymbol{\beta}_\mathrm{r}$.

In order to write the 2D form of the algorithm compactly, we expand our definitions of the extend and truncate operations from Eqs. (18) and (19) into 2D. In 2D the extend operator takes the form

$$\hat{\boldsymbol{w}} = \mathrm{extend}(\boldsymbol{w}, L_\mathrm{r}, L_\mathrm{c}) \tag{34}$$

so that $\hat{\boldsymbol{w}}$ has length $L_\mathrm{r}$ along the row dimension and $L_\mathrm{c}$ along the column dimension, and truncate takes the form

$$\boldsymbol{w} = \mathrm{truncate}(\hat{\boldsymbol{w}}, M_\mathrm{r}, M_\mathrm{c}), \tag{35}$$

where $\boldsymbol{w}$ has length $M_\mathrm{r}$ along the row dimension and $M_\mathrm{c}$ along the column dimension. Finally, we define a vector–matrix–vector triple product operator,

$$\boldsymbol{d} = \langle \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \rangle \Rightarrow d_{n_\mathrm{r},n_\mathrm{c}} = a_{n_\mathrm{r}} b_{n_\mathrm{r},n_\mathrm{c}} c_{n_\mathrm{c}}, \tag{36}$$

where $\boldsymbol{b}$ is 2D and $\boldsymbol{a}$ and $\boldsymbol{c}$ are 1D, making $\boldsymbol{d}$ 2D consisting of the entries of $\boldsymbol{b}$ with the rows rescaled by $\boldsymbol{a}$ and the columns rescaled by $\boldsymbol{c}$. The gradient rules for this operator are

$$\begin{aligned} \bar{\boldsymbol{b}} &= \langle \boldsymbol{a}^*, \bar{\boldsymbol{d}}, \boldsymbol{c}^* \rangle, \\ \bar{a}_{n_\mathrm{r}} &= \sum_{n_\mathrm{c}} c^*_{n_\mathrm{c}} b^*_{n_\mathrm{r},n_\mathrm{c}} \bar{d}_{n_\mathrm{r},n_\mathrm{c}}, \\ \bar{c}_{n_\mathrm{c}} &= \sum_{n_\mathrm{r}} a^*_{n_\mathrm{r}} b^*_{n_\mathrm{r},n_\mathrm{c}} \bar{d}_{n_\mathrm{r},n_\mathrm{c}}. \end{aligned} \tag{37}$$

Using these operators, we can write the 2D transform algorithm as

$$\begin{aligned} \boldsymbol{y} &= \langle \boldsymbol{\beta}_\mathrm{r}, \boldsymbol{x}, \boldsymbol{\beta}_\mathrm{c} \rangle, \\ \hat{\boldsymbol{y}} &= \mathrm{extend}(\boldsymbol{y}, L_\mathrm{r}, L_\mathrm{c}), \\ \boldsymbol{Y} &= \mathrm{FFT}(\hat{\boldsymbol{y}}), \\ \boldsymbol{H} &= \langle \mathbf{Z}_\mathrm{r}, \boldsymbol{Y}, \mathbf{Z}_\mathrm{c} \rangle, \\ \hat{\boldsymbol{h}} &= \mathrm{IFFT}(\boldsymbol{H}), \\ \boldsymbol{h} &= \mathrm{truncate}(\hat{\boldsymbol{h}}, M_\mathrm{r}, M_\mathrm{c}), \\ \boldsymbol{X} &= \langle \boldsymbol{\gamma}_\mathrm{r}, \boldsymbol{h}, \boldsymbol{\gamma}_\mathrm{c} \rangle. \end{aligned} \tag{38}$$

We can use reverse mode gradient rules to write the gradients of the error metric with respect to the input parameters, given the gradients of the error metric with respect to the outputs:

$$\begin{aligned} \bar{\boldsymbol{h}} &= \langle \boldsymbol{\gamma}_\mathrm{r}^*, \bar{\boldsymbol{X}}, \boldsymbol{\gamma}_\mathrm{c}^* \rangle, \\ \hat{\bar{\boldsymbol{h}}} &= \mathrm{extend}(\bar{\boldsymbol{h}}, L_\mathrm{r}, L_\mathrm{c}), \\ \bar{\boldsymbol{H}} &= \mathrm{GIFFT}(\hat{\bar{\boldsymbol{h}}}), \\ \bar{\boldsymbol{Y}} &= \langle \hat{\mathbf{Z}}_\mathrm{r}^*, \bar{\boldsymbol{H}}, \hat{\mathbf{Z}}_\mathrm{c}^* \rangle, \\ \hat{\bar{\boldsymbol{y}}} &= \mathrm{GFFT}(\bar{\boldsymbol{Y}}), \\ \bar{\boldsymbol{y}} &= \mathrm{truncate}(\hat{\bar{\boldsymbol{y}}}, N_\mathrm{r}, N_\mathrm{c}), \\ \bar{\boldsymbol{x}} &= \langle \boldsymbol{\beta}_\mathrm{r}^*, \bar{\boldsymbol{y}}, \boldsymbol{\beta}_\mathrm{c}^* \rangle. \end{aligned} \tag{39}$$

As in the 1D case, we still need to track the dependence on $\alpha$ through the algorithm above. All of the $\alpha$ dependence appears in the $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $z$ terms; the lines that require additional attention are all of those that involve a vector–matrix-vector product. For the post-factor, we get

$$\begin{aligned} \bar{\gamma}_{\mathrm{r},n_\mathrm{r}} &= \sum_{n_\mathrm{c}} \gamma^*_{\mathrm{c},n_\mathrm{c}} h^*_{n_\mathrm{r},n_\mathrm{c}} \bar{X}_{n_\mathrm{r},n_\mathrm{c}}, \\ \bar{\gamma}_{\mathrm{c},n_\mathrm{c}} &= \sum_{n_\mathrm{r}} \gamma^*_{\mathrm{r},n_\mathrm{r}} h^*_{n_\mathrm{r},n_\mathrm{c}} \bar{X}_{n_\mathrm{r},n_\mathrm{c}}, \end{aligned} \tag{40}$$

while for the pre-factor, we get

$$\begin{aligned} \bar{\beta}_{\mathrm{r},n_\mathrm{r}} &= \sum_{n_\mathrm{c}} \beta^*_{\mathrm{c},n_\mathrm{c}} y^*_{n_\mathrm{r},n_\mathrm{c}} \bar{y}_{n_\mathrm{r},n_\mathrm{c}}, \\ \bar{\beta}_{\mathrm{c},n_\mathrm{c}} &= \sum_{n_\mathrm{r}} \beta^*_{\mathrm{r},n_\mathrm{r}} y^*_{n_\mathrm{r},n_\mathrm{c}} \bar{y}_{n_\mathrm{r},n_\mathrm{c}}. \end{aligned} \tag{41}$$

For the kernel factors, we get

$$\begin{aligned} \bar{Z}_{\mathrm{r},n_\mathrm{r}} &= \sum_{n_\mathrm{c}} Z^*_{\mathrm{c},n_\mathrm{c}} Y^*_{n_\mathrm{r},n_\mathrm{c}} \bar{H}_{n_\mathrm{r},n_\mathrm{c}}, \\ \bar{Z}_{\mathrm{c},n_\mathrm{c}} &= \sum_{n_\mathrm{r}} Z^*_{\mathrm{r},n_\mathrm{r}} Y^*_{n_\mathrm{r},n_\mathrm{c}} \bar{H}_{n_\mathrm{r},n_\mathrm{c}}. \end{aligned} \tag{42}$$

Notice that $\bar{\boldsymbol{\gamma}}_\mathrm{r}$, $\bar{\boldsymbol{\gamma}}_\mathrm{c}$, $\bar{\boldsymbol{\beta}}_\mathrm{r}$, $\bar{\boldsymbol{\beta}}_\mathrm{c}$, $\bar{\mathbf{Z}}_\mathrm{r}$, and $\bar{\mathbf{Z}}_\mathrm{c}$ are all 1D and are exactly analogous to their counterparts in Eqs. (28), (29), and (30). Now that we have decomposed the $\alpha$ dependence into 1D components, we can use the formulas derived in the 1D case together with Eq. (31) to produce $\bar{\alpha}_\mathrm{r}$ and $\bar{\alpha}_\mathrm{c}$. We summarize the required gradient relations here. From Eqs. (28) and (29) we get

$$\bar{\alpha}_{\mathrm{r},\gamma} = \sum_{m_\mathrm{r}} [(i\pi \hat{\boldsymbol{m}}_\mathrm{r}^2) \circ \boldsymbol{\gamma}_\mathrm{r}^* \circ \bar{\boldsymbol{\gamma}}_\mathrm{r}]_{m_\mathrm{r}},$$
$$\bar{\alpha}_{\mathrm{c},\gamma} = \sum_{m_\mathrm{c}} [(i\pi \hat{\boldsymbol{m}}_\mathrm{c}^2) \circ \boldsymbol{\gamma}_\mathrm{c}^* \circ \bar{\boldsymbol{\gamma}}_\mathrm{c}]_{m_\mathrm{c}},$$
$$\bar{\alpha}_{\mathrm{r},\beta} = \sum_{n_\mathrm{r}} [(i\pi \hat{\boldsymbol{n}}_\mathrm{r}^2) \circ \boldsymbol{\beta}_\mathrm{r}^* \circ \bar{\boldsymbol{\beta}}_r]_{n_\mathrm{r}},$$
$$\bar{\alpha}_{\mathrm{c},\beta} = \sum_{n_\mathrm{c}} [(i\pi \hat{\boldsymbol{n}}_\mathrm{c}^2) \circ \beta_\mathrm{c}^* \circ \bar{\beta}_\mathrm{c}]_{n_\mathrm{c}}. \tag{43}$$

Following from Eq. (30), we get

$$\bar{\alpha}_{\mathrm{c,p}} = \sum_{p_\mathrm{c}=0}^{M_\mathrm{c}-1} \{-i\pi[p_\mathrm{c} - \Delta m_\mathrm{c} + \Delta n_\mathrm{c}]^2\} \circ \hat{z}_{\mathrm{c},q}^* \circ \hat{\bar{z}}_{\mathrm{c},q},$$
$$\bar{\alpha}_{\mathrm{c,q}} = \sum_{q_\mathrm{c}=L_\mathrm{c}-N_\mathrm{c}}^{L_\mathrm{c}-1} \{-i\pi[q_\mathrm{c} - L_\mathrm{c} - \Delta m_\mathrm{c} + \Delta n_\mathrm{c}]^2\} \circ \hat{z}_{\mathrm{c},q}^* \circ \hat{\bar{z}}_{\mathrm{c},q},$$
$$\bar{\alpha}_{\mathrm{r,p}} = \sum_{p_\mathrm{r}=0}^{M_\mathrm{r}-1} \{-i\pi[p_\mathrm{r} - \Delta m_\mathrm{r} + \Delta n_\mathrm{r}]^2\} \circ \hat{z}_{\mathrm{r},q}^* \circ \hat{\bar{z}}_{\mathrm{r},q},$$
$$\bar{\alpha}_{\mathrm{r,q}} = \sum_{q_\mathrm{c}=L_\mathrm{r}-N_\mathrm{r}}^{L_\mathrm{r}-1} \{-i\pi[q_\mathrm{r} - L_\mathrm{r} - \Delta m_\mathrm{r} + \Delta n_\mathrm{r}]^2\} \circ \hat{z}_{\mathrm{r},q}^* \circ \hat{\bar{z}}_{\mathrm{r},q}. \tag{44}$$

To find $\bar{\alpha}_\mathrm{c}$ and $\bar{\alpha}_\mathrm{r}$ we sum together the respective components:

$$\bar{\alpha}_\mathrm{r} = \bar{\alpha}_{\mathrm{r},\beta} + \bar{\alpha}_{\mathrm{r},\gamma} + \bar{\alpha}_{\mathrm{r,p}} + \bar{\alpha}_{\mathrm{r,q}},$$
$$\bar{\alpha}_\mathrm{c} = \bar{\alpha}_{\mathrm{c},\beta} + \bar{\alpha}_{\mathrm{c},\gamma} + \bar{\alpha}_{\mathrm{c,p}} + \bar{\alpha}_{\mathrm{c,q}}. \tag{45}$$

In the common case where $\alpha = \alpha_\mathrm{r} = \alpha_\mathrm{c}$, we have the final gradient relation:

$$\bar{\alpha} = \bar{\alpha}_\mathrm{r} + \bar{\alpha}_\mathrm{c}. \tag{46}$$

This gives us a straightforward, though somewhat involved, algorithm for computing the gradient with respect to $\alpha$. We should, however, emphasize that all of the operations required to compute these gradients are on 1D vectors, so the additional computational cost for these gradients is small compared to the full 2D algorithm.

## 7. PHASE RETRIEVAL ALGORITHM WITH $Q$ OPTIMIZATION

We can define the forward chirp $z$-transform from the previous section as

$$\boldsymbol{X} = \mathrm{CZT}(\boldsymbol{x}, \alpha). \tag{47}$$

and its gradient propagation form as

$$\bar{\boldsymbol{x}} = \overline{\mathrm{CZT}}_x(\boldsymbol{x}, \alpha; \bar{\boldsymbol{X}}),$$
$$\bar{\alpha} = \overline{\mathrm{CZT}}_\alpha(\boldsymbol{x}, \alpha; \bar{\boldsymbol{X}}), \tag{48}$$

where $\overline{\mathrm{CZT}}_x$ is defined by Eq. (23) and $\overline{\mathrm{CZT}}_\alpha$ is defined by Eqs. (43)–(46). We have denoted the gradients with respect to $x$ and $\alpha$ as separate functions because they will appear in different places in the phase retrieval algorithm. In practice they would be computed simultaneously, taking advantage of the same intermediate calculations in Eq. (39). By using these gradients we can modify our phase retrieval algorithm from Section 4 to optimize over $Q$. This results in a forward model that is the same as Eqs. (1)–(5) but replacing $\mathrm{FFT}\{\boldsymbol{g}\}$ in Eq. (3) with $\mathrm{CZT}(\boldsymbol{g}, \alpha)$. The corresponding gradient model modifies Eq. (7) by changing $\mathrm{IFFT}\{\bar{\boldsymbol{G}}\}$ to $\overline{\mathrm{CZT}}_x(\boldsymbol{g}, \alpha; \bar{\boldsymbol{G}})$ and adding

$$\bar{\alpha} = \overline{\mathrm{CZT}}_\alpha(\boldsymbol{g}, \alpha; \bar{\boldsymbol{G}}). \tag{49}$$

In order to find a solution, we jointly optimize with respect to $\boldsymbol{a}$ and $\alpha$ in our nonlinear optimizer.

## 8. COMPUTER SIMULATION

In order to measure the effectiveness of this algorithm and its ability to find $Q$ values that are significantly different than the starting guess, we simulated noise-free PSFs with known $Q$ values and wavefront aberrations with an RMS magnitude of 0.1 waves. We simulated 16 independent random wavefront realizations. We repeatedly ran the phase retrieval algorithm described in the previous section, including optimization over $Q$, starting with a constant wavefront and different initial guesses for $Q$. Figure 1 shows (left) the true and fit PSFs and (right) the true and fit wavefronts for a case where the true value for $Q$ was 2.5 and the initial guess was 2.2. Figure 2 shows the history of three quantities over the optimizer run: $Q$, the RMS error of the wavefront compared with the true solution, and the normalized mean-squared error (NRMSE) disagreement between the model and the data (the square root of the phase-retrieval error metric). We can see that $Q$ stayed relatively constant in the initial iterations. Around iteration 15, the algorithm began to make progress and quickly found the correct value of $Q$, during which time the error metric and wavefront error did not change very much. Only after an approximately correct $Q$ value was found did the algorithm begin to make good progress, reducing the error metric and fitting the wavefront. The final value for the estimate of $Q$ was 2.50000006585, essentially perfect due to the absence of noise. The data consistency metric for the simulation was a bias- and gain-invariant metric [17].

Figure 3 shows the results from a Monte Carlo study of different true $Q$ values and initial estimates for $Q$ over the 16 different wavefront realizations (all with 0.1 waves RMS wavefront error). The horizontal axis shows the true simulated value of $Q$, the vertical axis shows the initial guess for $Q$, and the gray levels show the fraction of cases that

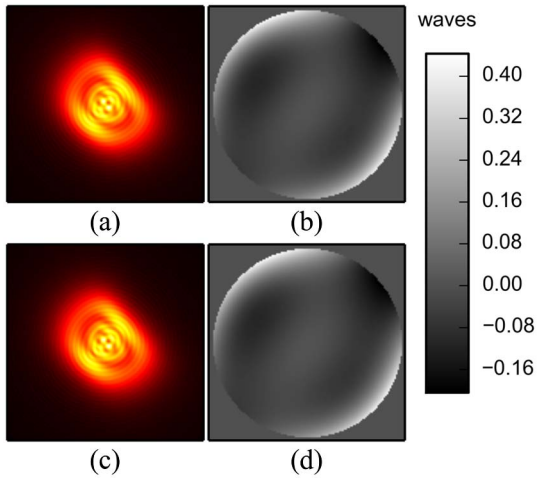


Fig. 1. Simulated (a) true PSFs, (b) true wavefront, (c) fit PSFs, and (d) fit wavefront. The color bar is in units of waves.

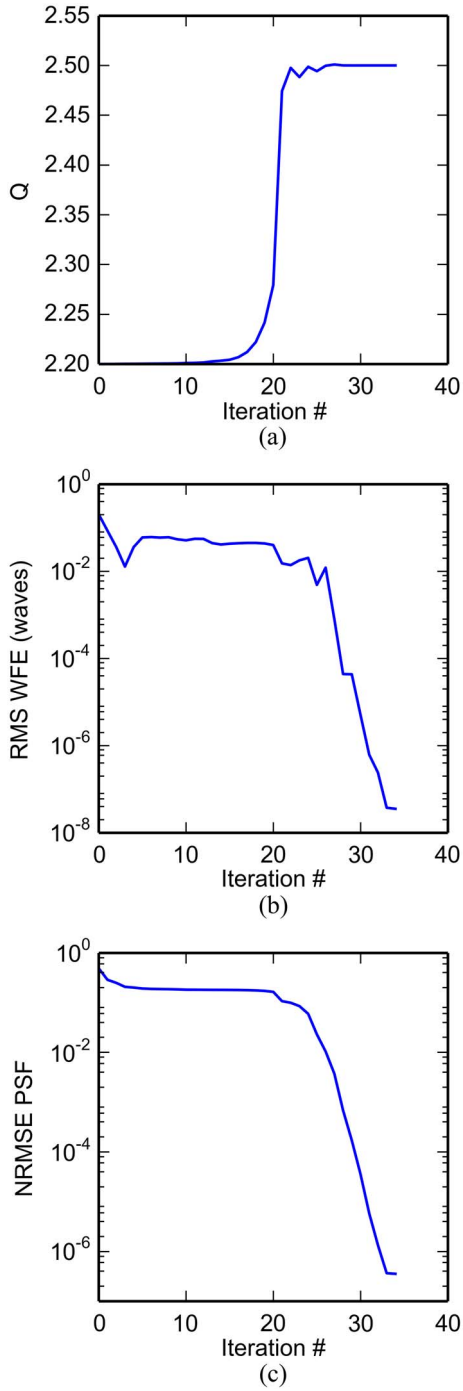


Fig. 2. (a) $Q$, (b) RMS wavefront error from truth, and (c) phase-retrieval error metric as a function of iteration number.

reached an excellent (accurate to within 0.01%) solution for $Q$. The simulated PSFs used for the Monte Carlo simulation included Poisson noise for 1,000,000 total photons and 10 photon-electrons RMS read noise. We note that the capture range is much larger than the few-percent level errors in knowledge of $Q$ we might expect in a reasonably well characterized system.

When attempting $Q$ retrieval simulations like these, one should be aware of a potential pitfall, which we encountered in earlier versions of these simulations. In the common definition of the DFT we used in Eq. (32), the total power in the output field depends on $\alpha$ (or on the pad length for a padded FFT). In simulations, we multiply the data by an appropriate constant to produce simulated intensity images at the desired signal level. If the phase retrieval model is constructed by multiplying the model data by the same constant used in the data simulation, then the constant will be known to the phase retrieval algorithm. This will give the algorithm a strong (but entirely nonphysical) signal to which it will fit $Q$: only the correct $Q$ will have a total power that agrees with the data. Using error metrics like that of [17] that presume no knowledge of the scaling between model PSFs and data avoids this problem.

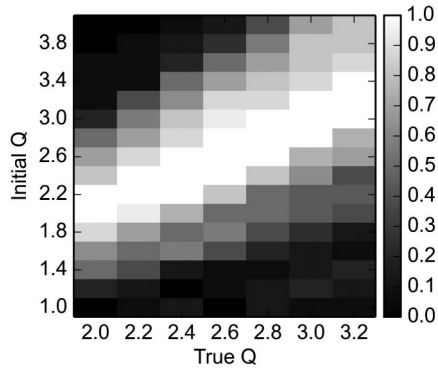


Fig. 3. Capture range for $Q$ retrieval. The color bar indicates the fraction of cases that converged to a correct solution for $Q$. The horizontal axis is the true value of $Q$ and the vertical axis is the initial guess for $Q$.

## 9. CONCLUSION

We have demonstrated a nonlinear-optimization phase retrieval algorithm for determining the sampling factor $Q$ based on a chirp $z$-transform numerical propagator and an algorithmic differentiation approach to constructing analytic gradients, including a derivation of the analytic gradients for the 2D case. This allows a direct optimization fit of $Q$ without using finite difference gradients or experiencing the unfavorable asymptotic scaling of the matrix-multiply DFT propagator. This enables $Q$ retrieval in problems involving high-fidelity models or large datasets. We have also briefly explored the capture range of the algorithm and observed that it is able to find the true $Q$ value even when the initial guess is significantly in error.

## REFERENCES

1. J. R. Fienup, J. C. Marron, T. J. Schulz, and J. H. Seldin, "Hubble space telescope characterized by using phase-retrieval algorithms," Appl. Opt. **32**, 1747–1767 (1993).
2. J. R. Fienup, "Phase retrieval algorithms: a comparison," Appl. Opt. **21**, 2758–2769 (1982).
3. J. R. Fienup, "Phase-retrieval algorithms for a complicated optical system," Appl. Opt. **32**, 1737–1746 (1993).
4. L. D. Feinberg, B. H. Dean, D. L. Aronstein, C. W. Bowers, W. Hayden, R. G. Lyon, R. Shiri, J. S. Smith, D. S. Acton, L. Carey, A. Contos, E. Sabatke, J. Schwenker, D. Shields, T. Towell, F. Shi, and L. Meza, "TRL-6 for JWST wavefront sensing and control," Proc. SPIE **6687**, 668708 (2007).
5. R. D. Fiete, "Image quality and λfn/p for remote sensing systems," Opt. Eng. **38**, 1229–1240 (1999).
6. T. P. Zielinski, B. H. Dean, J. S. Smith, D. L. Aronstein, and J. R. Fienup, "Determination of the sampling factor in a phase-diverse phase retrieval algorithm," in *Frontiers in Optics*,

OSA Technical Digest (online) (Optical Society of America, 2010), paper FWJ3.
7. T. P. Zielinski, "Robust image-based wavefront sensing," Ph.D. thesis (University of Rochester, 2011).
8. A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed. (SIAM, 2008).
9. A. S. Jurling and J. R. Fienup, "Applications of algorithmic differentiation to phase retrieval algorithms," J. Opt. Soc. Am. A **31**, 1348–1359 (2014).
10. A. S. Jurling and J. Fienup, "F/# estimation using a chirp z-transform and analytic gradients for wavefront sensing," in *Imaging and Applied Optics*, OSA Technical Digest (online) (Optical Society of America, 2013), paper OW3A.1.
11. M. D. Bergkoetter and J. R. Fienup, "A computational model for phase retrieval of narrow-band chromatic aberrations," in *Imaging and Applied Optics*, OSA Technical Digest (online) (Optical Society of America, 2013), paper OW3A.2.
12. L. Rabiner, R. Schafer, and C. Rader, "The chirp z-transform algorithm," IEEE Trans. Audio Electroacoust. **17**, 86–92 (1969).
13. L. Bluestein, "A linear filtering approach to the computation of discrete Fourier transform," IEEE Trans. Audio Electroacoust. **18**, 451–455 (1970).
14. D. Bailey and P. Swarztrauber, "The fractional Fourier transform and applications," SIAM Rev. **33**, 389–404 (1991).
15. R. Tong and R. W. Cox, "Rotation of NMR images using the 2D chirp-z transform," Magn. Reson. Med. **41**, 253–256 (1999).
16. R. P. Muffoletto, J. M. Tyler, and J. E. Tohline, "Shifted Fresnel diffraction for computational holography," Opt. Express **15**, 5631–5640 (2007).
17. S. T. Thurman and J. R. Fienup, "Phase retrieval with signal bias," J. Opt. Soc. Am. A **26**, 1008–1014 (2009).
18. S. J. Wright and J. Nocedal, "Quasi-Newton methods," in *Numerical Optimization* (Springer, 1999), Chap. 8, p. 198.