Research Article Vol. 28, No. 20 / 28 September 2020 / *Optics Express* 29590

Comparison of distributed memory algorithms for X-ray wave propagation in inhomogeneous media

SAJID ALI,[1] MING DU,[2] MARK F. ADAMS,[3] BARRY SMITH,[4] AND CHRIS JACOBSEN[2,5,6,*]

[1]*Applied Physics Program, Northwestern University, Evanston, Illinois 60208, USA*
[2]*Advanced Photon Source, Argonne National Laboratory, Argonne, Illinois 60439, USA*
[3]*Scalable Solvers Group, Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA*
[4]*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439, USA*
[5]*Department of Physics & Astronomy, Northwestern University, Evanston, Illinois 60208, USA*
[6]*Chemistry of Life Processes Institute, Northwestern University, Evanston, Illinois 60208, USA*
[*]*jacobsen@anl.gov*

Abstract: Calculations of X-ray wave propagation in large objects are needed for modeling diffractive X-ray optics and for optimization-based approaches to image reconstruction for objects that extend beyond the depth of focus. We describe three methods for calculating wave propagation with large arrays on parallel computing systems with distributed memory: (1) a full-array Fresnel multislice approach, (2) a tiling-based short-distance Fresnel multislice approach, and (3) a finite difference approach. We find that the first approach suffers from internode communication delays when the transverse array size becomes large, while the second and third approaches have similar scaling to large array size problems (with the second approach offering about three times the compute speed).

1. Introduction

Diffraction limited storage rings are providing the next advance in X-ray brightness from quasi-time-continuous synchrotron light sources [1]. These allow one to combine the high penetrating power and short wavelength of X-rays for nanoscale imaging of increasingly large specimens. Due to the overlap of features in a single view of an extended object, one must use tomography to obtain a 3D view of a specimen from a series of 2D projection images. However, as the transverse spatial resolution δ_{res} is improved, the depth of focus (DOF) decreases according to [2,3]

$$\mathrm{DOF} = 2\frac{\lambda}{\theta^2} = \frac{2}{0.61^2}\frac{\delta_{\mathrm{res}}^2}{\lambda} \simeq 5.4\frac{\delta_{\mathrm{res}}^2}{\lambda}, \tag{1}$$

where θ is the numerical aperture of the imaging optic, and $0.61 = 1.22/2$ comes from the Airy function for circular optics. Because of the depth of focus, features at different depths in an extended specimen are no longer sharply viewed in a single projection image. One way to overcome this limitation is to move to an optimization-based approach to image reconstruction, where one constructs a guess of the 3D object, calculates wavefield propagation through the object leading to an exit wave (and subsequently to predicted image intensities), and then adjusts the guess of the object until the difference between predicted and measured image intensities is minimized. Variations of such an approach have been demonstrated in electron microscopy [4,5], light microscopy [6–8], and X-ray microscopy [9–12].

In order to accurately represent the forward problem, these approaches all require one to implement computational wavefield modulation and propagation through a complex 3D object, and thus determine the complex exit wave leaving the object. This propagation is usually done with a multislice approach [13–15], where for one illumination direction one treats the object as

#400240 https://doi.org/10.1364/OE.400240
Journal © 2020 Received 16 Jun 2020; revised 11 Aug 2020; accepted 11 Aug 2020; published 21 Sep 2020

being comprised of a set of slices along the beam direction $\hat{z}$ with each slice being thinner than the DOF of Eq. (1). In this approach, the wavefield entering each slice is first modulated by the cumulative non-uniform refractive index variations of the slice along the beam direction, after which the resulting wavefield is transferred through the homogeneous average refractive index of the slice of material to the entrance of the next slice.

Once one has calculated the coherent exit wave leaving the specimen, one can model the subsequent transfer of this wave to measured intensities. This might be done using a lens to produce a direct measure of this exit wave in absorption contrast, or a lens with a Zernike phase ring to transfer weak phase variations to intensities, or holography over modest propagation distances, or far-field diffraction in methods such as coherent diffraction imaging and ptychography. Each of these approaches have their own relative merits, but in all cases, one needs to know the exit wave in order to calculate the expected intensities and compare them with measured intensities to improve one's guess of the complex refractive specimen.

Because of the potential for X-rays to be used for high resolution, beyond-DOF imaging of thick materials, we consider the question of the computational speed of various approaches for solving the forward problem when extended to large datasets. As an example, X-ray ptychography has been used to obtain $\delta_{\mathrm{res}} = 12$ nm images of integrated circuit features through 300 µm of silicon [16], and 8 nm resolution through 130 µm [17]. Both examples were of near-planar feature layers so that beyond-DOF imaging methods were not required. However, if one were to extend these results to a more general 3D object with a pixel size of half the achieved spatial resolution, one would need to propagate 2D wavefields with an array size of $[(300\ \mu\mathrm{m})/(6\ \mathrm{nm})]^2 = 50{,}000^2$ or $[(130\ \mu\mathrm{m})/(4\ \mathrm{nm})]^2 = 32{,}500^2$. Even larger array sizes are imaginable given that about 10% of a 15 keV incident X-ray beam is transmitted through 1 mm of silicon. It is therefore valuable to consider the computational costs of various approaches for large-array-size wave propagation through inhomogeneous materials.

The most commonly employed method [18–22] for computing evolution of an X-ray wavefield through an inhomogenous refractive object is to use the multislice (MS) method with Fresnel propagation to transfer the wavefield to the position of the next slice. As will be described in Sec. 2.1, this approach involves fast Fourier transforms (FFTs) and multiplication with the Fresnel propagator kernel of Eq. (8). However, another computational approach as described in Sec. 2.3 is to use the Helmholtz equation as a starting point, and solve for the exit wave using finite difference (FD) methods for solving partial differential equations (PDEs). In calculations for wave propagation in X-ray waveguides, this FD approach has been shown to offer speed and accuracy advantages [23,24].

We compare here multislice and finite difference based approaches for the calculation of large-array-size problems in X-ray wave propagation in inhomogeneous media. Comparisons of the two approaches at optical/UV wavelengths for fibres [25] and waveguides [26], and in the X-ray regime for waveguides as noted above, suggest that finite-difference methods are faster, and also more accurate as the propagation step size increases. However, those comparisons have been on problem sizes that fit on a single computational node, whereas for future X-ray experiments we wish to compare their performance on array sizes of $50{,}000^2$ or more pixels, as noted above. Therefore, we consider distributed memory parallel implementations of both methods. In the case of the multislice method using FFTs, we consider both a simple whole-array FFT approach, as well as a parallelized version of a short-distance tiling-based approach [27]. For the multislice and finite difference method, we make use of a well-developed software toolkit for solving partial differential equations PETSc/TS framework [28,29] on workstations as well as supercomputers. We first describe the mathematics of our approaches in Sec. 2, provide implementation details in Sec. 3, before discussing metrics in Sec. 4 and results in Sec. 6.

2. Algorithms

We consider here the forward problem of how to calculate the exit wave leaving a heterogeneous refractive index distribution for a large object. In the full-array Fresnel multislice approach, this is done by a sequence of wavefield propagation calculations for one guess of the object, which in turn, is nested within the iterative adjustment of the guess of the object to minimize the difference between predicted and measured image intensities. Because the calculation of wavefield propagation in a heterogeneous medium is undertaken repeatedly, we are interested in approaches that minimize computational time.

Specialized hardware has been developed that can calculate $(10{,}000)^2$ pixel holograms in just 100 ms [30], though this hardware only solves one step in the overall optimization problem, as noted above. Using a single workstation with a graphical processing unit (GPU), a solid-state drive (SSD) for rapid transfer of partial data to and from limited random access memory (RAM), and efficient tiling strategies as will be discussed in Sec. 2.2 below, single instances of short-distance wavefield propagation of $(131{,}072)^2$ pixel arrays have been demonstrated with an impressive calculation time of 3.6 minutes [27]. Doing calculations like this in a shorter computational time, and within the context of an optimization-based image reconstruction approach, can be achieved if one utilizes distributed memory parallelism in high performance computing clusters. These clusters typically consist of nodes that are connected by high-bandwidth, low-latency interconnects (with many advances on the latest supercomputers [31]), and protocols such as the message passing interface (MPI) for distribution and coordination of parallelized operations [32].

Because of our interest in X-ray microscopy applications, we limit ourselves to considering the refractive effect of an inhomogeneous medium. Most transmission imaging in X-ray microscopy is either done at soft X-ray photon energies around light element K absorption edges (0.2-0.8 keV), or energies of 2–15 keV where one obtains good penetration while still maintaining reasonable contrast from microscopic features [3,33]. In our energy range of interest, X-ray interactions are well described by a complex refractive index n of

$$n(x,y,z) = 1 - \delta(x,y,z) - i\beta(x,y,z) \tag{2}$$

where we have used the sign convention appropriate for writing forward wave propagation in the propagation direction $\hat{z}$ as $\exp[-iknz]$ with $k = 2\pi/\lambda$. The phase shifting part δ and absorptive part β of the refractive index are well tabulated [34], and are typically in the range of 10^{-3}–10^{-7}, with $\delta \gg \beta$ in most cases. When representing an object in a 3D array with slice thickness Δ_z along the illumination direction $\hat{z}$, the net refractive effect of the j^{th} slice is determined by

$$\begin{aligned} \delta_j(x,y) &= \frac{1}{\Delta_z}\int_{z_j}^{z_j+\Delta_z} \delta(x,y,z)\,dz \\ \beta_j(x,y) &= \frac{1}{\Delta_z}\int_{z_j}^{z_j+\Delta_z} \beta(x,y,z)\,dz \end{aligned} \tag{3}$$

leading to an advance in the phase φ of

$$\exp[i\varphi] = \exp[ik\delta_j(x,y)\,\Delta_z] \tag{4}$$

and a magnitude reduction $\exp[-a]$ of

$$\exp[-a] = \exp[-k\beta_j(x,y)\,\Delta_z] \tag{5}$$

for the slice. In visible light one might want to use the mean refractive index $\bar{n}$ for propagation to the plane of the next slice, and the refractive index variations $n(x,y,z) - \bar{n}$ the calculation of Eq. (3) for the effect of inhomogeneities within a slice. However, the small values of δ and β for X-rays make that unnecessary.

2.1. Full-array Fresnel multislice

As noted above, full-array Fresnel multislice is a well-known approach developed first in electron microscopy [13,14] and subsequently applied in visible light optics [15] and in X-ray microscopy [18–21] studies. With our particular interest in X-ray optics, this approach has been shown to produce results for very thick optics that are equivalent to those provided by coupled-wave equations [22], allowing for the simulation of the focusing properties of combinations of diffractive X-ray optics [35] as well as accurate modeling of the forward problem for image recovery of objects extending beyond the depth of focus limit [11,12]. What the approach leaves out is the ability to account for backscattered waves [36], but this effect is weak in X-ray interactions with non-crystallline media. Starting with a wave ψ_s incident on a slice, we first apply the phase advance and magnitude reductions of Eqs. (4) and (5) produced by the slice, giving

$$\psi_s^j(x,y) = \psi_s(x,y) \odot \left\{\exp\left[ik\Delta_z\delta_j(x,y)\right]\ \exp\left[-k\Delta_z\beta_j(x,y)\right]\right\} \tag{6}$$

as the modulated wavefield, where $\odot$ represents pointwise multiplication. We then propagate this modulated wavefield to the exit plane of this slice, giving a downstream wavefield $\psi_d(x,y)$ of

$$\psi_d(x,y) = \mathcal{F}^{-1}\left\{\mathcal{F}\left\{\psi_s^j(x,y)\right\} \odot H(u,v,\Delta_z)\right\} \tag{7}$$

$\mathcal{F}$ represents a Fourier transform and $\mathcal{F}^{-1}$ its inverse, and (u,v) are the transverse coordinates in the Fourier transform domain. The reciprocal space Fresnel propagation kernel $H(u,v,\Delta_z)$ of

$$H(u,v,\Delta_z) = \exp\left[-ik\Delta_z\sqrt{1-\lambda^2(u^2+v^2)}\right] \tag{8}$$

is preferred (rather than the equivalent kernel in real space) for short propagation distances to avoid aliasing artifacts [37,38]. One then has

$$N_z = t/\Delta_z \tag{9}$$

slices for the overall calcuation for a specimen with thickness t. This leads to Algorithm 1 for full-array Fresnel multislice.

How thin should the slices be in the multislice method? Based on Eq. (1), one would expect to require $\Delta_z \leq$ DOF. One comparison tested the full-array Fresnel multislice method (which can model arbitrary refractive index distributions) against coupled wave equation methods (which can be applied to easily defined, regular structures) [22]. This comparison used a parameterization that (in hindsight) is equivalent to the Klein–Cook parameter Q [39] of

$$Q = \frac{\pi}{2(1-\delta)}\frac{\lambda z}{\Delta_r^2} \tag{10}$$

for X-ray volume gratings with period $2\Delta_r$ and thickness z aligned to the propagation direction $\hat{z}$. When Q is well below 1, one can use simple scalar diffraction to describe the effects of the grating, whereas $Q \geq 1$ corresponds to the case where volume grating effects become pronounced. If we limit the slice thickness to a value such that $Q \leq 0.5$ and assume $1-\delta \simeq 1$, the slice thickness Δ_z should be kept below a value $z_{\text{K-C}}$ of

$$z_{\text{K-C}} \leq \frac{1}{\pi}\frac{\delta_{\text{res}}^2}{\lambda} \tag{11}$$

or $z_{\text{K-C}} \simeq 0.32\delta_{\text{res}}^2/\lambda$ instead of DOF $\simeq 5.4\delta_{\text{res}}^2/\lambda$. However, this is an extreme case that applies to regular gratings at the Nyquist limit, aligned to the beam propagation direction $\hat{z}$. In practice, a good approach is to start with Δ_z = DOF, and reduce it towards $\Delta_z = z_{\text{K-C}}$ while watching for asymptotic convergence of the exit wave.

Algorithm 1: Algorithm for implementation of full-array Fresnel multislice.

```
/* initialize                                                        */
$\psi(x,y) \leftarrow 1$
for $j$=1,$N$ do
    SliceDiff($j$)
    FreeProp($\Delta_z$)
end
Procedure SliceDiff($n$)
    /* Apply refractive effect of slice using Eq. 6                  */
    $\psi_s^j(x,y) = \psi_s(x,y) \odot \left\{\exp\left[ik\Delta_z\delta_j(x,y)\right]\,\exp\left[-k\Delta_z\beta_j(x,y)\right]\right\}$
    return
Procedure FreeProp($\Delta_z$)
    /* Free space propagation using Eq. 7 from source $s$ to destination $d$
       plane                                                         */
    $\psi_s^j(x,y) \xrightarrow{\mathcal{F}} \Psi(u,v)$
    $\Psi(u,v) = \Psi(u,v) \odot \exp\left[-ik\Delta_z\sqrt{1-\lambda^2(u^2+v^2)}\right]$
    $\Psi(u,v) \xrightarrow{\mathcal{F}^{-1}} \psi_d(x,y)$
    return
```

The refractive modulation step of Eq. (6) is a per pixel operation (*i.e.*, each pixel in the output array depends only on the corresponding pixel in the input array), leading it to be trivially parallelizable, and one could even distribute the set of $\delta_j(x,y)$ and $\beta_j(x,y)$ from all depth planes to the appropriate nodes prior to initiating the calculation. However, the Fourier transform of Eq. (7) is a whole-2D-array operation, so it is not trivially parallelizable. While there is considerable activity on developing efficient large-array parallel FFT implementations [40–42], inter-node communication requirements still set performance limits [43]. This motivates the use of other approaches for carrying out the operation of Eq. (7).

2.2. Tiling-based short-distance Fresnel multislice

In the mathematical definition of a discrete Fourier transform, the value of one input plane pixel affects all pixels in the transform, and vice versa. However, in short-distance wavefield propagation, information is localized due to the finite angle $\theta = \lambda/(2\Delta_r)$ of first-order diffraction from the finest features that are Nyquist sampled when using a pixel size of Δ_r [44,45]. For a propagation distance z_{prop}, this means that Nyquist-sampled diffraction information at the subsequent slice is contained within a radius r_1 of

$$r_1 = z_{\text{prop}}\tan(\theta) = z_{\text{prop}}\tan\left(\sin^{-1}\frac{\lambda}{2\Delta_r}\right) = \frac{\lambda z_{\text{prop}}}{\sqrt{(2\Delta_r)^2-\lambda^2}} \simeq \frac{\lambda z_{\text{prop}}}{2\Delta_r} \tag{12}$$

where the identity $\tan(\sin^{-1}(x)) = x/\sqrt{1-x^2}$ has been used; the approximate result applies to our case because the pixel size Δ_r is much larger than the X-ray wavelength. However, this does not incorporate the reality that interference fringes from weak features taper off in amplitude at large transverse distances. An alternative criterion is to consider diffraction from a half-edge, which can be characterized using the Cornu spiral [46] in terms of a dimensionless parameter $w = r_2\sqrt{2/(\lambda z_{\text{prop}})}$. This gives

$$r_2 = \frac{w}{\sqrt{2}}\sqrt{\lambda z_{\text{prop}}} \tag{13}$$

as an expression for the transverse distance r_2 for a given propagation distance z_{prop}. In half-edge Fresnel diffraction from a fully opaque object, one reaches the 8th dark fringe, where the intensity modulation is down to 8%, at a value of $w = 5.61$. Since X-ray microscopy usually involves weak phase objects, their effect at this transverse distance will be quite small; as a result, we use $w = 5.61$ to give

$$r_2 = 3.97\sqrt{\lambda z_{\mathrm{prop}}} \tag{14}$$

as a reasonable transverse distance beyond which there should be little effect from neighboring features.

Recognizing the limited transverse extent of diffraction from upstream features, one can use a tiling approach to parallelize the short-distance Fresnel multislice calculation [27]. In this approach, a large 2D array is split into a large series of much smaller tiles with buffer zones around their edges as shown in Fig. 1. One can then propagate these tiles separately, discard the buffer zones, and recombine the tiles to form the large 2D array at the downstream plane. These tiles can be sized to fit GPU memory [27] or other specialized hardware [30]. They can also be distributed to nodes on a high performance computing cluster, which is the approach we use here. Consider a large input wavefield array $\psi_s(N_x, N_y)$, and a refractive index array $\delta_{j=1\ldots N_z}(N_x, N_y) + i\beta_{j=1\ldots N_z}(N_x, N_y)$. The number of slices is N_z from Eq. (9), with Δ_z no larger than the depth of focus DOF of Eq. (1) and possibly as small as $z_{\text{K-C}}$ of Eq. (11). The tiles will have dimensions $\psi_s(N_{x,\mathrm{tile}}, N_{y,\mathrm{tile}})$ so that there are $N_x/N_{x,\mathrm{tile}}$ and $N_y/N_{y,\mathrm{tile}}$ tiles in the x and y dimensions, respectively. To any interior tile, one must add a buffer zone of physical width r_2 from Eq. (14), or pixel width

$$N_{\mathrm{buffer}} = r_2/\Delta_x, \tag{15}$$

to each side of the tile with information from neighboring tiles as shown in Fig. 1. This allows one to account for diffraction from features at the edge of nearby tiles into the field of view of the particular tile being processed. For a multislice calculation, one can choose between two alternative approaches to tiling-based Fresnel diffraction using these arrays:

- **2D tiling:** In this approach, at each slice one divides $\psi_s(N_x, N_y)$ into tiles $\psi_s(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$, and also the refractive index distribution $n = 1 - \delta - i\beta$ for slice j into tiles of $\delta_j(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$ and $\beta_j(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$. In this case, r_2 (Eq. (14)) and N_{buffer} (Eq. (15)) are calculated for the thickness of one slice, or $z = \Delta_z$. The tiles with their buffer zones are distributed to nodes. The refractive index modulation is then applied using Eq. (6), after which propagation by the slice thickness $z = \Delta_z$ is carried out using Eq. (7) to yield $\psi_d(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$. The buffer zone is then stripped, and $\psi_d(N_{x,\mathrm{tile}}, N_{y,\mathrm{tile}})$ is then returned. The various tiles of $\psi_d(N_{x,\mathrm{tile}}, N_{y,\mathrm{tile}})$ are then used to form the full input wavefield $\psi_s(N_x, N_y)$ entering the next slice, and the process is repeated. Because the free space propagation step of Eq. (7) is carried out over a small distance $z = \Delta_z$ ranging between $z =$ DOF and $z = z_{\text{K-C}}$, this approach has the advantage of requiring a smaller buffer zone size N_{buffer}. However, at each of the N_z slices of the calculation, it requires collecting the $\psi_d(N_{x,\mathrm{tile}}, N_{y,\mathrm{tile}})$ tiles from the computational nodes to re-form the full array $\psi_d(N_x, N_y)$ which then is used to distribute the set of $\psi_s(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$, $\delta_{j+1}(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$, and $\beta_{j+1}(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}})$ tile arrays to the computational nodes.

- **3D tiling:** In this approach, at the outset one divides the input wavefield $\psi_0(N_x, N_y)$ into tiles $\psi_0(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$ with r_2 (Eq. (14)) and thus N_{buffer} (Eq. (15)) calculated using $z = t$, the total sample thickness (a much larger value than the slice thickness Δ_z). One also generates 3D tilings of the refractive index arrays $\delta_{j=1\ldots N_z}(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$ and $\beta_{j=1\ldots N_z}(N_{x,\mathrm{tile}} + 2N_{\mathrm{buffer}}, N_{y,\mathrm{tile}} + 2N_{\mathrm{buffer}})$ for all the N_z slices, which are then distributed to computational nodes. The multislice calculation through all N_z slices can then be calculated on each node, after which the buffer zone is removed and the wavefield exiting the sample at each tile position is returned as $\psi_d(N_{x,\mathrm{tile}}, N_{y,\mathrm{tile}})$

so that the overall specimen exit wave $\psi_d(N_x, N_y)$ can be assembled. This approach has the advantage of not requiring any data transfer between computational nodes during the multislice calculation, but it involves each node carrying out its calculations on a larger array due to the increased size of N_{buffer} with $z = t$.

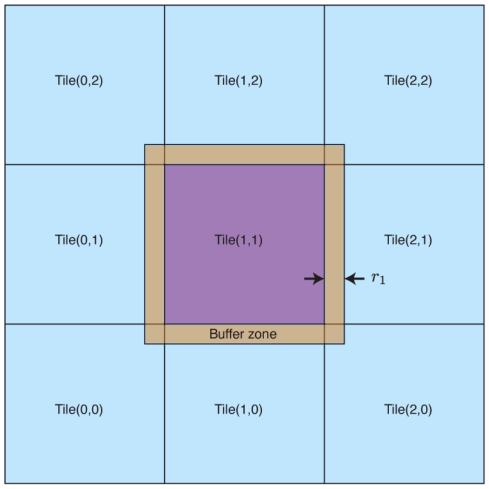

Fig. 1. For tiling-based short-distance Fresnel multislice, one can use a tiling approach to split a large 2D array of dimension $N_x \times N_y$ into a set of smaller arrays, each of size $N_{x,\text{tile}} \times N_{y,\text{tile}}$, so that these smaller arrays can be processed on separate computational nodes. When doing so, one must add a buffer zone of physical width r_2 (Eq. (14)), and pixel width $N_{\text{buffer}} = r_2/\Delta_x$ (Eq. (15)), to each side of the tile with information from neighboring tiles. This accounts for diffraction from features at the edge of nearby tiles coming into the field of view of the tile being processed.

We use the 3D tiling approach, as described in Algorithm 2. Before conducting numerical experiments using the choice of $r_2 = 3.97\sqrt{\lambda z_{\text{prop}}}$ as in Eq. (14), we conducted a validation test using a 256^3 voxel object that was also used in another publication [12]. The object array contains a hollow capillary tube positioned in the middle. We propagated a plane wave through the object, with the object divided into four 3D tiles of $128 \times 128 \times 256$ in a 2×2 grid. This way, each tile has a part of the non-vacuum object filling up to its edge; when the buffer zone width is too small, diffraction fringes of the object would wrap around and reenter from the opposite side, causing errors compared to the result given by full-array Fresnel multislice (the reference). We repeated the propagation simulation to sweep the value of $w/\sqrt{2}$ from 1 to 8, leading to the results shown in Fig. 2. When using $w/\sqrt{2} = 4$, which is very close to the value of 3.97 that we have chosen, the mean-squared-error (MSE) of the wavefield moduli between the output of tiling-based propagation and the reference falls to about 1×10^{-9}. Given that the variance of the reference modulus is 4×10^{-6}, this is a negligible error and should lead to sufficiently accurate results.

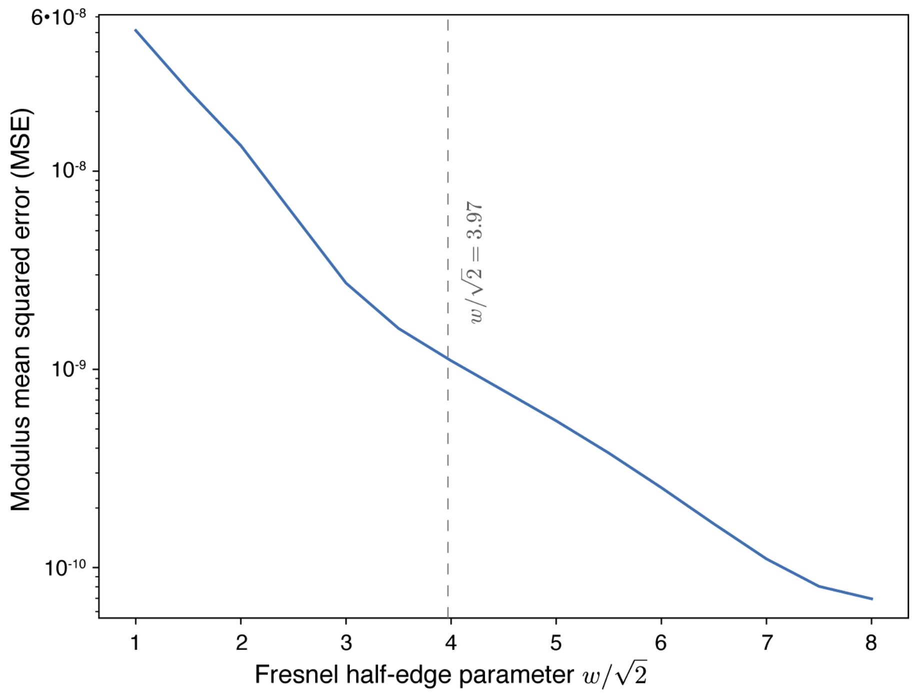

Fig. 2. Mean squared error of the exit wave of a subregion of a 3D object as a function of the buffer zone width $r_2 = (w/\sqrt{2})\sqrt{\lambda z_{\text{prop}}}$ of Eq. (13), showing that the choice of $r_2 = 3.97\sqrt{\lambda z}$ of Eq. (14) gives good results (a mean squared error of 10^{-9} compared to a variance in the reference modulus of 4×10^{-6}). Shown here is the result of using tiling-based short-distance propagation through a 256^3 voxel object as used in another publication [12]. The object was split into 4 128×128 tiles with the "seams" of the tiles running across the object, and buffer zones are added around each tile.

Algorithm 2: Algorithm for implementing the 3D tiling aproach to tiling-based short-distance Fresnel multislice. Variable names with lower case n, namely $n_{x,\text{tile}}$ and $n_{y,\text{tile}}$, denote the number of tiles in x and y. O_g and O_l are respectively the global and local (tile) object function, which symbolize both $\delta$and β. Similarly, ψ_g and ψ_l represent global and local wavefields.

```
forall i=mpi-ranks do
    /* initialize                                                                */
    $n_{x,\text{tile}}, n_{y,\text{tile}}, N_{x,\text{tile}}, N_{y,\text{tile}}$ = Init ($N_x, N_y, N_\text{ranks}$)
    $\boldsymbol{r}$ = DetermineTilePosition ($n_{x,\text{tile}}, n_{y,\text{tile}}, N_{x,\text{tile}}, N_{y,\text{tile}}, i_\text{rank}$)
    /* Initialize global wavefield array (assuming plane wave).                  */
    $\psi_g \leftarrow 1$
    $\psi_l$ = ExtractTile ($\boldsymbol{r}, \psi_g$)
    $O_l$ = ExtractTile ($\boldsymbol{r}, O_g$)
    for n=1,N do
        SliceDiff ($O_l$)
        FreeProp ($\Delta_z$)
    end
    WriteOutput ($\psi_l$)
end
Procedure Init ($N_x, N_y, N_\text{ranks}$)
    /* Determine work distribution.                                              */
    $n_{x,\text{tile}} = \text{int}\{\ [(N_x/N_y)\times N_\text{ranks}]^{1/2}\ \}$
    $n_{y,\text{tile}} = \text{int}\{\ [(N_y/N_x)\times N_\text{ranks}]^{1/2}\ \}$
    $N_{x,\text{tile}}, N_{y,\text{tile}} = \text{ceil}\{\ \max\ [N_y/n_{y,\text{tile}}, N_x/n_{y,\text{tile}}]\ \}$
    return $n_{x,\text{tile}}, n_{y,\text{tile}}, N_{x,\text{tile}}, N_{y,\text{tile}}$
Procedure DetermineTilePosition ($n_{x,\text{tile}}, n_{y,\text{tile}}, N_{x,\text{tile}}, N_{y,\text{tile}}, i_\text{rank}$)
    /* Determine local work area for this mpi rank                               */
    /* Return tile co-ordinates with the tile included                           */
    $\boldsymbol{r} \leftarrow [N_{y,\text{tile}}\times \text{floor}\{i_\text{rank}/n_{x,\text{tile}}\}, N_{x,\text{tile}}\times \text{mod}(i_\text{rank}, n_{x,\text{tile}})]$
    return $\boldsymbol{r}$
Procedure ExtractTile ($\boldsymbol{r}, O_g$)
    /* Fill local tile array with values from global array                       */
    $O_l \leftarrow O_g$
    return
Procedure WriteOutput ($\psi_l$)
    /* Combine the output at each rank to form global output while
       discarding buffer zone of size $N_\text{buffer}$, write it to HDF5 in parallel
       */
    return
```

2.3. Finite difference methods

The scalar Helmholtz equation of [2]

$$\nabla^2\psi + k^2\,(n(x,y,z))^2\,\psi = 0 \tag{16}$$

describes the propagation of a wave ψ with wavenumber $k = 2\pi/\lambda$ through an inhomogeneous medium with refractive index $n(x,y,z)$. To simplify its solution, the wave ψ is separated into two parts: a part $u(x,y,z)$ that varies in the weak refractive medium, and a part $\exp[-ikz]$ that is an unmodified forward-propagating wave in the propagation direction $\hat{z}$. This gives

$$\psi(x,y,z) = u(x,y,z)\exp[-ikz]. \tag{17}$$

Given the weak X-ray refractive index of Eq. (2), we can assume $\partial^2 u/\partial z^2 \ll \partial^2\psi/\partial z^2$ and approximate Eq. (16) with the parabolic wave equation [47,48] of

$$-2ik\frac{\partial u}{\partial z} + \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)u + k^2(n^2-1)u = 0. \tag{18}$$

If we make the definitions

$$\begin{aligned} A &\equiv \frac{-i}{2k} \\ F(x,y,z) &\equiv -\frac{ik}{2}\left[n^2(x,y,z)-1\right] \end{aligned} \tag{19}$$

$$= k\beta(x,y,z)(\delta(x,y,z)-1) + \frac{ik}{2}(\beta(x,y,z)^2 - \delta(x,y,z)^2), \tag{20}$$

we can write Eq. (18) as

$$A\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial u}{\partial z} + F(x,y,z)\,u = 0. \tag{21}$$

The expression of Eq. (21) presents a linear second order parabolic differential equation that describes a boundary value problem. Given that we know $u(x,y,z_s)$ (at the source plane) and require $u(x,y,z_d)$ (at the destination plane), it is more appropriate to rewrite Eq. (21) as an initial value problem [23] so that the equation being solved for at each plane is elliptic. Note that while a more recent formulation of an equivalent to Eq. (22) exists [24], the expression of Eq. (22) is sufficiently accurate for our purposes given the fact that we work at the hard X-ray energy regime. The formulation of Eq. (21) has also been used in prior studies of X-ray wave propagation in thick zone plates [49] and waveguides [23]. We can rewrite Eq. (21) as

$$\frac{\partial u}{\partial z} = A\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + F(x,y,z)\,u. \tag{22}$$

The expression of Eq. (22) can be discretized by the use of finite difference methods. Traditionally, the space derivatives are evaluated using a central difference scheme and the time integration is performed via implicit methods (where we have defined time to be the coordinate along the propagation axis). As noted in Sec. 1, this finite difference method has been shown to outperform the full-array Fresnel multislice algorithm when comparing compute time for the same degree of accuracy on single node computers [23,24].

The general Helmholtz equation problem is known to be challenging to solve using finite difference methods [50]. Previous implementations have favored methods that only require tridiagonal matrix inversions using the Thomas algorithm [51]. For one-dimensional systems, the Crank-Nicolson method [52] has been used, while two-dimensional problems have been tackled using Alternating Direction Implicit schemes [51,53] where the wave is propagated along one axis at a time to generate the familiar tridiagonal system of equations. The main disadvantage of ADI is poor scalability to large-scale problems [54].

Instead of formulations that require tridiagonal inversions, we employ iterative solvers along with preconditioners to enable the use of the Crank-Nicholson method for both one- and two-dimensional problems. As expanded upon in the implementation section, we are not required to program these algorithms since we express the problem using PETSc [28,29] which allows us to compose scalable solvers.

The recent availability of high-level discrete adjoint frameworks [55,56] offers another approach for the optimization problem. These frameworks allow one to access the sensitivity of the parameters necessary for an optimization-based inverse-problem reconstruction algorithm. These automatically generated adjoint solvers utilizing the same mode of parallelism as the equations, and potentially run faster than the forward problem (owing to the properties of adjoints and the fact that the adjoints are implemented as a series of linear solves). This is in contrast to the approach used by algorithmic differentiation [57,58], which operates on low-level operations and therefore does not offer quite as high performance.

3. Implementation

The full-array Fresnel multislice (Sec. 2.1) and finite difference (Sec. 2.3) algorithms described above have been implemented using the 3.13.1 release of the PETSc/TS framework [28,29] which is designed to support scalable solvers for partial differential equations (with code available [59]). PETSc supports distributed memory computing using the Message Passing Interface (MPI) [60] as well as the use of graphical processing units. The tiling-based short-distance Fresnel multislice algorithm (Sec. 2.2) was implemented in Python (with code available [61]) using the `mpi4py` package [62,63] for distributed memory parallelism, and the scientific Python stack `SciPy` [64] for multithread parallelism for each MPI task. All algorithms used the HDF5 library [65] for parallel disk I/O.

PETSc was installed on workstations and clusters using the spack package manager [66] with the Intel Compiler Collection to take full advantage of the underlying hardware. The Intel Math Kernel Library was chosen as the BLAS/LAPACK implementation for optimal performance for all algorithms.

Initial development and debugging was done on a Linux-based workstation “xrmlite.” Algorithm composition and tuning for optimal distributed memory performance were carried out on the cluster “bebop,” while final scaling studies were performed using the supercomputer “theta,” both at Argonne National Laboratory. The characteristics of these systems are listed in Table 1. PETSc does not use multi-threading, but benefits from higher memory bandwidth, which is available on the KNL processor at high process counts. The tiling-based short-distance Fresnel multislice method prefers the number of ranks per node to be a perfect square, which in this case, happens to match the maximum number of physical cores. Thus, for tests of all three approaches on “theta,” we set the CPU affinity to “depth”, used one thread per rank, and one thread per core. The terminology for the configuration options is given in [67]. We used the `balsam` workflow manager [68] to pack multiple jobs for queue submission.

Table 1. Compute systems used and their configuration. The machine “xrmlite” is a Linux workstation at Northwestern University. The cluster “bebop” is at the Laboratory Computing Resource Center (LCRC) at Argonne National Laboratory (with four Northwestern University nodes included), while the cluster “theta” is at the Argonne Leadership Computing Facility (ALCF) [69,70]. With both “bebop” and “theta,” we used only a fraction of the large number of available nodes for the strong scaling studies described in Sec. 6.2. We note that “xrmlite” has two Quadro P5000 GPUs, each connected to the CPU via PCIe3.0.

Compute System	Processor	Cores/node	Memory/node	Interconnect
xrmlite	2x Xeon E5-2620 v4	16	512 GB	N/A
bebop/LCRC	Xeon Phi 7250	68	192 GB	Omni-Path Fabric
theta/ALCF	Xeon Phi 7230	64	192 GB	Aries Dragonfly

3.1. Full-array Fresnel multislice

The full-array Fresnel multislice algorithm was implemented using the PETSc [28,71] framework which provides data structures for scalable and efficient linear algebra [72].

The PETSc application programmer interface (API) conceptualizes the fast Fourier transform (FFT) as a matrix multiplication by an "FFT" matrix, where FFT(x) is a matrix multiply $A * x$, but the A matrix is never explicitly constructed. Behind the scenes, the FFT is executed by the FFTW library [73] on CPUs. This matrix multiply can, however, only be performed on a specific class of vectors, since FFTW has its own requirements for distribution of data. PETSc also includes functionality to either create vectors that conform to the FFTW format (with the correct data distribution and padding), or the ability to scatter data from a regular MPI vector to a FFT-compatible vector.

We choose the "FFTW format" as the data structure for all of the vectors that are used in the full-array Fresnel multislice algorithm. This frees us from the tedious task of performing explicit data restructuring to switch between having the wave be FFTW-aligned and having it be distributed as a regular array for dot products (corresponding to Eq. (6)). The only downside to this approach is the poor scalability of distributed memory FFT for a large number of MPI ranks [40,41].

The above implementation of the multislice algorithm makes it straightforward to carry out the functions described in Algorithm 1 using the PETSc API.

3.2. Tiling-based short-distance Fresnel multislice

We used a hybrid programming model combining the message-passing interface (MPI) and multi-threading to implement the tiling-based short-distance Fresnel multislice algorithm. After propagation, the buffer zone of size N_{buffer} on each edge of a wavefield tile is discarded, and the valid region of the wavefield is written directly into the correct position of the output array. The output array is stored in an HDF5 file that is accessed in parallel by all ranks. The HDF5 [65] library was accessed via the Python interface `h5Py` [74,75]. Distributed memory programming was done via the `mpi4Py` package [62,63] which provides Python bindings to the MPI standard. Fast Fourier transforms (FFTs) were performed using the Intel-processor-optimized package `mkl-fft` [76] via its `NumPy` bindings.

3.3. Finite difference

For the finite difference approach, the TS ODE/DAE [29] integrator library (distributed as part of PETSc/TAO) provides a wide variety of scalable solvers for ordinary differential equations (ODEs) and differentiable-algebraic equations (DAEs), obviating the need to write explicit time integration algorithms. Therefore, we chose to implement the finite difference problem as a linear time-step (TS) object in PETSc. To manage the distributed memory grid, we used PETSc's data-management distributed-array (DMDA) object [28] which is designed for optimal performance when using logically rectangular grids (it re-orders the memory mapping to suit typical differential equation solver operations). As mentioned earlier, previous implementations [23,24] of parabolic wave equation solvers relied on algorithms that were not easily scalable to distributed parallel compute nodes. PETSc enables using a wide variety of preconditioners and Krylov solvers which can be tuned to the problem at hand, thus allowing us to design an algorithm with superior performance and scaling characteristics for parallel computing.

The discretization in space was performed via the central differencing scheme for space derivatives, and the time integration was performed by the TS object using either a first or second-order implicit method as dictated by the needs to the problem being solved. Because our eventual goal is to go from solving the forward propagation problem for a particular object guess (the forward problem), to solving for the object (the inverse problem), maintaining large propagation sizes per step is important as this ensures a minimal size of the refractive index grid while still accurately modelling the diffraction phenomenon. For this reason, we did not test explicit methods such as Euler or non-adaptive Runge-Kutta, as these are unstable at large step sizes. While a one-stage second order implicit method (known as the implicit midpoint method) gives the same result as a two-stage second order implicit method (with endpoint, known as Crank-Nicholson), the two-stage method is significantly faster. Therefore, we used the Crank-Nicholson scheme [52] in PETSc.

We used a GMRES linear solver [77,78] with preconditioning determined by the nature of diffracting object. When the object has some order to its structure (such as for simulating the focusing of thick Fresnel zone plates), algebraic multigrid preconditioning [79–81] was used. When the object of interest is better characterized as being irregular, we observed that an additive Schwarz preconditioner [82,83] was faster. For elliptic problems, one-level additive Schwarz methods are known to be non-optimal with increasing problem size (hence one needs multigrid methods). Depending on the ratio of the time-step size to the subdomain size squared for parabolic problems, it is possible to show the algorithms are optimal [84]; that is, the coarser level solvers of multigrid are not needed. This phenomena is similar to the fact one can replace the full-array Fresnel multislice method with the tiling-based short-distance Fresnel multislice method.

The general idea of multigrid schemes arises from the observation that low frequency residuals are challenging to eliminate using classical relaxation-based preconditioning schemes. Thus multigrid preconditioning works by transferring the residual to a coarser grid (where the residual now contains high frequency components), solving for which gives an estimate for the error which is then transferred back to the fine grid. Classical geometric multigrid preconditioning [85] uses interpolation operators based on the grid geometry to generate the coarse grids. However, algebraic multigrid preconditioning requires no information about the grid, and constructs coarse grids based on the system of equations being solved [80,81]. The selection of coarse grids (akin to graph partitioning) and construction of interpolation operators (with a Galerkin process) together form the “setup” phase. These coarse grids and interpolation operators can be reused for subsequent applications [80,81] of the preconditioner, thereby amortizing the cost of the setup phase. Algebraic multigrid preconditioning has been shown to work well for discretized Helmholtz operators [86].

The general idea of domain decomposition schemes is to split the task of solving the system of equations (arising from the partial differential equation discretization) from one large domain into smaller overlapping domains [82,83,87]. These sub-blocks are then solved independently and these solutions are then iteratively combined. In particular, we used the restricted additive Schwarz method as a preconditioner, which has been shown to improve performance when compared to using it as a solver [87].

4. Wavefield convergence metric

In order to numerically compare the step size sensitivity of each method, we measured the minimum number of slices $N_{z,\min}$ that each method takes to yield a converged result. In the limit of taking thinner slices (that is, as the number of slices N_z is increased towards $N_{z,\infty}$), multislice calculations converge on an exit wave ψ_d with magnitudes $A_{\infty,i}$ and phases $\phi_{\infty,i}$ at pixel positions i. However, if one were to calculate convergence using the phase $\phi_{\infty,i}$ of the exit wave ψ_d, one would possibly need to use phase unwrapping from the complex wavefield, which can be time-consuming and prone to error. The problem can be circumvented by calculating the root-mean-square (RMS) difference of the complex wavefields. Suppose the magnitude of the wavefield calculated using N_z steps is $A_{n,i}$ at pixel i, while the converged magnitude one would obtain using an infinite number of infintitessimally thin slices is $A_{\infty,i}$; also, suppose the phases in the two cases are $\phi_{n,i}$ and $\phi_{\infty,i}$, respectively. The complex wavefield RMS difference is given by

$$\begin{aligned}\xi_{\text{complex}} &= \left[\frac{1}{N}\sum_i \left[A_{n,i}\exp(i\phi_{n,i}) - A_{\infty,i}\exp(i\phi_{\infty,i})\right]^2\right]^{\frac{1}{2}} \\ &= \left[\frac{1}{N}\sum_i \left[A_{n,i}^2 + A_{\infty,i}^2 - 2A_{n,i}A_{\infty,i}\cos(|\phi_{n,i}-\phi_{\infty,i}|)\right]\right]^{\frac{1}{2}}.\end{aligned} \tag{23}$$

Because phase contrast dominates in hard X-ray imaging, we can assume $A_{n,i} \simeq A_{\infty,i} \simeq \bar{A}$ everywhere, with $\bar{A} \simeq \exp[-k\bar{\beta}t]$ using the Lambert-Beer law of $I = I_0\exp[-2k\beta t]$ with the X-ray refractive index $n = 1 - \delta - i\beta$ [3], and $\bar{\beta}$ indicating the spatial average within the inhomogenous specimen. Since $\cos(x) \approx 1 - x^2/2$, Eq. (23) reduces to

$$\begin{aligned}\bar{A}\,\xi_{\text{complex}} &= \bar{A}\sqrt{\frac{1}{N}\sum_i \left[2 - 2\cos(|\phi_{n,i}-\phi_{\infty,i}|)\right]} \\ &\simeq \bar{A}\sqrt{\frac{1}{N}\sum_i |\phi_{n,i}-\phi_{\infty,i}|^2} \\ &\simeq \bar{A}\,\xi_\phi.\end{aligned} \tag{24}$$

This approximation is illustrated in Fig. 3. Since the RMS phase error ξ_ϕ represents the standard deviation of a Gaussian distribution, the net reduction in the summation of amplitudes from many

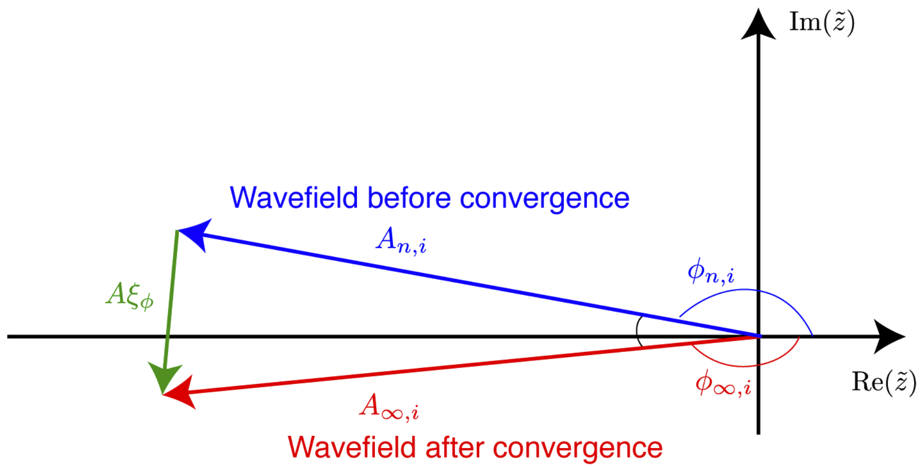

Fig. 3. Illustration of the metric for measuring the RMS average $\bar{A}\,\xi_\phi$ of the magnitude error at one pixel i between the complex value before convergence ($A_{n,i}\exp[i\phi_{n,i}]$; shown in blue) and after convergence ($A_{\infty,i}\exp[i\phi_{\infty,i}]$; shown in red). When obtaining a particular measure of the phase difference $\phi_{n,i} - \phi_{\infty,i}$ from a complex value $\tilde{z} = A\exp[i\phi]$ on the real (Re) and imaginary (Im) plane, one could obtain erroneous values in the case shown where the phase before convergence is reported as $\pi - \epsilon_{n,i}$ while the phase after convergence is reported as $-(\pi - \epsilon_{\infty,i})$, one would obtain an erroneous phase difference $\phi_{n,i} - \phi_{\infty,i}$ of near 2π. Calculating the RMS difference between complex wavefields (Eq. (24)) circumvents this problem by measuring the end-to-end distance between the red and blue vectors at individual pixels i, a result that does not require phase wrapping. When the moduli $A_{n,i}$ and $A_{\infty,i}$ are similar, the average modulus of the green vector (labeled here as $\bar{A}\,\xi_\phi$ using Eq. (24)) is approximately linearly related to the RMS average of $\bar{A}\,|\phi_{n,i} - \phi_{\infty,i}|$ subtended by the blue and red vectors.

waves [3,88,89] is given by $\exp[-\xi_\phi^2/2]$. Therefore if we set the requirement

$$\xi_\phi \leq (0.05)\, 2\pi, \tag{25}$$

we see that we obtain errors in the unattenuated amplitude ($\bar{A} \simeq 1$) of a wave no greater than 4.8%, since $\exp[-(0.05(2\pi))^2/2] = 0.952$. This is far more stringent than the usual Rayleigh quarter wave criterion for tolerance of phase errors. We therefore judge convergence by decreasing the slice thickness Δ_z (and thus increasing the number of slices N_z for a given specimen thickness t) until further decreases in Δ_z lead to changes in ξ_ϕ of less than $(0.05)2\pi = 0.31$ in accordance with Eq. (25). This gives us a measurement for the number of slices n_C required to reach convergence of

$$n_C = \min\{n|\bar{A}\, \xi_\phi(n) \leq \bar{A}\,(0.05)2\pi\} \tag{26}$$

which we will report for various tests of calculating X-ray wave propagation through thick inhomogeneous media.

For cases where the inhomogeneous object is surrounded by a featureless outer border (such as is the case for circular zone plates within a rectangular array), the calculation of the RMS amplitude error $\xi_{\text{complex}} \simeq \bar{A}\, \xi_\phi$ shown below will be from the feature-containing region, with featureless regions excluded.

5. Experiments

Our goal is to understand the characteristics of the full-array Fresnel multislice, tiling-based short-distance Fresnel multislice, and finite difference methods for propagating large area X-ray wavefields through thick inhomogeneous media. To do this, we carried out numerical tests using two different diffracting objects: a Fresnel zone plate thick enough that waveguide effects become apparent (Sec. 5.1), and a X-ray microtomography reconstruction of a charcoal specimen scaled to match the conditions of nanoscale imaging (Sec. 5.2). In order to understand the relative scattering power of these objects, we calculated the object's RMS phase deviation as

$$\sigma_\phi = \frac{2\pi\Delta_z}{\lambda}\left[\frac{1}{N_{x,y}}\sum_{x,y}\Big[\sum_z \delta(x,y,z) - \bar{\delta}(z)\Big]^2\right]^{\frac{1}{2}} \tag{27}$$

where $\bar{\delta}(z)$ refers to a uniform object with the phase shifting part of the refractive index set to the weighted mean of the refractive indices of the same slice (with axial coordinate z). For our calculations, we assumed a photon energy of $E = 15$ keV (giving $\lambda = 0.0827$ nm), and a transverse calculation grid size or pixel size of $\Delta_x = 2$ nm. Assuming that half-period features can be as small as $\delta_{\text{res}} = \Delta_x$, one can use Eq. (1) to find a calculation depth of focus of DOF = 26.0 nm, and Eq. (11) to find that the thickness at which the Klein-Cook parameter becomes $Q = 0.5$ is given by $z_{\text{K-C}} = 1.54$ nm (so that one can assume scalar diffraction, without waveguide effects, within one slice thickness). However, one can in fact use larger slice thicknesses Δ_z and still meet our convergence criteria $\xi_\phi \leq 0.05(2\pi)$ from Eq. (25), as will be shown below.

5.1. Fresnel zone plate test object

Fresnel zone plates (Fig. 4) are widely used as X-ray nanofocusing optics [3], since they offer normal incidence mounting and easy energy tunability. For conventional Fresnel zone plates, the spatial resolution is given by $\delta_{\text{res}} = 1.22 dr_N$ where dr_N is the width of the outermost, finest zone, and for gold zone plates at $E = 15$ keV a thickness along the X-ray beam direction $\hat{z}$ of about $t = 3$ µm is required to achieve focusing efficiencies that can in theory be as high as about 25%. Since we wish to test the ability of different wave propagation calculation methods to account for waveguide effects in thick structures, we chose to simulate a zone plate with a finest outermost zone width of $dr_N = 20$ nm and a thickness of $t = 30.81$ µm, giving a Klein-Cook parameter

(Eq. (10)) of $Q = 10$ for the zone width rather than the pixel size. For this zone plate, the depth of focus DOF corresponding to the spatial resolution of $\delta_{\text{res}} = 1.22 dr_N$ is DOF = 38.7 µm (Eq. (1)), while the distance over which the zone width would produce a value for the Klein-Cook parameter of $Q = 0.5$ is $z_{\text{K-C}} = 2.3$ µm (Eq. (11)). The magnitude of the exit wave for a zone plate averaged over the central region (≈ 25% side length) is $\bar{A} \approx 0.43$ for a plane wave input with a magnitude of unity. Thus, threshold for convergence (Eq. (24)) for the zone plate test object is

$$[\bar{A}\, \xi_\phi]_{\text{zp}} = \bar{A}\,(0.05)2\pi = 0.135. \tag{28}$$

Within this central region, the diffractive power (Eq. (27)) was calculated to be $\sigma_\phi = 13.60$.

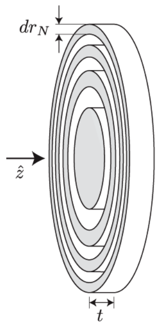

Fig. 4. Fresnel zone plate test object, with a thickness t and a finest zone width of dr_N. The beam propagation direction $\hat{z}$ is also indicated.

Because we wanted to explore the scaling of our calculation with increasing array size (N_x, N_y) with constant pixel size $\Delta_x = 2$ nm, at each array size we generated a zone plate with the above minimum zone width dr_N and thickness t, but with a diameter d equal to 80% of the array size, or $d = 0.8 N_x \Delta_x$. We used partial voxel filling of the zone plate material's refractive index to handle the cases where the boundary of a zone plate zone was within a voxel [22]. Since there is no variation of the zone structure along the direction of propagation $\hat{z}$, we only need to store one two-dimensional array for each (N_x, N_y) array which greatly simplifies storage issues. When using a plane wave ψ_s for illumination, we would be able to propagate the exit wave ψ_d to the focus position f given by

$$f = \frac{d \cdot dr_N}{\lambda} \tag{29}$$

which we have done in other studies [22,35]. However, for our present purposes, we just wish to find the minimum number of slices that lead to convergence of the exit wave to approximately the same value obtained with using a much larger number of slices in the calculation: that is, n_C as described by Eq. (26).

5.2. Porous aluminum test object

As noted above, a Fresnel zone plate provides a structure that can be extended axially to the case where waveguide effects come into play. However, a zone plate is also a highly regular structure, whereas more general specimens in X-ray microscopy are quite irregular.

For a test object that more accurately represents objects that are imaged at synchrotron sources, we used part of an X-ray tomographic reconstruction of an activated charcoal sample acquired in a previous study [90], and now available as the dataset "activated-charcoal" in TomoBank [91]. This 4 mm diameter specimen was imaged using 25 keV X-rays, with a reconstruction pixel size of 0.6 µm, resulting in object slices of 6613 × 6613 pixels, and a total number of 4198 object slices (or a 6613 × 6613 × 4198 voxel array). We then generated a baseline phantom from a subvolume from this array in a manner we now describe (and which is illustrated in Fig. 5). Each object slice in the original reconstruction had a slight ring artifact near the rotation axis center due to imperfect alignment of the data on the reconstruction rotation axis, and these rings could have contributed to a cylindrical waveguide artifact in the final phantom. Therefore, a $2448 \times 2448 \times 4198$ voxel subregion was selected that did not include this ring artifact. This subregion was then replicated into a 2×2 grid in the plane of the object slices, with pyramid blending [90,92] used at the tile overlaps, and the outermost 15% of the object slice area blended out to vaccum (that is, to a specimen density of zero). This resulted in a $4096 \times 4096 \times 4198$ voxel volume, which was then rotated so that the original tomographic rotation axis became the beam propagation direction. Since the multislice propagation slice thickness Δ_z is usually much larger than the transverse pixel size Δ_x, we then selected 51 tomographic reconstruction object slices (each separated from its neighbor by 50 slices, out of the center $51 \cdot 50 = 2550$ slices in the 4198 slice direction) from this volume. This way, the selected slices are sufficiently different to avoid waveguide effects. This yielded a $4096 \times 4096 \times 51$ voxel array, with 4096×4096 pixels in the transverse direction and 51 pixels along the beam propagation direction $\hat{z}$. Finally, the "baseline" phantom in our numerical study is assumed to contain 500 slices of thickness $\Delta_z = 147.46/500 = 0.295$ μm along the beam propagation direction $\hat{z}$. This $(N_x \times N_y \times nz) = (4096 \times 4096 \times 500)$ voxel baselone object was formed by looping the 51 object slices back and forth (*i.e.*, arranging the slices in the order 1, 2, . . ., 50, 51, 50, 49, . . ., 2, 1, 2, . . . and its repeat). For convergence tests where N_z was varied to find n_C of Eq. (26) in both Fresnel multislice methods, a smaller number of slices were obtained by linear interpolation of the baseline object along the propagation direction $\hat{z}$, leading to a modified phantom of $4096 \times 4096 \times N_z$ voxels.

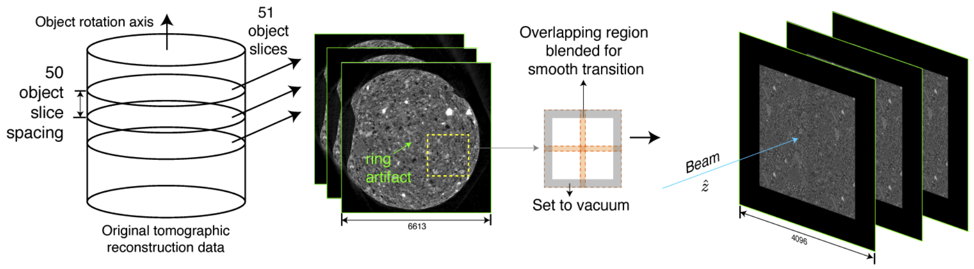

Fig. 5. The process used to generate the porous aluminum phantom object (right). A larger scale tomographic reconstruction of an activated charcoal specimen (left) was used as the data source. From the 4198 tomographic reconstruction slices of 6613×6613 pixels each, a $2448 \times 2448 \times 51$ voxel subregion was selected through all slices to avoid ring artifacts near the rotation axis. This subregion was then replicated into a 4×4 grid in the plane of the object slices, with pyramid blending used at the tile overlaps and the edges blended out to vaccum (that is, to a specimen density of zero). The resulting $4096 \times 4096 \times 51$ voxel array was then rotated so that the original data rotation axis (veritcal, at left) became the beam propagation direction $\hat{z}$ in the phantom object at right, after which both the pixel size and the contrast of the object were modified to yield the porous aluminum phantom object.

The original tomographic reconstruction was acquired using absorption contrast, whereas we wished to simulate a complex object. Therefore we normalized the absorption map so that it had a mean occupancy of 1, and multiplied it by the tabulated value [93] of $\delta + i\beta$ (Eq. (2)) for aluminum at 15 keV, effectively giving each voxel a different fractional filling with aluminum. The histogram of the resulting densities shown in Fig. 6 reveals that this led to some pixels having unphysically high densities (which we realized after our convergence and scaling tests were complete), but this only serves to make the object have slightly larger refractive properties with no impact on measuring convergence or calculating speeds of the algorithms tested. We also added to each slice 10–20 disk-shaped gold particles with radii ranging from 5 to 20 pixels (10–40 nm), and a thickness of one slice (or $\Delta_z = 0.295$ µm) to create strongly scattering features. This was accomplished by replacing the $\delta + i\beta$ of aluminum with that of gold at those positions. The total object thickness was set to $t = 147.5$ µm so that the object gave the same diffractive power $\sigma_\phi = 13.60$ (Eq. (27)) as the zone plate object. The object thus created had a magnitude of its exit wave of $A = 0.78$ averaged over the central region, leading to a threshold for convergence (Eq. (24)) of

$$[\bar{A}\,\xi_\phi]_{\mathrm{Al}} = \bar{A}\,(0.05)2\pi = 0.245. \tag{30}$$

We refer to this test specimen as the porous aluminum phantom.

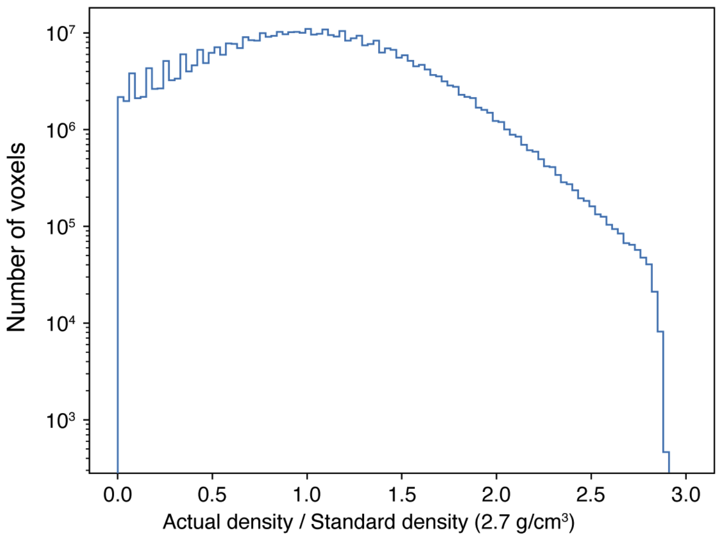

Fig. 6. Histogram of voxel densities of the porous aluminum test object. By setting the average occupancy of the charcoal test object to 1 and then multiplying by the refractive index of aluminum, the porous aluminum test object was inadvertently created with voxel densities exceeding the actual density of aluminum. This means that the test object was more strongly refracting than a true aluminum object would be, but this does not affect our measurement of the convergence or timing properties of the algorithms tested.

6. Results

6.1. Convergence results

Our first test was to compare the convergence of the three algorithms as a function of the number of slices $N_z = t/\Delta_z$ (Eq. (9)). For this test, we used the porous aluminum test object with thickness $t = 147.5$ µm and $(4096)^2$ transverse pixels. The object was re-sampled along the propagation direction $\hat{z}$ to vary Δ_z and thus N_z. The exit wave was then calculated with the three algorithms of Sec. 2 and successful convergence was measured using $\bar{A}\,\xi_\phi$ of Eqs. (24) and (25) using the value $[\bar{A}\,\xi_\phi]_{\mathrm{Al}} = 0.245$ of Eq. (30). As seen in Fig. 7, the Fresnel multislice approaches gave similar values for the minimum number of slices $n_C = 84$ for full-array Fresnel multislice and $n_C = 90$ for tiling-based short-distance Fresnel multislice. The corresponding maximum slice thicknesses of $\Delta_z = 1.76$ and $\Delta_z = 1.64$ µm, respectively, are both well beyond the minimum pixel value of DOF = 0.026 µm noted above. For the finite-difference algorithm, a much larger minimum number of slices of $n_C = 352$ was required, corresponding to $\Delta_z = 0.42$ µm for this irregular object.

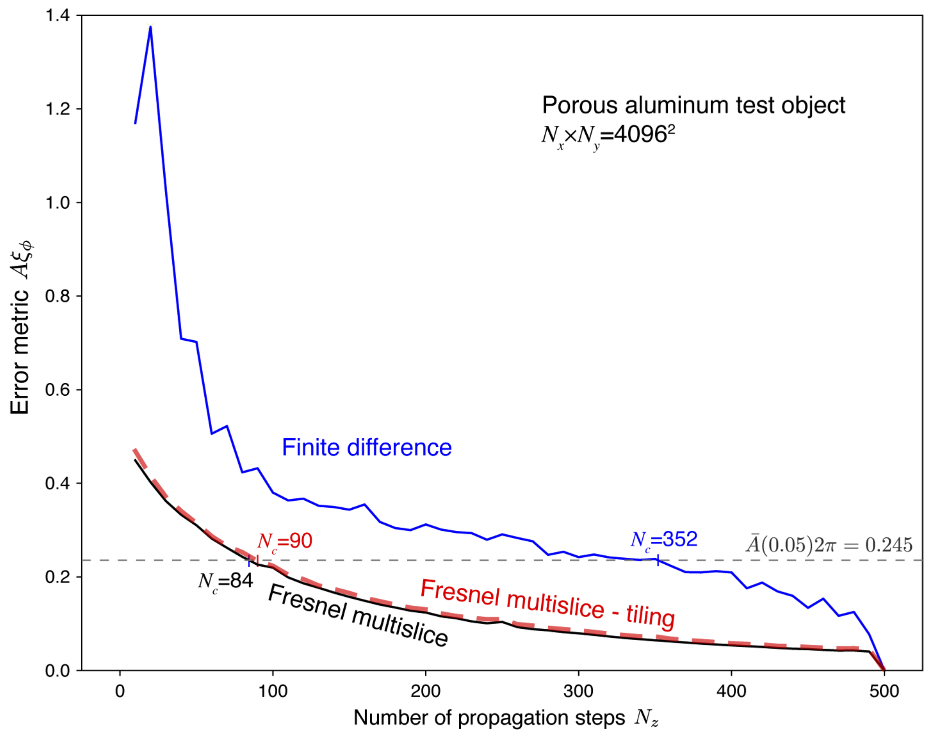

Fig. 7. Convergence test for the three algorithms of Sec. 2 using the porous aluminum test object. For this test, the 4096^2 transverse subarray of the object was selected, and the thickness t = 147.5 µm was bilinearly re-sampled onto a variable number of slices N_z. Using the convergence criterion of Eq. (26) giving a tolerance for this sample of $[\bar{A}\,\xi_\phi]_{\mathrm{Al}} = 0.245$ (Eq. (30)), the full-array Fresnel multislice approach reached convergence with $n_C = 84$ slices with $\Delta_z = 1.64$ µm while the tiling-based short-distance Fresnel multislice approach required $n_C = 90$ slices with $\Delta_z = 1.64$ µm. The finite difference method required $n_C = 352$ slices with $\Delta_z = 0.42$ µm for this irregular object.

We also used the porous aluminum object to test the performance of the finite difference algorithm as a function of the transverse array size $N_x \times N_y$, with results shown in Fig. 8. As the array size was decreased from 4096^2 to 512^2 transverse pixels, the minimum number of slices decreased from $n_C = 352$ to $n_C = 96$, with the $n_C = 96$ result corresponding to a slice thickness of $\Delta_z = 1.54$ µm which is more similar to what is required for the Fresnel multislice approaches.

We then tested the three algorithms on the $t = 30.81$ µm thick Fresnel zone plate test object, where one would expect to see the finest zone width $dr_N = 20$ nm give rise to depth of focus effects (Eq. (1)) at DOF = 38.7 µm and waveguide effects (Eq. (11)) at $z_{\text{K-C}} = 2.3$ µm. We tested the convergence of all three algorithms as a function of transverse array size as shown in Fig. 9, where the zone plate diameter d (and thus focal length f) was adjusted to fill 80% of the transverse array size in each case. With this highly regular object, the finite difference algorithm required fewer slices to converge ($n_C = 8$, giving $\Delta_z = 3.85$ µm) while the two Fresnel multislice algorithms required more slices ($n_C = 21$ and $\Delta_z = 1.47$ µm for full-array Fresnel multislice, and $n_C = 23$ and $\Delta_z = 1.34$ µm for tiling-based short-distance Fresnel multislice). Since the finite difference method has been shown to converge quickly in calculations of X-ray waveguides [23,24], it is not surprising that it performs better with the regular structure of thick zone plates. All three cases required slice thicknesses that are within a factor of 2 of the Klein-Cook thickness

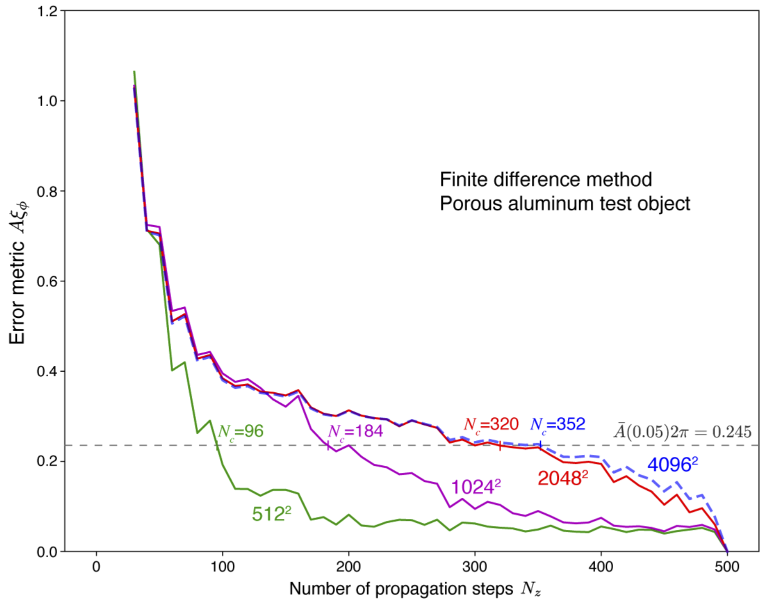

Fig. 8. Convergence of the finite difference approach as a function of transverse array size for the porous aluminum object. As in Fig. 7, the indicated size of transverse array (ranging from 512^2 to 4096^2 pixels) was extracted from the object, and the total object thickness t = 147.5 µm was bilinearly sampled along the propagation direction $\hat{z}$ to vary N_z. For each array size, the minimum number of slices n_C (Eq. (26)) was calculated using the convergence threshold $[\bar{A}\,\xi_\phi]_{\text{Al}} = 0.245$ of Eq. (30). As can be seen, the finite difference method converges more quickly with smaller transverse arrays, reaching n_C = 96 (with slice thickness Δ_z = 1.54 µm) at 512^2 transverse grid size with this irregular object.

$z_{\text{K-C}}$ = 2.3 µm estimated using Eq. (11), and all three cases had convergence properties that did not depend on the transverse grid size.

6.2. Scaling results

Having established the convergence of each approach, we then considered performance scalings on parallel computing systems as required for future image reconstruction problems with increasingly large array sizes. Because of the need to adjust transverse array sizes for these tests, we used the Fresnel zone plate test object described in Sec. 5.1, where the zone plate diameter was 80% of the transverse array size. Based on the results shown in Fig. 9, we used n_C = 8 slices for the finite difference approach, n_C = 21 for the full-array Fresnel multislice approach, and n_C = 23 slices for the tiling-based short-distance Fresnel multislice approach. We carried out these tests on the compute system “theta” with properties described in Table 1. The tiling-based short-distance Fresnel multislice approach requires the number of ranks per node to be a perfect square, so we were able to use 64 MPI ranks per compute node, matching the number of compute cores available on each node of “theta.”

The finite difference approach includes a “setup” phase that constructs a new preconditioner object each time a new calculation size is encountered (for the algebraic multigrid preconditioner, this involves setting up the coarse grids and interpolation matrices as described in Sec 3). This preconditioner can then be reused for subsequent applications on the same array size. In our tests below, a preconditioner was constructed for the first propagation step and used, as is, for all the following steps. In an optimization context where the time-stepping (TS) object (which

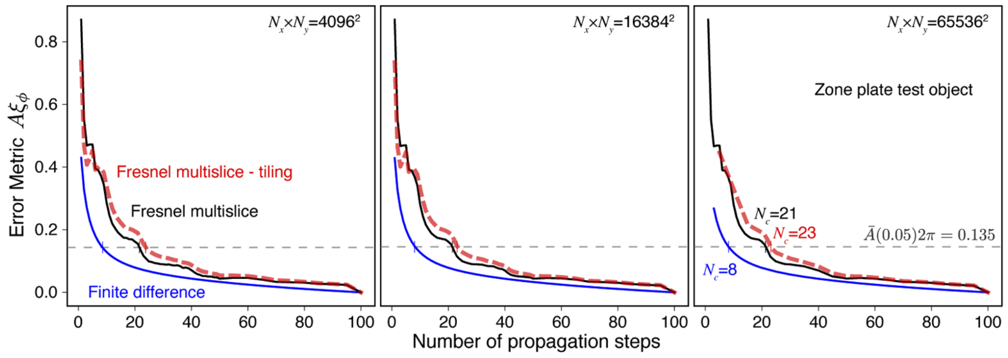

Fig. 9. Convergence test of the three algorithms for a Fresnel zone plate as a highly regular test object. In all cases, the zone plate thickness was t = 30.81 µm and the minimum zone width was dr_N = 20 nm, but the diameter d (and thus focal length f) of the zone plate was adjusted to match 80% of the transverse array size for 4096^2, 16384^2, and 65536^2 transverse pixels, respectively. Using the convergence threshold $[\bar{A}\,\xi_\phi]_{\text{zp}} = 0.135$ (Eq. (28)) for this object to find the minimum number of slices n_C (Eq. (26)), all three algorithms had minimum slice numbers n_C that were independent of transverse array size and that were within a factor of 2 of the thickness $z_{\text{K-C}}$ = 2.3 µm (Eq. (11)) at which waveguide effects would be expected for this specimen. The finite difference method required fewer slices with n_C = 8 and Δ_z = 3.85 µm, while the two Fresnel multislice methods required slightly more slices (n_C = 21 and Δ_z = 1.47 µm for full-array Fresnel multislice, and n_C = 23 and Δ_z = 1.34 µm for tiling-based short-distance Fresnel multislice).

handles the time integration, as described in Sec 3) is re-used, this setup cost would amount to a negligible amount of total runtime.

Our first test was to look at the total computation time for a fixed transverse array size of 32768^2 pixels while increasing the number of computational nodes used. In the case of 100% parallel computing efficiency, this so-called "strong scaling" test would be carried out in the time it takes one node to do the entire problem divided by the number of nodes used. As seen in Fig. 10, the full-array Fresnel multislice approach shows only a modest decrease in compute time when more nodes are used, until at 64 nodes and above the calculation time begins to increase, rather than decrease. The "strong scaling" details shown in Fig. 11 show why: the time for doing a fast Fourier transform (FFT) increases with these many nodes as internode communication speed limits outweigh the gains offered by using more cores when carrying out FFTs on large arrays. Because the tiling-based short-distance Fresnel multislice approach gives each node a subarray to work on with communication only at the calculation start (when each node is given its data) and end (when the full exit wave is assembled), it has improved scaling properties especially in terms of FFT time as shown in the bottom row of Fig. 11. The finite difference method implemented using PETSc takes a longer time even though a smaller number of slices n_C are required (Fig. 9), but using more nodes for the same problem gives a more rapid relative decrease in calculation time (that is, better "strong scaling") until there is no further gain when using more than about 100 nodes for this problem size.

Because our ultimate goal is to use parallel computing to calculate X-ray propagation through increasingly large objects, our next test was to consider array sizes that scaled up with increasing numbers of nodes used. Based on the observation in Figs. 10 and 11 that FFT performance decreases when each node is required to work with array sizes smaller than 4096^2, we used an array size of $(4096\sqrt{N_{\text{nodes}}})^2$ in this "weak scaling" test. We increased the number of nodes N_{nodes} by factors of 4 $(1, 4, 16, \ldots)$ for the full-array Fresnel multislice method in order to have radix-2 array sizes, and also for the finite difference method; since the tiling-based short-distance

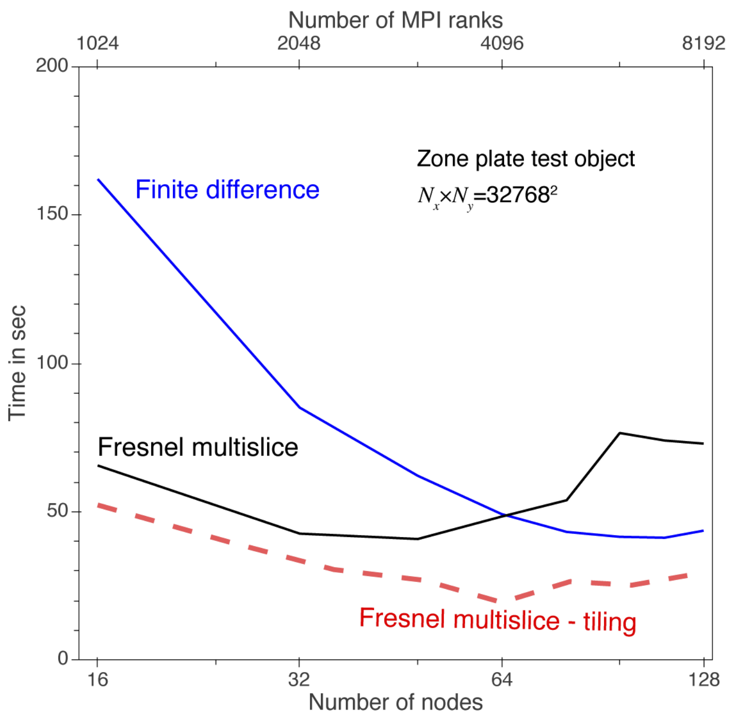

Fig. 10. Time for calculating the exit wave from the zone plate test object as a function of the number of nodes used. This “strong scaling” test was done with a constant transverse grid size of $N_x \times N_y = 32768^2$ pixels on the computational cluster “theta” (see Table 1), and using the number of slices n_C each algorithm required for convergence to the error tolerance $[\bar{A}\,\xi_\phi]_{\text{zp}}$ of Eq. (28) (the resulting values of n_C were consistently within 1 or 2 slices of the values shown in Fig. 9). While the finite difference method takes the longest amount of time with a small number of nodes, it benefits the most from increased parallelization so that the calculation time drops significantly by the time 128 nodes are employed. The full-array Fresnel multislice method shows only a modest time decrease as more nodes are employed, until at 64 nodes the calculation time begins to increase due to the requirement for considerable data communication between nodes. Because the tiling-based short-distance Fresnel multislice approach allows each node to proceed through to the exit wave plane before inter-node communication is again required, it takes the least time but after 64 nodes one again sees a slight increase in calculation time if additional nodes are used. Note that 64 nodes corresponds to a transverse array size of 4096^2 pixels per node. Further details on this “strong scaling” test are provided in Fig. 11.

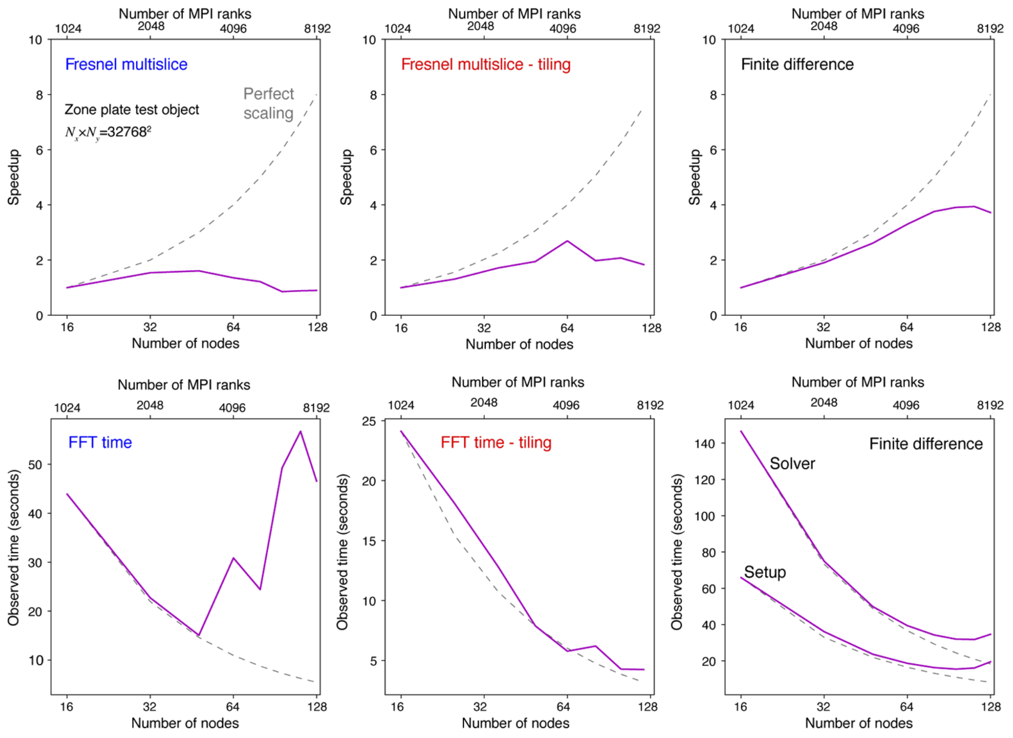

Fig. 11. Further details on the "strong scaling" test results shown in Fig. 10. These tests were of the zone plate test object on a $N_x \times N_y = 32768^2$ pixel transverse grid. For each of the three calculation methods, we show at top the speedup versus the number of nodes used (with a a linear "perfect scaling" trend showing up as a curved line on this log-linear plot). This shows that the finite difference method has the best scaling to calculation speedup with increased number of nodes. At bottom we show the time required for key operations in the various methods: the time required for a fast Fourier transform (FFT) in the full-array Fresnel multislice and tiling-based short-distance Fresnel multislice methods, and the time for problem setup and then problem solution for the finite difference method. With the full-array FFT approach, the advantage of having more processors is outweighed by data communication overhead when 64 or more nodes are used.

Fresnel multislice approach requires splitting a large transverse array into an integer number of tiles in each direction, for that method, we increased N_{nodes} by a series of perfect squares $(1^2, 2^2, 3^2, \ldots)$. As shown in Fig. 12, the total compute time for the full-array Fresnel multislice approach increases dramatically when the transverse array size goes beyond $N_x \times N_y = 32768^2$, with Fig. 13 making it clear that this is due to the time of calculating the FFT of a single large array with lots of internode data communication. The tiling-based short-distance Fresnel multislice approach is the fastest in this test, with the FFT time essentially unaffected by overall transverse array size as shown in Fig. 13; this is as expected given that in this approach each node works only on its local "tile" of the larger array. Once again, the finite difference approach is slower than the tiling-based short-distance Fresnel multislice approach, but it also shows more favorable scaling efficiency with larger problem size as shown in Fig. 13.

While the largest transverse array size used in the scaling tests described above was $N_x \times N_y = 65536^2$, we also carried out one calculation using the tiling-based short-distance Fresnel multislice approach on a 131072^2 pixel array. This was done on "theta" using 256 nodes and 64 ranks per node. This calculation took 12 seconds for data reading and tile division, 99 seconds for

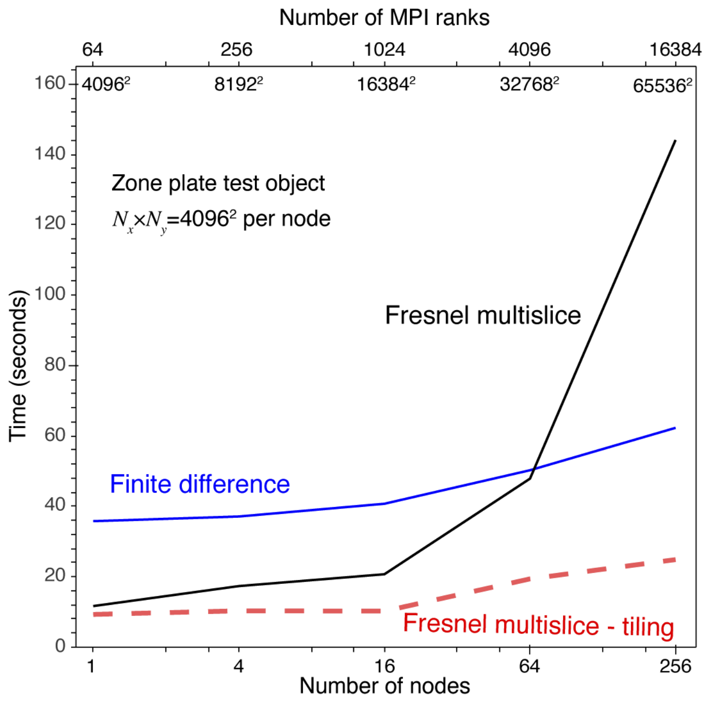

Fig. 12. Time for calculating the exit wave for the zone plate test object as a function of increasing the transverse array size along with the number of nodes, with each node given a transverse grid size of $N_x \times N_y = 4096^2$ (leading to a net array size of 65536^2 for 256 nodes, as indicated just below the top of the plot). For each algorithm, the number of slices n_C was as required for convergence to the error tolerance $[\bar{A}\,\xi_\phi]_{\text{zp}}$ of Eq. (28), giving values of n_C that were in all cases within 1 or 2 slices of the values shown in Fig. 9. This "weak scaling" test shows that both the finite difference and tiling-based short-distance Fresnel multislice approaches scale well as the problem size increases with the number of nodes used, consistent with the "strong scaling" test results of Fig. 10. With the full-array Fresnel multislice approach, the time required for data communication between nodes for full-array FFTs means that even with many nodes available large problems require considerably more time to compute.

writing the array out to an HDF5 file, and 25 seconds for a series of $n_C = 22$ slices, or about 1.1 seconds per slice. Recognizing that the scale of compute power available from "theta" makes this a completely unfair comparison, this is much faster than the 3.6 minute calculation time reported for the equivalent of 1 slice on a standard workstation [27]. We also carried out a tiling-based short-distance Fresnel multislice calculation on a 4096^2 zone plate test object broken into four tiles, with these tiles divided between two graphical processing units (GPUs) on the compute server "xrmlite" (Table 1). In this case, it took 3.68 seconds for CPU–GPU communication, and 0.14 seconds for the FFT calculation, confirming the benefits of GPU based parallelism for FFTs (as expected [94]). Therefore, distributed GPU parallelism is a viable approach for the tiling-based short-distance Fresnel multislice algorithm when the tiles fit in GPU memory.

While the main components of PDE solvers involve sparse linear algebra that are also memory-bound due to their low arithmetic intensity [28], optimal design and implementation of solvers for good performance on GPUs [95] remains a challenge that is being currently addressed [96,97]. In particular PETSc's algebraic multigrd (GAMG) can execute the solve phase entirely on the GPUs and there is ongoing development on performing the setup phase on the GPUs as well

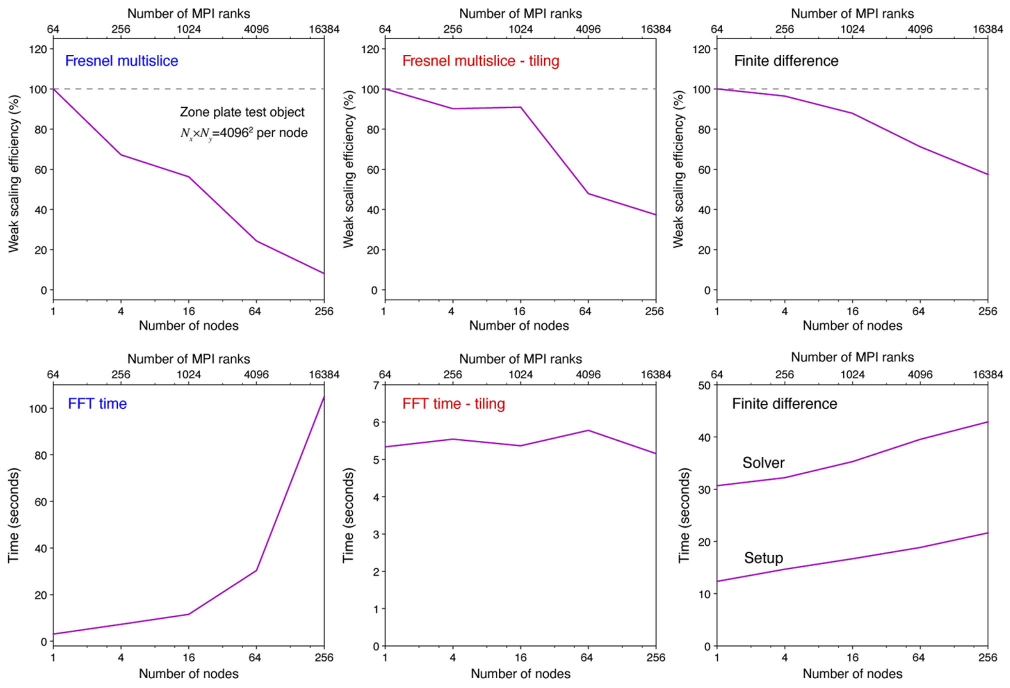

Fig. 13. Further details on the "weak scaling" test results shown in Fig. 12. These tests were of the zone plate test object with a constant array size of $N_x \times N_y = 4096^2$ per node, leading to a net array size of 65536^2 for 256 nodes. The top row shows the scaling efficiency for each of the three algorithms, which is the completion time compared to the 1 node result divided by the number of nodes used. The bottom row shows the time for key operations in each method: a fast Fourier transform or FFT for the Fresnel multislice approaches, and problem setup and solution for the finite difference method. As can be seen, the full-array Fresnel multislice approach has especially poor "weak scaling" performance due to the need for internode communcation at each slice position, while the tiling-based short-distance Fresnel multislice approach offers better parallel performance. The finite difference approach takes a longer time, but with less of a decrease in efficiency for larger transverse array size.

[97]. These performance benefits are passed onto users and do not require changes to application codes. Thus, we note that our finite difference solver will be able to use GPU parallelism when available with minimal code changes.

7. Conclusion

To reach the full potential of X-ray microscopy of combining high penetration in thick samples with nanoscale spatial resolution, it will become necessary to compute wave propagation through inhomogeneous media with a very large array size. This can be used both for calculating the performance of X-ray focusing optics, and also for creating a forward model that can be used for image reconstruction using numerical optimization approaches.

Two approaches used thus far in X-ray microscopy are the finite difference method [23,24], and the Fresnel multislice method [18–22]. We have implemented the finite difference method using the PETSc package [28,29], which offers efficient scaling on distributed memory compute systems. For the Fresnel multislice approach, we have used the fast Fourier transform interfaces built into PETSc for the full-array Fresnel multislice algorithm. We have also used `mpi4py` to implemented a parallelized version of the tiling-based short-distance Fresnel multislice approach that has been developed first for digital holography [27]. All three algorithms can compute moderately large transverse array problems in a reasonable time. In contrast to multislice methods, the finite difference approach requires fewer slices n_C for convergence when applied to the highly-regular, strong-waveguide-effect zone plate test object (Fig. 9), with more slices, n_C required for the irregular porous aluminum test object. Additionally, for an irregular object, the finite difference approach requires fewer slices to convergence at larger pixel sizes (Fig. 8), which is opposite to the behavior of full-array Fresnel multislice approach. This behavior of the finite difference approach could be used for robust reconstructions at downsampled resolutions.

With the full-array Fresnel multislice approach, one begins to suffer from internode communication bottlenecks at array sizes of $N_x \times N_y = 32768^2$ on the compute cluster “theta” described in Table 1. The finite difference method and the tiling-based short-distance Fresnel multislice approach offer much better scaling to large array sizes, as shown in Figs. 12 and 13, with the latter approach requiring roughly a third less compute time. We have also tested the tiling-based short-distance Fresnel multislice approach on array sizes as large as 131072^2 with good results. Together, these approaches show that parallelized software packages on high performance compute clusters allow one to calculate X-ray wave propagation in inhomogeneous media with reasonable execution times for very large array sizes. The combination of advances in bright X-ray sources [1] and in high performance computing therefore makes it possible to contemplate nanoscale X-ray imaging of macro-sized objects. The methods tested here show that the forward problem of simulating the exit wave for a guess of the object is computationally tractable, allowing for its use in an optimization approach for object reconstruction.

Funding

Basic Energy Sciences (DE-AC02-06CH11357); Advanced Scientific Computing Research (DE-AC02-06CH11357); National Institute of Mental Health (R01MH115265).

Acknowledgments

We thank Tim Salditt and Jacob Soltau for helpful discussions regarding the finite difference method. We thank Hong Zhang for programming advice and bug fixes for linear TS solver, and PETSc developers on the petsc-users mailing list for general guidance. We also thank Ed Bueler for providing a preprint of the forthcoming book “PETSc for Partial Differential Equations.” We also thank Argonne Leadership Computing Facility staff for help with optimizing the performance of the algorithms, including but not limited to Sudheer Chunduri for MPI tuning, and Kevin Harms and Taylor Childers for I/O tuning.

Disclosures

The authors declare no conflicts of interest.

References

1. M. Eriksson, J. F. van der Veen, and C. Quitmann, “Diffraction-limited storage rings – a window to the science of tomorrow,” J. Synchrotron Radiat. **21**(5), 837–842 (2014).
2. M. Born and E. Wolf, *Principles of Optics* (Cambridge University Press, Cambridge, 1999), seventh ed.
3. C. Jacobsen, *X-ray Microscopy* (Cambridge University Press, Cambridge, 2020).
4. W. Van den Broek and C. T. Koch, “Method for retrieval of the three-dimensional object potential by inversion of dynamical electron scattering,” Phys. Rev. Lett. **109**(24), 245502 (2012).
5. D. Ren, C. Ophus, M. Chen, and L. Waller, “A multiple scattering algorithm for three dimensional phase contrast atomic electron tomography,” Ultramicroscopy **208**, 112860 (2020).
6. A. M. Maiden, M. J. Humphry, and J. M. Rodenburg, “Ptychographic transmission microscopy in three dimensions using a multi-slice approach,” J. Opt. Soc. Am. A **29**(8), 1606–1614 (2012).
7. U. S. Kamilov, I. N. Papadopoulos, M. H. Shoreh, A. Goy, C. Vonesch, M. Unser, and D. Psaltis, “Learning approach to optical tomography,” Optica **2**(6), 517–522 (2015).

8. U. S. Kamilov, I. N. Papadopoulos, M. H. Shoreh, A. Goy, C. Vonesch, M. Unser, and D. Psaltis, "Optical tomographic image reconstruction based on beam propagation and sparse regularization," IEEE Trans. Comput. Imaging **2**(1), 59–70 (2016).
9. A. Suzuki, S. Furutaku, K. Shimomura, K. Yamauchi, Y. Kohmura, T. Ishikawa, and Y. Takahashi, "High-resolution multislice x-ray ptychography of extended thick objects," Phys. Rev. Lett. **112**(5), 053903 (2014).
10. E. H. R. Tsai, I. Usov, A. Diaz, A. Menzel, and M. Guizar-Sicairos, "X-ray ptychography with extended depth of field," Opt. Express **24**(25), 29089–29108 (2016).
11. M. A. Gilles, Y. S. G. Nashed, M. Du, C. Jacobsen, and S. M. Wild, "3D x-ray imaging of continuous objects beyond the depth of focus limit," Optica **5**(9), 1078–1086 (2018).
12. M. Du, Y. S. G. Nashed, S. Kandel, D. Gürsoy, and C. Jacobsen, "Three dimensions, two microscopes, one code: Automatic differentiation for x-ray nanotomography beyond the depth of focus limit," Sci. Adv. **6**(13), eaay3700 (2020).
13. J. M. Cowley and A. F. Moodie, "The scattering of electrons by atoms and crystals. I. A new theoretical approach," Acta Crystallogr. **10**(10), 609–619 (1957).
14. K. Ishizuka and N. Uyeda, "A new theoretical and practical approach to the multislice method," Acta Crystallographica A **33**(5), 740–749 (1977).
15. J. Van Roey, J. van der Donk, and P. E. Lagasse, "Beam-propagation method: analysis and assessment," J. Opt. Soc. Am. **71**(7), 803–810 (1981).
16. J. Deng, Y. P. Hong, S. Chen, Y. S. G. Nashed, T. Peterka, A. J. F. Levi, J. Damoulakis, S. Saha, T. Eiles, and C. Jacobsen, "Nanoscale x-ray imaging of circuit features without wafer etching," Phys. Rev. B **95**(10), 104111 (2017).
17. J. Deng, C. A. Preissner, J. A. Klug, S. Mashrafi, C. Roehrig, Y. Jiang, Y. Yao, M. J. Wojcik, M. D. Wyman, D. J. Vine, K. Yue, S. Chen, T. Mooney, M. Wang, Z. Feng, D. Jin, Z. Cai, B. P. Lai, and S. Vogt, "The Velociprobe: An ultrafast hard x-ray nanoprobe for high-resolution ptychographic imaging," Rev. Sci. Instrum. **90**(8), 083701 (2019).
18. A. R. Hare and G. R. Morrison, "Near-field soft X-ray diffraction modelled by the multislice method," J. Mod. Opt. **41**(1), 31–48 (1994).
19. Y. V. Kopylov, A. V. Popov, and A. V. Vinogradov, "Diffraction phenomena inside thick Fresnel zone plates," Radio Sci. **31**(6), 1815–1822 (1996).
20. Y. Wang and Jacobsen, "A numerical study of resolution and contrast in soft x-ray contact microscopy," J. Microsc. **191**(2), 159–169 (1998).
21. H. Yan, "X-ray nanofocusing by kinoform lenses: A comparative study using different modeling approaches," Phys. Rev. B **81**(7), 075402 (2010).
22. K. Li, M. Wojcik, and C. Jacobsen, "Multislice does it all: calculating the performance of nanofocusing x-ray optics," Opt. Express **25**(3), 1831–1846 (2017).
23. C. Fuhse and T. Salditt, "Finite-difference field calculations for two-dimensionally confined x-ray waveguides," Appl. Opt. **45**(19), 4603–4608 (2006).
24. L. Melchior and T. Salditt, "Finite difference methods for stationary and time-dependent x-ray propagation," Opt. Express **25**(25), 32090–32109 (2017).
25. R. Scarmozzino and R. M. Osgood Jr., "Comparison of finite-difference and Fourier-transform solutions of the parabolic wave equation with emphasis on integrated-optics applications," J. Opt. Soc. Am. A **8**(5), 724–731 (1991).
26. Y. Chung and N. Dagli, "An assessment of finite difference beam propagation method," IEEE J. Quantum Electron. **26**(8), 1335–1339 (1990).
27. D. Blinder and T. Shimobaba, "Efficient algorithms for the accurate propagation of extreme-resolution holograms," Opt. Express **27**(21), 29905–29915 (2019).
28. S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcín, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc users manual," Tech. Rep. ANL-95/11 - Revision 3.11, Argonne National Laboratory (2019).
29. S. Abhyankar, J. Brown, E. M. Constantinescu, D. Ghosh, B. F. Smith, and H. Zhang, "PETSc/TS: a modern scalable ODE/DAE solver library," arXiv:1806.01437 (2018).
30. T. Sugie, T. Akamatsu, T. Nishitsuji, R. Hirayama, N. Masuda, H. Nakayama, Y. Ichihashi, A. Shiraki, M. Oikawa, N. Takada, Y. Endo, T. Kakue, T. Shimobaba, and T. Ito, "High-performance parallel computing for next-generation holographic imaging," Nat. Electron. **1**(4), 254–259 (2018).
31. C. Zimmer, S. Atchley, R. Pankajakshan, B. E. Smith, I. Karlin, M. L. Leininger, A. Bertsch, B. S. Ryujin, J. Burmark, A. Walker-Loud, M. A. Clark, and O. Pearce, "An evaluation of the CORAL interconnects," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, (Association for Computing Machinery, New York, NY, USA, 2019), SC '19.
32. D. E. Bernholdt, S. Boehm, G. Bosilca, M. Gorentla Venkata, R. E. Grant, T. Naughton, H. P. Pritchard, M. Schulz, and G. R. Vallee, "A survey of MPI usage in the US exascale computing project," Concurrency Computat. Pract. Exper. **32**(3), e4851 (2020).
33. D. Attwood and A. Sakdinawat, *X-rays and Extreme Ultraviolet Radiation* (Cambridge University Press, Cambridge, 2017), 2nd ed.
34. B. L. Henke, E. M. Gullikson, and J. C. Davis, "X-ray interactions: Photoabsorption, scattering, transmission, and reflection at E=50–30,000 eV, Z=1–92," At. Data Nucl. Data Tables **54**(2), 181–342 (1993).

35. K. Li and C. Jacobsen, "More are better, but the details matter: combinations of multiple Fresnel zone plates for improved resolution and efficiency in x-ray microscopy," J. Synchrotron Radiat. **25**(4), 1048–1059 (2018).
36. M. Chen, D. Ren, H.-Y. Liu, S. Chowdhury, and L. Waller, "Multi-layer Born multiple-scattering model for 3D phase microscopy," Optica **7**(5), 394–403 (2020).
37. J. W. Goodman, *Introduction to Fourier optics* (W.H. Freeman, 2017).
38. K. Li and C. Jacobsen, "Rapid calculation of paraxial wave propagation for cylindrically symmetric optics," J. Opt. Soc. Am. A **32**(11), 2074–2081 (2015).
39. W. R. Klein and B. D. Cook, "Unified approach to ultrasonic light diffraction," IEEE Trans. Sonics Ultrason. **14**(3), 123–134 (1967).
40. F. Franchetti, D. G. Spampinato, A. Kulkarni, D. Thom Popovici, T. M. Low, M. Franusich, A. Canning, P. McCorquodale, B. V. Straalen, and P. Colella, "FFTX and SpectralPack: A first look," in *2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW)*, (2018), pp. 18–27.
41. D. Takahashi, "Implementation of parallel FFTs on cluster of Intel Xeon Phi processors," (2018). SIAM-PP-2018.
42. D. Takahashi, *Parallel FFT Algorithms for Distributed-Memory Parallel Computers* (Springer Singapore, 2019), pp. 77–112.
43. H. Ibeid, L. Olson, and W. Gropp, "FFT, FMM, and multigrid on the road to exascale: Performance challenges and opportunities," J. Parallel Distr. Com. **136**, 63–74 (2020).
44. H. Yoshikawa, "Fast computation of Fresnel holograms employing difference," Opt. Rev. **8**(5), 331–335 (2001).
45. T. Shimobaba, N. Masuda, and T. Ito, "Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane," Opt. Lett. **34**(20), 3133–3135 (2009).
46. F. A. Jenkins and H. E. White, *Fundamentals of Optics* (McGraw-Hill, 1976), 4th ed.
47. V. A. Fock, *Electromagnetic diffraction and propagation problems* (Pergamon, Oxford, UK, 1995).
48. S. N. Vlasov and V. I. Talanov, "The parabolic equation in the theory of wave propagation," Radiophys. Quantum Electron. **38**(1-2), 1–12 (1996).
49. Y. V. Kopylov, A. V. Popov, and A. V. Vinogradov, "Application of the parabolic wave equation to x-ray diffraction optics," Opt. Commun. **118**(5-6), 619–636 (1995).
50. O. G. Ernst and M. J. Gander, "Why it is difficult to solve Helmholtz problems with classical iterative methods," in *Numerical Analysis of Multiscale Problems*, I. G. Graham, T. Y. Hou, O. Lakkis, and R. Scheichl, eds. (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012), pp. 325–363.
51. J. W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods* (Springer New York, 1995).
52. J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type," Math. Proc. Cambridge Philos. Soc. **43**(1), 50–67 (1947).
53. D. W. Peaceman and H. H. Rachford Jr., "The numerical solution of parabolic and elliptic differential equations," J. Soc. Ind. Appl. Math. **3**(1), 28–41 (1955).
54. Y. Saad and H. A. van der Vorst, "Iterative solution of linear systems in the 20th century," J. Comput. Appl. Math. **123**(1-2), 1–33 (2000). Numerical Analysis 2000. Vol. III: Linear Algebra.
55. P. E. Farrell, D. A. Ham, S. W. Funke, and M. E. Rognes, "Automated derivation of the adjoint of high-level transient finite element programs," SIAM J. Sci. Comput. **35**(4), C369–C393 (2013).
56. H. Zhang, E. M. Constantinescu, and B. F. Smith, "PETSc TSAdjoint: a discrete adjoint ODE solver for first-order and second-order sensitivity analysis," arxiv:1912.07696 (2019).
57. A. Griewank and A. Walther, *Evaluating Derivatives* (Society for Industrial and Applied Mathematics, 2008), 2nd ed.
58. U. Naumann, *The Art of Differentiating Computer Programs* (Society for Industrial and Applied Mathematics, 2011).
59. S. Ali, https://github.com/s-sajid-ali/xwp_petsc.
60. M. P. I. Forum, "A message-passing interface standard, version 3.1," https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.
61. M. Du, https://github.com/mdw771/beyond_dof.
62. L. Dalcín, R. Paz, and M. Storti, "MPI for Python," J. Parallel Distr. Com. **65**(9), 1108–1115 (2005).
63. L. Dalcín, R. Paz, M. Storti, and J. D'Elía, "MPI for Python: Performance improvements and MPI-2 extensions," J. Parallel Distr. Com. **68**(5), 655–662 (2008).
64. T. E. Oliphant, "Python for scientific computing," Comput. Sci. Eng. **9**(3), 10–20 (2007).
65. The HDF Group, "Hierarchical Data Format, version 5," http://www.hdfgroup.org/HDF5/ (1997–2019).
66. T. Gamblin, M. LeGendre, M. R. Collette, G. L. Lee, A. Moody, B. R. de Supinski, and S. Futral, "The Spack package manager: bringing order to HPC software chaos," in *SC15: International Conference for High-Performance Computing, Networking, Storage and Analysis*, (IEEE Computer Society, Los Alamitos, CA, USA, 2015), pp. 1–12.
67. A. L. C. Facility, "Affinity on theta," https://www.alcf.anl.gov/support-center/theta/affinity-theta.
68. M. A. Salim, T. D. Uran, J. T. Childers, P. Balaprakash, V. Vishwanath, and M. E. Papka, "Balsam: Automated scheduling and execution of dynamic, data-intensive HPC workflows," arxiv:1909.08704 (2019).
69. A. L. C. Facility, "Theta machine overview," https://www.alcf.anl.gov/support-center/theta/theta-machine-overview.
70. K. Harms, T. Leggett, B. Allen, S. Coghlan, M. Fahey, C. Holohan, G. McPheeters, and P. Rich, "Theta: Rapid installation and acceptance of an XC40 KNL system," Concurrency Computat. Pract. Exper. **30**(1), e4336 (2018).
71. S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcín, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc Web page," https://www.mcs.anl.gov/petsc (2019).

72. S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds. (Birkhäuser Press, 1997), pp. 163–202.
73. M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," Proc. IEEE **93**(2), 216–231 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation".
74. A. Collette, "HDF5 for Python," (2008).
75. "HDF5 for Python, https://www.h5py.org/.
76. O. Pavlyk, D. Nagorny, A. Guzman-Ballen, A. Malakhov, H. Liu, E. Totoni, T. A. Anderson, and S. Maidanov, "Accelerating scientific Python with Intel optimizations," in *Proceedings of the 16th Python in Science Conference*, K. Huff, D. Lippa, D. Niederhut, and M. Pacer, eds. (2017), pp. 106–112.
77. Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," SIAM J. Sci. and Stat. Comput. **7**(3), 856–869 (1986).
78. Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," SIAM J. Sci. Comput. **14**(2), 461–469 (1993).
79. W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial: Second Edition*, Other Titles in Applied Mathematics (Society for Industrial and Applied Mathematics, 2000).
80. K. Stüben, "A review of algebraic multigrid," J. Comput. Appl. Math. **128**(1-2), 281–309 (2001). Numerical Analysis 2000. Vol. VII: Partial Differential Equations.
81. U. M. Yang, "Parallel algebraic multigrid methods — high performance preconditioners," in *Numerical Solution of Partial Differential Equations on Parallel Computers*, A. M. Bruaset and A. Tveito, eds. (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 209–236.
82. O. Widlund and M. Dryja, *An additive variant of the Schwarz alternating method for the case of many subregions*, Technical Report 339, Ultracomputer Note 131 (Department of Computer Science, Courant Institute, 1987).
83. B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations* (Cambridge University Press, Cambridge, UK, 1996).
84. X.-C. Cai, "Additive Schwarz algorithms for parabolic convection-diffusion equations," Numer. Math. **60**(1), 41–61 (1991).
85. F. Hülsemann, M. Kowarschik, M. Mohr, and U. Rüde, "Parallel geometric multigrid," in *Numerical Solution of Partial Differential Equations on Parallel Computers*, A. M. Bruaset and A. Tveito, eds. (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 165–208.
86. M. F. Adams, "Algebraic multigrid methods for direct frequency response analyses in solid mechanics," Comput. Mech. **39**(4), 497–507 (2007).
87. M. J. Gander, "Schwarz methods over the course of time," Electronic Transactions on Numerical Analysis **31**, 228–255 (2008).
88. A. Maréchal, "Étude des effects combinés de la diffraction et des aberrations géométriques sur l'image d'un point lumineux," Revue D'Optique Théorique et Instrumentale **26**, 257–277 (1947).
89. J. Ruze, "The effect of aperture errors on the antenna radiation pattern," Nuovo Cimento **9**(S3), 364–380 (1952).
90. R. F. C. Vescovi, M. Du, V. De Andrade, W. Scullin, D. Gürsoy, and C. Jacobsen, "*Tomosaic*: efficient acquisition and reconstruction of teravoxel tomography data using limited-size synchrotron x-ray beams," J. Synchrotron Rad. **25**(5), 1478–1489 (2018).
91. F. De Carlo, D. Gürsoy, D. J. Ching, K. J. Batenburg, W. Ludwig, L. Mancini, F. Marone, R. Mokso, D. M. Pelt, J. Sijbers, and M. Rivers, "TomoBank: a tomographic data repository for computational x-ray science," Meas. Sci. Technol. **29**(3), 034004 (2018).
92. E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. . Burt, and J. M. Ogden, "Pyramid methods in image processing," RCA Engineer **29**, 33–41 (1984).
93. T. Schoonjans, A. Brunetti, B. Golosio, M. Sanchez del Rio, V. A. Solé, C. Ferrero, and L. Vincze, "The xraylib library for x-ray–matter interactions. Recent developments," Spectrochim. Acta, Part B **66**(11-12), 776–784 (2011).
94. P. Steinbach and M. Werner, "gearshifft – the FFT benchmark suite for heterogeneous platforms," in *High Performance Computing* J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, eds. (Springer International Publishing, Cham, 2017), pp. 199–216.
95. H. Anzt, E. Boman, R. Falgout, P. Ghysels, M. Heroux, X. Li, L. Curfman McInnes, R. Tran Mills, S. Rajamanickam, K. Rupp, B. Smith, I. Yamazaki, and U. Meier Yang, "Preparing sparse solvers for exascale computing," Phil. Trans. R. Soc. A **378**(2166), 20190053 (2020).
96. R. T. Mills, "Progress with PETSc on manycore and GPU-based systems on the path to exascale," https://www.mcs.anl.gov/petsc/meetings/2019/slides/mills-petsc-2019.pdf.
97. B. Smith, R. T. Mills, T. Munson, S. Wild, M. Adams, S. Balay, G. Betrie, J. Brown, A. Dener, S. Hudson, M. Knepley, J. Larson, O. Marin, H. Morgan, J.-L. Navarrow, K. Rupp, P. Sanan, H. Zhang, H. Zhang, and J. Zhang, "Recent PETSc/TAO enhancements for exascale," https://ecpannualmeeting.com/assets/overview/posters/petsc-ecp2020-poster.pdf.