Check for updates

SOFTWARE TOOL ARTICLE

REVISED The abTEM code: transmission electron microscopy from first principles

[version 2; peer review: 2 approved]

Jacob Madsen, Toma Susi

Faculty of Physics, University of Vienna, Vienna, 1090, Austria

v2 **First published:** 24 Mar 2021, **1**:24
https://doi.org/10.12688/openreseurope.13015.1
Latest published: 21 May 2021, **1**:24
https://doi.org/10.12688/openreseurope.13015.2

Abstract

Simulation of transmission electron microscopy (TEM) images or diffraction patterns is often required to interpret experimental data. Since nuclear cores dominate electron scattering, the scattering potential is typically described using the independent atom model, which completely neglects valence bonding and its effect on the transmitting electrons. As instrumentation has advanced, new measurements have revealed subtle details of the scattering potential that were previously not accessible to experiment.

We have created an open-source simulation code designed to meet these demands by integrating the ability to calculate the potential via density functional theory (DFT) with a flexible modular software design. abTEM can simulate most standard imaging modes and incorporates the latest algorithmic developments. The development of new techniques requires a program that is accessible to domain experts without extensive programming experience. abTEM is written purely in Python and designed for easy modification and extension.

The effective use of modern open-source libraries makes the performance of abTEM highly competitive with existing optimized codes on both CPUs and GPUs and allows us to leverage an extensive ecosystem of libraries, such as the Atomic Simulation Environment and the DFT code GPAW. abTEM is designed to work in an interactive Python notebook, creating a seamless and reproducible workflow from defining an atomic structure, calculating molecular dynamics (MD) and electrostatic potentials, to the analysis of results, all in a single, easy-to-read document.

This article provides ongoing documentation of abTEM development. In this first version, we show use cases for hexagonal boron nitride, where valence bonding can be detected, a 4D-STEM simulation of molybdenum disulfide including ptychographic phase reconstruction, a comparison of MD and frozen phonon modeling for convergent-beam electron diffraction of a 2.6-million-atom silicon system, and a

Open Peer Review

Approval Status ✓ ✓

	1	2
version 2 (revision) 21 May 2021	✓ view	
version 1 24 Mar 2021	✓ view	✓ view

1. **Colin Ophus**, Lawrence Berkeley National Laboratory, Berkeley, USA

2. **Hamish Brown**, University of Melbourne, Melbourne, Australia

Any reports and responses or comments on the article can be found at the end of the article.

performance comparison of our fast implementation of the PRISM algorithm for a decahedral 20000-atom gold nanoparticle.

Keywords
transmission electron microscopy, electron scattering, image simulation, density functional theory, molecular dynamics, open source, Python



This article is included in the European Research Council (ERC) gateway.



This article is included in the Horizon 2020 gateway.

Corresponding authors: Jacob Madsen (jacob.madsen@univie.ac.at), Toma Susi (toma.susi@univie.ac.at)

Author roles: Madsen J: Conceptualization, Formal Analysis, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Susi T**: Conceptualization, Formal Analysis, Funding Acquisition, Project Administration, Supervision, Visualization, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No [756277]), (project ATMEN).
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2021 Madsen J and Susi T. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Madsen J and Susi T. **The abTEM code: transmission electron microscopy from first principles [version 2; peer review: 2 approved]** Open Research Europe 2021, **1**:24 https://doi.org/10.12688/openreseurope.13015.2

First published: 24 Mar 2021, **1**:24 https://doi.org/10.12688/openreseurope.13015.1

REVISED **Amendments from Version 1**

All reviewer comments have been addressed. Updates to the code include: interpolation with Fourier-space padding, corrections to use case examples (including improved line styling to accommodate impaired color vision), and added repository examples including PACBED and partial coherence in 4D-STEM.

Any further responses from the reviewers can be found at the end of the article

1 Introduction

Transmission electron microscopy (TEM) is one of the most versatile and powerful experimental tools for the imaging and diffraction of structures ranging from the micrometer scale with sub-Ångström resolution now routinely achievable in modern aberration corrected instruments. In TEM, information about the sample structure is encoded in the scattering of the electron waves by the full electromagnetic potential of the specimen, which is dominated by the atomic electrostatic potentials. These potentials include the contribution of the screened nuclear cores as well as the valence electron density of the sample, and since valence electrons are responsible for binding the material together, studying them is of significant scientific interest.

The modern electron microscope should be an ideal tool for the high-resolution imaging of charge redistribution caused by chemical bonding, but these measurements are a challenge because only a small fraction of the total electrons in a material participate in bonding, and because the dense cores dominate the scattering signal. However, as improvements in instrumentation and techniques continue rapidly, this is likely to increasingly change, as evidenced by the surging popularity of techniques such as four-dimensional scanning transmission electron microscopy (4D-STEM) combined with ptychography in materials science, and cryogenic microcrystal electron diffraction in structural biology.

To reliably quantify subtle differences in the scattering signal, precise alignment of the instrument and a careful comparison between theoretical models and experiments are required. The use of image simulations has long aided this process, and many excellent codes have been developed. However, these have exclusively relied on the independent atom model (IAM), which approximates the specimen potential as a superposition of isolated atoms, completely neglecting chemical bonding. A growing number of studies are going beyond the IAM by calculating the potential using density functional theory (DFT)[1–7]. As expected, these studies find a better agreement for a range of different materials when comparing to measurements that are sensitive to valence electron density, such as holography and various forms of phase-contrast imaging.

The most common image simulation method is the multislice algorithm, and there is no shortage of codes implementing it [8–19], though the degree of support and documentation varies widely. Implementations have remained largely similar for a good while, apart from some iterative improvements. One significant recent development has been to use one (or more) graphics processing units (GPUs) for accelerating the calculations. In terms of methods, the most important recent advancement is the development of the PRISM algorithm, which massively accelerates scanning TEM (STEM) simulations[20].

Here, we present the abTEM code, which is a new multislice image simulation package created to seamlessly merge DFT and other atomistic modeling methods with electron scattering simulations, providing a much easier way of performing TEM simulations with an *ab initio* description of bonding. We have implemented both the multislice and PRISM algorithms to simulate all the standard imaging modes.

abTEM is further distinguished from existing codes by its pure Python implementation and focus on user extendability. We show that thanks to the effective utilization of open-source libraries, the performance of abTEM is as good or even better compared to similar codes implemented in programming languages that have traditionally been considered superior in performance, such as C or Fortran. abTEM already includes fast GPU implementations of all of its algorithms on widely used NVidia GPUs, and support for products from AMD is planned for the near future.

abTEM is an open-source project under the GPLv3 license, and we welcome contributions via our open Github repository[21]. Documentation and code examples are found online. The code was announced at the M&M 2020 meeting[22], and it supersedes and replaces our earlier PyQSTEM code[6].

This article is organized as follows: in Section 2, we discuss the physical methodology behind abTEM, including its implementation of the multislice electron scattering algorithm, IAM and *ab initio* potentials, the PRISM algorithm, and inelastic and thermal diffuse scattering. Next, in Section 3 we give brief remarks on our implementation details, followed by illustrative code examples. In Section 4, we discuss in turn programmatic modularity, dependencies and availability, interactive dashboards, and finally GPU memory management. We then turn to actual use cases in Section 5, providing novel results on systems that highlight how easy it is to use *ab initio* electrostatic potentials and how fast and efficient abTEM is. Finally, we end with a brief conclusion in Section 6, including features that we plan to develop next.

2 Methods and algorithms

2.1 Multislice

The multislice algorithm can be used to simulate any kind of TEM measurement: what differs is only how the input wave function of the electron probe is defined and how the scattered exit wave function is detected. Since the multislice algorithm is conceptually quite simple and discussed in detail elsewhere[8], we provide only a brief sketch of the method here.

In the multislice algorithm, the specimen electrostatic potential is divided into thin slices along the beam propagation direction (which by abTEM convention is the z axis). Scattering is calculated by alternating so-called phase object transmission through each slice with propagation of the wave to the next. For an electron wave ψ_n impinging on slice number n, this can be expressed as

$$\psi_{(n+1)} = p * (\psi_n t_n),$$

where p is the Fresnel free-space propagator, * represents the convolution operation, and the transmission function t_n is defined as

$$t_n(x,y) = \exp\left(i\sigma V_n(x,y)\right),$$

where i is the imaginary unit, σ is the interaction constant dependent on the electron wavelength λ, and

$$V_n(x,y) = \int_{z_n}^{z_n+\Delta z} V(x,y,z)\,dz \tag{1}$$

is the n'th projected potential slice with thickness Δz.

The convolution is conveniently calculated as a multiplication in Fourier space via a succession of a forward and an inverse Fourier transform. Another Fourier transform is needed to bandwidth-limit the transmission function and the Fresnel propagator to 2⁄3 of the Nyquist frequency to avoid aliasing artifacts[8].

The Fourier transforms are carried out efficiently using the fast Fourier transform algorithm implemented in the efficient FFTW[23] and cuFFT libraries on CPU and CUDA-enabled GPUs, respectively.

2.2 Potentials

The electrostatic potential of a specimen determines how transmitting electrons scatter and thus connects the properties of the material to the resulting images or diffraction patterns. Conversely, by directly analyzing scattered intensities or comparing them to simulations of the specimen electrostatic potential, the properties of the sample can be deduced. Fundamentally, the electrostatic potential is directly derived from the electron density of the atoms in a specimen, which is described by their quantum mechanical many-body wave function.

Since this cannot be analytically solved except for the very simplest of molecules, various approximations have been developed, such as early Hartree-Fock-Slater and Dirac-Fock formalisms. In the context of electron scattering, these highly accurate but computationally extremely expensive techniques can be used to parametrize (IAMs). However, there is increasing interest in a fully *ab initio* approach to calculating the specimen potential[22]. The most widely used and powerful such method is DFT, where the many-body problem of N electrons with $3N$ spatial coordinates is reduced to a variational solution for the three spatial coordinates of the electron density.

While abTEM is designed explicitly for *ab initio* potentials, IAM potentials can in many cases be sufficient or useful: for example, typical annular dark-field contrast in STEM is dominated by nuclear scattering and thus

well described by the IAM, and in general, comparing images or diffraction patterns simulated with *ab initio* potentials to the IAM allows the effects of chemical bonding to be elucidated. Thus, ABTEM supports common parametrized IAM potentials in addition to its integration with DFT methods.

2.3 Parametrized potentials

A potential parametrization, as pioneered by Doyle and Turner[24], is a numerical fit to atomic electron scattering factors calculated from first principles, describing the radial dependence of the potential for each element. ABTEM supports two of the most accurate recent parametrizations: that by Lobato and Van Dyck[25] (which is the ABTEM default) and by Kirkland[8].

Building upon earlier work Weickenmeier and Kohl[26] as well as Peng *et al.*[27], Kirkland accurately fitted a combination of Gaussians and Lorentzians to Dirac-Fock scattering factors, providing a widely used and robust parametrization that was distributed digitally from 1998 onwards. In 2014, Lobato and Van Dyck improved the quality of the fit further, using hydrogen's analytical non-relativistic electron scattering factors as basis functions to enable the correct inclusion of all physical constraints, providing to date the most accurate universal neutral atom parameter set[25]. For a comparison of several parametrizations to each other as well as quantitative experimental data for 2D materials, please see Ref. 6.

2.3.1 ab initio potentials. Although the ground-state electron density for all electrons could be numerically solved in DFT, the description of electron wave functions near the nuclei is computationally very expensive, and thus most practical approaches have adopted some partition between the core and valence regions. While reference methods such as linearized full-potential augmented plane waves (FLAPW) are very accurate, their computational expense limits them to around a hundred atoms, which is insufficient for most TEM simulations.

In recent years, pseudopotential[28] and projector-augmented wave (PAW) methods[29] have offered much greater computational efficiency. In both, the core electrons are not described explicitly but replaced by a smooth pseudo-density in the former, and in the latter by smooth analytical projector functions in the core region. The PAW method is arguably better suited for obtaining efficient and accurate *ab initio* all-electron electrostatic potentials, since inverting the projector functions allows the true core electron density to be analytically recovered (for an extended discussion, see Ref. 22). As such, this is the approach chosen for ABTEM, specifically via the grid-based DFT code GPAW.

In the PAW formalism[29], the total charge density $\rho(\mathbf{r})$ is a sum of the squared explicitly computed all-electron valence wave functions, the (frozen) core electron density derived from the PAW projector functions, and the nuclei that are treated as point charges. The charge density is divided into a smooth part $\tilde{\rho}(\mathbf{r})$ plus corrections for each atom a: $\rho^a(\mathbf{r}) - \tilde{\rho}^a(\mathbf{r})$, where the smooth part is given as pseudo wave functions and pseudo core charges. We can obtain the electrostatic potential $v(\mathbf{r})$ by solving the Poisson equation in two separated steps for the pseudo part and within the atomic augmentation spheres, and finally adding smeared nuclear charges to avoid the corrections diverging as $-Z^a/r$ near the nuclei in the total charge density.

The wavefunctions, electron density, and potential are described on real-space numerical grids, whose density controls the accuracy of the calculation (though not in a variational manner, unlike in traditional plane-wave bases also supported by GPAW). A detailed description of the code is given in Refs. 30,31, while the all-electron electrostatic potentials derived from it are described in our earlier work in Ref. 6.

It is worth noting that like other codes, ABTEM currently uses only the electro*static* potential, whereas electrons in truth interact with both electric and magnetic fields via the Lorentz force. This approximation is well justified by the fact that magnetic interactions are much weaker, but there is emerging interest in multislice simulations that also account for magnetism[32]. An *ab initio* approach is obviously ideally suited for such modeling, and our integration with the highly scalable GPAW code makes large magnetic domains or nanostructures feasible to treat from first principles.

2.3.2 Potential slicing. The multislice method requires a mathematical slicing of the potential into *xy* planes as given by Equation 1. ABTEM implements two alternative methods for projecting the potential: the formally correct finite numerical integrals, and a less accurate but commonly used and much faster method of using analytically solvable infinite integrals.

The finite projection method is based on evaluating the singular potential functions in real space. The singularities are removed by effectively convolving the 3D potential with a Gaussian whose standard deviation is equal to the real-space grid spacing. This is implemented using the convolution theorem; the 3D scattering factor is multiplied by a Gaussian, and the resulting radial function is Fourier-transformed to obtain the a radial potential function without a singularity.

We evaluate the the finite numerical projection integrals in Equation 1 following the work of Lobato and Van Dyck[13]. The integrals are handled by the double exponential Tanh–Sinh quadrature[33], which is designed for accurate evaluation of functions with endpoint singularities. Integrals across the sharply peaked cores are split into two integrals at the atomic position and the number of weights and nodes of the quadrature are determined such that the error is smaller than a given tolerance using a worst-case error estimate.

Instead of performing the expensive integral at each grid point, it is calculated along a radial line and interpolated on the simulation grid. Due to the rapid change of the atomic potentials near the core, the evaluation points of the integrals are geometrically spaced. The potential is set to zero outside a cutoff radius, determined for each atomic species such the error is smaller than a given tolerance. The use of the cut-off radius creates a discontinuity, hence we use a tapering cut-off function.

For simulations with thousands of atoms, a large proportion of the integrals are likely to be almost identical, and thus a cache is automatically used for saving the integrals and reusing their results if integration limits are identical within a given tolerance.

The potential slices are calculated in parallel using multithreading on both GPUs and CPUs. If the entire potential is too large to hold in memory, it is possible to calculate a smaller number of slices as they are needed. The maximum number of slices calculated in parallel is automatically estimated based on available system memory.

Most other codes use only the infinite projection method for calculating the projected potential, which assigns the infinite projection of each atom to a single slice. The fastest method of implementing this is by placing delta functions at the atomic positions and convolving the superposition with analytical potentials[11]. The convolution is efficiently evaluated by multiplying the Fourier-transformed superposition of delta functions with the 2D scattering factors, thereby also avoiding the issue of real-space singularities. The result is divided by a Sinc function to compensate for the finite size of the discretized delta functions.

This infinite projection method is up to 100 times faster than using finite projection integrals. The error can be up to a few percent in regions between the atoms, but much less when nuclear scattering is dominant. An example in our online repository explores these differences[21].

2.4 PRISM

Although it is universally applicable and conceptually simple, the multislice algorithm is not very efficient for large-scale STEM simulations, where the electron probe scans across the specimen: while the atomic potentials can be reused for different probe positions, the remainder of the calculation must be run independently. For STEM simulations consisting of thousands or even millions of probe positions, this is very costly. This problem is directly addressed by the recently developed PRISM algorithm[16,20].

The PRISM algorithm takes advantage of the fact that the electron probe can be expressed as a plane-wave expansion. Using the linearity of the multislice algorithm with respect to the wave function, the individual plane-wave basis functions can be propagated through the specimen independently. The set of plane waves at the exit surface, defined as the scattering matrix, can be coherently summed to calculate a scattered probe. The position and aberrations of the scattered probe can be chosen by multiplying each plane wave, ϕ_{nm}, by an appropriate complex coefficient prior to summation:

$$\psi(r, r_0) = \sum_{nm} \alpha_{nm}(r_0)\phi_{nm}(r), \quad \phi_{nm} = \exp(-2\pi i \boldsymbol{q}_{nm} \cdot \boldsymbol{r}), \tag{2}$$

where the coefficient, $\boldsymbol{\alpha}_{nm}$, depends on the aberrations and the desired probe position r_0. The wave vector, q_{nm}, must fulfill

$$\boldsymbol{q}_{nm} = (nf\Delta_q, mf\Delta_q), \tag{3}$$

where Δ_q is the Fourier space pixel size, the integer f is the interpolation factor, and the integers m and n represents the plane-wave index fulfilling

$$\sqrt{m^2+n^2}\, f\, \lambda \Delta q \leq \alpha_{max}, \tag{4}$$

where α_{max} is the probe convergence semi-angle. The size of the basis is naturally limited by the aperture and can be further reduced by a factor of f^2 at the cost of accuracy. Increasing the interpolation factor, and thus keeping only every f'th coefficient in Fourier space, corresponds to tiling the probe f times in real space, hence the interpolation factor should be chosen such that the tiled probes does not interact. To aid users ABTEM implements methods for estimating the error and choosing an appropriate level of interpolation.

When the interpolation factor is greater than one, the result needs to be cropped so only the desired probe and not its repetitions are included. Due to the memory-size of the scattering matrix the computational cost of the cropping operation is non-trivial. In the original implementation[16], the scattering matrix is cropped before the sum reduction in Equation 2 is performed, ensuring that a minimum number of operations are necessary. Our algorithm differs by cropping after performing the reduction for a batch of positions: a chunk of the scattering matrix, large enough for calculating all positions in the batch, is selected and if necessary, is transferred to the GPU. The coefficients for the entire batch are calculated, and the reduction is performed simultaneously for all positions in the batch using multithreaded parallelization. Lastly, the cropping is performed on the much smaller reduced array.

Our implementation requires more floating-point operations than strictly necessary, but calculations are performed with fewer memory transfers and expensive kernel launches from Python. To minimize excess work, probe-position batches are taken as compact squares in order to have as much overlap of the necessary parts of the scattering matrix as possible. This implementation has the unusual feature that its speed does not necessarily increase monotonically with the batch size: The amount of excess work increases with the batch size, so once full thread utilization is reached, further increase of the batch size may degrade the performance.

In Section 5, we directly compare the ABTEM implementation of PRISM to the Prismatic code[16], demonstrating the highly competitive performance of our pure Python code.

2.5 Inelastic scattering

The basic version of the multislice algorithm assumes that the electron beam scatters only elastically. In real materials, the electrons also undergo inelastic energy loss, which affects their interference and contributes to backgrounds. Furthermore, the atoms of the target are not static but in constant motion due to zero-point vibrations and thermal phonon occupations.

Inelastic scattering can be approximately modeled using an absorptive potential[34], where the imaginary part of a complex electrostatic potential is used to describe the loss of electrons from the elastic channel. However, in this approach, the inelastically interacting electrons are simply removed from an otherwise purely elastic scattering calculation. Although this is computationally efficient, the method's serious weaknesses are that the electron flux is not conserved, and high-angle scattering is underestimated. For the case of phonon scattering, this limitation is overcome in the widely used frozen phonon model (see below).

The frozen phonon model has become standard in most multislice programs. However, there is also increasing interest in explicitly simulating spectra for low energy losses, which can be theoretically quite challenging[35,36]. Very recently, Zeiger and Rusz developed a novel method for modeling low-loss electron energy-loss spectroscopy (EELS) based on molecular dynamics and multislice simulations[37], for which our code is ideally suited. Moving up in energy, plasmon scattering is, to our knowledge, not included in any publicly available code, perhaps because they are less important for high-angle scattering[38]. Plasmon modeling following a Monte Carlo method by Mendis[39] is planned for a future release.

Finally, inner-shell ionization is of particular interest as a spectroscopic fingerprint of elements and their bonding. EELS, including dynamical scattering, can be modeled by combining multislice simulations with the transition potentials for the elements of interest[12]. Simulating EEL spectrum images used to be extremely computationally demanding, but recently a much faster method was developed by Brown[40]. These algorithms are currently being implemented into ABTEM, further improving its *ab initio* capabilities.

2.5.1 Thermal diffuse scattering. Thermal diffuse scattering (TDS), or electron-phonon scattering, is important both in STEM and high-resolution TEM (HRTEM) and is responsible for features including diffuse backgrounds and Kikuchi lines as well as for a large part of the annular dark-field signal[41]. The most common and successful method for simulating TDS with dynamical scattering is the frozen phonon model. Its basic idea relies on a rather classical picture where each electron sees a different configuration of atoms displaced from equilibrium by thermal vibrations. Despite its simplicity, the model's accuracy has been substantiated both theoretically[42] and numerically by comparison to a fully quantum mechanical model in the standard Born–Oppenheimer approximation[43].

The frozen phonon structures are usually created by independently displacing each atom according to a Gaussian distribution with a fixed width depending on the element, i.e., following the approximate Einstein model. A more accurate thermal ensemble can be directly generated through molecular dynamics simulations at a given temperature[44]. In abTEM this is facilitated through the Atomic Simulation Environment (ASE)[45], which interfaces with several popular MD codes. abTEM's example library include a single-worksheet demonstration of using the high-performance classical MD code LAMMPS[46] through ASE in conjunction with abTEM.

3 Implementation

3.1 Python

Most multislice codes have a somewhat rigid character, using a graphical user interface or input files to control the execution of binaries written in compiled languages such as Fortran and C. A distinguishing feature of abTEM is that all tasks are accomplished by writing and running Python scripts. Python is a dynamically typed programming language with clear and expressive syntax. It can be used for writing everything from small scripts to large programs and libraries like abTEM itself. The Python language has continually gained popularity for scientific computing for the past two decades, thanks in particular to its extensive and broad base of open-source libraries. In the TEM community, this includes packages such as HyperSpy[47] and Py4DSTEM[48].

Some multislice codes have a Python or MATLAB scripting interface[13,16], but the level of interactivity rarely goes beyond automation of input and output of fixed simulation modes. Our previous effort in providing a Python interface to the powerful multislice code QSTEM went somewhat further, but the actual simulations were still not performed by PyQSTEM itself[6]. Now, abTEM relies on external Python libraries only to handle atoms and DFT calculations, or to enhance its numerical performance.

abTEM also differs from earlier codes by not directly implementing *any* common imaging modes, but invites the user instead to mix and match objects to construct the desired simulation. The design patterns used by abTEM thus take inspiration from object-oriented scientific codes, particularly ASE[45], which is also used for importing and manipulating atomic structures and interfacing with other atomistic simulation codes. The idea behind this approach is that the user operates using understandable concepts from physics instead of computational details. abTEM provides Python classes like `Waves` that store the wave function as a NumPy array, and implements methods like `multislice` for propagating the wave function. In Figure 1, we show common objects the user can expect to interact with.

3.2 Modularity

A key design principle in object-oriented programming is to keep different objects independent, improving readability and simplifying further development as contributors only need familiarity with a fraction of the codebase. The objects in Figure 1 are divided into categories, and the objects within a category are generally made to be interchangeable. To use similar classes interchangeably in conjunction with others requires that the classes follow a template. A considerable amount of thought has gone into creating templates that allow flexibility to implement new functionality with the minimum effort. Further, since all code is open source in Python, this is highly accessible even for non-expert users.

Each of the classes shown in Figure 1 is a subclass of an abstract base class. Any new class within that category is required to implement some basic methods and properties. For example, any detector should implement a minimum of two methods: the `allocate_measurement` method for creating an array in memory or as an HDF5 file for storing the results, and the `detect` method for taking a wave function object and returning a corresponding measurement. A detector following this pattern is automatically compatible with any other algorithm implemented in abTEM.

3.3 Performance and dependencies

Dynamic interpreted languages such as Python are attractive for domain experts and scientists trying out new ideas. However, the performance of the interpreter is often a barrier to high performance. To mitigate this, NumPy

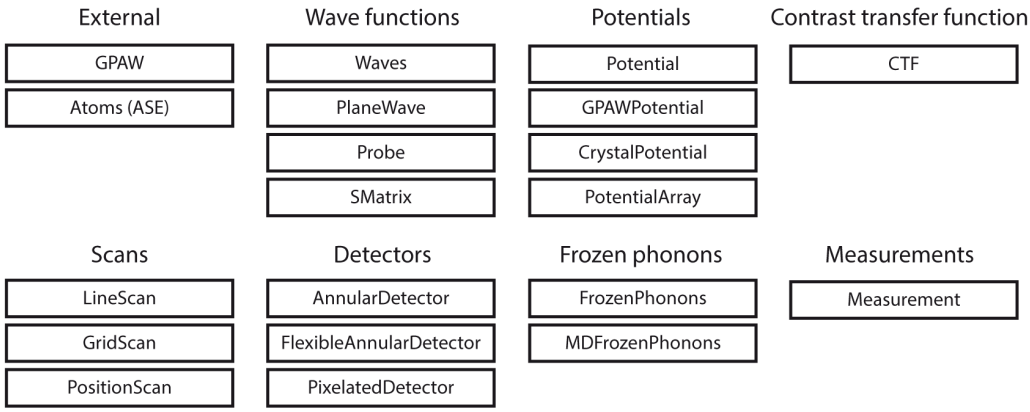


Figure 1. The modular design of abTEM is enabled by Python classes that implement physically meaningful concepts. Consistent object-oriented design allows new instances of these classes to be easily implemented as our code grows or new techniques or instrumentation become available.

is one of the most important scientific libraries in the Python ecosystem as it provides a multi-dimensional array (`ndarray`) object that has become the foundation of efficient numeric computation in Python. If an operation can be performed with a single call to a NumPy function, the performance matches the underlying implementation in C. However, this is not always possible, and scientific codes often rely on Python C extensions to efficiently implement custom computation. For example, the DFT code GPAW consists of about 90% Python with a small amount of computationally demanding functionality implemented in C.

The process of writing a Python C extension can be error-prone due to the difficulty of manually managing the reference counts of Python objects and generally requires a lot of 'boilerplate' code, even for simple use cases. Further, the inclusion of compiled code complicates the release of cross-platform programs, as it requires the user to be familiar with code-compilation or the maintainer to provide precompiled binaries. abTEM achieves high performance without C extensions by using the Numba library[49], which is a just-in-time Python compiler that focuses on scientific and array-oriented computing. Numba analyzes and optimizes Python code and then uses the LLVM compiler library to generate machine code with a performance similar to that of C.

Python's recent popularity in machine learning has led to the release of high-performance libraries for GPU-accelerated calculations. According to our benchmarks, pre-eminent GPU libraries' speed is largely identical for performance-critical FFT and matrix multiplication operations. Our choice to use CuPy[50] comes down to its compatibility with Numpy: CuPy can in most cases be used as a drop-in replacement, while users without access to a compatible GPU can install abTEM without it. Due to CuPy limitations, abTEM currently only supports Nvidia's CUDA toolkit on Windows and Linux, requiring an NVidia graphics card – all GPU multislice programs currently require CUDA, apart from the STEMcl package[18], which uses the non-proprietary OpenCL framework. However, OpenCL is, according to our testing, slower than CUDA. abTEM will soon expand to support AMD ROCm GPUs on Linux following the next release of CuPy, which is expected to deliver AMD support. We hope that support for Apple Silicon will soon follow.

abTEM currently offers only limited multi-CPU/GPU parallelization, requiring the user to distribute the work across multiple CPU/GPU workers and gather the results afterwards. For example, one can assign each worker a fraction of the frozen phonon configurations and add up the results after they are done. Our online repository[21] demonstrates multi-CPU parallelization using the commonly available Message Passing Interface (MPI), utilizing the excellent mpi4py library and parallel HDF5 to support parallelized filesystem access. We expect to further improve parallelization in the near future.

3.4 GPU memory management

The arrays used to store the potential or scattering matrix may be too large to fit into limited GPU memory. Hence, recent multislice applications have emphasized the importance of asynchronous memory transfer, whereby memory transfer and kernel execution are performed simultaneously[16,18].

Since we prefer to keep abTEM as simple as possible, we use serial memory transfer and kernel execution. We focus instead on using batching to limit the memory transfers and ensure high GPU utilization. Both FFTW and cuFFT support batch FFTs, whereby multiple Fourier transforms of the same size can be computed simultaneously. Hence, a number of wave functions are propagated simultaneously in the multislice algorithm. For small batch sizes, memory transfer is indeed a considerable overhead, but this generally disappears as the batch size grows. In HRTEM, only a single wave function is propagated, hence batching is not possible; however, in this case there is no memory to transfer, since each slice is only used once and calculated on the fly on the GPU. For large simulations, memory constraints may limit the batch size, but we generally did not find memory transfer to be a significant bottleneck.

4 Operation

4.1 Minimum system requirements

Although a powerful system is recommended for real work, any modern laptop will meet the minimum requirements for running abTEM. It is also possible to run for teaching and demonstration purposes using free cloud computing services, such as Binder[51].

4.2 Code examples

To demonstrate the use of abTEM, we show a few basic examples below, with many more found in our online repository online repository[21]. We start with a simple example showing how to combine objects together for a HRTEM simulation:

```
from ase.io import read
from abTEM import PlaneWave, CTF
atoms = read('atoms.cif')
plane_wave = PlaneWave(sampling=0.01, energy=300e3)
exit_wave = plane_wave.multislice(atoms)
ctf = CTF(defocus=200, focal_spread=40)
image_wave = ctf.apply(exit_wave)
image = image_wave.intensity()
```

A structure file is imported as an ASE `Atoms` object; the `read` method supports a multitude of popular file formats. The incoming wave function is defined as a plane wave and propagated through the structure using the multislice algorithm, with the default IAM potential implicitly applied. abTEM follows the ASE unit convention with Å used for spatial variables and eV for energies. The wave function is transferred to the image plane using a contrast transfer function (CTF), where the intensity is calculated to obtain the final image. In such functions, the angular unit of mrad is used.

The example could be easily modified to instead simulate electron diffraction by using the `diffraction_pattern` method as the final step, and further to simulate convergent-beam electron diffraction (CBED) by using `Probe` to define the incoming wave function. The contrast transfer function includes partial coherence through the widely-used quasi-coherent approximation[8]; examples of the correct incoherent summation can be found in the online repository.

Next, we show the code for a basic annular dark field STEM simulation:

```
from ase import Atoms
from abTEM import Probe, Potential, GridScan, AnnularDetector
atoms = Atoms('C')
atoms.center(vacuum=10)
potential = Potential(atoms, slice_thickness=0.5)
probe = Probe(sampling=0.01, energy=300e3, semiangle_cutoff=30)
scan = GridScan(start=(0,0), end=potential.extent, sampling=0.1)
detector = AnnularDetector(inner=70, outer=180)
adf_signal = probe.scan(scan, detector, potential)
```

This example creates a single C atom at the center of a 10×10×10 Å simulation cell. Whereas previously the default method for calculating the potential was used, we explicitly set the `Potential` object to define how this is done. The incoming wave function is now defined as a probe, and for detecting the exit wave

functions, an annular dark field detector is defined. Using a `PixelatedDetector` in place of an `AnnularDetector` can change a simulation from annular dark field STEM to 4D-STEM, and the PRISM algorithm could be used instead of multislice by simply replacing the `Probe` object with `SMatrix`.

The downside of our approach is that it may not be immediately obvious for new users how to combine the objects to achieve the desired simulation mode. To assist these users, the ABTEM repository contains an expanding example library implementing common (and some less common) image simulations. Users can download the notebooks and modify them; this could be as easy as changing the imported structure file.

4.3 Cross-platform interactive notebooks

Given the visual nature of image simulations, a particularly attractive way of using ABTEM is through Jupyter notebooks. This web application allows users to create and share documents that contain live code, visualizations and narrative text. The notebooks are composed of cells, each of which may contain the code for accomplishing a subtask. To facilitate a visual workflow, most of the classes in ABTEM implement a method for creating a quick visualization of each subtask; for example a heatmap of the projected potential or the profile of an electron probe. The mix of code, explanatory text and visualizations immediately provides documentation of the simulation, allowing others to understand and reproduce it.

Thus, ABTEM is designed to be used via scripts, in an interactive Python session, or in a Jupyter notebook. Since these are cross-platform tools, support for popular operating systems including Linux, MacOS and Windows is easy to offer and maintain. Nonetheless, we recognize that this is not always the optimal way of interacting with code. Relying on new tools from the data science community developed for interactive data analysis, ABTEM also supports the creation of small web applications embedded in a Jupyter notebook.

For example, the `apply` method of the `CTF` class has an "interact" keyword that, if set to true, opens an interactive visualization of the resulting image. The ability to change the defocus and spherical aberrations using a simple slider and seeing the image update provides quick intuitive understanding of the effects and interplay of the parameters.

The tools for creating notebook-embedded interactive visualizations can with little effort also be used for creating simple web applications that are straight-forward to deploy on a server. This could be useful for teaching, as it enables users to experiment with simulating TEM images with only a web browser.

5 Use cases

Finally, we turn to realistic simulation use cases. These calculations were performed using a consumer-grade desktop PC running Linux (Ubuntu 20.04.1) equipped with an 8-core Intel Core i9-9900K processor and 32 GB of memory and a NVidia RTX 2080 Ti graphics card with 2944 cores and 11 GB of graphics memory. Jupyter notebooks for completely recreating each example are available online, where we plan to collect all publication-related open code.

5.1 Comparing IAM to *ab initio* for hBN

In our first example, we illustrate the effect of valence bonding by comparing IAM and DFT potentials for the simulation of hexagonal boron nitride (hBN), whose strong ionicity makes it an ideal test case. As has been previously noted, while annular dark field (ADF) contrast is rather insensitive to bonding, both bright field (BF) contrast (previously measured using HRTEM[2]) and electron diffraction (ED) intensities (measured with parallel illumination HRTEM[6]) do show marked quantitative differences; very recently, this was also confirmed by ptychography[52]. ABTEM makes such comparisons extremely easy to make, not to mention facilitating the use of *ab initio* potentials in the first place.

We create an orthogonal periodic unit cell of hBN using the ASE constructor for graphene with the lattice constant of hBN and replacing the C atoms with B and N, and create both IAM and DFT scattering potentials, the latter requiring a cheap GPAW plane wave calculation for converging the valence electron density. We then simulate a STEM experiment at an electron energy of 80 keV and a probe convergence semi-angle of 31.5 mrad, with two detectors: a BF detector with a semi-angular range of 0 to 16 mrad, and a high-angle ADF detector for the range 95 to 126 mrad.

We scan over the entire cell at the Nyquist sampling for the highest computational efficiency, here corresponding to a real-space spacing of 0.3 Å, allowing us to interpolate the measured images down to the sampling of the potential (0.04 Å) without introducing artifacts. Images can further be easily tiled to display larger fields of view or calculate diffraction patterns, and interpolated line profiles plotted to facilitate quantitative comparisons.

Figure 2 shows our simulated results. As expected, the ADF contrast is almost entirely insensitive to the potential, but a reduction of the N atom scattering intensity by about 5% due to charge redistribution is visible in the BF contrast. The same effect also reduces the intensity of the first diffraction order in the simulation based on the DFT potential. Thus, as is being increasingly recognized[22], effects due to charge redistribution can be measured using modern TEM methods – and readily simulated using ABTEM.

5.2 4D-STEM ptychography of MoS_2

In our second example, we attempt to computationally reproduce recent 4D-STEM experiments on MoS_2, where different measurements are reconstructed from a fast pixelated direct electron detector[53]. The modular design of ABTEM makes it easy to directly model 4D-STEM experiments, and we have implemented both integrated center of mass (iCoM) measurements[54] and the popular ePIE phase reconstruction algorithm[55] in Python to facilitate such simulations.

We create a periodic orthogonal 4×4×1 MoS_2 supercell, and create a vacancy by deleting one S atom from the top layer. We then create an IAM scattering potential and a scanning probe with parameters corresponding to the experiment[53]: an electron energy of 80 keV, probe convergence semi-angle of 21.4 mrad, defocus corresponding to the given chromatic aberration parameters, a reasonable amount of residual spherical aberration and focal spread, and some residual astigmatism to try and mimic the experimental contrast. The scan area is chosen to closely reproduce the experimental field of view.

We define two `PixelatedDetectors` with different spatial samplings (0.21 Å for BF, ADF, and iCoM, and 0.45 Å for ptychography). The Experimental signal is emulated using Poisson noise with an electron

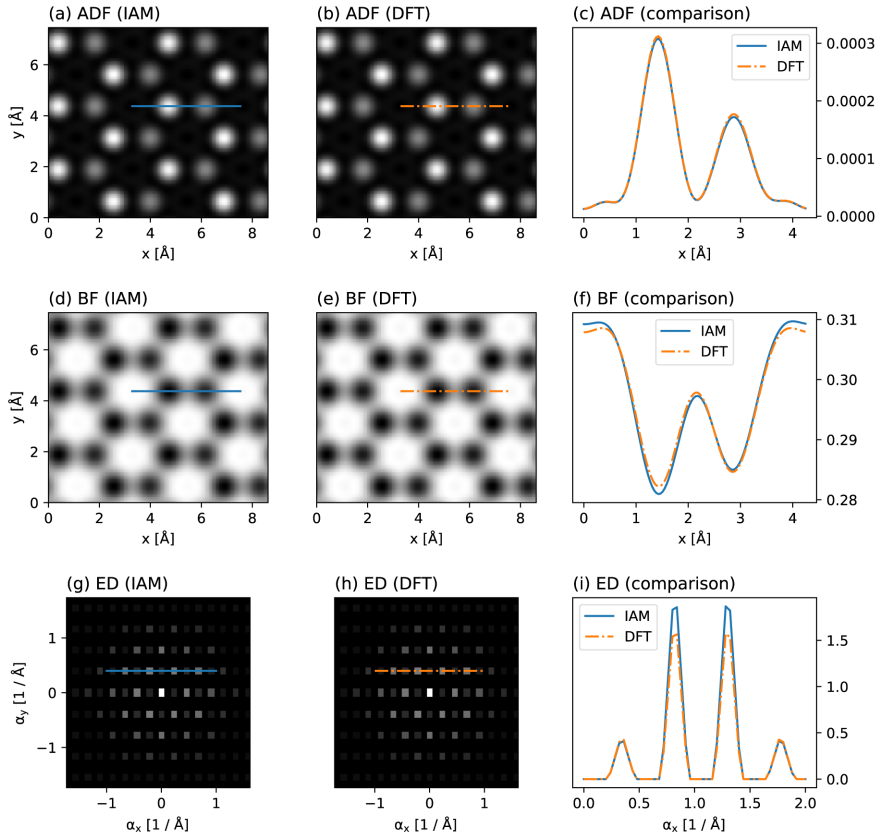


Figure 2. Comparison of independent atom model (IAM) and density functional theory (DFT) scattering potentials for hBN. Annular dark-field (ADF) images based on the (**a**) IAM and (**b**) DFT, with (**c**) line profiles plotted over the N and B sites. (**d**,**e**) Corresponding bright-field (BF) images and (**f**) line profiles. (**g**,**h**) Corresponding electron diffraction (ED) patterns displayed on a logarithmic scale derived from tiled images, and (**i**) line profiles through the first two diffraction orders. While the ADF contrast is nearly completely insensitive to valence bonding, both BF contrast (see Ref. 2 for an experimental measurement) and the ED intensities (see Ref. 6 for an experimental measurement) do show a marked difference due to charge redistribution in this highly ionic compound that is not described by the IAM.

dose of 6×10^6 e^- / Å2. Additionally, we define a BF detector with a semi-angular range of 0 to 21.4 mrad and an ADF detector for the range 64.2 to 85.6 mrad, which are post-integrated from the 4D-STEM measurement alongside the iCoM signal. For the phase reconstruction, we run five iterations of the ePIE algorithm. Diffractograms are calculated from the measured images using a built-in method.

Figure 3 shows our simulated results, which mimic the experimental Figure 2 in Ref. 53. According to our testing, for a rigid semiconducting material such as MoS_2, neither the inclusion of structural relaxation or using an *ab initio* electrostatic potential would affect the results significantly. Note that we did not apply any thermal averaging in this simple example, which would reduce the contrast of S_2 columns compared to the S vacancy, bringing the results into better agreement with the experimental data[53].

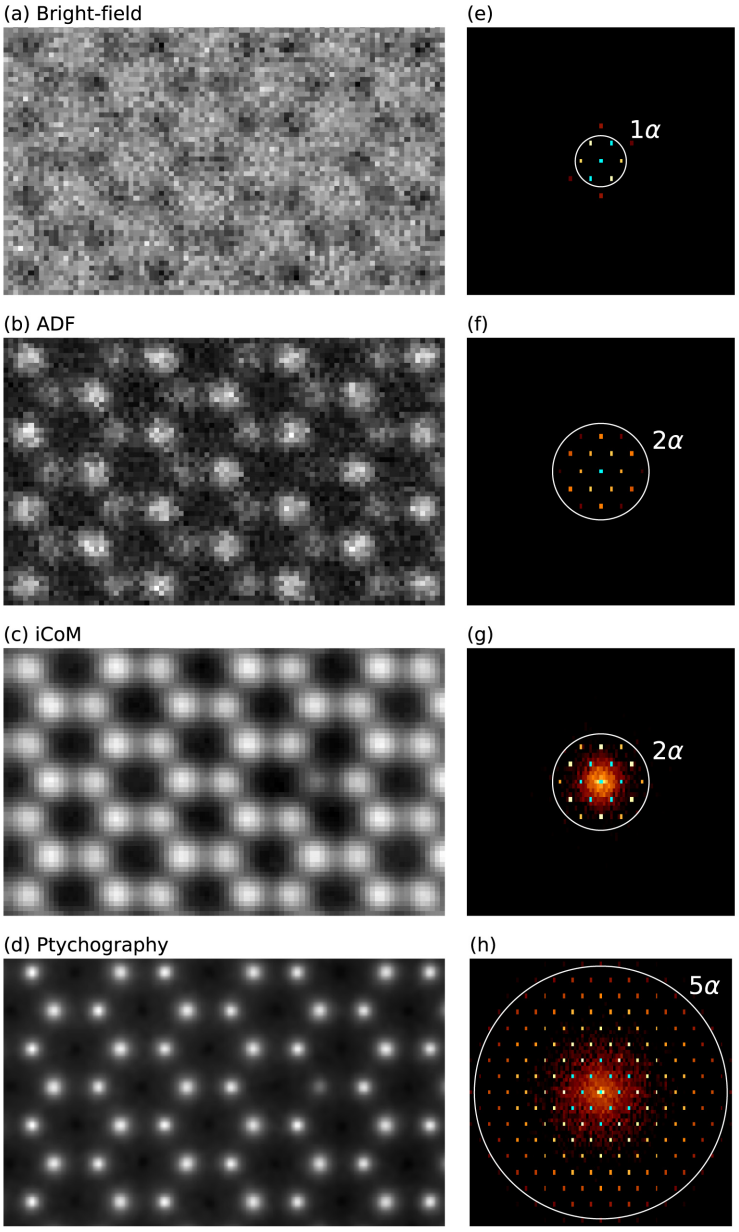


Figure 3. 4D-STEM simulations of MoS_2 with a single S vacancy. (**a**)–(**c**) Bright-field, annular dark field (ADF), and integrated center of mass (iCoM) (0.21 Å sampling) measurements and (**d**) ptychographic phase reconstruction (0.45 Å sampling), all derived from pixelated detector measurements with applied Poisson noise. (**e**)–(**h**) Diffractograms displayed on logarithmic scale calculated from the images and colored with the color map 'afmhot' with a cyan color set for the brightest values. For the experimental work that inspired this example, see Ref. 53.

5.3 CBED and ED with MD phonons

In our third example, we examine the effect of phonon correlations on TDS in CBED and ED patterns. The example demonstrates how abTEM can be used in conjunction with a MD code to calculate accurate thermal ensembles for use in a frozen phonon calculation. The example also demonstrates the performance of abTEM with respect to imaging modes that only require a single wave propagation, and hence where calculating the IAM potential is a substantial part of the overall computational cost.

We create a sample structure of crystalline silicon in the [100] zone axis, with a 200×200×1303 $Å^3$ supercell containing 2.6 million atoms. The thermal ensemble is simulated using molecular dynamics with the Tersoff[56] force field implemented in LAMMPS, and periodic boundary conditions are applied in all directions. We obtain 16 phonon snapshots after equilibration at 300K for 50 ps, chosen at time intervals exceeding the expected phonon correlation time (10 ps). The available four rotation and two inversion symmetries are used to effectively simulate a total of 128 configurations.

The wave function is sampled at 0.025 Å/pixel and the potential is sliced with a thickness 0.1 Å, corresponding to sampling the potential on a voxel grid of size 4096×4096×13000. The slices are chosen so thin to accurately portray features that depend on the 3D symmetry of the crystal. This could also be accomplished by slicing the potential at the crystal planes[57], but that would neglect the 3D nature of the phonons. The potential is calculated on-the-fly using standard infinite projection integrals. On GPU, a batch size of six slices is automatically selected by the program, whereas all CPU threads are already fully engaged without slice parallelization.

The multislice simulation of each frozen phonon configuration took 182 s on GPU and 85 min on CPU, not including the MD simulation. The time spent on calculating the potential and wave propagation was approximately equal, as was reported in the description of the potential algorithm[11]. Note that in this comparison, the calculation of the transmission function from the potential is counted as wave propagation.

For comparison, we calculate the equivalent CBED pattern using the Einstein model. The standard deviation of the Gaussian distributions of each atomic position are set to 0.126 Å to match that found for the MD thermal ensembles. Figure 4 shows that the Einstein model and the more accurate MD model are quantitatively different, as has been discussed before[44,58]. The correlation of atomic vibrations in the crystal results in more scattering around the Bragg reflections, and the Einstein model overestimates the low-angle diffuse background. At medium-to-high scattering angles the Einstein model provides good agreement with the MD result, whereas a fully elastic model without any phonon images severely underestimates high-angle scattering.

5.4 Large-scale STEM simulations

In our final example, we show how abTEM is capable of performing extremely large STEM simulations using the PRISM algorithm. We examine the speed of this algorithm compared to multislice, as well as the speed of abTEM's pure Python PRISM algorithm to Prismatic, which is an open-source C++/CUDA package using asynchronous memory transfer. To our knowledge, of the two existing codes implementing PRISM, Prismatic and PyMultislice[16,19], the former released by the original developers of the algorithm is faster.

We simulate a STEM image of a decahedral gold nanoparticle with 20324 atoms. The supercell is 136×136×118 $Å^3$, and a grid of 4096×4096 pixels was used, corresponding to a real-space sampling of 0.033 Å or maximum anti-aliased scattering angle of 419 mrad; the slice thickness was 1.0 Å. The convergence semi-angle is set to 25 mrad, thus this probe requires a basis set of 20849 plane waves to represent (or more than 1 Tb of memory) and hence interpolation is required. We set the interpolation factor to 16, resulting in a basis set of just 81 plane waves, with an effective probe window of 8.5×8.5 $Å^2$ and a maximum error of 1.2% compared to standard multislice, which is about the same error as is introduced by using 1.0 Å as opposed to 0.5 Å slices.

The probe is scanned at the Nyquist frequency (here 0.42 Å per pixel), resulting in a total of 106276 probe positions. The calculations took 73 s per frozen phonon configuration on GPU and 1224 s on CPU. The scattering matrix was stored in CPU memory in both calculations and the potential was calculated on the fly, as keeping both the potential and scattering matrix in memory was not possible. For comparison, using multislice the same calculation took around 96000 s (~1 day) on GPU and 2220000 s (~25 days) on CPU, though with a precalculated potential. This shows that PRISM was 1315 times faster on GPU and 1813 times on CPU, in line with the findings of the original authors[20].

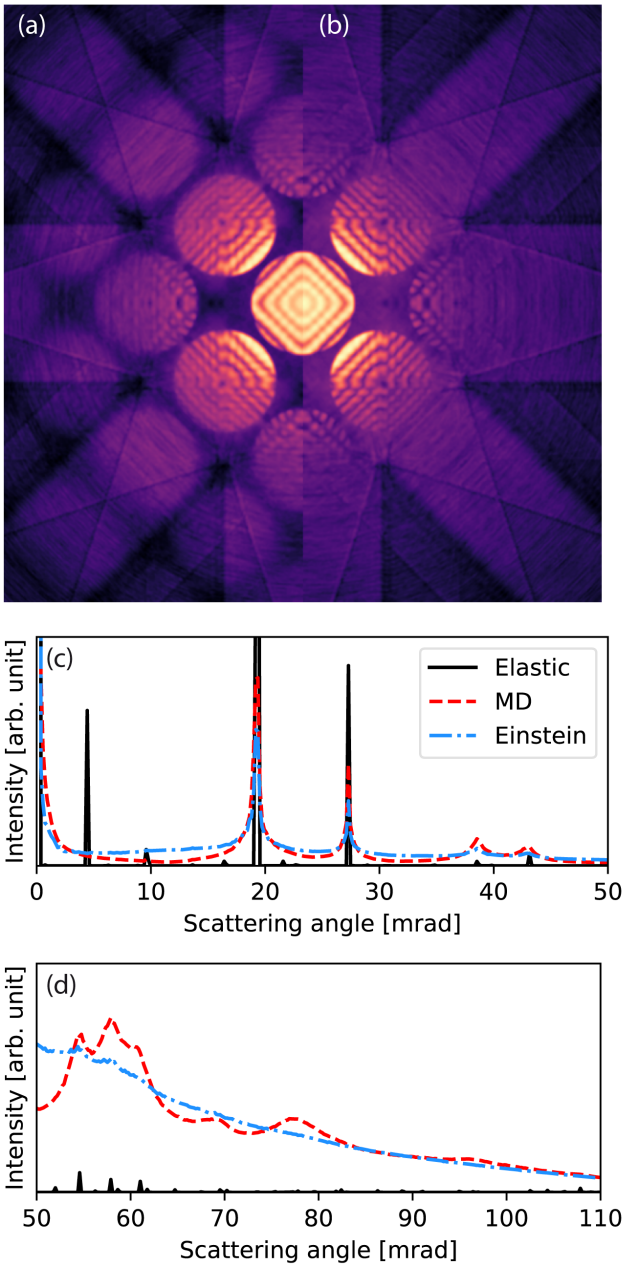


Figure 4. Convergent-beam electron diffraction (CBED) simulation with thermal diffuse scattering. (**a**)–(**b**) Calculated CBED patterns for Si[100] at 100 keV with an aperture of 9.4 mrad. (**a**) Calculated using molecular dynamics (MD) and (**b**) calculated using an Einstein model. In (**a**) more of the higher order disks are visible. (**c**)–(**d**) The rotationally averaged electron diffraction patterns for Si[100] at 100 keV.

The PRISM algorithm requires two steps: propagating the scattering matrix using the multislice algorithm and reducing the scattering matrix at each probe position. In all of the calculations above, the multislice propagation was the more expensive step. For large simulations, this step is unlikely to suffer much from the overhead of calling the FFTW and cuFFT libraries from Python and hence not suffer from our pure Python approach. It is also difficult to see how to significantly further improve the performance of this step.

In Figure 5, we benchmark abTEM against Prismatic for the simulation of differently sized decahedral nanoparticles. The size of the simulation supercell was varied, while keeping the real space sampling and number of plane waves in the probe constant at the values given above. abTEM was consistently up to twice as fast compared to Prismatic on our benchmark system using either GPU and CPU algorithms.

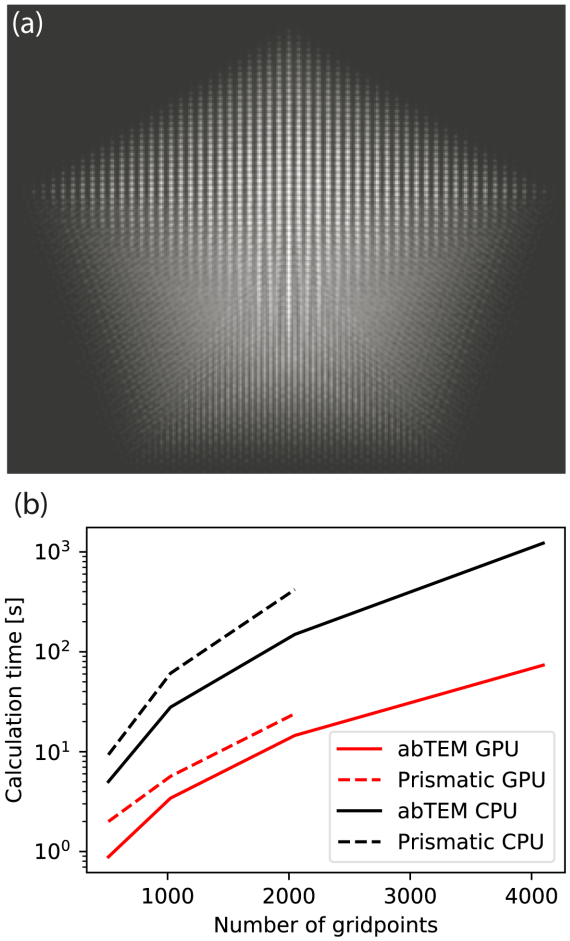


Figure 5. (**a**) High-angle annular dark-field STEM image of a decahedral nanoparticle consisting of more than 10^5 atoms, simulated in 15 s using abTEM's GPU-accelerated PRISM algorithm. (**b**–**c**) The time required for simulating a nanoparticle as a function of the size of system. (**b**) Calculation time for simulating a decahedral nanoparticle, as a function of system size given as as the number of pixels on each side of the simulation grid. The calculation for the largest nanoparticle is left out for Prismatic as this simulation was not possible due to running out of memory.

Note that Prismatic uses a factor of 2 antialiasing aperture, while abTEM uses a factor of 3⁄2. Hence, the maximum scattering angle in abTEM is 25% higher. It also has the ability to calculate the potential on the fly, whereas Prismatic always precalculates the potential: the simulation of largest nanoparticle system shown in Figure 5(a) was not possible using Prismatic on our hardware as the potential could not fit into memory. We do acknowledge that Prismatic has multi-CPU and -GPU support, and thus is likely to outperform abTEM on other hardware. Nonetheless, it is clear from this comparison that abTEM performs exceedingly well despite its pure Python implementation.

6 Conclusion

We have presented a new multislice simulation code called abTEM. The program was created to integrate atomistic simulation codes with the multislice algorithm in order to meet the demands of improved accuracy and flexibility posed by ongoing experimental advances. abTEM is written entirely in Python and has all the benefits that entail, and we have demonstrated that it is as fast as a highly optimized C++ code thanks to effective use of modern open-source software libraries. Python is also becoming important on the experimental side thanks to the Nion Swift microscope control software, enabling full integration of all aspects of the modern TEM research workflow. Notably, Python scripting has also been recently enabled for the Gatan Microscopy Suite[59].

abTEM is under active open development and is made to be easily extendable. We expect to significantly improve multi-CPU and -GPU parallelization in the near future, including support for AMD graphics cards

following the next release of CuPy. In terms of new simulation modes, our current focus is on improving the description of inelastic energy loss, and we are specifically working on implementing core losses following the work of Brown[40] and phonon losses following that of Zeiger and Rusz[37]. We have also taken some initial steps towards implementing plasmon losses following the approach of Mendis[39].

The code is now fully functional and ready for scientific work. We hope to entice experts in the field to try it out and join us in implementing new features, but are especially keen to enable experimentalists to complement their work by easily accessible simulations. Our ultimate goal is a fully *ab initio* description of all aspects of TEM simulation.

Data availability

All data underlying the results are available as part of the article and no additional source data are required.

All presented figures can be reproduced with the open code available from:

https://github.com/jacobjma/abTEM/tree/master/articles/2021_ORE_abTEM[21].

Software availability

Documentation and code examples available from: https://abtem.readthedocs.io/en/latest/index.html

Source code available from: https://github.com/jacobjma/abTEM

Archived source code at time of publication: https://doi.org/10.5281/zenodo.4767986[21].

License: GNU General Public License version 3

Acknowledgements

J.M. and T.S. acknowledge computational resources provided by the Vienna Scientific Cluster. We are deeply indebted to the many authors that contribute to the vibrant scientific Python ecosystem.

References

1. Deng B, Marks LD, Rondinelli JM: **Charge defects glowing in the dark.** *Ultramicroscopy.* 2007; **107**(4–5): 374–381.
PubMed Abstract | Publisher Full Text
2. Meyer JC, Kurasch S, Park HJ, *et al.*: **Experimental analysis of charge redistribution due to chemical bonding by high-resolution transmission electron microscopy.** *Nat Mater.* 2011; **10**(3): 209–215.
PubMed Abstract | Publisher Full Text
3. Kurasch S, Meyer JC, Künzel D, *et al.*: **Simulation of bonding effects in HRTEM images of light element materials.** *Beilstein J Nanotechnol.* 2011; **2**: 394–404.
PubMed Abstract | Publisher Full Text | Free Full Text
4. Müller-Caspary K, Krause FF, Grieb T, *et al.*: **Measurement of atomic electric fields and charge densities from average momentum transfers using scanning transmission electron microscopy.** *Ultramicroscopy.* 2017; **178**: 62–80.
PubMed Abstract | Publisher Full Text
5. Borghardt S, Winkler F, Zanolli Z, *et al.*: **Quantitative Agreement between Electron-Optical Phase Images of WSe_2 and Simulations Based on Electrostatic Potentials that Include Bonding Effects.** *Phys Rev Lett.* 2017; **118**(8): 086101.
PubMed Abstract | Publisher Full Text
6. Susi T, Madsen J, Ludacka U, *et al.*: **Efficient first principles simulation of electron scattering factors for transmission electron microscopy.** *Ultramicroscopy.* 2019; **197**: 16–22.
PubMed Abstract | Publisher Full Text
7. Madsen J, Susi T: **abTEM: *ab Initio* Transmission Electron Microscopy Image Simulation.** *Microscopy and Microanalysis.* 2020; **26**(Supplement S2): 448–450.
Publisher Full Text
8. Kirkland EJ: **Advanced Computing in Electron Microscopy.** SpringerLink: Springer e-Books, Springer US, 2010.
Publisher Full Text
9. Koch C: **Determination of Core Structure Periodicity and Point Defect Density along Dislocations.** PhD Thesis, Arizona State University, 2002.
Reference Source
10. Rosenauer A, Schowalter M: **STEMSIM—a New Software Tool for Simulation of STEM HAADF Z-Contrast Imaging.** In: Cullis, A. G. & Midgley P. A. (eds.) *Microscopy of Semiconducting Materials 2007. Springer Proceedings in Physics.* Springer Netherlands, Dordrecht. 2008; **120**: 170–172.
Publisher Full Text
11. Van den Broek W, Jiang X, Koch CT: **FDES, a GPU-based multislice algorithm with increased efficiency of the computation of the projected potential.** *Ultramicroscopy.* 2015; **158**: 89–97.
PubMed Abstract | Publisher Full Text
12. Allen LJ, Alfonso AJD, Findlay SD: **Modelling the inelastic scattering of fast electrons.** *Ultramicroscopy.* 2015; **151**: 11–22.
PubMed Abstract | Publisher Full Text
13. Lobato I, Van Dyck D: **MULTEM: A new multislice program to perform accurate and fast electron diffraction and imaging simulations using Graphics Processing Units with CUDA.** *Ultramicroscopy.* 2015; **156**: 9–17.
PubMed Abstract | Publisher Full Text
14. Singh S, Ram F, De Graef M: **EMsoft: open source software for electron diffraction/image simulations.** *Microscopy and Microanalysis.* 2017; **23**(S1): 212–213.
Publisher Full Text
15. Oelerich JO, Duschek L, Belz J, *et al.*: **STEMsalabim: A high-**

performance computing cluster friendly code for scanning transmission electron microscopy image simulations of thin specimens. *Ultramicroscopy.* 2017; **177**: 91–96.
PubMed Abstract | Publisher Full Text

16. Pryor A Jr, Ophus C, Miao J: **A streaming multi-GPU implementation of image simulation algorithms for scanning transmission electron microscopy.** *Adv Struct Chem Imaging.* 2017; **3**(1): 15.
PubMed Abstract | Publisher Full Text | Free Full Text

17. Barthel J: **Dr. Probe: A software for high-resolution STEM image simulation.** *Ultramicroscopy.* 2018; **193**: 1–11.
PubMed Abstract | Publisher Full Text

18. Radek M, Tenberge JG, Hilke S, *et al.*: **STEMcl-A multi-GPU multislice algorithm for simulation of large structure and imaging parameter series.** *Ultramicroscopy.* 2018; **188**: 24–30.
PubMed Abstract | Publisher Full Text

19. Brown H, Pelz P, Ophus C, *et al.*: **A Python Based Open-source Multislice Simulation Package for Transmission Electron Microscopy.** *Microscopy and Microanalysis.* 2020; **26**(S2): 2954–2956.
Publisher Full Text

20. Ophus C: **A fast image simulation algorithm for scanning transmission electron microscopy.** *Adv Struct Chem Imaging.* 2017; **3**(1): 13.
PubMed Abstract | Publisher Full Text | Free Full Text

21. Madsen J, Susi T, Janssen J: **jacobjma/abTEM 1.0.0beta14.** *Zenodo.* Version 1.0.0beta14, 2021.
http://www.doi.org/10.5281/zenodo.4767986

22. Madsen J, Pennycook TJ, Susi T: ***ab initio* description of bonding for transmission electron microscopy.** arXiv:2010.07145. 2020.
Reference Source

23. Frigo M, Johnson SG: **The Design and implementation of FFTW3.** *Proceedings of the IEEE.* 2005; **93**(2): 216–231.
Publisher Full Text

24. Doyle PA, Turner PS: **Relativistic Hartree-Fock X-ray and electron scattering factors.** *Acta Cryst Sec A.* 1968; **24**(3): 390–397.
Publisher Full Text

25. Lobato I, Van Dyck D: **An accurate parameterization for scattering factors, electron densities and electrostatic potentials for neutral atoms that obey all physical constraints.** *Acta Cryst Sec A.* 2014; **70**: 636–649.
Publisher Full Text

26. Weickenmeier A, Kohl H: **Computation of absorptive form factors for high-energy electron diffraction.** *Acta Cryst Sec A.* 1991; **47**: 590–597.
Publisher Full Text

27. Peng LM, Ren G, Dudarev SL, *et al.*: **Robust Parameterization of Elastic and Absorptive Electron Atomic Scattering Factors.** *Acta Cryst Sec A.* 1996; **52**: 257–276.
Publisher Full Text

28. Schwerdtfeger P: **The Pseudopotential Approximation in Electronic Structure Theory.** *ChemPhysChem.* 2011; **12**(17): 3143–3155.
PubMed Abstract | Publisher Full Text

29. Blöchl PE: **Projector augmented-wave method.** *Phys Rev B Condens Matter.* 1994; **50**(24): 17953–17979.
PubMed Abstract | Publisher Full Text

30. Mortensen JJ, Hansen LB, Jacobsen KW: **Real-space grid implementation of the projector augmented wave method.** *Phys Rev B.* 2005; **71**: 035109.
Publisher Full Text

31. Enkovaara J, Rostgaard C, Mortensen JJ, *et al.*: **Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method.** *J Phys Condens Matter.* 2010; **22**(25): 253202.
PubMed Abstract | Publisher Full Text

32. Nguyen K, Huang J, Karigerasi M, *et al.*: **Multislice Electron Scattering Simulations for Angstrom-scale Magnetic Measurements with 4D-STEM.** *Microscopy and Microanalysis.* 2020; **26**(Suppl S2): 22–23.
Publisher Full Text

33. Takahasi H, Mori M: **Double Exponential Formulas for Numerical Integration.** *Publications of the Research Institute for Mathematical Sciences.* 1973; **9**(3): 721–741.
Publisher Full Text

34. Bird DM, King QA: **Absorptive form factors for high-energy electron diffraction.** *Acta Crystallographica Section A.* 1990; **46**: 202–208.
Publisher Full Text

35. Senga R, Suenaga K, Barone P, *et al.*: **Position and momentum mapping of vibrations in graphene nanostructures.** *Nature.* 2019; **573**(7773): 247–250.
PubMed Abstract | Publisher Full Text

36. Hage FS, Radtke G, Kepaptsoglou DM, *et al.*: **Single-atom vibrational spectroscopy in the scanning transmission electron microscope.** *Science.* 2020; **367**(6482): 1124–1127.
PubMed Abstract | Publisher Full Text

37. Zeiger PM, Rusz J: **Efficient and Versatile Model for Vibrational STEM-EELS.** *Phys Rev Lett.* 2020; **124**(2): 025501.
PubMed Abstract | Publisher Full Text

38. Beyer A, Krause FF, Robert HL, *et al.*: **Influence of plasmon excitations on atomic-resolution quantitative 4D scanning transmission electron microscopy.** *Scientific Reports.* 2020; **10**(1): 17890.
PubMed Abstract | Publisher Full Text | Free Full Text

39. Mendis BG: **An inelastic multislice simulation method incorporating plasmon energy losses.** *Ultramicroscopy.* 2019; **206**: 112816.
PubMed Abstract | Publisher Full Text

40. Brown HG, Ciston J, Ophus C: **Linear-scaling algorithm for rapid computation of inelastic transitions in the presence of multiple electron scattering.** *Phys Rev Res.* 2019; **1**: 033186.
Publisher Full Text

41. Hage FS, Kepaptsoglou DM, Ramasse QM, *et al.*: **Phonon Spectroscopy at Atomic Resolution.** *Phys Rev Lett.* 2019; **122**(1): 016103.
PubMed Abstract | Publisher Full Text

42. Van Dyck D: **Is the frozen phonon model adequate to describe inelastic phonon scattering?** *Ultramicroscopy.* 2009; **109**(6): 677–682.
PubMed Abstract | Publisher Full Text

43. Forbes BD, D'Alfonso AJ, Findlay SD, *et al.*: **Thermal diffuse scattering in transmission electron microscopy.** *Ultramicroscopy.* 2011; **111**(12): 1670–1680.
PubMed Abstract | Publisher Full Text

44. Krause FF, Bredemeier D, Schowalter M, *et al.*: **Using molecular dynamics for multislice TEM simulation of thermal diffuse scattering in AlGaN.** *Ultramicroscopy.* 2018; **189**: 124–135.
PubMed Abstract | Publisher Full Text

45. Larsen AH, Mortensen JJ, Blomqvist J, *et al.*: **The atomic simulation environment-a Python library for working with atoms.** *J Phys Condens Matter.* 2017; **29**(27): 273002.
PubMed Abstract | Publisher Full Text

46. Plimpton S: **Fast Parallel Algorithms for Short-Range Molecular Dynamics.** *J Comput Phys.* 1995; **117**(1): 1–19.
Publisher Full Text

47. de la Peña F, Prestat E, Fauske VT, *et al.*: **hyperspy: Release v1.6.1**. 2020.
Publisher Full Text

48. bsavitzky, Zeltmann SE, Barnard E, *et al.*: **py4dstem/py4DSTEM: DOI release**. 2019.
Publisher Full Text

49. Lam SK, Pitrou A, Seibert S: **Numba: A LLVM-Based Python JIT Compiler.** In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC,* LLVM'15, Association for Computing Machinery, New York, NY, USA, 2015.
Publisher Full Text

50. Okuta R, Unno Y, Nishino D, *et al.*: **CuPy: A NumPy- Compatible Library for NVIDIA GPU Calculations.** In: *31st Conference on Neural Information Processing Systems (NIPS 2017)*. 2017.

51. Jupyter P, Bussonnier M, Forde J, *et al.*: **Binder 2.0 - Reproducible, interactive, sharable environments for science at scale.** *Proceedings of the 17th Python in Science Conference.* 2018.
Publisher Full Text

52. Martinez GT, Naginey TC, Jones L, *et al.*: **Direct Imaging of Charge Redistribution due to Bonding at Atomic Resolution via Electron Ptychography.** *arXiv:1907.12974,* 2019.
Reference Source

53. Jiang Y, Chen Z, Han Y, *et al.*: **Electron ptychography of 2D materials to deep sub-ångström resolution.** *Nature.* 2018; **559**(7714): 343–349.
PubMed Abstract | Publisher Full Text

54. Lazić I, Bosch EGT, Lazar S: **Phase contrast STEM for thin samples: Integrated differential phase contrast.** *Ultramicroscopy.* 2016; **160**: 265–280.
PubMed Abstract | Publisher Full Text

55. Maiden AM, Rodenburg JM: **An improved ptychographical phase retrieval algorithm for diffractive imaging.** *Ultramicroscopy.* 2009; **109**(10): 1256–1262.
PubMed Abstract | Publisher Full Text

56. Tersoff J: **Modeling solid-state chemistry: Interatomic**

potentials for multicomponent systems. *Phys Rev B Condens Matter.* 1989; **39**(8): 5566–5568.
PubMed Abstract | Publisher Full Text

57. Chuvilin A, Kaiser U: **On the peculiarities of CBED pattern formation revealed by multislice simulation.** *Ultramicroscopy.* 2005; **104**(1): 73–82.
PubMed Abstract | Publisher Full Text

58. Biskupek J, Chuvilin A, Kaiser U: **Evaluation of Frozen Phonons Models for Multislice Calculation of TDS.** *Microscopy and Microanalysis.* 2007; **13**(S03): 130–131.
Publisher Full Text

59. Miller B, Mick S: **Real-Time Data Processing using Python in DigitalMicrograph.** *Microsc Microanal.* 2019; **25**(Suppl 2): 234–235.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 2

Reviewer Report 21 May 2021

https://doi.org/10.21956/openreseurope.14825.r26947

© **2021 Ophus C.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

✔ **Colin Ophus**
National Center for Electron Microscopy, Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, CA, USA

All of my comments have been addressed. Great paper!

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Transmission electron microscopy, electron scattering simulation, data analysis, materials science.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 09 April 2021

https://doi.org/10.21956/openreseurope.14085.r26646

© **2021 Brown H.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

✔ **Hamish Brown**
[1] Advanced Microscopy Facility, Bio21 Molecular Science and Biotechnology Institute, University of Melbourne, Melbourne, Australia
[2] Advanced Microscopy Facility, Bio21 Molecular Science and Biotechnology Institute, University of

Melbourne, Melbourne, Australia

The field of TEM image simulation is now very mature with many free and open source simulation codes available. However many of these codes are written in low-level languages such as Fortran and C++ which means that they can be inflexible and difficult to modify, especially for the average microscopist who is not an expert programmer. Additionally there has been insufficient research on developing workflows for including the electrostatic potential of materials generated by "first principals" methods such as density functional theory (DFT) and molecular dynamics in TEM simulations. This paper addresses both of these challenges by introducing a Python TEM simulation toolkit that is straight-forward to use, implements more recent advances such as the reciprocal-space interpolated scattering matrix (PRISM) approach and Tanh-Sinh quadrature for the correct treatment of 3D potentials and interfaces very conveniently with the GPAW DFT library and the LAMMPS molecular dynamics packages.

The paper well written, with clear and concise discussions of the code furnished with instructive and relevant examples. The code itself is a well finished product and is intuitive in the way that it is put together with Python notebooks providing useful examples of how to run key simulations. I whole heartedly recommend indexing of the work with only a few minor points I would like to see addressed:

In the introduction suggest replacing "reaching nowadays sub-Ångström resolution" with "with sub-Ångström resolution now routinely achievable in modern aberration corrected instruments"

In the second to last paragraph of sec. 1, suggest replacing "such succession" with "Fourier transform"

In sec. 2.4 the brief description of the PRISM algorithm is a little vague (understandable given the complexity of the concept). Suggest changing to "the electron probe can be expressed using a plane-wave basis and the individual plane waves are propagated through the specimen using a pre-calculated scattering matrix and coherently summed at the exit surface to calculate the scattered probe". In the next sentence I suggest modifying "are then multiplied by a coefficient α_{nm}" by being explicit that α_{nm} is a complex component of the scattering matrix. A short sentence explaining why the introduction of a PRISM interpolation factor leads to a tiling out in real space, where keeping only every fth coefficient in Fourier space is equivalent to creating an f x f tiling of that function (I believe this is referred to as periodic summation) and hence the need for real space cropping, would be useful.

In sec 4.2 Defocus spread of conventional TEM images can be taken into account by integrating the measured intensity for a given defocus over the probability distribution of the defocus spread. See for example Eq. (3.20) in Kirkland's textbook, reference 8 in the manuscript. However the first code example of 4.2 seems to apply this spread to the complex electron wave function before the intensity is evaluated. For a weak phase object the two approaches should be equivalent however this is not true in general. Is this another example the quasi-coherent approximation discussed with respect to partial coherence earlier in this chapter?

In Sec 5.1 I was initially alarmed by the sentence "We create an orthogonal periodic unit cell of hBN using the ASE constructor for graphene", inspection of the code (hBN_DFT_IAM.ipynb) convinced me that indeed the graphene constructor is only used to create a unit cell structure,

since graphene shares the symmetry of hBN, and the atoms and unit cell are modified post-hoc to reflect the correct atomic structure of hBN. Suggest mentioning this in the text.

In section 5.2, I am curious as to how well probe correction worked in this simulated example. If the aberrations in the probe are discernable by eye it would be instructive to place the aberrated probes used in the simulation and reconstructed next to the ptychographic reconstruction.

In sec 2.3.2 the authors discuss how Tanh-Sinh quadrature is used to accurately evaluate integrals over the z coordinate, this was first discussed in the context of TEM simulation by Lobato and Van Dyck (reference 13 in the current manuscript) so work should also be referenced in this section.

In reference 48. the author name bsavitzky seems to be inappropriately formatted

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Competing Interests: Non-financial competing interests: I am author of the open-souorce py_multislice python TEM simulation package and contributor to the prismatic and muSTEM TEM simulation packages.

Reviewer Expertise: TEM simulation and experiment, condensed matter physics.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 18 May 2021

Toma Susi

Original comments by the reviewer are reproduced in *italics* below, followed by our point-by-point replies.

The field of TEM image simulation is now very mature with many free and open source simulation codes available. However many of these codes are written in low-level languages such as Fortran and C++ which means that they can be inflexible and difficult to modify, especially for the average microscopist who is not an expert programmer. Additionally there has been insufficient

research on developing workflows for including the electrostatic potential of materials generated by "first principals" methods such as density functional theory (DFT) and molecular dynamics in TEM simulations. This paper addresses both of these challenges by introducing a Python TEM simulation toolkit that is straight-forward to use, implements more recent advances such as the reciprocal-space interpolated scattering matrix (PRISM) approach and Tanh-Sinh quadrature for the correct treatment of 3D potentials and interfaces very conveniently with the GPAW DFT library and the LAMMPS molecular dynamics packages.
The paper well written, with clear and concise discussions of the code furnished with instructive and relevant examples. The code itself is a well finished product and is intuitive in the way that it is put together with Python notebooks providing useful examples of how to run key simulations. I whole heartedly recommend indexing of the work with only a few minor points I would like to see addressed:

We thank you very much for your evaluation and constructive comments. We provide point-by-point responses below.

In the introduction suggest replacing "reaching nowadays sub-Ångström resolution" with "with sub-Ångström resolution now routinely achievable in modern aberration corrected instruments"
In the second to last paragraph of sec. 1, suggest replacing "such succession" with "Fourier transform"

Thank you for these suggestions, the text has been revised.

In sec. 2.4 the brief description of the PRISM algorithm is a little vague (understandable given the complexity of the concept). Suggest changing to "the electron probe can be expressed using a plane-wave basis and the individual plane waves are propagated through the specimen using a pre-calculated scattering matrix and coherently summed at the exit surface to calculate the scattered probe". In the next sentence I suggest modifying "are then multiplied by a coefficient α_{nm}" by being explicit that α_{nm} is a complex component of the scattering matrix. A short sentence explaining why the introduction of a PRISM interpolation factor leads to a tiling out in real space, where keeping only every fth coefficient in Fourier space is equivalent to creating an f x f tiling of that function (I believe this is referred to as periodic summation) and hence the need for real space cropping, would be useful.

We agree that the section was difficult to understand for readers without prior knowledge of PRISM. We have now expanded it, incorporating many of the suggestions. We would argue that the scattering matrix, as defined in the PRISM paper, is just the set of exit waves produced by running the multislice algorithm on the set of plane waves present in the probe-forming aperture, and hence the coefficients, α_{nm}, are not as such components of the scattering matrix; this definition is slightly different from the original concept of the scattering matrix. But this is perhaps more a matter of semantics.

In sec 4.2 Defocus spread of conventional TEM images can be taken into account by integrating the measured intensity for a given defocus over the probability distribution of the defocus spread. See for example Eq. (3.20) in Kirkland's textbook, reference 8 in the manuscript. However the first code example of 4.2 seems to apply this spread to the complex electron wave function before the intensity is evaluated. For a weak phase object the two approaches should be equivalent however

this is not true in general. Is this another example the quasi-coherent approximation discussed with respect to partial coherence earlier in this chapter?

Yes, the quasi-coherent approximation is applying the focal spread to the wave function, which only works for a relatively small focal spread; abTEM is essentially using Eq. 3.37 in Kirkland's textbook. We have uploaded an example exploring the accuracy of the quasi-coherent approximation, where we see that it may be adequate for HRTEM, but, of course, very inadequate for CBED. It is possible, although currently a little complicated, to simulate partial coherence using the correct incoherent summation.
https://github.com/jacobjma/abTEM/blob/master/examples/other/quasicoherent_approximation.ipynb
https://github.com/jacobjma/abTEM/blob/master/examples/other/partial_coherence_4DSTEM.ipynb

In Sec 5.1 I was initially alarmed by the sentence "We create an orthogonal periodic unit cell of hBN using the ASE constructor for graphene", inspection of the code (hBN_DFT_IAM.ipynb) convinced me that indeed the graphene constructor is only used to create a unit cell structure, since graphene shares the symmetry of hBN, and the atoms and unit cell are modified post-hoc to reflect the correct atomic structure of hBN. Suggest mentioning this in the text.

This was admittedly confusing, but luckily as stated, the code is clear on this point. We have now also updated the text accordingly.

In section 5.2, I am curious as to how well probe correction worked in this simulated example. If the aberrations in the probe are discernable by eye it would be instructive to place the aberrated probes used in the simulation and reconstructed next to the ptychographic reconstruction.

In this simple example, our focus was on showing how easy it is to reproduce an experimental dataset from the literature with multiple detectors, and to create a corresponding purely simulated image. Thus we prefer not to add further information into that figure. However, we did use a probe_guess for ePIE that discarded the (in the simulation) known aberrations, but it did not appreciably change the final outcome whether we supplied full knowledge of the probe to the reconstruction or not.

In sec 2.3.2 the authors discuss how Tanh-Sinh quadrature is used to accurately evaluate integrals over the z coordinate, this was first discussed in the context of TEM simulation by Lobato and Van Dyck (reference 13 in the current manuscript) so work should also be referenced in this section.

Thank you for pointing out this significant oversight, the work by Lobato and Van Dyck has now been properly referenced.

In reference 48. the author name bsavitzky seems to be inappropriately formatted

Good catch, it has been fixed.

Competing Interests: No competing interests were disclosed.

Reviewer Report 09 April 2021

https://doi.org/10.21956/openreseurope.14085.r26649

© 2021 Ophus C. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Colin Ophus
[1] National Center for Electron Microscopy, Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[2] National Center for Electron Microscopy, Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, CA, USA

In this manuscript, Madsen and Susi present their image simulation code abTEM for transmission electron microscopy. This code has many things to recommend it:

- completely open source
- all python implementation for easy understanding and modification
- GPU acceleration for fast calculations
- support for calculating the electrostatic atomic potentials from first principles (to include bonding, charge transfer, etc.) through the GPAW DFT code.
- support for atomic cell construction and manipulation via the ASE code.
- implementation of both classic algorithms such as multislice and newer methods like PRISM.

This paper is very well written, and the codes are well documented and user-friendly. I had a lot of trouble in windows, but the binder notebooks all appear to work well. I tested as many of the paper use cases as I could – all worked well, though it would be helpful for the authors to include more sample code for different sample geometries. I should note that I attempted to break the code by checking sampling densities and slice thicknesses that were far outside the range of ordinary simulation parameters (both too large and too small), and I was unable produce results that seemed unphysical – this shows the robustness of the potential integration pipeline developed in this code.

I also think the implementation of PRISM on the GPU is very clever – the batching of adjacent probes is a very clean way to efficiently use however much GPU RAM is available. It's also pretty cool to see an all-python GPU implementation that is faster than Prismatic!

In summary, I recommend indexing of this work after minor revision. Below I have listed a few questions I have about the paper as well as minor comments that the authors are free to ignore.

Questions

- " ... allowing us to interpolate the measured images down to the sampling of the potential (0.04 Å) ..." How is the interpolation performed? Fourier space padding I assume? If it's real space, even bicubic interpolation may not be accurate enough.

- The images in Figure 2 look very strange to me. I checked the aspect ratio and it is correct

(though it's quite strange to label one axis in steps of 2.0 and the other in steps of 2.5 Angstroms). However, the atoms are definitely not round, which I assume is due an interpolation error. This can be seen if I saturate the image and draw a perfect circle: Link to image

- I also don't quite understand the results in Figure 2 d, e, f. The largest deviation in the BF signal is in the middle of the BN rings, furthest from the atomic potentials – I can rationalize this since this probe position is strongly affected by the degree of ionic screening. However, I would still like to see the quantitative intensity measurements, i.e. dimensions on the vertical axes of b, e, h in units the probe intensity.

- Figure 5a caption or the paper text should probably mention the image is HAADF (I assume) with the detector annular ranges.

Minor suggestions

- Figure 1 shows most of the important objects in abTEM. I would suggest adding a few more: the center-of-mass (along x and y direction) of the probe output intensity would make it easy to perform differential phase contrast (DPC) simulations (which might be already included as an output since DPC results are shown later in the paper). I would also like to see some kind of method for outputting a thickness series – it would be useful for researchers who wish to compare HRTEM images, single STEM probes, or PACBED images with experiments. Finally, it might be useful to include some kind of post-processing as a class, for example numerical integration over a range of angles or defocus values to model limited coherence (maybe this would belong in the Measurement class?).

- Is the choice to go with MPI fixed? I might recommend Dask instead of (or in addition to) since it seems to be easier to scale from local parallelization to large scale HPC parallelization.

- "... is encoded in the scattering of the electron waves by the electrostatic potential of the specimen." The electron wave can also scatter from electrostatic and magnetic fields too. I am particularly interested in using abTEM to simulate samples with significant charge transfer leading to long range electrostatic fields (in excess of the atomic potential due to the specimen). Note – reading further I see this possibility is covered in the manuscript in section 2.3.1.

- In the code blocks, many numerical terms are included without a leading zero, i.e. .5 or .1 rather than 0.5 and 0.1. The APA style guide recommends always including the leading zero for numbers than can exceed 1.0.

- In the show_atoms(...) command, I wanted to tilt the cell to arbitrary angles to make sure I was constructing my sample correctly. It's possible I am just missing something, but perhaps the argument "plane= arg" could be extended to allow arbitrary 3 element project vectors (u, v, w)?

- In the conclusion, the authors explicitly mention NION Swift's python implementation. The other vendors are significantly behind on python implementation, but have made some progress which could also be mentioned (e.g. GMS python scripting tinyurl.com/kykj38bj).

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Transmission electron microscopy, electron scattering simulation, data analysis, materials science.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 18 May 2021

Toma Susi

Original comments by the reviewer are reproduced in *italics* below, followed by our point-by-point replies.

In this manuscript, Madsen and Susi present their image simulation code abTEM for transmission electron microscopy. This code has many things to recommend it:

- *completely open source*
- *all python implementation for easy understanding and modification*
- *GPU acceleration for fast calculations*
- *support for calculating the electrostatic atomic potentials from first principles (to include bonding, charge transfer, etc.) through the GPAW DFT code.*
- *support for atomic cell construction and manipulation via the ASE code.*
- *implementation of both classic algorithms such as multislice and newer methods like PRISM.*

This paper is very well written, and the codes are well documented and user-friendly. I had a lot of trouble in windows, but the binder notebooks all appear to work well. I tested as many of the paper use cases as I could – all worked well, though it would be helpful for the authors to include more sample code for different sample geometries. I should note that I attempted to break the code by checking sampling densities and slice thicknesses that were far outside the range of ordinary simulation parameters (both too large and too small), and I was unable produce results that seemed unphysical – this shows the robustness of the potential integration pipeline

developed in this code.
I also think the implementation of PRISM on the GPU is very clever – the batching of adjacent probes is a very clean way to efficiently use however much GPU RAM is available. It's also pretty cool to see an all-python GPU implementation that is faster than Prismatic!
In summary, I recommend indexing of this work after minor revision. Below I have listed a few questions I have about the paper as well as minor comments that the authors are free to ignore.

We are very grateful for this very thorough and fair evaluation of our work! Below we address the points you raise in order.
" … allowing us to interpolate the measured images down to the sampling of the potential (0.04 Å) …" How is the interpolation performed? Fourier space padding I assume? If it's real space, even bicubic interpolation may not be accurate enough.
Indeed, bicubic interpolation is not accurate enough, we used biquintic interpolation, which seems to be good enough in this case. Your comment prompted us to implement the option of doing the interpolation with Fourier space padding. We see that it does perform a bit better than biquintic interpolation, as you may have anticipated. We have a small example in our repository that demonstrates how well interpolation compares to a finer scan sampling:
https://github.com/jacobjma/abTEM/blob/master/examples/other/is_nyquist_enough.ipynb

The images in Figure 2 look very strange to me. I checked the aspect ratio and it is correct (though it's quite strange to label one axis in steps of 2.0 and the other in steps of 2.5 Angstroms). However, the atoms are definitely not round, which I assume is due an interpolation error. This can be seen if I saturate the image and draw a perfect circle.

Thank you for flagging these issues. This kind of unequal adaptive tick spacing is unfortunately the default behavior in matplotlib. We have now implemented an "equal_ticks" argument for our show() methods, but leave this off by default since it will not play nicely with any scale updates. Perhaps a better solution can be found in the future, and we have updated the ticks of Figure 2 to have equal spacing.
Regarding the non-round atoms: this seems to have been an issue with the cell we used for this example being slightly too small to fit the probe; we have now tiled the potential to avoid this and have updated the figure.

I also don't quite understand the results in Figure 2 d, e, f. The largest deviation in the BF signal is in the middle of the BN rings, furthest from the atomic potentials – I can rationalize this since this probe position is strongly affected by the degree of ionic screening. However, I would still like to see the quantitative intensity measurements, i.e. dimensions on the vertical axes of b, e, h in units the probe intensity.

That is indeed the case, but not only due to ionic screening: charge rearrangement due to bonding is most easily visibly at positions furthest away from the atoms, as the differences to the independent atom model are – in relative terms – the largest there. We also observed the same for graphene in our previous study (doi:10.1016/j.ultramic.2018.11.002). In the case of hBN, there is also a noticeable difference in the BF intensity of the N site, where in the case of the *ab initio* potential, the greater electronegativity of N in the bonded ionic compound leads to significant charge transfer from the B atoms, enhancing the screening

of the N nucleus and thus lower its contrast (as originally shown in doi:10.1038/nmat2941). We have now added scales to the line plot vertical axes to facilitate quantitative comparisons, which better show the relatively small magnitudes of the differences. We would add though that this is a simple example meant to illustrate how easy it is to use *ab initio* potentials; in a real study one would want to properly converge the DFT parameters and use a more realistic probe!

Figure 5a caption or the paper text should probably mention the image is HAADF (I assume) with the detector annular ranges.

Indeed, this has been added.

Figure 1 shows most of the important objects in abTEM. I would suggest adding a few more: the center-of-mass (along x and y direction) of the probe output intensity would make it easy to perform differential phase contrast (DPC) simulations (which might be already included as an output since DPC results are shown later in the paper).

The SegmentedDetector object already allows you to run DPC simulations using a circular detector. We also implement a center_of_mass function which allows you to calculate the center of mass and integrated COM from a 4D-STEM dataset. However, having a dedicated center-of-mass detector is a good idea, since writing the intermediate results of huge 4D-STEM datasets to disc may be bottleneck for computational performance. The center-of-mass detector will be added to abTEM in the near future.

I would also like to see some kind of method for outputting a thickness series – it would be useful for researchers who wish to compare HRTEM images, single STEM probes, or PACBED images with experiments.

In the example linked below, we show how performing a PACBED thickness series might be accomplished. While we believe the current implementation may be preferred by experienced Python users due to the amount of flexibility it allows, we also see that the current option is not very beginner-friendly. In a future release, we will aim to develop a simpler and more compact solution.
https://github.com/jacobjma/abTEM/blob/master/examples/other/pacbed_thickness_series.ipynb

Finally, it might be useful to include some kind of post-processing as a class, for example numerical integration over a range of angles or defocus values to model limited coherence (maybe this would belong in the Measurement class?).

We uploaded an example showing how you might do this. Our above comment related to PACBED applies here as well, we will try to make the code for doing this more user-friendly.
https://github.com/jacobjma/abTEM/blob/master/examples/other/partial_coherence_4DSTEM.ipynb

Is the choice to go with MPI fixed? I might recommend Dask instead of (or in addition to) since it seems to be easier to scale from local parallelization to large scale HPC parallelization.

No, we actually reconsidered and decided to go with Dask shortly after the submission of

the article. We have started development on this, but it is not ready to include in this version of the code, but we expect to include it in the next major release.

"... is encoded in the scattering of the electron waves by the electrostatic potential of the specimen." The electron wave can also scatter from electrostatic and magnetic fields too. I am particularly interested in using abTEM to simulate samples with significant charge transfer leading to long range electrostatic fields (in excess of the atomic potential due to the specimen). Note – reading further I see this possibility is covered in the manuscript in section 2.3.1.

We believe you are thinking in the context of the independent atom model, where electrostatic potentials are something additional to the atomic potentials. In terms of the physics, the electrostatic potential of the specimen naturally includes all contributions, atomic or charge transfer leading to long-range electrostatic fields, whose natural inclusion is one of the benefits of using an *ab initio* description!

In the code blocks, many numerical terms are included without a leading zero, i.e. .5 or .1 rather than 0.5 and 0.1. The APA style guide recommends always including the leading zero for numbers than can exceed 1.0.

This has now been corrected.

In the show_atoms(...) command, I wanted to tilt the cell to arbitrary angles to make sure I was constructing my sample correctly. It's possible I am just missing something, but perhaps the argument "plane= arg" could be extended to allow arbitrary 3 element project vectors (u, v, w)?

You can now set the plane normal using the azimuth and elevation angles in a tuple of length two, which will give you a 3D projection of the structure. However, since matplotlib does not have a real 3D engine, this implementation is a bit constrained. You may want to try ASE's visualize.view() method, which opens an interactive 3D view. This also has some limitations, as it does not work seamlessly in remote environments and has trouble with large systems. In such cases you can use the nglview backend: with nglview installed you can specify "view(atoms, viewer="nglview)". We have made nglview available in our Binder environment.

In the conclusion, the authors explicitly mention NION Swift's python implementation. The other vendors are significantly behind on python implementation, but have made some progress which could also be mentioned (e.g. GMS python scripting tinyurl.com/kykj38bj).

That is a fair point, and GMS is now briefly mentioned in our conclusion

Competing Interests: No competing interests were disclosed.