Adorym: a multi-platform generic X-ray image reconstruction framework based on automatic differentiation

MING DU,[1,*] **SAUGAT KANDEL,**[2] **JUNJING DENG,**[1] **XIAOJING HUANG,**[3] **ARNAUD DEMORTIERE,**[4] **TUAN TU NGUYEN,**[4] **REMI TUCOULOU,**[5] **VINCENT DE ANDRADE,**[1] **QIAOLING JIN,**[6,7] **AND CHRIS JACOBSEN**[1,6,7]

[1]*Advanced Photon Source, Argonne National Laboratory, Lemont, Illinois 60439, USA*
[2]*Applied Physics Program, Northwestern University, Evanston, Illinois 60208, USA*
[3]*National Synchrotron Light Source II, Brookhaven National Laboratory, Upton, New York 11973, USA*
[4]*Laboratoire de Réactivité et Chimie des Solides (LRCS), CNRS UMR 7314, Université de Picardie Jules Verne, Hub de l'Energie, 15 Rue Baudelocque, 80039 Amiens Cedex, France*
[5]*European Synchrotron Radiation Facility, 71 Avenue des Martyrs, 38000 Grenoble, France*
[6]*Department of Physics & Astronomy, Northwestern University, Evanston, Illinois 60208, USA*
[7]*Chemistry of Life Processes Institute, Northwestern University, Evanston, Illinois 60208, USA*
[*]*mingdu@anl.gov*

Abstract: We describe and demonstrate an optimization-based X-ray image reconstruction framework called Adorym. Our framework provides a generic forward model, allowing one code framework to be used for a wide range of imaging methods ranging from near-field holography to fly-scan ptychographic tomography. By using automatic differentiation for optimization, Adorym has the flexibility to refine experimental parameters including probe positions, multiple hologram alignment, and object tilts. It is written with strong support for parallel processing, allowing large datasets to be processed on high-performance computing systems. We demonstrate its use on several experimental datasets to show improved image quality through parameter refinement.

1. Introduction

Most image reconstruction problems can be categorized as inverse problem solving. One begins with the assumption that a measurable set of data y arises from a forward model $F(x, \theta)$, which in turn depends on an object function x and parameters θ, giving

$$y = F(x, \theta). \tag{1}$$

With experimental data y_{meas} and additive noise ϵ, imaging experiments build a relation of

$$y_{\mathrm{meas}} = F(x, \theta) + \epsilon. \tag{2}$$

When F is a non-linear function of x, or when the problem size is very large, a non-iterative solution of Eq. (2) is either intractable or non-existent at all. In other scenarios where a computationally feasible direct solution does exist, the quality of the solution can degrade when the known information is noisy or incomplete. A common example is standard filtered-backprojection tomography, which is known to produce significant artifacts when the projection images are sparse in viewing angles [1]. These issues motivate the use of iterative methods in solving Eq. (2), where x is gradually adjusted to find a minimum of a loss function L that is often formulated as

$$L = \|y_{\mathrm{meas}} - y\|^2 = \|y_{\mathrm{meas}} - F(x, \theta)\|^2, \tag{3}$$

where the $\|\cdot\|^2$ operation computes the Euclidean distance between $\boldsymbol{y}_{\mathrm{meas}}$ and $F(\boldsymbol{x})$. Loss function L formulations other than Eq. (3) are also frequently used [2], but they all include a metric measuring the mismatch between $\boldsymbol{y}_{\mathrm{meas}}$ and $F(\boldsymbol{x})$. The object function $\boldsymbol{x}$ can be a collection of coefficients for a certain basis set (such as for Zernike polynomials [3]), which might yield a relatively small number of unknowns with easy solution. However, letting $\boldsymbol{x}$ be a complex 2D pixel or 3D voxel array provides a more general description for complicated objects. Furthermore, the parameter $\boldsymbol{\theta}$ may include a complex finite-sized illumination wavefield (a probe function) with a variety of positions or angles that sample the object array, and a variety of propagation distances leading to a set of intensity measurements $\boldsymbol{y}_{\mathrm{meas}}$. In some cases, $\boldsymbol{\theta}$ is also unknown and need to be solved along with $\boldsymbol{x}$. For example, in more complicated imaging methods such as far-field ptychography [4,5] and its near-field counterpart [6], the probe function might itself be finite sized but unknown, but it can be recovered along with the object as part of recovering $\boldsymbol{x}$. In other methods such as holotomography [7], at each object rotational angle one might have a set of intensity measurements $\boldsymbol{y}_{\mathrm{meas}}$ acquired at different Fresnel propagation distances from the object $\boldsymbol{x}$, and each measurement might have experimental errors in position or rotation, collectively symbolized by $\boldsymbol{\theta}$.

The fact that a wide range of imaging problems can be framed in terms of minimizing the loss function L of Eq. (3) suggests that object reconstruction algorithms should be generic to some degree, or at least highly modular. One example of this can be found in multislice ptychography [8], which can be thought of as combining the probe-scanning of ptychography with multiple Fresnel propagation distances as in holotomography. There has also been an evolution in approaches to ptychographic tomography. It was first demonstrated using 2D ptychography to recover a complex exit wave at each rotation angle, followed by phase unwrapping, followed by standard tomographic reconstruction of these set of projection images [9]. An extension of this approach to objects that are thick enough that propagation effects arise was done by using multislice ptychography reconstruction to generate a synthesized projection, followed by phase unwrapping and standard tomographic reconstruction [10]. Since ptychographic image reconstruction [5] by itself can be treated as a nonlinear optimization problem [11], an alternative approach is to do the entire ptychographic tomography reconstruction as a nonlinear optimization problem. One can then be flexible on the ordering of scanned probe positions versus rotations [12], and one can incorporate multislice propagation effects [13,14]. In this case, one uses the full set of intensities recorded at all probe positions and object rotations as $\boldsymbol{y}_{\mathrm{meas}}$, and recovers the 3D object as well as the probe function using the loss function minimization approach of Eq. (3). One can also include a noise model in the loss function, which might be the Gaussian approximation to the Poisson distribution at higher photon count, or a Poisson model at lower photon count [2]. The point of these examples is that a nonlinear optimization approach allows one to change the forward model F and the loss function L to match the conditions of the experiment, and use a sufficiently flexible optimization approach to minimize L and thus find a solution for $\boldsymbol{x}$.

The loss function L of Eq. (3) is a scalar, but it depends on a large number of unknowns in $\boldsymbol{x}$, and furthermore the forward model $F(\boldsymbol{x})$ may by itself be nonlinear. In addition, the loss function L can also include many regularizers R with weighting λ [15] to incorporate known or desired properties of the image such as sparsity [16]. Minimizing a multivariate scalar quantity can be done using gradient-based methods, which requires one to calculate the partial derivative of L with regards to each element of $\boldsymbol{x}$ to give $\frac{\partial L}{\partial x_i}$.

For an example, we can look at the gradient-based minimization strategy for the ptychography application (the PIE algorithm [5] is one such procedure). PIE can be shown [2] to be the equivalent of gradient descent minimization of a loss function $L = \|F_1(\boldsymbol{o}, \boldsymbol{p}) - \boldsymbol{y}_{\mathrm{meas}}\|^2$. This loss function is the l_2-norm of the difference between $F_1(\boldsymbol{x})$ and $\boldsymbol{y}_{\mathrm{meas}}$. In this case F_1 is the ptychographic forward model as a function of the object $\boldsymbol{o}$ and probe $\boldsymbol{p}$, with the forward model accounting for object sampling, probe modulation, and far-field diffraction. The structure of

the problem is illustrated schematically at left in Fig. 1, where the gradient of L with regards to both o and p (denoted as $\frac{\partial L}{\partial o}$ and $\frac{\partial L}{\partial p}$) is used to optimize the guesses of both the object and the probe. Now, let us consider optimization of probe positions (so as to account for differences between intended and actual positions) along with reconstructing the object [11,17–19]. One way to incorporate probe position corrections Δr into the forward model is to use the Fourier shift theorem [20], an operation denoted by F_2. The new forward model is then constructed by placing F_2 before F_1 – that is, first apply Fourier shift to get the shifted probe p' at the current scan position, then perform object modulation and far-field propagation. This is shown schematically at right in Fig. 1. Modifying the forward model can be done trivially by simply inserting the Fourier shift code at the right location, but we would also have to re-derive the gradients in order to solve the unknowns. Since o enters the forward model after F_2, we still know the gradient $\frac{\partial L}{\partial o}$. However, our previous knowledge of $\frac{\partial L}{\partial p}$ no longer holds, since we now must differentiate through F_2 to get the new gradient with regards to p. In other words, the updated $\frac{\partial L}{\partial p}$ now requires additional derivatives $\frac{\partial y}{\partial p'}$ and $\frac{\partial p'}{\partial p}$ (Fig. 1), which were not known before. Moreover, the gradient with regards to Δr should also be derived in order to optimize Δr. Modifying the gradients requires much more effort than modifying the forward model! The former can be tedious, and prone to errors, and often needs to be redone when changes to the forward model are made. The disproportional difficulty of re-deriving the gradients introduces heavy burdens to algorithm development tasks beyond the parameter refinement example described above – for example, testing out new noise models, developing more complicated physical models (*e.g.*, one that accounts for multiple scattering), and extending existing algorithms from one imaging setup to another.
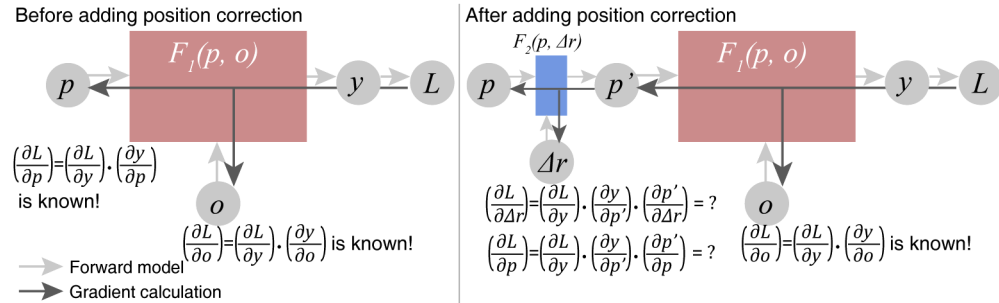
Fig. 1. When modifying an existing forward model, the gradient of the new loss function often needs to be re-derived by going into the depth of the model.

These complexities have inspired the use of an alternative approach for image reconstruction problems: the use of automatic differentiation (AD) [21] for loss function minimization. Automatic differentiation involves the estimation of partial derivatives of mathematical operations as expressed in computer code, by storing intermediate results from small variations of the input vector x on a forward pass and using these in a backward pass to accumulate the final derivative. The use of AD was suggested for iterative phase retrieval at a time before easy-to-use AD tools were available [22]. However, even in the short time since that suggestion, the explosion in machine learning (where AD is often used to train a neural network [23]) has led to the development of powerful AD toolkits, which have then been exploited for parallelized ptychographic image reconstruction [24]. Because one does not have to manually calculate a new set of derivatives as the forward problem is modified, AD has subsequently been used for image reconstruction in near-field and Bragg ptychography [25], and for comparing near-field and far-field ptychography with near-field holography with both Gaussian and Poisson noise models [26]. Automatic differentiation has also been used to address 3D image reconstruction beyond the

depth-of-focus limit by using multislice propagation for the forward model [14]. These successes have motivated our development of the software framework we report here, whose primary aim is to extend the application of AD beyond a specific imaging modality.

In designing an AD-based framework for image reconstruction, we have attempted to follow these guidelines:

- The framework should be generic and able to work with a variety of imaging techniques by assuming the general abstract model of image reconstruction. That is, the essential components of the framework should include the object function, the probe, the forward model, the loss, the optimizer, and the AD-based differentiator, but each of them should be subject to as few assumptions as possible, and whenever an assumption is necessary, it should be preferably a general one. For example, the object function is assumed to be 3D, which is downward-compatible with 2D image reconstruction problems. Another example is in regard to the forward model: one of the several forward model classes provided in the framework is a ptychotomography model assuming multiple viewing angles and multiple diffraction patterns per angle. By setting the number of diffraction patterns per angle to 1 and using a larger probe size, it can also work for full-field holography.

- The framework should be able to not only optimize the object function, but also refine experimental parameters such as probe positions or projection alignments in order to address practical issues one would encounter in experiments.

- The forward model part should have a plug-and-play characteristic, which would allow users to conveniently define new forward models to work with the framework, or to modify the existing ones. We will show in Section 2.2 that we achieved this by packaging each forward model as an individual Python class that contains the prediction function and the loss function. As long as new variants of the class are created with both components following the input/output requirements, users can get it to work with AD immediately after telling the program to calculate the loss using that forward model. The same plug-and-play readiness also applies to refinable parameters and optimizers.

- The framework should not be bound to a certain AD library. There are a variety of AD packages available, and each of them are uniquely advantageous in some aspects. We have thus created a unified frontend in our framework that wraps two AD libraries, namely *PyTorch* and *Autograd*, and used the application programming interfaces (APIs) from that frontend to build all forward models. In this way, the user can switch between both AD backends by simply changing an option. The frontend may also be expanded to other AD backends as long as they are added to the frontend following the stipulated input/output of each function.

- The framework should be able to run on both single workstations and high-performance computers (HPCs), which allows one to freely choose the right platform and optimally balance convenience and scalability. This means that it should support parallelized processing using a widely available protocol. As will be introduced in Section 2.5, we used the MPI protocol [27] in our case. We have also implemented several strategies for parallelization, ranging from data parallelism [28] to more memory-efficient schemes based on either MPI communication or parallel HDF5 [29].

- The framework also needs to be implemented in a language that has a large community of scientific computation users, and has the API support of most popular AD libraries. The language itself should also be portable, so that the program can be run on a wide range of platforms. As such, we choose to implement our framework in Python.

Our design following these criteria leads to Adorym, which stands for **A**utomatic **D**ifferentiation-based **O**bject **R**etrieval with D**y**namical **M**odeling. The same name appeared in our early publication on the algorithm for reconstructing objects beyond the depth-of-focus limit [14], where the notion of "dynamical modeling" means both that the forward model accounts for dynamical scattering in thick samples using multislice propagation, and that the forward model can be dynamically adjusted for various scenarios thanks to AD. The source code of Adorym is openly available on our GitHub repository (https://github.com/mdw771/adorym). We have also prepared a detailed documentation (https://adorym.readthedocs.io/). The rest of this paper will describe the overall architecture of Adorym and each component of it. We will then show reconstruction results of both simulated and experimental, and demonstrate Adorym's performance for varying experimental types and imaging techniques.

2. Methods and theories

With the aim of maximizing the applicability of Adorym to a wide range of X-ray imaging techniques, we adhered to modular and object-oriented programming principles when designing it. In this section, we briefly introduce the most crucial modules and infrastructures of Adorym – namely, its data format, its forward models and loss functions, its AD backends, and its parallelization scheme. These parts are directly relevant to the realization of Adorym's versatility, flexibility, and scalability. While we will limit our narrative to these key features of Adorym in the main text, readers can refer to Supplement 1 for further detail. A full picture illustrating its architecture is shown in Fig. 2.
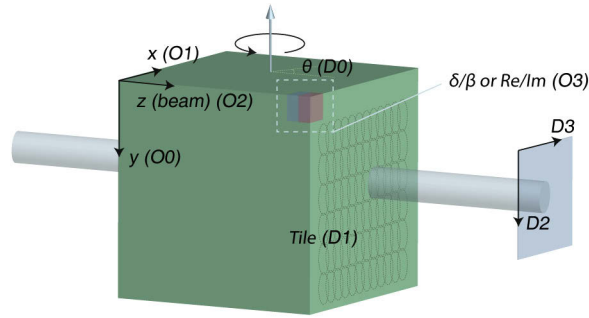
2.1. Data format and array structures

Adorym uses HDF5 files [29] for input and output, where acquired images are saved as a 4D dataset with shape `[num_angles, num_tiles, len_detector_y, len_detector_x]`. This is a general data format that is compatible towards more specific imaging types: for example, 2D ptychography has `num_angles = 1` and `num_tiles > 1`, while full-field tomography has `num_tiles = 1` and `num_angles > 1`. The last two dimensions specify a 2D image which is referred to as a "tile." For ptychography, a tile is just a single diffraction pattern. When dealing with full-field data from large detector arrays, Adorym provides a script to divide each image into several subblocks or tiles, so that the divided image data can be treated in a way just like ptychography, where only a small number of these tiles are processed each time. When applied to wavefield propagation, this is known as a "tiling-based" approach; it allows single workstations to work with very large arrays [30], or parallel computation on large arrays when using HPCs [31].

During reconstruction, the object function being solved is stored as a 4D array, with the first 3 dimensions representing the spatial axes. Depending on user setting, the last dimension represents either the δ/β-parts of the object's complex refractive indices, or the real/imaginary parts of the object's multiplicative modulation function (*i.e.*, for an incident wavefield $\psi(\boldsymbol{r}_{x,y})$ on a 2D plane xy, the wavefield modulated by object function $O(\boldsymbol{r}_{x,y})$ is given by $\psi(\boldsymbol{r}_{x,y}) \cdot \left(\Re[O(\boldsymbol{r}_{x,y})] + i\Im[O(\boldsymbol{r}_{x,y})] \right)$). While the real/imaginary representation is more commonly used in coherent diffraction imaging with non-uniform illumination such in ptychography [5], the δ/β representation can potentially remove the need for phase wrapping when a proper regularizer is used (this is discussed in Section S1.2 of Supplement 1). The array geometry of the raw data file and the object function is shown in Fig. 3.

2.2. Forward model and refinable parameters

Adorym comes with a `propagate` module for simulating wave propagation. Near-field and far-field propagation can be done using the built-in Fresnel propagation and Fraunhofer functions. Furthermore, Adorym can also model diffraction in the object using the multislice method [32,33],

Fig. 2. Architecture of Adorym, listing the relationships between all modules, classes, and child classes.

Fig. 3. Representation of experimental coordinates in Adorym's readable dataset (*D*) and object function array (*O*). Directions and quantities are labeled with the index of dimension in the corresponding array; for example, *O*2 means that the associated object axis is stored as the 2nd dimension of the object array.

so that multiple scattering in thick samples beyond the depth of focus can be accounted for in object reconstruction [14]. In Adorym, two variants of the multislice algorithm are implemented: the first assumes a constant slice spacing, so that the Fresnel convolutional kernel can be kept constant. This implementation is intended for performing joint ptychotomography reconstruction for thick samples with isotropic voxel size. The other variant allows the slices to be separated by unequal spacing, which is more suitable for single-angle multislice ptychography [8,34,35]. For the latter, Adorym allows the slice spacings to be refined in the AD framework. Moreover, Adorym allows one to use and reconstruct multiple mutually incoherent probe modes, which can be used to account for partial coherence [36] or continuous motion scanning [37–39].

Adorym also allows other experimental parameters to be refined along with the minimization of the loss function, as long as these parameters can be incorporated in the forward model as differentiable functions. An example is probe position refinement in ptychography, as shown in Algorithm 1 and demonstrated in Sec. 3.4. A full list of refinable parameters currently provided by Adorym can be found in Section S1.D.2 of Supplement 1.

The fact that Adorym assumes a 4D data format, and a 3D object function with 2 channels, means that it can use a ptychotomographic forward model to work with many imaging techniques. Holography data, for instance, is interpreted by Adorym as a special type of ptychography with 1 tile per angle, with detector size equal to the *y*- and *x*-dimensions of the object array, and with a near-field propagation from the sample plane to the detector plane. However, this does not prevent us from creating forward models dedicated for more specific imaging techniques. In fact, since all forward models are packaged as individual classes, users will find it easy to define new forward models, and then use them in the reconstruction routine.

2.3. Loss functions

For the final loss function, Adorym has two built-in types of data mismatch term. The first is the least-square error (LSQ), expressed as

$$\mathcal{D}_{\text{LSQ}} = \frac{1}{N_d} \sum_{m}^{N_d} \left\| \sqrt{I_{\text{pred},m}} - \sqrt{I_{\text{meas},m}} \right\|^2, \tag{4}$$

where N_d is the number of detector (or tile) pixels. The second is the Poisson maximum likelihood error, expressed as [2]:

$$\mathcal{D}_{\text{Poisson}} = \frac{1}{N_d} \sum_{m}^{N_d} I_{\text{pred},m} - I_{\text{meas},m} \log(I_{\text{pred},m}). \tag{5}$$

Algorithm 1: An example `predict` function in the forward model of far-field ptychoto-mography. This is for data parallelism mode, with probe position refinement and multiple probe modes.

Input:

detector size $[l_y, l_x]$, object size $[L_y, L_x, L_z]$, full object function O, a list of probe modes $\boldsymbol{\Psi}$, probe positions $\boldsymbol{R}$, probe position corrections $\boldsymbol{\Delta R}$, wavelength λ, pixel size δ, current rotation angle θ

Initialization

> $\boldsymbol{o} \leftarrow []$ // List of object chunks each in shape of $[l_y, l_x, L_z]$
> $\boldsymbol{\Psi}_{\text{detector}} \leftarrow []$ // List of detector-plane waves
> $\boldsymbol{\Psi}_{\text{shifted}} \leftarrow []$ // Shifted probe functions

end

Procedure

> $O' \leftarrow \texttt{Rotate}(O)$
> **for** r in $\boldsymbol{R}$ **do**
>> /* Get local object chunks corresponding to
>>   diffraction patterns (tiles).   */
>> $\boldsymbol{o}.\texttt{append}(\texttt{TakeObjectChunk}(O', \boldsymbol{r}))$
>> /* Shift probes according to the list of position
>>   corrections.   */
>> $\boldsymbol{\Psi}_{\text{shifted}}.\texttt{append}(\texttt{FourierShift}(\boldsymbol{\Psi}, \boldsymbol{\Delta R}))$
> **end**
> /* Do propagation for all probe modes.   */
> **for** ψ_i in $\boldsymbol{\Psi}_{shifted}$ **do**
>> $\psi_i \leftarrow \texttt{MultislicePropagate}(\psi_i, \boldsymbol{o}, \lambda, \delta)$
>> $\psi_i \leftarrow \texttt{FarFieldPropagate}(\psi_i)$
>> $\boldsymbol{\Psi}_{\text{detector}}.\texttt{append}(\psi_i)$
> **end**

end

Output: $\boldsymbol{\Psi}_{\text{detector}}$

While both loss functions have their own merits in terms of numerical robustness and noise resistance [2,26], we provide both of them for users' choice. Furthermore, Adorym also allows users to conveniently add new loss function types (for example, mixed Gaussian-Poisson).

Adorym allows users to provide a finite support constraint to supply prior knowledge about the sample. This is commonly used in coherent diffraction imaging [40–42]. Also, the finite support mask can be contracted during reconstruction using the shrinkwrap algorithm [42] according to the user-defined threshold. Furthermore, Adorym comes with `Regularizer` classes which contains several types of Tikhonov regularizers [43,44]. These include the l_1-norm [16], the reweighted l_1-norm [45], and total variation [46]. A detailed description of these regularizers can be found in Supplement 1. Additional regularizers can be added by creating new child classes of `Regularizer`.

2.4. AD engines and optimizers

The automatic differentiation (AD) engine is the cornerstone of Adorym: it provides the functionality to calculate the gradients of the loss function with regards to the object, the probe, and other parameters. Adorym uses two AD backends: *Autograd* [47], and *PyTorch* [48]. *Autograd* builds its data type and functions on the basis of the popular scientific computation

package NumPy [49] and SciPy [50]; these in turn can make use of hardware-tuned libraries such as the Intel Math Kernel library [51]. However, *Autograd* does not have built-in support for graphical processing units (GPUs). This shortcoming is addressed in *PyTorch*, which also has a larger user community. Both libraries are dynamic graph tools that allow the computational graph (representing the forward model) to be altered at runtime [52]. This allows for more flexible workflow control, and easier debugging. In order for users to easily switch between both backends, we have built a common front end for them in the `wrapper` module. This module provides a unified set of APIs to create optimizable or constant variables (arrays in *Autograd*, or tensors in *PyTorch*), call functions, and compute gradients.

Optimizers define how gradients should be used to update the object function or other optimizable quantities. Adorym currently provides 4 built-in optimizers: gradient descent (GD), momentum gradient descent [53], adaptive momentum estimation (Adam) [54], and conjugate gradient (CG) [55]. Moreover, Adorym also has a `ScipyOptimizer` class that wraps the `optimize.minimize` module of the SciPy library [50], so that users may access more optimization algorithms coming with SciPy when using the *Autograd* backend on CPU. More details about these optimizers can be found in Supplement 1.

2.5. Parallelization modes

Adorym supports parallelized processing based on the message passing interface (MPI) [27], which allows the software to spawn multiple processes using different GPUs, or many nodes on an HPC. Parallel processing based on MPI can be implemented in several different ways, each of which differs from others in terms of computational overhead and memory consumption. On the most basic level, Adorym allows users to run in its data parallelism (DP) mode, where each MPI rank saves a full copy of the object function and all other parameters (Fig. 4(a)) [28]. Before updating the parameters, the gradients are averaged over all ranks. While the DP mode is low in overhead, it has high memory consumption. This in turn limits the ability to reconstruct objects when one has limited memory resources. Thus, Adorym also provides another two modes of parallelization, namely the distributed object (DO) mode, and the HDF5-file-mediated low-memory (H5) mode.

2.5.1. Distributed object (DO) mode

In distributed object (DO) mode, only one object function is jointly kept by all ranks. This is done by seperating the object along the vertical axis into several slabs, and letting each rank store one of them in its available RAM as 32-bit floating point numbers. In this way, standard tomographic rotation can be done independently by all ranks in parallel. The stored object slab does not enter the device memory as a whole when GPU acceleration is enabled; instead, only partial chunks of the object that are relevant to the tiles being processed (*i.e.*, on the beam paths of these tiles) are extracted from the object slabs in the CPU, and only these partial chunks are sent to GPU memory for gradient computation. Since the shape of a slab is usually different from an object chunk (whose *x-y* cross sections should match the size of a tile), we use MPI's AlltoAll communication [27] for a rank to gather the voxels it needs to assemble an object chunk from other ranks. This is illustrated in Fig. 5, where we assume the MPI command spawns 8 ranks, and the object function is evenly distributed among these ranks. For simplicity, we assume a batch size of 1, meaning a rank processes only 1 tile at a time. Before entering the differentiation loop, the object slab kept by each rank is duplicated as a new array, which is then rotated to the currently processed viewing angle. In our demonstrative scenario, rank 0 processes the top left tile in the first iteration, and the vertical size of the tile spans four object slabs. As ranks 0–3 have knowledge about rank 0's job assignment, they extract the part needed by rank 0 from their own slabs, and send them to rank 0. Rank 0 then assembles the object chunk from the partial slabs by concatenating them along *y* in order. Since rank 0 itself contains the object slab needed by

other ranks, it also needs to send information to them. The collective send/receive can be done using MPI's AlltoAll communication in a single step. The ranks then send the assembled object chunks to their assigned GPUs (if available) for gradient computation, yielding the gradient chunks. These gradient chunks are scattered back to the "storage spaces" of relevant ranks on RAM – but this time to "gradient slab" arrays that are one-to-one related to the object slabs. After all tiles on this certain viewing angle are processed, the gradient slabs kept by all ranks are reverted back to 0°. Following that, the object slab is updated by the optimizer independently in each rank. When reconstruction finishes, the object slabs stored by all ranks are dumped to a RAM-buffered hard drive, so that they can be stacked to form the full object. The optimization workflow of the DO mode is shown in Fig. 4(b).

While the DO mode has the advantage of requiring less memory per rank, a limitation is that tilt refinement about the x- and z-axis is hard to implement. That is because tilting about these axes requires a rank to get voxel values from other slabs, which not only induces excessive MPI communication, but also requires AD to differentiate through MPI operations. The latter is not impossible, but existing AD packages may need to be modified in order to add that feature. Another possible approach among the slabs kept by different MPI ranks is to introduce overlapping regions along y; in this case, tilting about x and z can be done individually by each rank, though the degree of tilt is limited by the length of overlap. This is not yet implemented in Adorym, but could be added in the future.

2.5.2. HDF5-file-mediated low-memory mode (H5)

When running on an HPC with hundreds or thousands of computational nodes, the DO mode can significantly reduce the memory needed by each node to store the object function if one uses many nodes (and only a few ranks per node) to distribute the object. If one is instead using a single workstation or laptop, the total volume of RAM is fixed, and very large-scale problems are still difficult to solve even if one uses DO. For such cases, Adorym comes with an alternative option of storing the full object function in a parallel HDF5 file [29] on the hard drive that is accessible to all MPI ranks. This is referred to as the H5 mode of data storage. In the H5 mode, each rank reads or writes only partial chunks of the object or gradient that are relevant to the current batch of raw data it is processing, similar to the DO mode except that the MPI communications are replaced with hard drive read/write instructions. While the H5 mode allows the reconstruction of large objects on limited memory machines, I/O with a hard drive (even with a solid state drive) is slower than memory access. Additionally, writing into an HDF5 file with multiple ranks is subject to contention, which may be mitigated if a parallel file system with multiple object storage targets (OSTs) is available [56]. This is more likely to be available at an HPC facility than on a smaller, locally managed system, and precise adjustment of striping size and HDF5 chunking are needed to optimize OST performance [57]. Despite these challenges, the HDF5 mode is valuable because it enables reconstructions of large objects and/or complex forward models on limited memory machines. It also provides future-proofing because, as fourth-generation synchrotron facilities deliver higher brightness and enable one to image very thick samples [58], we may eventually encounter extra-large objects which might be difficult to reconstruct even in existing HPC machines.

A comparison of the three parallelization modes is shown in Fig. 4.
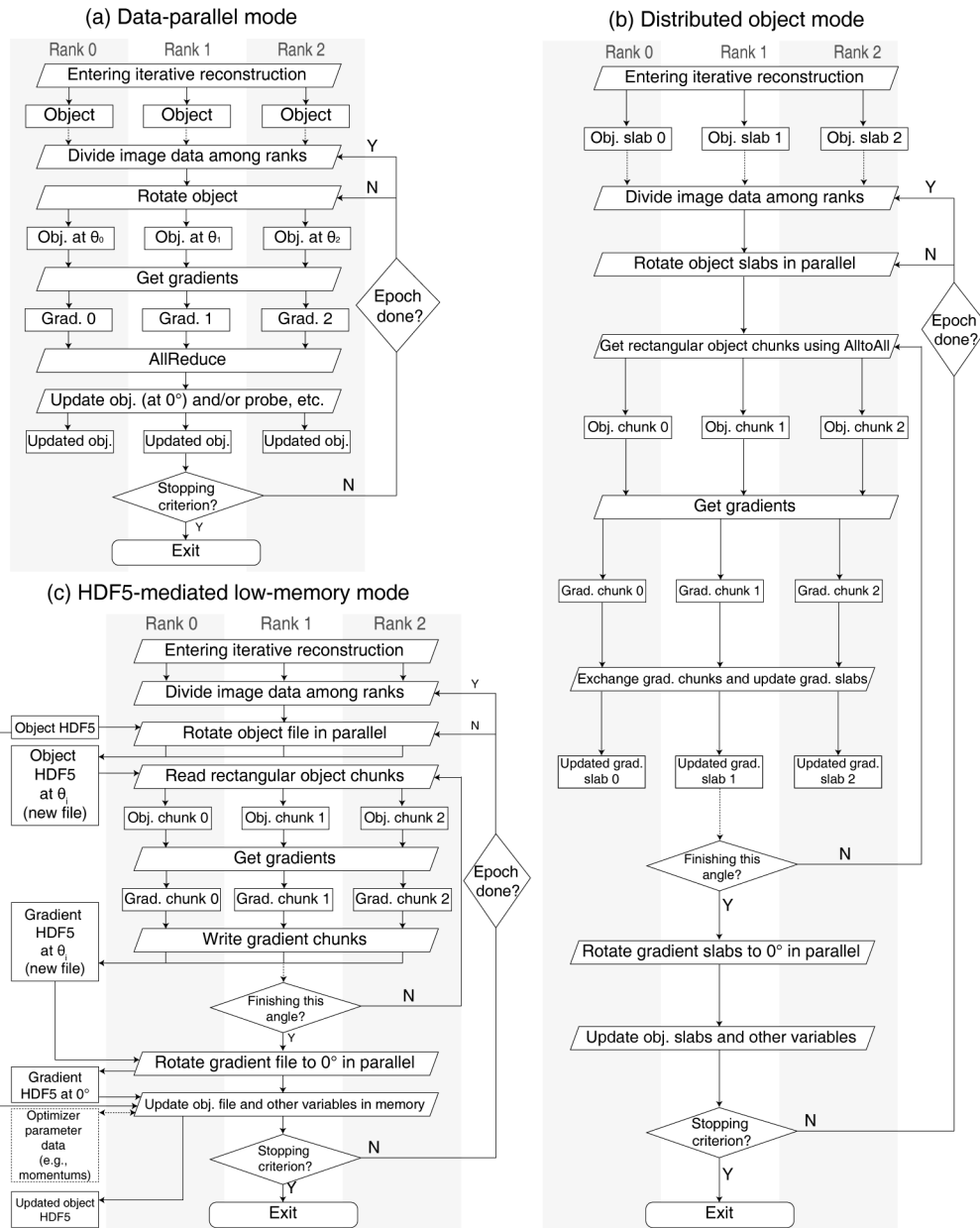
Fig. 4. Workflow diagram of Adorym in DP mode, DO mode, and H5 mode.
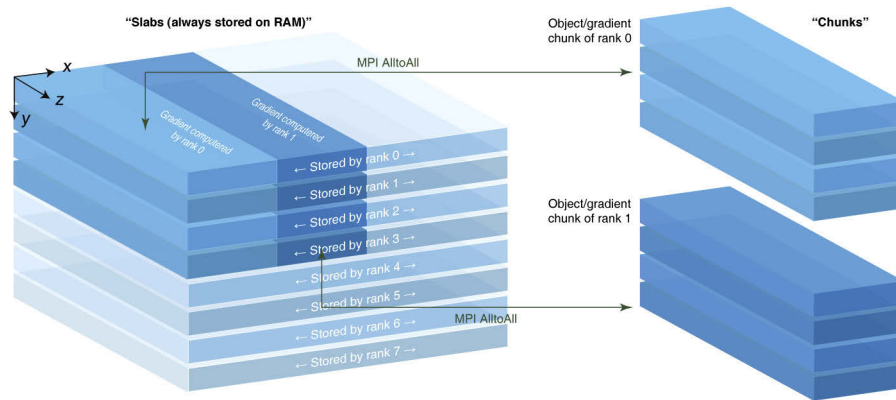
Fig. 5. Illustration of the distributed object (DO) scheme. With multiple MPI ranks, the object is divided into a vertical stack of slabs, which are kept separately by all ranks. When a rank processes a diffraction image, it gets data from ranks that possess the parts of the object function that it needs, after which it assembles these partial slabs into the object chunk corresponding to the beam path of the diffraction image that it is processing. After gradient of the chunk is calculated, it is scattered back into the same positions of the gradient slabs kept by relevant ranks. Object update is done by each rank individually using the gathered gradient array.

3. Results

Having described the key elements of Adorym, we now demonstrate its use for reconstructing several simulated and experimental datasets.

3.1. Computing platforms

This section involves several different computing platforms, on which Adorym or other referenced packages were run:

- The machine "Godzilla" is a HP Z8 G4 workstation with two Intel Xeon Silver 4108 CPUs (8 cores each), 768 GB DDR4 RAM, one NVIDIA P4000 GPU (8 GB), a 256 GB solid state drive for the Red Hat Enterprise Linux 7 operating system and swap files, and a 18-TB, 3-drive RAID 5 disk array.

- The machine "Blues" is an HPC system at Argonne's Laboratory Computing Resource Center (LCRC). It includes 300 nodes equipped with 16-core Intel Xeon E5-2670 CPUs (Sandy Bridge architecture), and 64 GB of memory per node.

- The machine "Cooley" is an HPC system at the Argonne Leadership Computing Facility (ALCF). It has 126 compute nodes with Intel Haswell architecture. Each node has two 6-core, 2.4-GHz Intel E5–2620 CPUs, and NVIDIA Tesla K80 dual GPUs.

- The machine "Theta" is a HPC system at the Argonne Leadership Computing Facility (ALCF). This Cray XC40 system uses the Aries Dragonfly interconnect for inter-node communication [59]. Each node of Theta possesses a 64-core Intel Xeon Phi processor (Intel KNL 7230) and 192 GB DDR4 RAM. An additional 16 GB multi-channel dynamic RAM (MCDRAM) is also available with each node; this is used as the last-level cache between L2 and DDR4 memory in our case. Up to 4392 nodes are available, though we used no more than 256 in the work reported here. The storage system of Theta uses the

Lustre parallel file system, which allows data to be striped among 56 of its object storage targets (OSTs). This system was used to analyze the charcoal dataset shown in Fig. 13.

We indicate the platform used for each demonstration of Adorym below, and show compute time metrics in Table 1.

Table 1. Performance data of all test cases shown in Sec. 3., using the compute platforms described in Sec. 3.1. Walltimes shown in seconds do not include the time spent for saving intermediate results, or for providing diagnostic checkpoint results after each epoch.

Case	Backend	Platform	Tile size	Number of tiles per batch	Mean walltime per batch (s)	Data size (GB)
MDH (simulation)	*PyTorch*	Godzilla	512×512	4 (1 set of 4 distances)	0.08	0.004
MDH (experimental)	*PyTorch*	Godzilla	2160×2560	3 (1 set of 3 distances)	1.4	166 (all angles)
2D ptychography w/ pos. refin.	*PyTorch*	Godzilla	256×256 (5 modes)	35	0.4	2.6
Sparse MS ptychography	*PyTorch*	Godzilla	128×128	101	0.3	0.19
Ptychotomography	*PyTorch*	Godzilla	64×64 (5 modes)	50	5.2/2.2 (w/wo. tilt refin.)	29.5
Tomography with DO	*Autograd*	Theta	25×52	1	5.8	4.8

3.2. Multi-distance holography for heterogeneous samples: a simulation case study

Multi-distance holography (MDH) [7,60] is an in-line coherent imaging technique, where holograms are collected at multiple distances to provide diversity for robust phase retrieval. These distances should be chosen to minimize the overlap of spatial frequencies for which the contrast transfer function (CTF) from Fresnel propagation crosses zero [61]. The recorded data are non-linearly related to both the magnitude and phase of the object, but a linear approximation can be taken if one assumes that the object is pure-phase [7] or weak in absorption, and has uniform composition so that $\delta(\boldsymbol{r}) = (1/\kappa)\beta(\boldsymbol{r})$ (where $\delta(\boldsymbol{r})$ and $\beta(\boldsymbol{r})$ are terms in the complex refractive index $n(\boldsymbol{r}) = 1 - \delta(\boldsymbol{r}) - i\beta(\boldsymbol{r})$ as a function of voxel coordinate $\boldsymbol{r}$, and κ is a constant coefficient) [62]. In this case the linearized CTF can be easily inverted, and the phase of the object can be found directly from a combination of the back-propagated holograms [7]. Even when these conditions are not fully satisfied, the multiple hologram distances provide phase diversity, allowing object reconstruction using an iteratively regularized Gauss–Newton (IRGN) method with the addition of either a finite-support constraint [63] or the pure-phase object approximation [60].

In order to test the ability of Adorym to reconstruct multi-distance holography data, we created a 512×512 2D simulated object shown in Fig. 6. The well-known "cameraman" image was used as the magnitude of the object modulation function, in which the magnitude ranges from 0.6 to 1.0. The "mandrill" image was used for the phase, with a range of -0.5 to $+0.5$ radians. Because an optimization approach allows one to use an accurate forward model, there is also no need to require compositional uniformity or weak object conditions. By using separate images for magnitude and phase, we effectively gave individual pixels wildly differing ratios of δ/β in the X-ray refractive index of $n = 1 - \delta - i\beta$. Thus the simulated object satisfied neither the weak absorption nor the uniform composition approximations. As a simplification, we assumed that the object and detector had the same 1 μm pixel size (a simplification from the common practice of using geometric magnification from a smaller source), which means a hologram recorded using 17.5 keV X rays would have a depth of focus of about 7.3 cm. We then simulated the recording of holograms at $z = 40, 60, 80,$ and 100 cm distance from the object, so these holograms are noticeably different from each other as shown in Fig. 6.
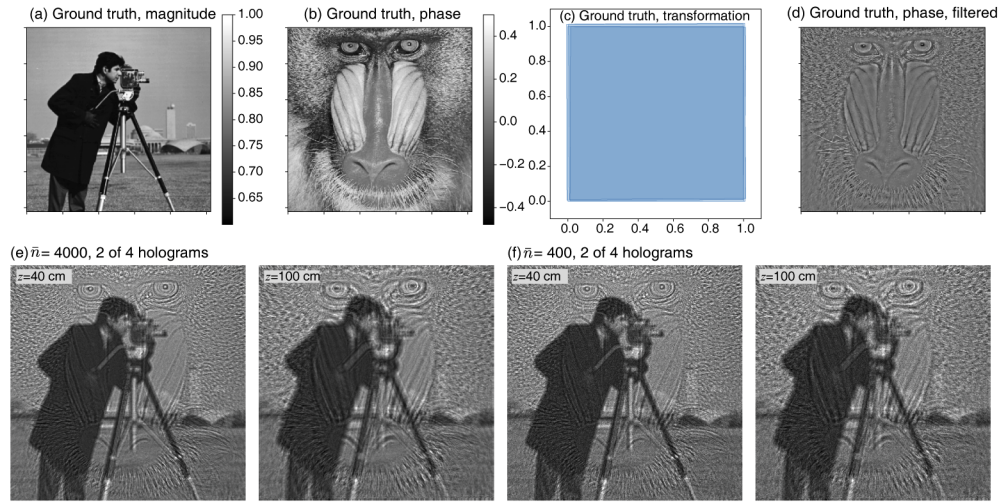
Fig. 6. Ground truth images used in the simulation. Images (a) and (b) show the magnitude and phase of the simulated object, which are uncorrelated. In (c) we show the artificial affine errors added to the holograms. Transformation matrices used for all 4 holograms at different defocusing distances (where the one for the first hologram is an identity transformation matrix) are applied to the vertices of 4 1×1 squares, and the distorted squares are shown in the figure. In (d) we show a high-pass filtered version of (b), which is used for calculating the SSIM of reconstructed phase maps. The first (40 cm distance) and last (100 cm) of four recorded holograms are shown in (e) with $\bar{n} = 4000$ photons per pixel incident, and in (f) with $\bar{n} = 400$ photons per pixel incident.

The intensity of an in-line hologram is unable to record the zero-spatial-frequency phase of an object; in fact, phase contrast is only transferred at spatial frequencies that appear outside the first Fresnel zone of radius $r_1 = \sqrt{\lambda z}$ in the hologram. Therefore we show in Fig. 6(d) a version of the "mandrill" phase object which has been highpass filtered to include only those spatial frequencies above $1/4\sqrt{\lambda \bar{z}}$, with $\bar{z} = 70$ cm representing the mean hologram recording distance (and with the filter cutoff smoothed using a Gaussian with $\sigma = 2.5$ pixels). Combined with the "cameraman" magnitude image, this highpass-filtered phase image serves as a modified ground-truth object when evaluating multi-distance hologram reconstructions. We then incorporated several types of measurement error into our simulated hologram recordings:

1. When recording holograms at multiple distances, it is possible for the translation stage of the detector system (typically a scintillator screen followed by a microscope objective and a visible light camera) to introduce slight translation and orientation errors between the recordings, and a scaling error can arise from imperfect knowledge of the source-to-object and object-to-detector distances used to provide geometric magnification. These misalignment factors can be collectively described and refined by Adorym as an affine transformation matrix. We therefore applied random affine transformations to the second, third, and fourth holograms, resulting in misalignment in translation, tilt, and non-uniform scaling. Figure 6(c) shows the distortion that would result if the same affine transformations were applied to full 512×512 μm hologram recordings in Cartesian coordinates. To measure the error in the recovered inverse affine transformation matrix A_r versus the actual affine matrix A_0, we used a vector $r_0 = (1, 1)$ in a Euclidian distance metric d_{affine} of

$$d_{\text{affine}} = \frac{|A_r A_0 r_0 - r_0|}{|r_0|}, \tag{6}$$

which has a value of zero if the recovered affine matrix matches the actual matrix.

2. In order to test refinement of propagation distances, we gave Adorym deliberately erroneous starting values of $z = 38, 58, 78$ and 98 cm for the hologram recording distances, rather than the actual distances of $z = 40, 60, 80$, and 100 cm.

3. When indicated, we also added Poisson-distributed noise corresponding to $\bar{n} = 4000$ or $\bar{n} = 400$ incident photons per pixel, so that each of the four holograms had an average of either 1000 or 100 photons per pixel before accounting for absorption in the object.

These errors were incorporated into the set of simulated holograms, with two of the four holograms shown in Figs. 6(e) and (f). We then set the object size to $512 \times 512 \times 1$, and ran 1000 epochs of reconstruction using *PyTorch* as the GPU-enabled backend on our workstation "Godzilla." This led to the results shown in Fig. 7. With both projection affine transformation refinement and propagation distance refinement turned on with the LSQ loss function (Eq. (4)), a clear and sharp magnitude image was obtained as shown in Fig. 7(a). The propagation distances were refined to $z = 39.7, 59.7, 79.9$, and 100.1 cm, which are close to the true values of $z = 40, 60, 80$, and 100 cm, especially considering the 7.3 cm depth of focus. When we instead used the Poisson loss function (Eq. (5)), we obtained the reconstructed image of Fig. 7(b), which is as good as the LSQ result; in addition, the refined hologram distances of $z = 39.9, 59.9, 80.0$, and 100.0 cm were also very accurate. When we turned off affine transformation refinement while using LSQ, the reconstruction was significantly degraded, showing both significant magnitude/phase crosstalk and also fringe artifacts as shown in Fig. 7(c). When we turned off distance refinement but kept transformation refinement, the reconstructed images showed higher fidelity than in the distance-refinement-only case, as shown in Fig. 7(d). This indicates that affine transformation refinement plays a crucial role in this example.

In order to quantitatively compare these outcomes, we calculated the structural similarity indices (SSIMs) [64] of the various reconstructed magnitude and phase images in comparison with the modified ground truth object as described above. Readers could consult the cited paper, or Supplement 1, for the detailed definition. We used the magnitude image of Fig. 6(a) and the highpass filtered phase image of Fig. 6(d) as ground truth images, with the phase image normalized using its mean $\bar{\varphi}$ and standard deviation σ_φ as $\varphi' = (\varphi - \bar{\varphi})/\sigma_\varphi$ so as to eliminate phase offsets and scalings from the SSIM calculation. The results of SSIM comparisons of these ground truth images with various reconstructions are presented in Fig. 8(a), showing significant improvement with affine transformation refinement turned on. The influence of distance error, on the other hand, is relatively small.

We now consider a more realistic situation where the measured images are subject to photon noise. For this, the holograms were created with Poisson noise assuming two incident photon fluence levels: $\bar{n} = 4000$ photons per pixel total over the 4 holograms, and $\bar{n} = 400$; examples of 2 of the 4 holograms at each fluence are shown in Figs. 6(e) and (f). For each noise level, we repeated the reconstruction using both LSQ and Poisson loss functions, with both transformation and distance refinement turned on. In addition to that, we also include in our test scenarios the use of LSQ along with a total variation (TV) regularization described in Table S1 of Supplement 1 (TV regularization is often used for noise suppression by guiding the reconstruction algorithm towards a spatially smoother solution [46,65]). The TV regularizer weighting hyperparameter was set to $\gamma = 0.01$, since higher weightings resulted in "blocky" looking reconstructions with significant loss of high-frequency information. From the results shown in Fig. 7, we can see that the Poisson loss function leads to the best visual appearance for both fluence levels. The addition of the TV term reduces the contrast of crosstalk artifacts between phase and magnitude, but it also affects high-spatial-frequency features.

For distance refinement under noisy imaging conditions, LSQ+TV yielded the worst performance: the reconstruction from dataset with $\bar{n} = 4000$ total photons/pixel ended up with refined
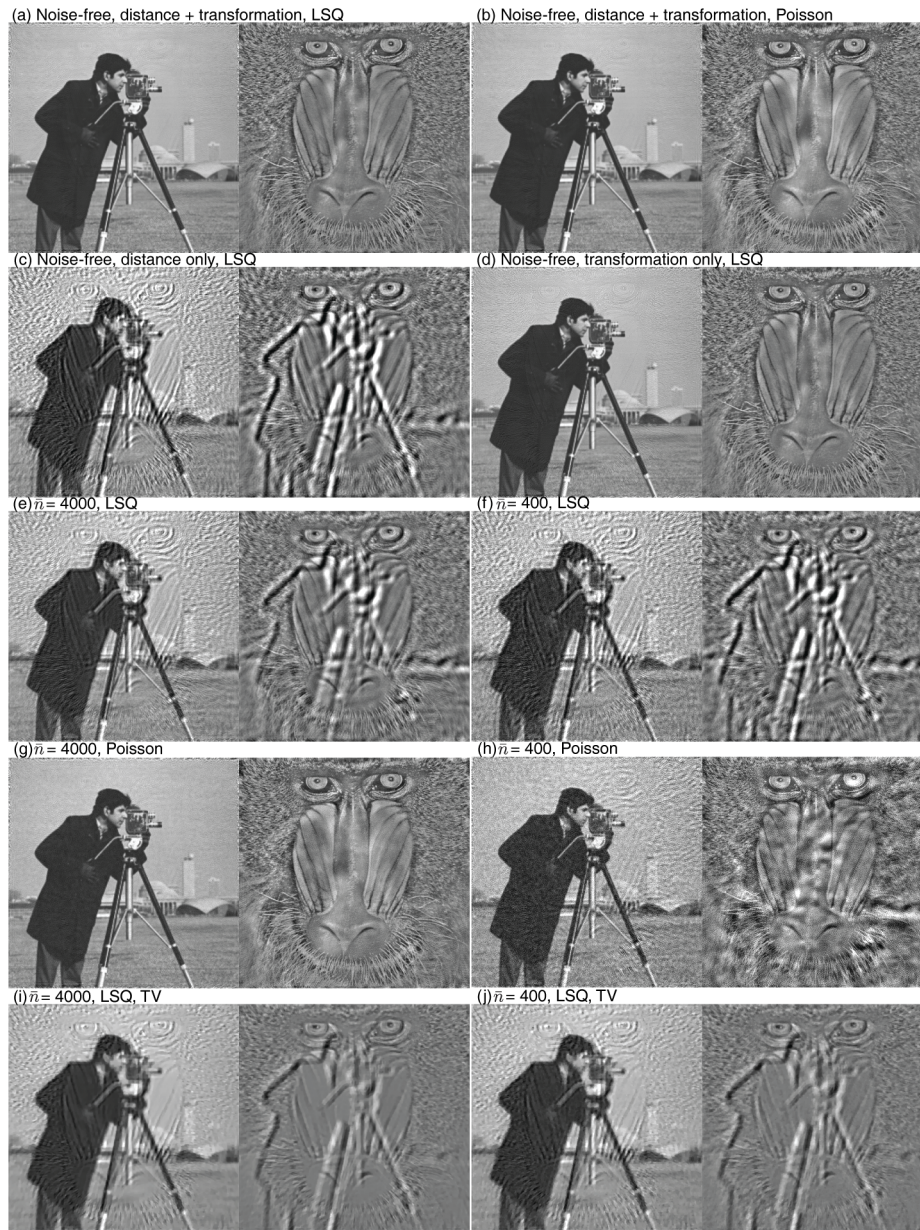
Fig. 7. Multi-distance holography test using the "cameraman" image for magnitude and the "mandrill" image for phase. Images (a-d) show Adorym reconstructions from noise-free holograms, either with both affine transformation and distance refinement enabled, or with just one of them. In the former case, both LSQ (Eq. (4)) and Poisson (Eq. (5)) loss function results are shown. We then show magnitude and phase reconstructions that include both distance and transformation refinement with limited per-pixel photons of $\bar{n} = 4000$ using LSQ (e) and Poisson (g) noise models, and for $\bar{n} = 400$ using LSQ (f) and Poisson (h) noise models – in both cases, use of the Poisson noise model reduces crosstalk between the magnitude "cameraman" and phase "mandrill" images. If we add a total variation (TV) regularizer to the reconstruction, the image quality is degraded (with significant crosstalk between magnitude and phase images) because both the "cameraman" and "mandrill" images contain high spatial frequency detail. This is shown in (i) for $\bar{n} = 4000$ and in (j) for $\bar{n} = 400$ with the LSQ noise model.
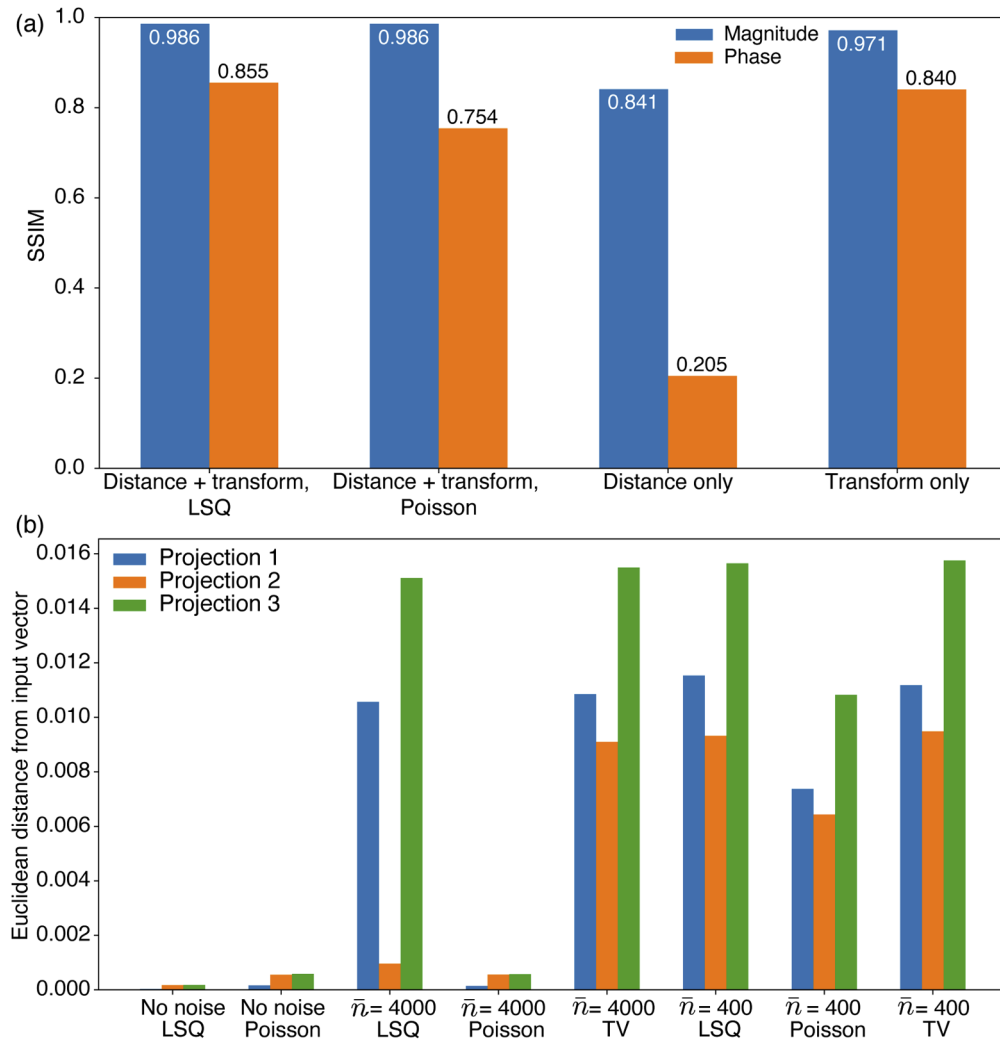
Fig. 8. (a) Structural similarity indices (SSIMs) between ground truth and various reconstructed images from noiseless data. The reconstructed phase contrast images were compared against the highpass filtered phase image shown in Fig. 6(d). (b) Euclidian distance error d_{affine} (Eq. (6)) for the recovered affine transformation matrix A_r, under different choices for loss function and regularizer, and different fluences $\bar{n}$.

distances of z = 37.8, 45.2, 107.1, and 116.1 cm. The average distance error of LSQ+TV at $\bar{n}$ = 400 is about 20% lower than that with $\bar{n}$ = 4000, which is counterintuitive. Because distance refinement is highly dependent on the gradient yielded from the mismatch of Fresnel diffraction fringes, it may be incompatible with the preference for smoothing provided by TV. However, this observation should not discredit the effectiveness of TV in improving reconstruction quality. By further tuning the weight γ, one might be able to reach a much better balance between image smoothness and refinement accuracy, but such exploration is beyond the scope of this paper. Use of LSQ and Poisson error functions without TV regularization work well: for $\bar{n}$ = 400 photons/pixel total, LSQ yielded z = 38.8, 56.1, 79.0, and 101.8 cm, while Poisson gave z = 39.2, 59.2, 79.1, and 99.1 cm. With $\bar{n}$ = 4000 photons/pixel, distance refinement using Poisson is almost as accurate as the noise free case.

Now we compare the results of affine transformation refinement for both noise-free and noisy cases in Fig. 8(b). For noise-free data, the transformation refinement results are almost the exact inverse of the original distorting matrix, giving a normalized Euclidean error (Eq. (6)) of $d_{\text{affine}} \simeq 10^{-3}$ which, for an image less than 512 pixels across, corresponds to a residual error of less than one pixel. When noise is present, using affine refinement together with Poisson loss function leads to significant improvements.

3.3. Experimental multidistance holography with material homogeneity constraint

The numerical experiment shown in Sec. 3.2 assumed an extreme case where the magnitude and phase of the object modulation function are completely uncorrelated. In reality, it is more often the case that these two parts show significant spatial correlation, and this can be exploited as an additional constraint. As noted above, if one can make the heterogeneous material assumption of $\delta(\boldsymbol{r}) = (1/\kappa)\beta(\boldsymbol{r})$, then both the contrast transfer function (CTF) approach as well as a transport of intensity equation (TIE) based approach can be used in a linear approximation to relax the requirement on the diversity of Fresnel recording distances [62]. This assumption of fixed κ can also be incorporated into the Fresnel diffraction model of Adorym, so it is our forward model of choice for multi-distance holography (MDH).

We used Adorym to reconstruct a MDH dataset of a battery electrode that contained porous lithium peroxide (Li_2O_2) formed as the product of a cathodic reaction. The sample was prepared following the procedure used for a transmission X-ray microscope study of the same type of materials [66]. The MDH data were collected at the ID16B beamline of the European Synchrotron Radiation Facility (ESRF) using a beam energy of 29.6 keV and a point-projection microscope setup. The source-to-detector distance was fixed at about 75 cm, and holograms were collected at four sample-to-detector distances so as to minimize zero-crossings in the CTF phase retrieval equation [61]. We used only three of the four holograms for our Adorym reconstruction in order to challenge it with reduced information redundancy. The selected sample-to-detector distances are z_{sd} = 69.61, 69.51, and 69.11 cm, which convert to effective distances of z_{eff} = 5.366, 5.450, and 5.785 cm if a plane-wave illumination were to have been used based on the Fresnel scaling theorem [67]. The holograms were rescaled so that they have the same pixel size of 50 nm. Before reconstruction, the holograms were coarsely aligned using phase correlation. Reconstruction of the phase image was performed first on the workstation "Godzilla" which was run for 200 epochs with κ initialized to be 0.01. After this, refinement was enabled for κ, propagation distance, and hologram affine transformation. The resulting phase contrast image is shown in Fig. 9(a), and performance metrics are shown in Table 1. For comparison purposes, we also obtained a phase contrast image using the CTF algorithm in the weak phase and weak absorption assumption, and where the pre-reconstruction alignment was done separately for both translation and tilt; this image is shown in Fig. 9(b). In comparing these two images, the Adorym reconstruction of Fig. 9(a) allows one to distinguish many individual particles even at the center of the deposition cluster where the particles are most densely stacked. In addition, because the raw holograms are

slightly misaligned in tilt angle while the simple phase correlation method we used only fixes translational misalignment, our input images to Adorym were well aligned at the center but not near the edges. The affine transformation refinement feature of Adorym was able to effectively fix the misalignment issue: in the final result of Fig. 9(a), we do not see any duplicated "ghost features" that would typically come from the misalignment of multiple transmission images taken from the same viewing angle. Rather, the reconstructed phase contrast image shows good contrast, and standalone particles at the boundary of the deposited Li_2O_2 cluster appear to be well defined. Minor fringe artifacts remain visible in the empty region, which can be attributed to the slight deviation of the sample from the homogeneous object assumption. By comparing the intermediate phase map at different stages of the reconstruction process, we do observe the effect of misalignment far away from the image center. Figure 9(c-g) tracks the evolution of a local region near the lower right corner of the phase map from epoch 40 to 200; the location of the region in the whole image is indicated by the green dashed box in Fig. 9(a). At epoch 40, this part of the phase map is severely impacted by feature duplication resulting from hologram misalignment. As the reconstruction continues, the correcting transformation matrices of the second and third hologram are refined further, and feature duplication keeps getting reduced (the first hologram was used as the reference upon which the other two were aligned). The refinement process also adjusted the phase-absorption correlation coefficient κ from 0.01 to 0.1, with convergence achieved before epoch 40; the supplied propagation distances on the other hand were already sufficiently accurate and the refinement did not significantly alter their values.
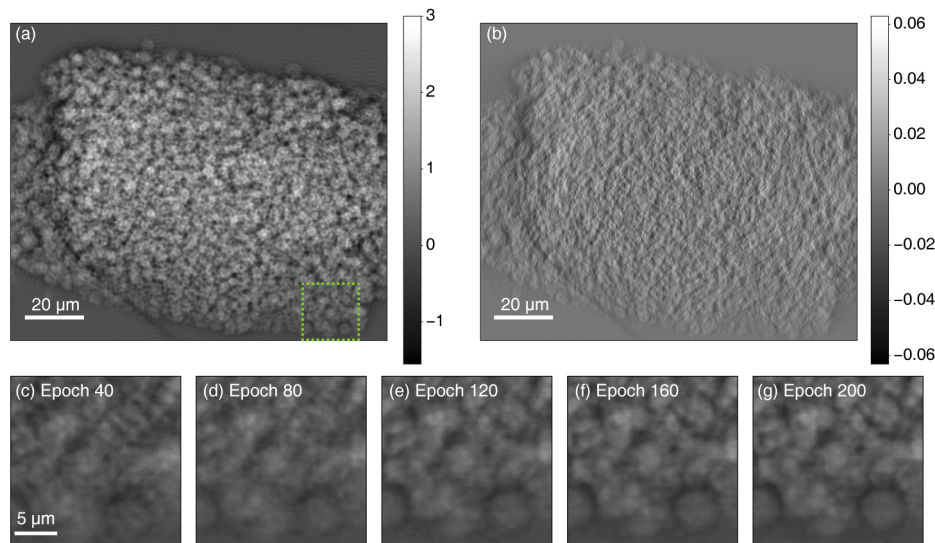
Fig. 9. Phase retrieval and tomographic reconstruction results of multi-distance holography (MDH) data of a Li-O$_2$ battery electrode. (a) Phase contrast projection image retrieved with Adorym using 3 holograms at slightly different distances. (b) A reference reconstruction obtained with the contrast transfer function (CTF) algorithm, using the same data as (a) plus an additional hologram at another distance. (c-g) Evolution of the reconstruction for a region indicated by the green dashed box in (a) from epoch 40 to 200, sampled every 40 epochs. Misalignment of holograms produces errors that are more severe away from the image center, so that the patch at epoch 40 exhibits obvious feature duplication. With transformation refinement, the feature duplication gradually diminishes throughout the reconstruction process [66].

3.4.   2D fly-scan ptychography with probe position refinement

While multidistance holography does not require any beam scanning, ptychography has emerged as a powerful imaging technique for extended objects with no optics-imposed limits on spatial resolution and no approximations required on specimen material homogeneity [68,69]. However, practical experiments often encounter two complications. The first is that the probe positions (the scanned positions of the spatially-limited coherent illumination spot) might be imperfectly controlled or known; this error can be corrected for using either correlative alignment-based [17–19] or optimization-based [11,70,71] approaches. The second is that the probe is often in motion during the recording of a diffraction pattern in a so-called "fly scan" or continuous motion scanning approach; one can used mixed-state pychographic reconstruction methods [36] to compensate for probe motion in ptychography [37–39,72]. One can also correct for both complications together in one optimization-based approach [70]. We show here that Adorym can handle both issues without explicitly implementing the closed-form gradient.

A Siemens star test pattern of 500 nm thick gold was imaged using a scanning X-ray microscope at the 2-ID-D beamline of the Advanced Photon Source (APS). A double-crystal monochromator was used to deliver a 8.8 keV X-ray beam with a bandwidth of $\Delta E/E \approx 1.4 \times 10^{-4}$ and an estimated flux in the focus of 1×10^9 photons/s. The beam was focused on the sample using a gold zone plate with 160 μm diameter, 70 nm outermost zone width, and 700–750 nm thickness; a 30 μm diameter pinhole combined with a 60 μm beam stop is placed downstream serving as an order-selecting aperture to isolate the first diffraction order focus. This produces a modified Airy pattern at the focus, and a ring-shaped illumination pattern at far upstream and downstream planes. The detector was placed 1.75 m downstream the sample, which resulted in a sample-plane pixel size of 13.2 nm. A 70×70 scan grid with a step size about 48 nm was used to collect 4900 diffraction patterns in fly-scan mode, each with an exposure time of 50 ms. A Dectris Eiger 500K detector was used to collect the diffraction patterns, and all images were cropped to 256×256 pixels before reconstruction. This dataset included deliberate errors in illumination probe positions for a demonstration of position refinement using an established approach [73].

We used Adorym on the workstation "Godzilla" (Sec. 3.1) to reconstruct this dataset to yield an object of $618 \times 606 \times 1$ pixels, adding 50 pixels on each side to accommodate specimen regions that were illuminated by the tails of the probe. The probe was initialized using an aperture-defocusing approach: a 256×256 annular wavefront with an outer radius of 10 pixels and an inner radius of 5 pixels was first created to mimic the ring-shaped aperture used in experiment. It was then defocused by 70 μm using Fresnel propagation, which constituted the initial guess for the first probe mode; the other modes were seeded with duplicates of the first mode with Gaussian noise added to create perturbations. To accommodate the fly-scan scheme, we used 5 probe modes, and we used the Fourier shift method to optimize each probe position as described in Sec. 2.2. To prevent drifting of the entire image, we subtracted the mean value of all probe positions from the set after they were updated at the end of each iteration. With both multi-mode reconstruction and probe position refinement turned on, we obtained the reconstructed phase map shown in Fig. 10(a), which has had the surrounding empty regions cropped out. The reconstruction appears sharp, and the spokes of the Siemens star stay straight as expected for this sample fabricated using a precision electron beam lithography system. The image quality remains high even beyond the rectangular region of the main scan (indicated by the green dashed box) due to the tails of the beam, with decreasing quality only seen at the outer edges due to reduced fluence. To illustrate probe position correction, we show in Fig. 10(b) the initial guesses of the probe positions (which were known to be in disagreement with the actual positions [73]) as hollow black circles, with lines connecting to solid blue circles representing the refined positions. If instead we turn off probe position refinement but still use 5 probe modes, the spokes of the Siemens star in the reconstructed image appear distorted, as shown in Fig. 10(c). For the whole image, the mean distance by which the refined probe positions are moved is

6.24 pixels or about 80 nm, which can be seen by the shift of the yellow-boxed region between Fig. 10(a) and (c). Alternatively, if we keep probe position refinement but use only 1 probe mode, we end up with the result of Fig. 10(d) yielding straight spokes as in Fig. 10(c), but with lower image resolution due to the probe motion being incorporated into the image. In addition, the central part of the test pattern exhibits lower contrast, indicating a increased reconstruction error at high spatial frequencies.
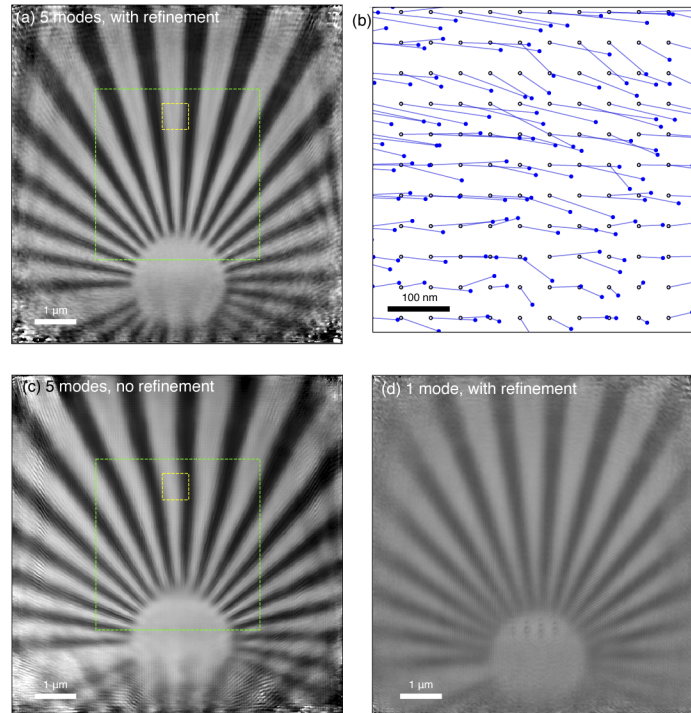
Fig. 10. Ptychography reconstruction results of the Siemens star sample with deliberately-included errors in actual probe position. (a) Reconstruction with 5 probe modes, and probe position refinement. The area covered by the scan grid is shown in the green dashed boxes; the larger reconstructed image area arises due to the large finite size of the illumination probe. In (b), we show the probe position refinement results within the region bounded by the smaller yellow box of (a); here the assumed positions are represented by open dots on a square grid, with the refined positions shown by blue dots. Also shown is the reconstruction with probe modes included but no position refinement (c), or with position refinement included but no probe modes (d). All reconstructed images are shown over the same dynamic range. Comparing the yellow boxes in (a) and (c), one can see that the bounded segment of the Siemens star spoke in (c) is shifted slightly to the left. Thus, the refined probe positions are mostly to the right of the original positions within that window, which agrees with (b).
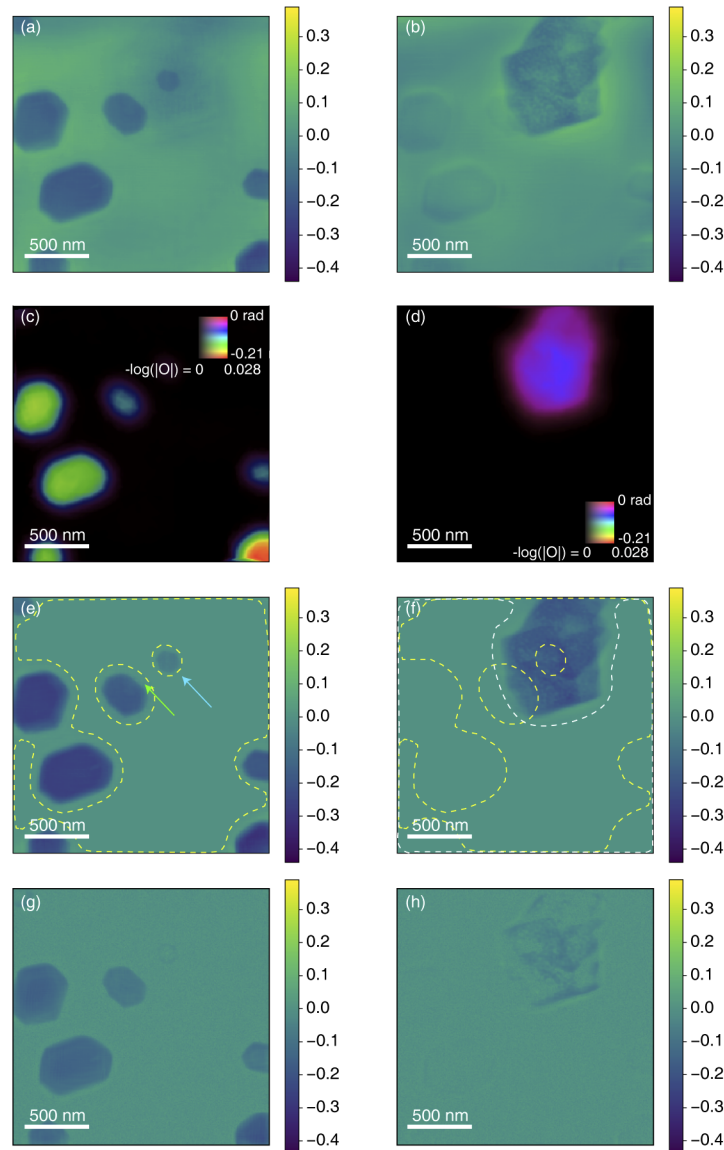
3.5. 2D multislice ptychography

Adorym can also handle a mixture of near-field and far-field propagation in the forward model. While we have shown that automatic differentiation can be used for multislice ptychotomography [14], we show here the simpler case of multislice ptychography of reconstructing several planes at different distances from the probe focus using data collected from a single viewing angle [8,35]. We term it "sparse multislice ptychography" to distinguish it from the multislice ptychotomography in [14] which reconstructs a dense and continuous object. For validation of

our approach, we used a dataset that has already been analyzed using SMP [74]. The specimen is of Au and Ni nanoflakes on two sides of a 10 μm thick silicon window (the silicon window was largely transparent at the 12 keV photon energy used). The ptychography dataset consisted of 1414 128×128 diffraction patterns acquired using a 12 nm FWHM beam focus from a multilayer Laue lens (MLL), with the specimen located 20 μm downstream of the focus (the depth of focus for this optic of 3.9 μm). The sample plane pixel size is 7.3 nm. The object shape in Adorym was $501 \times 500 \times 2$, where the last dimension represents the number of slices. In previous work, the probe function reconstructed from a ptychographic dataset with only a gold particle in the field of view was used as the initial probe function [74]; with Adorym, we used the inverse Fourier transform of the average magnitude from all diffraction patterns to initialize the probe function. While slice spacing refinement was enabled in this reconstruction, the spacing was known sufficiently well from both the previous SMP reconstruction, and from electron microscopy, that no significant adjustment was made from the 10 μm separation specified at reconstruction initiation. The Adorym reconstruction was run for 50 epochs with the probe function fixed, after which it was run for an additional 1950 epochs with probe optimization, all using the Adam optimizer. This led to a reconstructed phase of both slices shown in Fig. 11(a) and (b), which closely match the SMP results reported previously [74].

Both this reconstruction and the previous reconstruction [74] show low-spatial-frequency crosstalk between the two slices, which is a well-known issue [35] in the case where the separation between slices is not much larger than the depth-of-focus. For this particular sample with known differences in material between the two planes, one can use X-ray fluorescence maps acquired simultaneously with the ptychography data to greatly reduce this crosstalk [75]. We employed an approach in Adorym of creating finite support constraints based on the X-ray fluorescence images. We first convolved the X-ray fluorescence images (Au L_α for slice 1, and Ni K_α for slice 2) with a Gaussian with $\sigma = 5$ pixels or about 36.5 nm, and then creating a binary mask by thresholding the XRF maps using a value determined through k-means clustering [76] implemented in the SciPy package [50]. We also used the XRF maps to generate the initial guess for the object function. For this, the XRF images were normalized to their maxima, and multiplied with an approximate phase shifting value estimated from the refractive indices of Au and Ni. The magnitude and phase guesses for both slices are shown in Fig. 11(c) and (d). These guessed slices roughly match the shape of the reconstructed nanoflakes in Fig. 11(a) and (b), and do not contain any crosstalk from the other slice, but they also do not show detail beyond that enabled by the focusing optic, unlike the case of a ptychographic reconstruction. Therefore, a subsequent ptychographic reconstruction of 2000 epochs was performed using Adorym with the addition of finite support constraint masks at each epoch, yielding the crosstalk-free images of Fig. 11(e) and (f) with the boundaries of the respective finite support constraint masks indicated by dashed lines. Additionally, the "gap" region of the finite support in Fig. 11(e), indicated by the green arrow, is uniform and clean. This is not achievable if one simply multiplies the support mask with (a). This reconstruction also reveals a small Au flake on the upstream plane which was previously obscured by larger, higher contrast Ni flake on the downstream plane. Using the XRF-derived initial image guesses alone did not lead to much improvement over the images (a) and (b); it was mainly the finite support constraint that led to the improved results of images (e) and (f).

The loss-function based approach of Adorym allows one to test other reconstruction approaches with ease. We therefore tested out the use of a reweighted l_1-norm regularizer [45] as a Tikhonov regularizer [43] in the reconstruction, without using the finite support constraint. The reweighted l_1-norm seeks to suppress weak pixel values, and can therefore take advantage of the initial guess of the object. The reconstruction results using $\alpha_1 = \alpha_2 = 0.01$ as regularizer weights for planes 1 and 2 respectively are shown in Fig. 11(g) and (h). For both slices, the regularizer is indeed effective in suppressing the crosstalk coming from the other slice, when comparing Figs. 11(a, b) with (g, h). However, on slice 1 the small Au flake pointed to with a the blue arrow in Fig. 11(e)

Fig. 11. Reconstruction results of a multislice ptychography dataset [74] consisting of gold (upstream sample surface; left panels) and nickel (downstream sample surface; right panels) nanoflakes on either side of a 10 μm thick silicon window. The initial retrieved phase images shown in (a) and (b) are from slice 1 and 2, respectively, where features appear to be sharp and clearly defined, yet with crosstalk at low spatial frequencies due to separation by only a small multiple of the 3.9 μm depth of focus of the illuminating optic. To improve the reconstruction, simultaneously-obtained X-ray fluorescence data was used to provide an initial estimate at lower resolution of the magnitude (whose minus-logarithm is coded by brightness) and phase (coded by hue) of each slice, as shown in (c) and (d). With this improved starting guess, as well as a finite support constraint derived from the X-ray fluorescence maps, we obtained improved reconstructions as shown in (e) and (f) with the finite support constraint boundaries shown as dashed lines (yellow for Au, and white for Ni). An alternative approach without finite support is to impose sparsity by incorporating a reweighted l_1-norm regularizer leading to images (g) and (h); in these images, crosstalk is suppressed at the cost of degrading the contrast and sharpness of the reconstructed images.

becomes hardly visible in (g) because its values were overly penalized. On slice 2, the Ni flake was correctly reconstructed in terms of its overall shape, but at lower contrast and a loss of internal detail so that it had a "hollow" appearance. This is likely due to over-regularization, and one may tune the regularizer weights to better balance reconstruction fidelity and crosstalk suppression, but this investigation is beyond the scope of this paper. Besides, the images show increased high-spatial-frequency "salt and pepper" noise. This type of noise comes from the XRF-derived initial guess, which contains a slight scattering-based signal even in empty regions, and the noise level of these pixels are slightly magnified by the reweighted l_1-norm during reconstruction since low-voxel pixels were subject to heavier suppression while high-value pixels could grow in value. This undesirable effect is again likely to be prevented through further tuning of the regularization weights.

The ability of Adorym to explore a wide range of constraints in an automatic-differentiation-based image reconstruction approach is illustrated by the above examples. One can try other regularizers beyond what is demonstrated here, change the forward models, and alter the workflow in a flexible manner.

3.6. *Joint ptychotomography*

Ptychotomography was originally done in a two-step fashion, where ptychography was done to obtain high-resolution 2D projection images at each viewing angle, and those projections were then used in a conventional tomographic reconstruction algorithm to obtain a 3D volume [9]. More recently, a joint ptychotomography approach has been developed where the 3D object is directly reconstructed from the set of diffraction patterns acquired over scanned positions and rotations [12]. This approach can relax the probe position overlap requirements in 2D ptychography at one rotation angle, since there is sufficient information overlap in position and angle to obtain a 3D reconstruction as demonstrated in simulations [12] and in experiments [78].

We show here a joint ptychotomography reconstruction obtained using Adorym to analyze previously-published data [77] from the frozen-hydrated single-cell algae *Chlamydomonas reinhardtii*. This data was acquired using the Bionanoprobe at the Advanced Photon Source at Argonne [79] at 5.5 keV photon energy and a Fresnel zone plate with an approximate Rayleigh resolution of 90 nm. 2D scans were acquired using 50 nm spacing in continuous scan mode, with typically 38,000 diffraction patterns acquired to give a field of view of nearly $(10 \ \mu\text{m})^2$. An Hitachi Vortex ME-4 detector mounted at 90° was used to record the X-ray fluorescence signal from intrinsic elements, and a Dectris Pilatus 100K hybrid pixel array detector recorded the ptychographic diffraction data with the center 256^2 pixels used in the subsequent reconstruction with voxel size of 10.2 nm. 2D images were recorded at a total of 63 angles of a tilt range of -68° to +56°. The original reconstruction in [77] was done following the two-step method, using PtychoLib [80] for 2D ptychographic reconstruction, and GENFIRE [81], a Fourier iterative reconstruction algorithm, for tomographic reconstruction. The projection images used by GENFIRE were twice downsampled so that the pixel size was about 21 nm, and the 3D reconstruction was done using projections in a range of 124°.

Since X-ray fluorescence (XRF) data are available along with the ptychographic diffraction patterns, we can use them to do a preliminary cross-angle alignment before conducting reconstruction. Similar to [77], we estimated the positional offsets at different viewing angles using the P-channel XRF maps which have the best signal-to-noise ratio. Before alignment, we performed a series of quick (3 epochs) 2D ptychography reconstructions for all viewing angles using downsampled ptychographic data, and selected 43 viewing angles from -68° to 56° whose 2D reconstructions exhibited acceptable quality; the data quality for the rest is worse due to beam profile fluctuation. Correspondingly, P maps from these 43 angles were used in our own processing. The pre-alignment was performed using the iterative reprojection method [82] implemented in TomoPy [83]. After acquiring the alignment data, we first did a tomography

reconstruction on the aligned P-channel XRF maps using Adorym in order to examine the consistency with the images shown in the original report [77]. The reconstruction was run for 50 epochs using the CG optimizer. The reconstructed P map projected through the volume is shown at 0° (a) and 90° (b) in Fig. 12(a) (the corresponding images obtained previously are Figs. 2(A) and 4G in [77]). During reconstruction, we allowed Adorym to refine both the cross-angle positional offsets, and the tilt angles of the object along all 3 axes at each nominal specimen tilt about *y*. As the pre-alignment provided by iterative reprojection was already highly accurate, the alignment refinement returned only marginal corrections. On the other hand, the tilt refinement yielded interesting results: as shown in Fig. 12(c), the change of *x*- and *z*-axis tilt corrections are relatively continuous with the nominal viewing angle, instead of showing random fluctuations around 0°. This is the expected outcome when the actual rotation axis is not strictly vertical, or when it precesses continuously during data acquisition.

We then applied the results of tilt refinement and the reprojection-based pre-alignment to the joint ptychotomography reconstruction. Since joint ptychotomography reconstruction is more computationally intensive than tomography, we downsampled the size of diffraction patterns and the object function by 4 times, where all diffraction patterns were cropped to 64×64 (while the original 2D ptychography reconstruction reported in [77] cropped the diffraction patterns to 256×256 but tomography reconstruction in that work downsampled the phase maps by twice), and the object size was set to $256 \times 256 \times 256$ with a voxel size of 42 nm. We used 5 probe modes to account for fly-scan data acquisition; this probe function is commonly used and jointly updated by all specimen tilt angles. Although this may not properly account for beam fluctuation during acquisition, holding onto the same probe further improves the utilization of information coupling among different angles, which is favorable under the condition of angular undersampling [78]. The reconstruction was run for 3 epochs using the Adam optimizer on the "Godzilla" workstation, using the *PyTorch* backend with GPU acceleration. Using the Adam optimizer, the learning rate was set to 1×10^{-6}. The probe function was initialized to be the one retrieved from an individual 2D ptychographic reconstruction, but was constantly optimized during reconstruction to account for the fluctuation of beam profile throughout acquisition. The probe and all refined parameters were held fixed for the first 100 minibatches, and then optimized until the reconstruction finished.

The ptychotomography reconstruction results are shown in Fig. 12(d) and (e). To generate (d), the reconstructed object array was thresholded with voxels below the threshold assigned with zero opacity, so that the rendered 3D volume reveals a sharp spherical outline of the cell, with the internal organelles clearly visible. Next, we rotated the 3D array about the *y*-axis by 30° (so that its orientation matches what is written as a 60° orientation in Fig. 2(D) of [77]). We then show 6 *x*–*y* plane optical slices at indicated *z* positions in this new orientation in Fig. 12(e). In order to estimate the reconstructed spatial resolution, we also plot its power spectra along all the 3 spatial axes in Fig. 12(f). Each power spectrum is expected to be composed of 2 segments: a power-law decline in from low to middle spatial frequencies of $0.008f_{\text{Ny}}$ to $0.31f_{\text{Ny}}$, and a more level section at higher frequencies of $0.47f_{\text{Ny}}$ to $1f_{\text{Ny}}$ corresponding to noise uncorrelated from one pixel to the next (where f_{Ny} is spatial frequency corresponding to the Nyquist sampling limit). The intersection between the two fitting lines represents the point where high-frequency noise starts to dominate, and can be considered as an approximate measure of the resolution. With this, we found the average *xy*-resolution is approximately 70 nm, and the *z*-resolution is approximately 100 nm. The previous reconstruction [77] reports the resolutions to be 45 and 55 nm, respectively; however, Given that we downsampled the data twice more in our case, the joint ptychotomography capability of Adorym is able to provide reasonably good reconstruction quality compared to the more computationally intensive iterative method of image reconstruction steps followed by reprojection alignment steps.
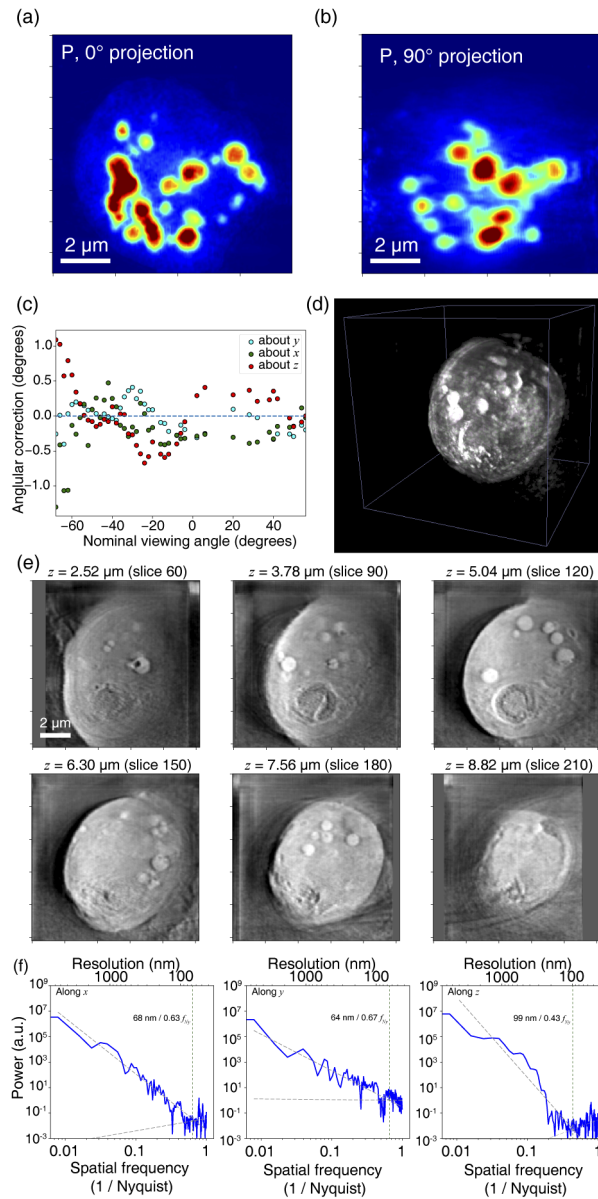
Fig. 12. Adorym reconstruction of a tomographic X-ray fluorescence and ptychography dataset of a frozen hydrated alga cell *Chlamydomonas reinhardtii* acquired using 5.5 keV X rays [77]. The phosphorus *K* fluorescence projection (a) at $0°$ through the 3D volume highlights polyphosphate bodies, which are also seen at somewhat lower resolution in the $90°$ projection through the reconstruction due to the limited tomographic tilt range of $-68°$ to $+56°$. As in a previously-reported reconstruction of this dataset [77], this phosphorus reconstruction used refinement of the tomographic tilt angles, leading to angular corrections about all three axes as shown in (c). These refined angles were then employed in a subsequent ptychotomography reconstruction, yielding a 3D volume rendered in (d). Sub-images in (e) show optical sections from the reconstructed volume cut at indicated *z*-positions after it is rotated about the *y*-axis by $30°$. In (f), the power spectra of the reconstructions in the *x*-, *y*-, and *z*-direction are shown. For each case, two lines with the same color are fitted respectively using datapoints with spatial frequency in the range of $0.008–0.31 f_{Ny}$ and $0.47–1.0 f_{Ny}$. As the latter fits the noise plateau, the intersection between both lines (marked by dotted vertical lines) provides a measure of spatial resolution.
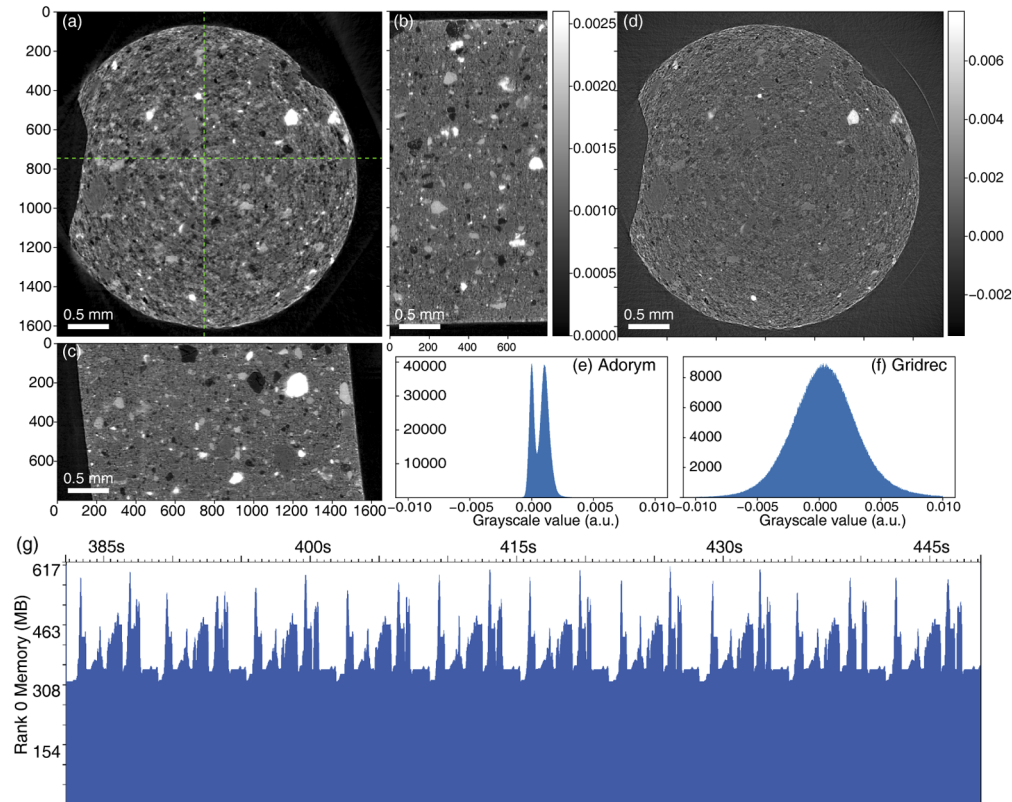
3.7. Conventional tomography on HPC

Adorym's optimization framework can also work with conventional tomography. When using the LSQ loss function and GD optimizer, Adorym's tomography reconstruction is equivalent to the basic algebraic reconstruction technique (ART) [1], but more advanced optimizers like Adam can be used to accelerate the reconstruction.

The tomographic data to be reconstructed was an activated charcoal dataset [84]. The original data were collected at the APS 32-ID beamline using a beam energy of 25 keV, filtered using a crystal monochromator, and the images were recorded by a CMOS camera (1920 × 1200 GS3-U3-23S6M-C) after the X-ray beam was converted into visible light by a LuAG:Ce scintillator. This resulted in a sample-plane pixel size of 600 nm. Since the size of the pellet (about 4 mm in diameter) was larger than the detector's field of view, full rotation sets were taken with the beam located at different offsets from the center of rotation in a *Tomosaic* approach [84]. Each assembled projection image was 4204 × 6612 pixels in size. For this demonstration, we used 900 projections evenly distributed between 0 and 180°, downsampled the projections by 4 times in both *x* and *y*, and then selected the first 800 lines after downsampling for each projection angle, so that the size of each projection image is 800 × 1653 pixels with an angular sampling that is coarser than required by the Crowther criterion [85]. Before reconstruction, all projection images were divided into tiles that are 25 × 52 in size, so each viewing angle contained a tile grid of 32 × 32, or 1024 tiles in total.

Tomographic reconstruction was performed on the supercomputer "Theta" (Sec. 3.1). We used 256 nodes with 4 MPI ranks per node, so that the total number of ranks exactly matched the number of tiles per angle. We used Adorym's distributed object (DO) scheme to parallelized the reconstruction; in this case, each of first 800 MPI ranks saved one slice of the object, and the corresponding gradient, with a horizontal size 1653 × 1653. The remaining 224 ranks did not serve as containers for object or gradient slabs, but were still responsible for calculating the gradients of their assigned tiles. Since each tile contained 25 pixels in the *y* direction, assembling an object chunk on each rank required the incoming transfer of 25 ranks (including itself); this rank, at the same time, also sent parts of its own object slab to 32 ranks, which is the number of tiles horizontally across the object. Therefore the array size that a rank sent to another rank is $1 \times 52 \times 1653 \times 2$, where the last dimension is for both $\delta(\mathbf{r})$ and $\beta(\mathbf{r})$ (although only $\beta(\mathbf{r})$ is reconstructed in conventional absorption tomography). With this configuration, the `AlltoAll` MPI communication for object chunk assembly took 0.7 s on average. The gradient synchronization, which is essentially the reverse process and requires each rank to send a $25 \times 52 \times 1653 \times 2$ gradient array to 25 ranks, took an average of 0.5 s. Using a LSQ loss function and the Adam optimizer, gradient calculation and object update took about 0.7 s and 0.3 s on average. Along with rotation (1 s for both object rotation and gradient rotation) and other overheads, each batch (which involved 1024 tiles or exactly one viewing angle) took roughly 5.8 s to process, excluding the time spent for occasionally saving checkpoints for the object and other parameters.

After 3 epochs of reconstruction, we obtained the 3D volume whose orthogonal slices are shown in Fig. 13, where sub-image (a) displays the *xz*-cross section cut from the vertical center of the volume (slice 400 out of the 800 slices). Figures 13(b) and (c) respectively show the *xy*- and *yz*-cross sections cut at the positions indicated by the dashed lines in (a). Since Crowther criterion [85] requires $1653\pi = 5193$ projection angles for complete coverage in reciprocal space, while we used only 900 angles for this reconstruction, it is to be expected that our ART-like approach would outperform filtered backprojection methods such as *Gridrec* [86–88], with a *Gridrec* reconstructed middle slice shown in Fig. 13(d). Comparing Fig. 13(a) and (d), we observed that the result of Adorym shows a much better contrast than *Gridrec*. The difference is made clearer when we compare the histograms of the reconstructed slice. The histogram of optical densities in the Adorym's reconstruction shown in Fig. 13(e) exhibits 2 distinct peaks,

which correspond to the vacuum background surrounding the charcoal pellet and the pellet itself. The background peak on the left is much sharper than the charcoal peak, indicating that the background is uniformly reconstructed and numerically well separated from the non-vacuum features. The histogram of the *Gridrec* result shown in Fig. 13(f), on the other hand, exhibits only one broad Gaussian peak, meaning that the background and the sample features are somewhat intermingled.

Fig. 13. Reconstruction of conventional projection tomography data from an activated charcoal pellet [84]. In (a-c), we show slices from the 3D volume reconstructed by Adorym: (a) shows the horizontal cross section cut at slice 400, and (b, c) show the cross sections cut from locations indicated by the green dashed lines. In (d), we show the same slice as in (a) but reconstructed using conventional filtered backprojection (FBP), exhibiting much lower contrast. The histogram of the Adorym reconstruction slice (a), and the filtered backprojection slice (d), are shown in (e) and (f) respectively; they indicate that Adorym better represents the density differences between different features in the sample. In (g), we show the memory profile during 10 minibatches of Adorym reconstruction.

We utilized the Intel VTune^{TM} profiler available on "Theta" to obtain the memory profile of Adorym during runtime. Figure 13(g) shows the memory consumption of rank 0 through a 66-second window of the reconstruction process, where 10 minibatches were computed sequentially (the average per-batch walltime was slightly longer than 5.8 s due to the added overhead of VTune itself). The profile shows obvious periodicity, with each period corresponding to a minibatch. The two highest peaks of around 650 MB in each minibatch are caused by rotating the object slice and the gradient slice, respectively, as both rotation operations involve considerable intermediate memory usage (contributed by loading the transformation coordinates, performing

bilinear interpolation, and so on). The other peaks result from gradient calculations and the update step of Adam, which involves several intermediate variables that have the same size as the object array. Beyond that, the baseline memory is jointly comprised of the stored object array, gradient array, Adam parameter arrays, and auxiliary data including the tile allocation over all ranks. The overall memory usage is steady, and no memory leakage is found. If data parallelism was used, each rank would then require at least 70 GB of memory to store the whole object function and the associated gradient and optimizer parameters, not to mention the additional usage caused by intermediate and auxiliary variables. The reconstruction times are included in Table 1.

4. Discussion

We have shown that Adorym provides a reconstruction framework for several imaging methods, including multi-distance near-field holography (MDH), sparse multislice ptychography (SMP) and 2D ptychography, and projection as well as ptychographic tomography. This framework allows for the refinement of imperfectly-known experimental parameters, resulting in substantial improvements in reconstruction quality. The forward model class `ForwardModel` allows for the treatment of ptychotomography with multiple viewing angles and multiple images per angle; a simpler variant allows for simple projection tomography with a per-angle tile number of 1 and a single illumination mode. The ptychotomography forward model is compatible with many other imaging techniques, like near-field 2D holography, but even if a technique does not fit in that model, adapting Adorym to the new technique typically only requires minor changes. Among the techniques demonstrated in Section 3, the most "different" pair might be 3D ptychotomography and MDH, but the core modifications (which are mainly on incorporating multiple propagation distances and adding "buffer zones" for preventing wrapping of diffraction fringes, and do not include optional features like tiling, affine transformation, distributed processing *etc.*) take only about 30 lines of code. The code can be visited on the GitHub repository noted in Section 1.

This flexibility is made possible through the use of automatic differentiation (AD) for gradient derivation. While AD drastically reduces the development cost, does it mean that the runtime speed has to be sacrificed as a trade-off? There are several reasons to think this might be the case:

- Some AD libraries (like *PyTorch* and *Autograd*) create dedicated data types for node variables, so that each of them knows the operations that connect it with other nodes, as well as the list of nodes that it is connected to. This genre of AD is known as "operator overloading" [24], and is favorable for its flexibility and ease of implementation, but logging these interconnections at runtime means that each operation causes a slight additional overhead [89].

- When a gradient-based algorithm is manually derived, the expression of the gradient can be often simplified to reduced the number of numerical evaluations needed. In contrast, it is difficult for AD to fully take this advantage because each step in backpropagation is local to a primitive function. An illustrative example is a scalar that contains $x \log(x)$: the derivative with regards to x is $x(1/x) + \log(x)$, so that in a manual differentiation one would simplify the first term to 1 while AD would mindlessly follow the backpropagation procedure and evaluate both x and $1/x$. Nevertheless, for many important problems this loss in performance is small, and one can define custom functions in many AD libraries to explicitly inform AD of the derivatives of these special composite functions.

- AD works under a gradient-based optimization strategy, which is not the only strategy to solve inverse problems. In some cases, it may be possible to exploit the specific structure of an inverse problem to provide a more efficient solution. For example, alternating projection algorithms like hybrid input-output (HIO) [40] and difference map [90] can

potentially deal better with the non-convexity of certain problems, while gradient-based methods including those based on AD can likely encounter the issue of being stuck in local minima. Nevertheless, AD as a general tool is not exclusive with these ostensibly non-gradient strategies. Some methods like the HIO can in fact be reformulated as a minimization/maximization problem and solved using AD [91]. Also, alternating projection methods, or variable splitting methods like ADMM [92] and FISTA [93] could also involve projection operators or subproblems that cannot be easily solved with a closed-form expression. For example, to solve the regularization subproblem within a splitting algorithm, one can often use the proximal operator [65], which is formulated as a minimization problem and can be solved using a gradient-based approach. AD can extend its application to these strategies, although such strategies are not yet implemented in Adorym.

However, the runtime penalties are not necessarily unacceptably high. In the case of multislice X-ray ptychotomography of a 256^3 voxel simulated object that extends beyond the depth-of-focus limit, a C code with manually derived gradients took 6.48 core hours per illumination angle to arrive at a solution [13] using the HPC system "Blues" (Sec. 3.1), while a Python code using *TensorFlow* for the AD implementation [14] took 8.25 core hours for a similar reconstruction step using the CPUs only on the HPC system "Cooley" (Sec. 3.1). This may be in part due to AD's "cheap derivative principle" [21], where the cost of evaluating the gradient with reverse-mode AD is generally bound to small multiple of the cost needed to evaluate the forward model. Moreover, well-developed AD libraries can incorporate low-level optimizations, which can help them relative to basic manual differentiation implementations. For example, when working with GPUs, *PyTorch* uses the CUDA stream mechanism to queue on-device instructions to the GPU's command buffer, so that tensor operations on GPU can be concurrent with Python interpretation on CPU [48]. The most effective way to accelerate image reconstruction computations is to increase the degree of parallelization and improve interprocess communication – a principle that applies to both AD and manually coded implementations. Finally, it is known that interpreted languages like Python and MATLAB might have inferior execution speeds compared to compiled languages, but the advent of just-in-time (JIT) compilers [94] is likely to change the situation. We are aware of a few AD libraries which are already using JIT, such as JAX [95]; meanwhile, efforts on incorporating JIT support into *PyTorch* are also ongoing [89], which Adorym can gain from when it becomes available.

The memory consumption of AD is another concern, though this can be offset in part by the use of well-developed AD libraries. In some cases, one might have similar storage requirements for a manual differentiation implementation. For example, in multislice imaging where the wavefield at the *j*-th slice, $\psi_j(\boldsymbol{r}_{x,y})$, is modulated by this object slice $O_j(\boldsymbol{r}_{x,y})$, the modulated wavefield is given by $\psi_{j+1}(\boldsymbol{r}_{x,y}) = \psi_j(\boldsymbol{r}_{x,y})O_j(\boldsymbol{r}_{x,y})$. The derivative of $\psi_{j+1}(\boldsymbol{r}_{x,y})$ with regards to $O_j(\boldsymbol{r}_{x,y})$ is given by $\psi_j^*(\boldsymbol{r}_{x,y})$. That is, to update the object slice $O_j(\boldsymbol{r}_{x,y})$, one needs to know $\psi_j(\boldsymbol{r}_{x,y})$ which is calculated in the forward pass. Both AD and manual differentiation may inevitably choose to store $\psi_j(\boldsymbol{r}_{x,y})$ during the forward pass and reuse it for gradient calculation, especially when recalculating $\psi_j(\boldsymbol{r}_{x,y})$ in the differentiation stage is too costly. Most AD libraries do not blindly store all intermediate variables, but only those needed to be reused for differentiating a non-linear operation, or a linear operation with a variable operator. The advanced memory management strategies in well-developed AD libraries also help. For example, *PyTorch* features a reference counting scheme which deallocates the memory of variables immediately when their usage counts down to zero, preventing the extra memory consumption in the traditional garbage collection scheme [48]. Additionally, while in manual differentiation one can freely adjust the workflow of the program, it is also possible in AD to not store some intermediate variables, but recalculate them in the backward pass. This strategy, known as checkpointing [96], leads to a tradeoff between memory consumption and computation time. Manually scheduled

checkpointing is available with *PyTorch* [97] and can be conveniently added into the forward model if desirable; on the other hand, automatic scheduling of checkpointing has also been proposed and implemented [98], which could potentially reach a better balance between storage and recomputation.

AD has a profound value when developing new methods, as making improvements on existing forward models or devising new forward models are made much less laborious without needing to hand-derive gradients. Beyond the parameter refinement capabilities demonstrated in Section 3, one could also consider adding the following example capabilities:

- Optimizing structured illumination function for ptychographic or holographic imaging. Structured illumination is often used as a way to improve the resolution and reconstruction robustness of these techniques by enhancing the contrast transfer function of the imaging system, and choosing the optimal illumination function for a certain imaging technique or sample condition is usually of interest [6,99,100]. With AD, one possible way to get an optimal illumination function would be to add the structured modulator into the forward model as an optimizable variable, and set the loss function to measure the disparity between the predicted intensity and the desired far-field intensity distribution.

- Speckle tracking. For imaging techniques like point-projection microscopy, where X-ray beams are focused using optics such as multi-layer Laue lenses, it is often important to accurately retrieve the illuminating wavefield in order to reconstruct the phase images with good quality. When the reconstruction algorithm is based on the generalized Fresnel-scaling theorem [101], the forward model involves resampling of the measured intensity following a function without a closed-form expression. While an optimization-based speckle tracking algorithm applicable in this case has been proposed [101], AD can potentially address the problem using a similar strategy, but without the need of hand-deriving and implementing the differentiation through the resampling operation. One could instead use the `grid_sample` function in *PyTorch* for a well-optimized implementation.

- In-situ 3D imaging. In many cases, it is of interest to track the structural evolution over time inside a 3D sample. This means that conventional tomography reconstruction algorithms must be modified to account for the continuous deformation of the sample during acquisition. One way to address this issue is to represent the time-space dependence of the object function as the linear superposition of a series of spatial basis functions, where the superposition coefficients depends only on time [102]. With AD, this strategy can be conveniently implemented by adding the time-dependent coefficient as a refinable variable. Alternatively, AD may also solve the time-dependency by refining a vector field deformer that dictates the distortion of the object function [103]. This approach involves grid-based resampling, but as mentioned above, AD libraries like *PyTorch* provide easy-to-use interfaces for this operation. This application is very closely related to the optical flow problem in computer vision, where AD has been demonstrated to provide good performance [104].

In Adorym, these additional capabilities can be achieved by modifying or creating new forward models.

In addition to extending towards more forward models, another field of future improvement for Adorym or any optimization-based reconstruction methods is the underlying optimization algorithm. First-order methods like Adam and CG are fast and robust in many cases, but they are not advantageous compared to second-order methods when the loss is a well structured convex function. Second order algorithms based on Gauss-Newton approximation provide a relatively easy and low-cost way to utilize the curvature of the loss function, with the Curveball algorithm [105] being a particularly light-weight variant. As Gauss-Newton methods approximate the

Hessian matrix using the "internal" Jacobian of the forward model (*i.e.*, the derivatives of the prediction function), the power of AD can be again exploited in this case to calculate the desired vector-Jacobian products and Jacobian-vector products without manual derivation. Finally, since the capability of AD is often part of common machine learning tools like *PyTorch*, it is also easy to seamlessly combine a physical model and a neural network into one pipeline and optimize the parameters in both parts. This has already been demonstrated in full-field coherent diffraction imaging [106].

AD toolkits are often built and optimized by operators of supercomputer systems, allowing an Adorym user to piggyback on those efforts. Thus the DO parallelization scheme described in Sec. 2.5.1 can provide a well-supported approach to reconstructing gigavoxel datasets on HPCs.

5. Conclusion

We have developed Adorym as a generic image reconstruction framework that utilizes the power of automatic differentiation. It works with a variety of imaging techniques, ranging from near-field 2D holography to far-field 3D ptychographic tomography. It is able to refine a series of experimental parameters along reconstructing the object. It has a modular architecture so that it can be extended to cover new forward models or refinable parameters without lengthy manual derivations of gradients. It can be operated on platforms ranging from workstations to HPCs, and comes with several parallelization schemes that provide different balances between computational overhead and memory usage. Future efforts will be aimed at improving speed and memory consumption. Adding JIT support, and incorporating "smarter" checkpointing strategies, are both potential approaches towards this objective.

Disclosures. The authors declare no conflicts of interest.

Supplemental document. See Supplement 1 for supporting content.

References

1. A. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging* (IEEE, 1988).
2. P. Godard, M. Allain, V. Chamard, and J. M. Rodenburg, "Noise models for low counting rate coherent diffraction imaging," Opt. Express **20**(23), 25914–25934 (2012).
3. J. C. Zingarelli and S. C. Cain, "Phase retrieval and Zernike decomposition using measured intensity data and the estimated electric field," Appl. Opt. **52**(31), 7435–7444 (2013).
4. W. Hoppe, "Beugung im Inhomogenen Primärstrahlwellenfeld. I. Prinzip einer Phasenmessung," Acta Cryst A **25**(4), 495–501 (1969).
5. J. M. Rodenburg and H. M. Faulkner, "A phase retrieval algorithm for shifting illumination," Appl. Phys. Lett. **85**(20), 4795–4797 (2004).
6. M. Stockmar, P. Cloetens, I. Zanette, B. Enders, M. Dierolf, F. Pfeiffer, and P. Thibault, "Near-field ptychography: phase retrieval for inline holography using a structured illumination," Sci. Rep. **3**(1), 1927 (2013).
7. P. Cloetens, W. Ludwig, J. Baruchel, J. Baruchel, D. Van Dyck, J. Van Landuyt, J. P. Guigay, and M. Schlenker, "Holotomography: Quantitative phase tomography with micrometer resolution using hard synchrotron radiation x rays," Appl. Phys. Lett. **75**(19), 2912–2914 (1999).
8. 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).
9. M. Dierolf, A. Menzel, P. Thibault, P. Schneider, C. M. Kewish, R. Wepf, O. Bunk, and F. Pfeiffer, "Ptychographic x-ray computed tomography at the nanoscale," Nature **467**(7314), 436–439 (2010).
10. P. Li and A. Maiden, "Multi-slice ptychographic tomography," Sci. Rep. **8**(1), 2049 (2018).
11. M. Guizar-Sicairos and J. R. Fienup, "Phase retrieval with transverse translation diversity: a nonlinear optimization approach," Opt. Express **16**(10), 7264–7278 (2008).
12. D. Gürsoy, "Direct coupling of tomography and ptychography," Opt. Lett. **42**(16), 3169–3172 (2017).
13. 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), 1078–1085 (2018).

14. 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).

15. A. Neumaier, "Solving ill-conditioned and singular linear systems: A tutorial on regularization," SIAM Rev. **40**(3), 636–666 (1998).

16. R. Tibshirani, "Regression shrinkage and selection via the lasso," J. Royal Stat. Soc. B **58**, 267–288 (1996).

17. A. M. Maiden, M. J. Humphry, M. C. Sarahan, B. Kraus, and J. M. Rodenburg, "An annealing algorithm to correct positioning errors in ptychography," Ultramicroscopy **120**, 64–72 (2012).

18. F. Zhang, I. Peterson, J. Vila-Comamala, A. Diaz, F. Berenguer, R. Bean, B. Chen, A. Menzel, I. K. Robinson, and J. M. Rodenburg, "Translation position determination in ptychographic coherent diffraction imaging," Opt. Express **21**(11), 13592–13606 (2013).

19. A. Tripathi, I. McNulty, and O. G. Shpyrko, "Ptychographic overlap constraint errors and the limits of their numerical recovery using conjugate gradient descent methods," Opt. Express **22**(2), 1452–1466 (2014).

20. J. S. Lim, *Two-Dimensional Signal and Image Processing* (Prentice-Hall, 1990).

21. A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (Society for Industrial and Applied Mathematics, 2008), 2nd ed.

22. A. S. Jurling and J. R. Fienup, "Applications of algorithmic differentiation to phase retrieval algorithms," J. Opt. Soc. Am. A **31**(7), 1348–1359 (2014).

23. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT, 2017).

24. Y. S. G. Nashed, T. Peterka, J. Deng, and C. Jacobsen, "Distributed automatic differentiation for ptychography," Procedia Comput. Sci. **108**, 404–414 (2017).

25. S. Kandel, S. Maddali, M. Allain, S. O. Hruszkewycz, C. Jacobsen, and Y. S. G. Nashed, "Using automatic differentiation as a general framework for ptychographic reconstruction," Opt. Express **27**(13), 18653–18672 (2019).

26. M. Du, D. Gürsoy, and C. Jacobsen, "Near, far, wherever you are: simulations on the dose efficiency of holographic and ptychographic coherent imaging," J. Appl. Crystallogr. **53**(3), 748–759 (2020).

27. Message Passing Interface Forum, "A message-passing interface standard, version 3.1," (2015). https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.

28. E. P. Xing, Q. Ho, P. Xie, and D. Wei, "Strategies and principles of distributed machine learning on big data," Eng. **2**(2), 179–195 (2016).

29. The HDF Group, "Hierarchical Data Format, version 5," http://www.hdfgroup.org/HDF5/.

30. D. Blinder and T. Shimobaba, "Efficient algorithms for the accurate propagation of extreme-resolution holograms," Opt. Express **27**(21), 29905–29915 (2019).

31. S. Ali, M. Du, M. Adams, B. Smith, and C. Jacobsen, "Comparison of distributed memory algorithms for x-ray wave propagation in inhomogeneous media," Opt. Express **28**(20), 29590–29618 (2020).

32. 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).

33. K. Ishizuka and N. Uyeda, "A new theoretical and practical approach to the multislice method," Acta Cryst A **33**(5), 740–749 (1977).

34. 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).

35. 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).

36. P. Thibault and A. Menzel, "Reconstructing state mixtures from diffraction measurements," Nature **494**(7435), 68–71 (2013).

37. P. M. Pelz, M. Guizar-Sicairos, P. Thibault, I. Johnson, M. Holler, and A. Menzel, "On-the-fly scans for x-ray ptychography," Appl. Phys. Lett. **105**(25), 251101 (2014).

38. J. Deng, Y. S. G. Nashed, S. Chen, N. W. Phillips, T. Peterka, R. Ross, S. Vogt, C. Jacobsen, and D. J. Vine, "Continuous motion scan ptychography: characterization for increased speed in coherent x-ray imaging," Opt. Express **23**(5), 5438–5451 (2015).

39. X. Huang, K. Lauer, J. N. Clark, W. Xu, E. Nazaretski, R. Harder, I. K. Robinson, and Y. S. Chu, "Fly-scan ptychography," Sci. Rep. **5**(1), 9074 (2015).

40. J. R. Fienup, "Reconstruction of an object from the modulus of its Fourier transform," Opt. Lett. **3**(1), 27–29 (1978).

41. J. Miao, P. Charalambous, J. Kirz, and D. Sayre, "An extension of the methods of x-ray crystallography to allow imaging of micron-size non-crystalline specimens," Nature **400**(6742), 342–344 (1999).

42. S. Marchesini, H. He, H. Chapman, S. Hau-Riege, A. Noy, M. Howells, U. Weierstall, and J. Spence, "X-ray image reconstruction from a diffraction pattern alone," Phys. Rev. B **68**(14), 140101 (2003).

43. A. N. Tikhonov, "On the solution of ill-posed problems and the method of regularization," Doklady Akademii Nauk SSSR **151**, 501–504 (1963).

44. M. T. McCann and M. Unser, *Biomedical Image Reconstruction: From the Foundations to Deep Neural Networks* (IEEE, 2019).

45. E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," J. Fourier Anal. Appl. **14**(5-6), 877–905 (2008).

46. Markus Grasmair and Frank Lenzen, "Anisotropic total variation filtering," Appl. Math. Optim. **62**(3), 323–339 (2010).
47. "Autograd tutorial," (2020). https://github.com/HIPS/autograd/blob/master/docs/tutorial.md.
48. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds. (Curran Associates, 2019), pp. 8024–8035.
49. S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," Comput. Sci. Eng. **13**(2), 22–30 (2011).
50. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nat. Methods **17**(3), 261–272 (2020).
51. E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, and Y. Wang, *High-Performance Computing on the Intel Xeon Phi* (Springer International Publishing, 2014).
52. M. Looks, M. Herreshoff, D. Hutchins, and P. Norvig, "Deep learning with dynamic computation graphs," arXiv, arXiv:1702.02181 (2017).
53. N. Qian, "On the momentum term in gradient descent learning algorithms," Neural Netw. **12**(1), 145–151 (1999).
54. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv, arXiv:1412.6980 (2014).
55. B. Polyak, "The conjugate gradient method in extremal problems," USSR Comput. Math. Math. Phys. **9**(4), 94–112 (1969).
56. B. Behzad, J. Huchette, H. V. T. Luu, R. Aydt, S. Byna, Y. Yao, Q. Koziol, and Prabhat, "A framework for auto-tuning HDF5 applications," in *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing*, (Association for Computing Machinery, 2013), pp. 127–128.
57. M. Howison, Q. Koziol, D. Knaak, J. Mainzer, and J. Shalf, "Tuning HDF5 for Lustre file systems," https://digital.library.unt.edu/ark:/67531/metadc837191/.
58. 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).
59. S. Parker, V. Morozov, S. Chunduri, K. Harms, C. Knight, and K. Kumaran, "Early evaluation of the Cray XC40 Xeon Phi system "theta" at Argonne," in *2017 Cray Users Group meeting*, (Cray, 2017).
60. M. Krenkel, M. Toepperwien, F. Alves, and T. Salditt, "Three-dimensional single-cell imaging with x-ray waveguides in the holographic regime," Acta Cryst A **73**(4), 282–292 (2017).
61. S. Zabler, P. Cloetens, J. P. Guigay, J. Baruchel, and M. Schlenker, "Optimization of phase contrast imaging using hard x rays," Rev. Sci. Instrum. **76**(7), 073705 (2005).
62. L. D. Turner, B. B. Dhal, J. P. Hayes, A. P. Mancuso, K. A. Nugent, D. Paterson, R. E. Scholten, C. Q. Tran, and A. G. Peele, "X-ray phase imaging: Demonstration of extended conditions with homogeneous objects," Opt. Express **12**(13), 2960–2965 (2004).
63. S. Maretzke, M. Bartels, M. Krenkel, T. Salditt, and T. Hohage, "Regularized Newton methods for x-ray phase contrast and general imaging problems," Opt. Express **24**(6), 6490–6506 (2016).
64. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Trans. on Image Process. **13**(4), 600–612 (2004).
65. 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).
66. Z. Su, V. D. Andrade, S. Cretu, Y. Yin, M. J. Wojcik, A. A. Franco, and A. Demortière, "X-ray nanocomputed tomography in Zernike phase contrast for studying 3D morphology of Li–O$_2$ battery electrode," ACS Appl. Energy Mater. **3**(5), 4093–4102 (2020).
67. D. Paganin, *Coherent X-ray Optics* (Oxford University, 2006).
68. H. M. L. Faulkner and J. Rodenburg, "Movable aperture lensless transmission microscopy: A novel phase retrieval algorithm," Phys. Rev. Lett. **93**(2), 023903 (2004).
69. J. Rodenburg, A. Hurst, A. Cullis, B. Dobson, F. Pfeiffer, O. Bunk, C. David, K. Jefimovs, and I. Johnson, "Hard-x-ray lensless imaging of extended objects," Phys. Rev. Lett. **98**(3), 034801 (2007).
70. M. Odstrčil, A. Menzel, and M. Guizar-Sicairos, "Iterative least-squares solver for generalized maximum-likelihood ptychography," Opt. Express **26**(3), 3108–3123 (2018).
71. P. Dwivedi, S. Konijnenberg, S. Pereira, and P. Urbach, "Lateral position correction in ptychography using the gradient of intensity patterns," Ultramicroscopy **192**, 29–36 (2018).
72. J. N. Clark, X. Huang, R. J. Harder, and I. K. Robinson, "A continuous scanning mode for ptychography," Opt. Lett. **39**(20), 6066–6069 (2014).

73. J. Deng, Y. Yao, Y. Jiang, S. Chen, J. A. Klug, M. Wojcik, F. S. Marin, C. Preissner, C. Roehrig, Z. Cai, S. Vogt, and B. Lai, "Instrumentation and method developments of x-ray ptychography at the Advanced Photon Source," Proc. SPIE **11112**, 111120E (2019).

74. H. Öztürk, H. Yan, Y. He, M. Ge, Z. Dong, M. Lin, E. Nazaretski, I. K. Robinson, Y. S. Chu, and X. Huang, "Multi-slice ptychography with large numerical aperture multilayer Laue lenses," Optica **5**(5), 601–607 (2018).

75. X. Huang, H. Yan, Y. He, M. Ge, H. Öztürk, Y.-L. L. Fang, S. Ha, M. Lin, M. Lu, E. Nazaretski, I. K. Robinson, and Y. S. Chu, "Resolving 500 nm axial separation by multi-slice x-ray ptychography," Acta Crystallogr. Sect. A **75**(2), 336–341 (2019).

76. N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using k-means clustering algorithm and subtractive clustering algorithm," Procedia Comput. Sci. **54**, 764–771 (2015).

77. J. Deng, Y. H. Lo, M. Gallagher-Jones, S. Chen, A. Pryor, Q. Jin, Y. P. Hong, Y. S. G. Nashed, S. Vogt, J. Miao, and C. Jacobsen, "Correlative 3D x-ray fluorescence and ptychographic tomography of frozen-hydrated green algae," Sci. Adv. **4**(11), eaau4548 (2018).

78. M. Kahnt, J. Becher, D. Brückner, Y. Fam, T. Sheppard, T. Weissenberger, F. Wittwer, J.-D. Grunwaldt, W. Schwieger, and C. G. Schroer, "Coupled ptychography and tomography algorithm improves reconstruction of experimental data," Optica **6**(10), 1282–1289 (2019).

79. S. Chen, J. Deng, Y. Yuan, C. Flachenecker, R. Mak, B. Hornberger, Q. Jin, D. Shu, B. Lai, J. Maser, C. Roehrig, T. Paunesku, S.-C. Gleber, D. J. Vine, L. Finney, J. Von Osinski, M. Bolbat, I. Spink, Z. Chen, J. Steele, D. Trapp, J. Irwin, M. Feser, E. Snyder, K. Brister, C. Jacobsen, G. Woloschak, and S. Vogt, "The Bionanoprobe: hard x-ray fluorescence nanoprobe with cryogenic capabilities," J. Synchrotron Radiat. **21**(1), 66–75 (2014).

80. Y. S. Nashed, D. J. Vine, T. Peterka, J. Deng, R. Ross, and C. Jacobsen, "Parallel ptychographic reconstruction," Opt. Express **22**(26), 32082–32097 (2014).

81. A. Pryor, Y. Yang, A. Rana, M. Gallagher-Jones, J. Zhou, Y. H. Lo, G. Melinte, W. Chiu, J. A. Rodríguez, and J. Miao, "GENFIRE: A generalized Fourier iterative reconstruction algorithm for high-resolution 3d imaging," Sci. Rep. **7**(1), 10409 (2017).

82. D. Gürsoy, Y. P. Hong, K. He, K. Hujsak, S. Yoo, S. Chen, Y. Li, M. Ge, L. M. Miller, Y. S. Chu, V. De Andrade, K. He, O. Cossairt, A. K. Katsaggelos, and C. Jacobsen, "Rapid alignment of nanotomography data using joint iterative reconstruction and reprojection," Sci. Rep. **7**(1), 11818 (2017).

83. D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "TomoPy: a framework for the analysis of synchrotron tomographic data," J. Synchrotron Radiat. **21**(5), 1188–1193 (2014).

84. R. 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 Radiat. **25**(5), 1478–1489 (2018).

85. R. A. Crowther, D. J. DeRosier, and A. Klug, "The reconstruction of a three-dimensional structure from projections and its application to electron microscopy," Proc. Royal Soc. Lond. A **317**(1530), 319–340 (1970).

86. J. D. O'Sullivan, "A fast sinc function gridding algorithm for Fourier inversion in computer tomography," IEEE Trans. Med. Imaging **4**(4), 200–207 (1985).

87. B. A. Dowd, G. H. Campbell, R. B. Marr, V. Nagarkar, S. Tipnis, L. Axe, and D. P. Siddons, "Developments is synchrotron x-ray computed microtomography at the National Synchrotron Light Source," Proc. SPIE **3772**, 224–236 (1999).

88. F. Marone and M. Stampanoni, "Regridding reconstruction algorithm for real-time tomographic imaging," J. Synchrotron Radiat. **19**(6), 1029–1037 (2012).

89. B. van Merriënboer, O. Breuleux, A. Bergeron, and P. Lamblin, "Automatic differentiation in ML: Where we are and where we should be going," arXiv, arXiv:1810.11530 (2018).

90. V. Elser, "Phase retrieval by iterated projections," J. Opt. Soc. Am. A **20**(1), 40–55 (2003).

91. S. Marchesini, "A unified evaluation of iterative projection algorithms for phase retrieval," Rev. Sci. Instrum. **78**(1), 011301 (2007).

92. S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," FNT in Machine Learning **3**(1), 1–122 (2010).

93. A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," SIAM J. Imaging Sci. **2**(1), 183–202 (2009).

94. S. K. Lam, A. Pitrou, and S. Seibert, "Numba: a LLVM-based Python JIT compiler," in *LLVM 2015: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, (Association for Computing Machinery, 2015), pp. 1–6.

95. S. S. Schoenholz and E. D. Cubuk, "Jax, m.d.: End-to-end differentiable, hardware accelerated, molecular dynamics in pure python," arXiv, arXiv:1912.04232 (2019).

96. C. C. Margossian, "A review of automatic differentiation and its efficient implementation," arXiv, arXiv:1811.05031 (2018).

97. "PyTorch documentation," (PyTorch, 2020). https://pytorch.org/docs.

98. A. Griewank and A. Walther, "Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation," ACM Trans. Math. Softw. **26**(1), 19–45 (2000).

99. M. Odstrčil, M. Lebugle, M. Guizar-Sicairos, C. David, and M. Holler, "Towards optimized illumination for high-resolution ptychography," Opt. Express **27**(10), 14981–14997 (2019).

100. D. J. Ching, S. Aslan, V. Nikitin, M. J. Wojcik, and D. Gursoy, "Evaluation of modified uniformly redundant arrays as structured illuminations for ptychography," arXiv, arXiv:2004.01766 (2020).
101. A. J. Morgan, K. T. Murray, M. Prasciolu, H. Fleckenstein, O. Yefanov, P. Villanueva-Perez, V. Mariani, M. Domaracky, M. Kuhn, S. Aplin, I. Mohacsi, M. Messerschmidt, K. Stachnik, Y. Du, A. Burkhart, A. Meents, E. Nazaretski, H. Yan, X. Huang, Y. S. Chu, H. N. Chapman, and S. Bajt, "Ptychographic x-ray speckle tracking with multi-layer Laue lens systems," J. Appl. Crystallogr. **53**(4), 927–936 (2020).
102. V. V. Nikitin, M. Carlsson, F. Andersson, and R. Mokso, "Four-dimensional tomographic reconstruction by time domain decomposition," IEEE Trans. Comput. Imaging **5**(3), 409–419 (2019).
103. M. Odstrčil, M. Holler, J. Raabe, A. Sepe, X. Sheng, S. Vignolini, C. G. Schroer, and M. Guizar-Sicairos, "Ab initio nonrigid x-ray nanotomography," Nat. Commun. **10**(1), 2600 (2019).
104. Z. Shen, F.-X. Vialard, and M. Niethammer, "Region-specific diffeomorphic metric mapping," arXiv, arXiv:1906.00139 (2019).
105. J. Henriques, S. Ehrhardt, S. Albanie, and A. Vedaldi, "Small steps and giant leaps: Minimal Newton solvers for deep learning," arXiv, arXiv:1805.08095 (2018).
106. H. Chan, Y. Nashed, S. Kandel, S. Hruszkewycz, S. Sankaranarayanan, R. Harder, and M. Cherukara, "Real-time 3D nanoscale coherent imaging via physics-aware deep learning," arXiv, arXiv:2006.09441 (2020).