JOURNAL OF APPLIED CRYSTALLOGRAPHY

ISSN 1600-5767

Received 30 April 2026
Accepted 19 July 2026

Edited by Th. Proffen, Oak Ridge National Laboratory, USA

Keywords: computational modeling; materials modeling; molecular simulation; nanostructure; materials science; nanoscience.

AES-Debye: an accurate, efficient and scalable engine for Debye scattering calculations

Navid Panchi,[a,b]* Sebastian Kuckuk,[b] Markus Wittmann,[b] Michael Engel[a] and Alberto Leonardi[c,d]*

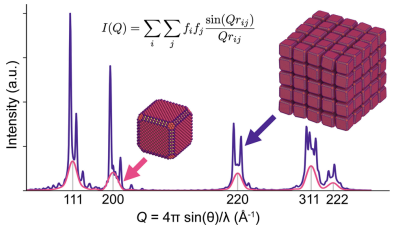
[a]Institute for Multiscale Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany, [b]Erlangen National High Performance Computing Center (NHR@FAU), Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany, [c]Technische Fakultät, Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany, and [d]UKRI-STFC Rutherford Appleton Laboratory, Diamond Light Source, Harwell Science and Innovation Campus, Didcot OX11 0DE, United Kingdom. *Correspondence e-mail: navid.panchi@fau.de, alberto.leonardi@fau.de

Total scattering models are essential for characterizing the structure and disorder of nanoscale materials. The Debye scattering equation (DSE) provides a rigorous route to elastic total scattering, but its direct evaluation is computationally demanding because pairwise contributions must be accumulated at every scattering vector, whereas common acceleration strategies based on binned pair-distance distributions or gridded fast Fourier transforms can introduce discretization and aliasing artifacts that compromise diffuse-scattering accuracy. Here, we present *AES-Debye*, an accuracy-preserving DSE framework that aggregates pair distances into a pair distribution function (PDF) using corrected bin centers and numerically robust accumulation to suppress discretization and summation errors. A data-locality-aware parallel design enables efficient execution on CPUs and GPUs. We demonstrate strong scalability by computing a high-resolution total scattering profile for a system of 90 million atoms, $(0.1\ \mu m)^3$, in minutes on a distributed-memory CPU platform. These capabilities extend accurate elastic total scattering calculations to large complex systems while simultaneously providing high-resolution PDFs for downstream structural analysis.

1. Introduction

Powder diffraction is widely used for characterizing the structure and microstructure of materials (Kaduk *et al.*, 2021). Experimental diffraction profiles are analyzed using a range of modeling approaches to extract statistically meaningful information about the sample. Extracting this information is particularly critical for nanostructured materials, where structural features such as crystalline domain size, domain shape and lattice distortions dictate macroscopic physical and chemical properties (Zhang *et al.*, 2011; Cernuto *et al.*, 2011; Zhang *et al.*, 2017). Because these characteristics manifest as line broadening and diffuse scattering, robust total scattering models are essential for reliable interpretation.

Traditionally, powder diffraction data have been analyzed using line profile methods such as Rietveld refinement (Rietveld, 1967) and whole powder pattern modeling (WPPM) (Scardi & Leoni, 2004). These approaches determine peak positions and relative integrated intensities from the crystal structure, while peak shapes are described by parameterized functions whose parameters are optimized against experimental data. Although computationally efficient, such methods rely on the Bragg approximation within long-range-ordered domains and therefore do not fully capture the



structural disorder responsible for diffuse scattering. Peak broadening from lattice distortions is typically incorporated through approximate models that are specialized to specific types of disorder (Gelisio & Scardi, 2016); only a limited number of such models are available for highly disordered materials.

To compute powder total scattering profiles while accounting for arbitrary structural disorder, two broad classes of approach are commonly used: calculations based on the Fourier transform of the scattering density, and evaluations of the Debye scattering equation (DSE). Fast Fourier transform (FFT)-based methods typically exhibit favorable asymptotic scaling, $O(M \log M)$ with the number of grid points M, and are widely used for diffraction calculations of large systems. However, for powder scattering the orientational averaging required to obtain the one-dimensional profile requires interpolation between Cartesian reciprocal-space grids and spherical shells (Ross *et al.*, 2014; Kieffer & Wright, 2013). Moreover, FFT-based approaches require discretization of the scattering density on a finite real-space grid, making the accuracy dependent on the chosen grid resolution and sampling strategy. With memory requirements a critical sampling factor, the mapping of the scattering intensities from Cartesian space onto radial bins involves a non-unique rebinning procedure, with the final profile depending on the specific interpolation and pixel weighting strategy employed. Crucially, eliminating directional sampling approximations during this step requires a number of unique angular projections that scales linearly with the number of atoms (Leonardi, 2021). Sampling these dense reciprocal-space directions explicitly forces the overall scaling back to $O(N^2)$, effectively neutralizing the efficiency gains of the underlying transform. Consequently, the powder pattern represents an approximation whose accuracy depends on the reciprocal-space sampling and integration strategy, with discrepancies generally becoming more pronounced at high Q. In contrast, the DSE operates directly on atomic coordinates and avoids density discretization, at the expense of a formally $O(N^2)$ pairwise summation (Debye, 1915). Without assuming periodicity or long-range order, it is well suited to disordered and nanoscale systems (Scardi & Gelisio, 2016; Gelisio & Scardi, 2016).

In its common form, the Debye scattering equation is given as

$$I(Q) = \sum_i \sum_j f_i f_j \frac{\sin(Q r_{ij})}{Q r_{ij}}, \tag{1}$$

where $r_{ij} = |\mathbf{r}_j - \mathbf{r}_i|$ is the interatomic distance and f_i and f_j are atomic scattering factors. The scattering vector magnitude is $Q = (4\pi/\lambda)\sin\theta$, where λ is the incident wavelength and θ is the scattering angle (Warren, 1990). This formulation mathematically enforces an exact orientational average of the powder profile. Consequently, while the DSE fully captures the internal structure and morphological anisotropy of individual nanoparticles, including finite size and shape effects, it does not natively describe directional correlations or texture, such as those arising in aligned systems under flow, external fields or deformation. Specialized extensions have been developed to incorporate preferred orientation into total scattering formalisms (Cervellino & Frison, 2020) which can be included in a future extension.

Evaluating equation (1) requires one $\mathrm{sinc}(x) = \sin(x)/x$ evaluation for every atomic pair, giving $O(N^2)$ complexity for N atoms. Since $I(Q)$ is typically sampled on many Q points, the total cost increases to $O(\ell N^2)$, where ℓ denotes the number of sampled Q values. In practice, this cost is compounded by the large number of transcendental evaluations and by floating point accumulation of terms spanning a wide range of magnitudes, which can introduce non-negligible numerical error.

A variety of strategies have been explored to improve efficiency:

(i) *Algorithmic modifications* reduce computational cost by changing the underlying physics or sampling strategy. These include pair distance cutoffs (*Debyer*, https://github.com/wojdyr/debyer), which introduce arbitrary correlation length and shape artifacts; golden ratio orientation sampling for powder averaging (Watson & Curtis, 2013), which is most accurate at low scattering angles; and methods that exploit lattice periodicity (Grover & McKenzie, 2001; Thomas, 2010; Leonardi, 2021), which are inherently unsuitable to disordered systems.

(ii) *Full Debye methods* include brute-force summation, sometimes accelerated on GPUs (Gelisio *et al.*, 2010; Johansen *et al.*, 2024), as well as pair distribution function (PDF)-based approaches in which a PDF is constructed first and then the scattering intensity is computed (Neverov, 2017; Rudolph *et al.*, 2019; Reuter & Köfinger, 2019).

However, many existing implementations gain speed at the expense of accuracy by relying on coarse histograms or 32-bit arithmetic, and they remain susceptible to numerical errors associated with floating point accumulation and binning approximations (Hall & Monot, 1991).

Leonardi & Bish (2016) improved numerical accuracy through fine binning, dynamic bin center correction and integer arithmetic to suppress floating point noise. However, their implementation (*Rose-X*) exhibits poor performance in disordered systems because random access into the PDF degrades memory locality and cache efficiency. Building on these observations, we identify the principal bottlenecks and redesign the computation around domain decomposition to achieve more local and predictable data access. The resulting engine, *AES-Debye*, retains the numerical rigor of *Rose-X* while introducing a hybrid OpenMP/MPI/CUDA parallel framework for CPUs and GPUs, yielding speedups of up to 20×.

2. Implementation details

This section describes *AES-Debye*, whose design is centered on three objectives: accuracy, efficiency and scalability. *Accuracy* is achieved through a PDF-based formulation with bin center correction and precision-aware integer accumulation, which suppresses discretization and summation artifacts.

Efficiency is improved through cache-friendly data layouts, compact histograms, strip mining, and a cell-list-based domain decomposition that enhances localized memory locality and reduces latency. *Scalability* is provided by a hybrid OpenMP/MPI/CUDA design that exploits shared memory, and device-level and distributed memory parallelism across CPUs, GPUs and multi-node systems.

2.1. PDF formulation with bin center correction

We reformulate equation (1) by grouping recurring pair distances into a PDF, allowing the DSE to be evaluated more efficiently. Expressing the intensity in terms of representative distances ν_k and their multiplicities N_k gives

$$I(Q) = \sum_a \sum_b \left[f_a f_b \sum_k N_k \frac{\sin(Q\nu_k)}{Q\nu_k} \right], \tag{2}$$

where a and b denote atomic species.

This two-step formulation decouples the atomic pair enumeration from the reciprocal-space evaluation. Constructing the PDF still requires computing all interatomic distances and therefore scales as $O(N^2)$. Once the histogram has been built, however, evaluating equation (2) scales only as $O(\ell n_{\rm bins})$, where ℓ is the number of sampled Q points and $n_{\rm bins}$ is the number of populated histogram bins. The PDF-based approach is therefore advantageous whenever $N^2 + \ell n_{\rm bins} < \ell N^2$. In practice, for large systems and dense Q grids, $n_{\rm bins} \ll N^2$, so the cost of reciprocal-space evaluations is reduced substantially relative to direct pairwise summation at every Q point.

Distinct pair distances are accumulated into uniformly spaced histogram bins of width Δ. The choice of Δ determines the numerical resolution and introduces artifacts at Q intervals proportional to $1/\Delta$. A second source of error arises when the bin center ν_k is a poor proxy for the average distance of the pairs assigned to that bin, which can lead to unphysical negative intensities (Hall & Monot, 1991).

Leonardi & Bish (2016) addressed this by storing, in addition to the bin counts, the accumulated difference between the squared pair distances and the squared bin center. After the PDF is constructed, this error in squared pair distance (ESPD) is used to correct the bin center so it more accurately represents the average distance within the bin. Let ρ be the corrected center and ν the original center, and define $\delta = \rho - \nu$ and $\psi = \rho^2 - \nu^2$. From $(\nu + \delta)^2 - \nu^2 = \psi$, one obtains

$$\delta^2 + 2\nu\delta - \psi = 0, \tag{3}$$

which admits a closed-form solution for bins containing a single (monodisperse) distance.

For bins containing multiple contributing distances, a series expansion provides an accurate estimate of the mean shift,

$$\langle \delta_k \rangle = \frac{1}{N_k} \left[\frac{\psi_k}{2\nu_k} - \frac{\psi_k^3}{4\nu_k^2} + \frac{\psi_k^4}{8\nu_k^3} + O(\ldots) \right], \tag{4}$$

where N_k is the number of pairs in bin k and ψ_k is the accumulated ESPD. The corrected representative distance is $\hat{\nu}_k = \nu_k + \langle \delta_k \rangle$, which replaces ν_k in equation (2). Full derivations and error propagation details are given by Leonardi & Bish (2016).

2.2. Numerical precision and accumulation

Double-precision floating point (`double`) provides a wide dynamic range (up to 10^{308}) but limited mantissa precision: integers are represented exactly only up to $2^{53} \simeq 9 \times 10^{15}$. Beyond this threshold, intermediate values can no longer be represented exactly, which compromises the accuracy of large accumulations. Because bin center correction depends on precise tracking of the squared error term ψ, we perform all bin indexing and ψ accumulations in 64-bit integers (`int64`).

Atomic positions are first normalized to the simulation box, mapped to the interval [0, 1), scaled by 10^9 and stored as `int64`. This preserves the required spatial precision. For example, coordinates typically contain no more than six significant digits in common simulation outputs from *LAMMPS* (Thompson *et al.*, 2022) and *HOOMD-blue* (Anderson *et al.*, 2020). The resulting maximum squared interatomic distance is of the order of $(10^9)^2$, which remains well within the `int64` range, so ψ can be accumulated exactly up to `INT64_MAX`. To handle extreme cases safely, we record potential over- and underflow events in an auxiliary PDF. This introduces one additional conditional per update while preserving correctness.

For uniformly spaced PDF bins, the worst-case accumulation occurs in the final bin, where the center ν is farthest from the upper boundary $\nu_{\rm ul}$. If all contributing pairs lie exactly at $\nu_{\rm ul}$, the maximum safe count is

$$n_{\rm max} = \frac{\texttt{INT64\_MAX}}{\nu_{\rm ul} - \nu}, \tag{5}$$

which is typically $\sim 5.6 \times 10^6$ for the bin widths used here. Because explicit overflow checks on every update would introduce branching and misprediction overhead, we instead maintain a per-bin counter initialized to $n_{\rm max}$ and decrement it with each update. When the counter reaches zero, a single overflow check is performed on ψ. If an overflow is detected, it is recorded and the counter is reset to $n_{\rm max}$; otherwise the counter is refreshed according to the remaining safe headroom. This strategy preserves numerical robustness while keeping runtime overhead low.

2.3. Data structures and memory layout

Atomic positions are stored in a structure-of-arrays (SoA) layout, with three separate arrays for x, y and z, so that sequential coordinate access $(i, i+1,)$ remains cache friendly and does not become a bottleneck. Profiling with Intel *VTune* shows that the dominant CPU limitation is latency from random updates to the PDF histogram; the same access pattern is also a major performance limiter on GPUs. Because memory access becomes progressively slower from L1 to L2 to L3 caches and finally to DRAM, large high-accuracy PDFs often exceed L1/L2 capacity and reside in L3 or main memory. Structural disorder worsens this effect by spreading updates

across many bins and thereby increasing the number of random accesses. As a result, crystalline systems run fastest, whereas disordered systems tend to become latency bound.

To mitigate this behavior, we employ two complementary strategies:

(i) *Strip mining (CPU only).* We buffer bin indices k and ESPD values ψ in two small contiguous arrays of size 512: k is stored in 32-bit unsigned integers (`uint32`) and ψ in `int64`. Once full, the buffers are flushed to the global PDF in a dedicated update loop. This form of write combining reduces random stores and improves cache locality, consistent with earlier observations (Reuter & Köfinger, 2019).

(ii) *Reducing memory footprint.* To lower access latency, each PDF bin is compacted to 128 bits: `int64` for ψ, together with `int32` values for the count N_k and the per-bin headroom counter $n_{\max}$. When $n_{\max}$ reaches zero, an overflow or underflow check is triggered for ψ; if necessary, the current N_k is offloaded to a secondary `int64` accumulator. On GPUs, pair count overflows are instead tracked with a separate auxiliary counter to avoid large scattered writes.

2.4. Cell-list-based domain decomposition

To improve memory access patterns during PDF construction, we partition the simulation box into a regular grid of equally sized cells. A prescribed number of cells is defined along each spatial dimension and atoms are assigned to cells according to their position. The coordinates are reordered by cell so that atoms belonging to the same cell occupy contiguous memory. For spatially non-uniform systems, the number of cells along each axis can be scaled in proportion to the corresponding system dimensions, promoting a more even distribution of atom pairs across cells and thereby improving load balance.

We then construct a grid of cell centers and compute center-to-center distances for all unique cell pairs, storing these distances together with the corresponding pair indices. The resulting list is sorted by distance, as illustrated in Fig. 1.

The PDF is computed by iterating over the sorted cell pair list and accumulating contributions from all atomic pairs associated with each cell pair. Because the list is ordered by center-to-center distance, successive iterations tend to update nearby histogram regions, which improves cache reuse and reduces memory latency. This strategy is particularly effective for large or disordered systems, where otherwise random bin updates would dominate runtime.

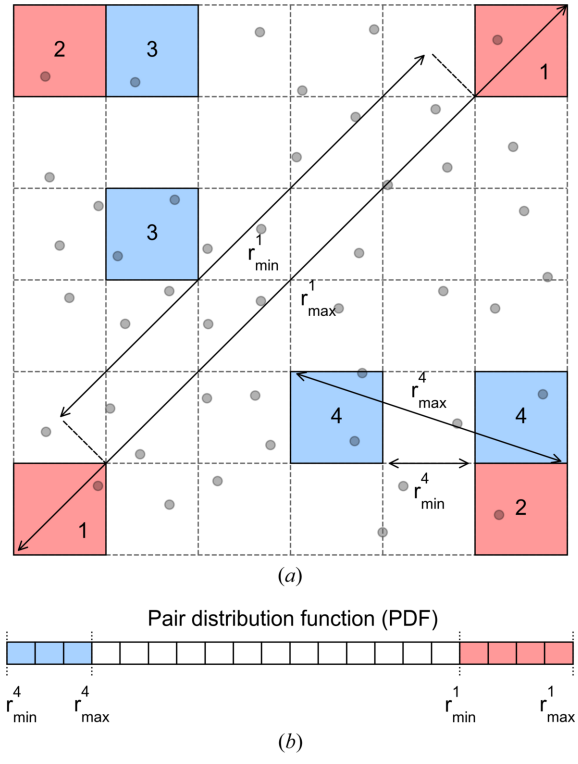


Figure 1
Schematic diagram of the cell-list-based domain decomposition in a 2D toy system. (Top) Representative cell pairs are color coded and grouped by center-to-center distance; the minimum ($r_{\min}$) and maximum ($r_{\max}$) possible interatomic distances within a pair are indicated. (Bottom) Memory access pattern during PDF computation. Grouping cell pairs with similar center-to-center distances improves cache reuse and localizes updates to the nearby PDF bin.

2.5. Parallelization

We accelerate PDF construction through parallelization (Fig. 2), using both CPU and GPU backends to support a wide range of hardware platforms. Although the library is designed for supercomputing clusters, it also runs efficiently on conventional workstations, desktops and laptops.

2.5.1. Shared memory (OpenMP)

On CPUs, we parallelize the loop over the sorted cell pair list. Each thread accumulates PDF contributions from its assigned cell pairs into a private buffer, thereby avoiding race

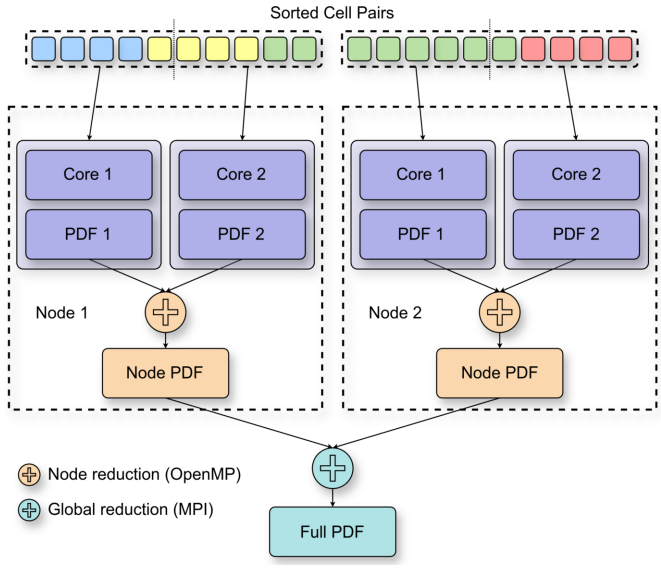


Figure 2
CPU parallelization scheme using a hybrid MPI + OpenMP model. Each node runs a single MPI process that spawns multiple OpenMP threads (one per core in this example). This schematic shows two MPI processes, each with two OpenMP threads.

conditions; the thread-local PDFs are combined in a final bin-wise reduction. OpenMP is used for portability and compatibility with major C++ compilers. The core loop for CPU-based PDF construction is shown in Algorithm 1.

Algorithm 1 CPU-based PDF computation using cell pairs

```
N_cell-pairs ← number of cell pairs
for m, n ≤ N_cell-pairs do                          ▷ Parallel loop
    N_a ← number of atoms of species a in cell m
    N_b ← number of atoms of species b in cell n
    n_bins ← number of bins in the PDF
    r_max ← maximum pair distance
    for i ≤ N_a do
        for j ≤ N_b do
            δr = r_j − r_i
            k = ⌊||δr|| n_bins/r_max⌋
            N_k += 1
            ψ_k += (δr · δr − ν_k^2)
        end for
    end for
end for                                             ▷ Reduce all thread-local PDFs
```

2.5.2. GPU acceleration (CUDA)

GPU parallelization is implemented using custom CUDA kernels. Kernel launch parameters are selected to maximize occupancy while respecting register and shared memory limits. Each thread evaluates a pairwise distance and updates the global PDF structure in device memory. In contrast to the CPU implementation, a private per-thread buffer is impractical on GPUs because of the very large number of concurrent threads; therefore synchronization is handled through atomic updates to shared global memory locations.

Alongside a baseline kernel in which each thread performs both bin counting and ψ accumulation, we implement a coalesced-access variant. In this variant, two consecutive threads operate on the same atomic pair: one updates the bin count and the other updates ψ. This organization improves memory throughput and delivers better performance across different GPU architectures.

The intensity evaluation is likewise parallelized on both CPUs and GPUs. On CPUs, reductions are used to avoid atomic writes. On GPUs, the work is distributed over the intensity array, with each thread computing a single intensity value by iterating over all PDF bins. This removes the need for atomic operations in the reciprocal-space evaluation.

2.5.3. Distributed memory (MPI)

To enable execution across multiple nodes or GPUs in a supercomputing environment (Table 1), we implement distributed memory parallelism with the message passing interface (MPI). The sorted cell pair list is divided among MPI processes, each of which executes its assigned workload independently, while OpenMP threads provide additional shared memory parallelism. After thread-local reduction, a global MPI reduction merges the partial PDFs into the final result. Although this collective communication introduces overhead, it is negligible for the large systems targeted here.

Table 1
Parallelization strategies supported for different hardware configurations.

Hardware	Parallelization strategy
Single CPU	OpenMP
Multiple CPUs	MPI + OpenMP
Single GPU	CUDA
Multiple GPUs	MPI + CUDA

For multi-GPU configurations, we assign one MPI process to each GPU. Since the current GPU implementation does not use cell lists for workload decomposition, the outer loop of the pair distance computation is partitioned across MPI processes instead. Each GPU evaluates its assigned segment independently, and a final reduction step merges the partial PDFs into a single global result. Further details on optimization strategies that were explored but proved ineffective are given in Appendix *A*.

3. Accuracy and numerical validation

Establishing a reliable reference for accuracy validation is challenging for large-scale Debye scattering calculations. While direct brute-force evaluation of the Debye sum is formally exact, for the system sizes considered here it can be affected by significant numerical summation errors, particularly at small scattering vector magnitudes where cancelation effects become severe. To assess the accuracy of the PDF-based implementation, we therefore compared the computed intensity profiles against those obtained with *Rose-X* (Leonardi & Bish, 2016) [Fig. 3(*a*)]. Although *Rose-X* employs approximations, it provides quantitative estimates of its calculation error, making it a suitable high-accuracy benchmark for large-scale validation. The two profiles are nearly indistinguishable, with a maximum relative error of $\sim 1.3 \times 10^{-7}$. This is well within the error limits calculated using *Rose-X* through full error propagation, thus confirming that *AES-Debye* reproduces the numerical accuracy of the *Rose-X* formulation while enabling the architectural and performance improvements described above.

To illustrate the role of bin center correction, we computed the powder diffraction profile of a crystalline test system using two versions of the PDF: one with corrected bin centers and one without. As shown in Fig. 3(*b*), the corrected profile eliminates unphysical intensities, whereas the uncorrected profile exhibits several. More importantly, the two profiles diverge in the diffuse background region between the peaks. Because diffuse scattering carries information about disorder, inaccuracies in this regime can bias the interpretation of both crystalline and disordered materials. Bin center correction is therefore not merely a cosmetic improvement but an essential ingredient for high-accuracy total scattering calculations.

4. Performance benchmarks

We benchmarked *AES-Debye* on cubic Pd face-centered cubic (f.c.c.) crystals with edge lengths from 4 to 40 nm, corresponding to systems ranging from 4000 to 4 million atoms. All

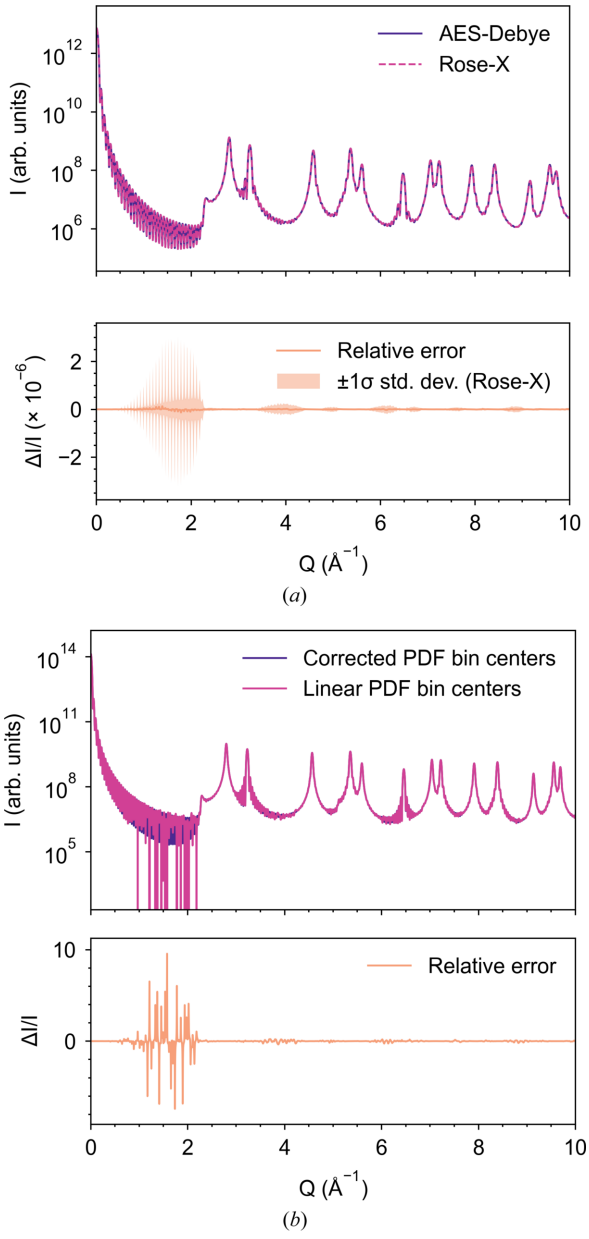


Figure 3
Validation of the PDF-based implementation and bin center correction. (*a*) Comparison of the intensity profile computed with *Rose-X* and with the present implementation for a Pd nanocube (side length 9.3 nm). The lower panel shows the relative error between the two profiles, together with the standard deviation of the intensity error estimated from *Rose-X*. (*b*) Powder diffraction profiles computed with and without bin center correction for a crystalline Pd nanocube (side length 15.1 nm). The correction removes unphysical negative intensities and improves the diffuse background.

runs used the same Q grid, PDF bin width Δ and incident wavelength to ensure direct comparability. CPU throughput is reported in million pair distances per second per core (MPD s^{-1} $core^{-1}$) and GPU throughput in billion pair distances per second (BPD s^{-1}).

To probe sensitivity to structural disorder, we use the Debye–Einstein model (Vaccari & Fornasini, 2006) by adding Gaussian displacements with standard deviation σ to the ideal

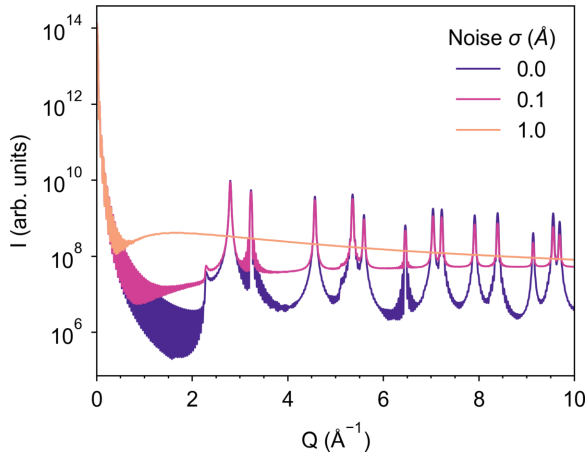


Figure 4
Diffraction profiles for a Pd nanocube (side length 10 nm) with increasing Gaussian displacement noise σ, used to mimic thermal disorder. Increasing σ broadens and attenuates the Bragg peaks, consistent with increasing structural disorder. The high-frequency oscillations in the background are expected finite-sized features associated with the parallel facets of the cube and monodisperse size, rather than numerical artifacts.

lattice positions, varying σ from 0 Å (perfect crystal) to 1 Å. While this approach adds uncorrelated noise directly to perfectly crystalline systems, we verified that introducing structural disorder via molecular dynamics simulations, which captures short-range correlations, produces comparable effects. This provides a controlled spectrum of workloads, from cache-friendly crystalline systems to latency-dominated disordered ones, and is used throughout the scaling studies. The corresponding evolution of the diffraction profile is shown in Fig. 4.

4.1. CPU scaling

CPU benchmarks were performed on a single node of the Fritz cluster at the Erlangen National High Performance Computing Center (NHR@FAU) (https://doc.nhr.fau.de/), using dual-socket Intel Xeon Platinum 8360Y processors (36 cores per socket, 54 MB L3 per socket, SMT disabled). All runs were executed at a fixed frequency of 2.2 GHz to eliminate variability from turbo boost and dynamic frequency scaling. The code was compiled with Intel oneAPI C++ (2023).

Fig. 5(*a*) shows the per-core PDF throughput as a function of system size. For the baseline implementation without cell lists, crystalline systems reach a maximum near 500000 atoms and then decline, whereas disordered systems saturate earlier at substantially lower throughput. This drop reflects increasing cache pressure as more PDF bins are updated and the working set spills beyond L3 cache into main memory.

The cell-list implementation largely removes this large-size degradation. Its throughput increases with system size, surpasses the baseline at approximately 750000 atoms and approaches the single-core roofline for large crystalline structures. For both crystalline and disordered systems, the cell-list implementation overtakes the baseline and sustains higher throughput across the full benchmark range. The

precise crossover depends on the system size, cell count, disorder level and CPU architecture.

Strong-scaling results are shown in Fig. 5(*b*). The cell-list implementation consistently outperforms the baseline and improves the scaling in both crystalline and disordered cases, with speedups of up to 1.5× and 5.5×, respectively. In the baseline implementation, the performance of disordered systems degrades with increasing core count because more threads contend for shared L3 cache resources. By localizing bin updates, the cell-list implementation improves cache reuse and reduces traffic, substantially reducing the loss.

4.2. GPU scaling

GPU benchmarks were performed on NVIDIA A100 and A40 devices on the Alex cluster at NHR@FAU (https://doc.nhr.fau.de/). The A100 provides 40 GB HBM2e memory with 108 SMs, while the A40 provides 48 GB GDDR6 memory with 84 SMs. The GPU implementation was compiled with CUDA 12.4.

On both GPUs, throughput increases with system size and then saturates. We also compared kernels with coalesced and non-coalesced memory access patterns. On the A40, crystalline systems achieve up to 30 BPD s^{-1} [Fig. 5(*c*)], but the throughput in disordered systems falls to less than half this value, reflecting the limited 6 MB L2 cache and the resulting sensitivity to irregular memory access. In the strongly disordered regime, A40 performance becomes comparable to that of a single Fritz node with 72 cores tested in the previous section.

On the A100, throughput reaches 40 BPD s^{-1} for crystalline systems and 43 BPD s^{-1} for disordered systems [Fig. 5(*d*)], highlighting the benefit of the larger 25 MB L2 cache under irregular memory access. Coalesced access consistently outperforms non-coalesced access in crystalline systems despite the additional arithmetic, because the gain in memory efficiency outweighs the extra work. In disordered systems, the A100 shows similar performance for both access modes, whereas the A40 performs worse with the coalesced variant, indicating stronger cache contention on that architecture. For this reason, both kernel variants are provided.

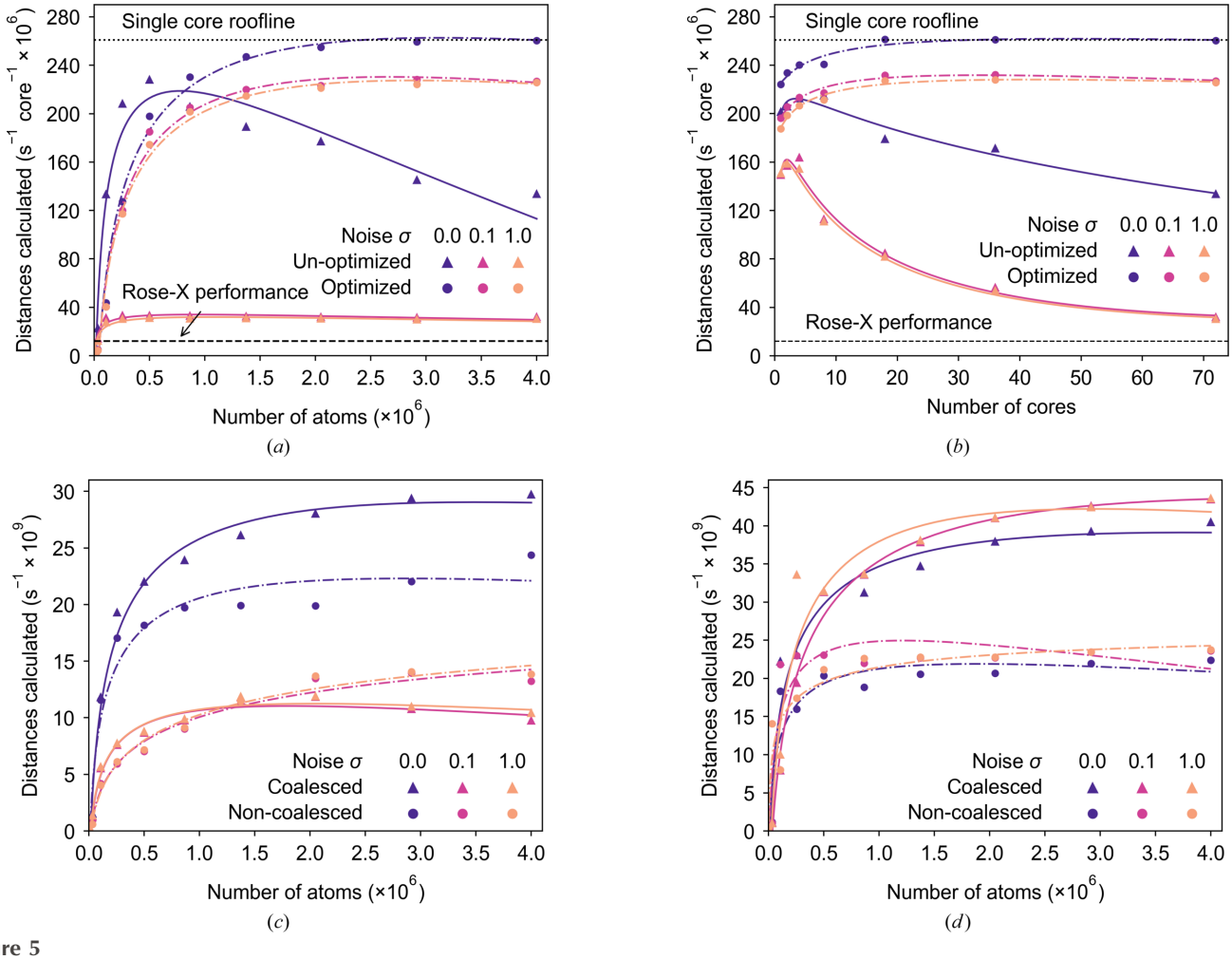


Figure 5
Performance scaling across CPUs and GPUs for different disorder levels σ. (*a*) Throughput versus atom count on 72 CPU cores; the dashed lines indicate the single-core roofline and a comparison with the existing earlier implementation in *Rose-X*. (*b*) Strong scaling on a single CPU node. (*c*) Asymptotic performance on an NVIDIA A40 (48 GB GDDR6, 6 MB L2 cache). (*d*) Asymptotic performance on an NVIDIA A100 (40 GB HBM2e, 25 MB L2 cache).

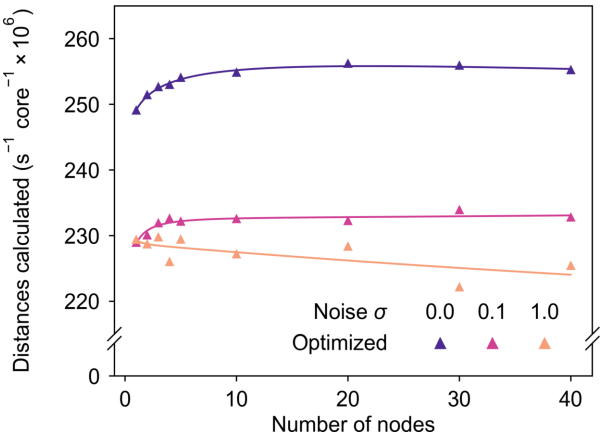


Figure 6
Strong-scaling performance as a function of node count for a four-million atom system. Only the optimized implementation is shown. The performance loss due to communication overhead remains minimal (<5%) over the tested range.

4.3. Node scaling (MPI)

To assess strong scaling across multiple nodes, we computed a diffraction profile for a large atomistic model (Fig. 6). The implementation scales efficiently up to 40 nodes on the Fritz cluster because the dominant cost, pair accumulation, is distributed across MPI rank, with communication limited to a final global reduction.

For load balancing, we estimate the work associated with each cell pair from the product of the occupancies of the two cells. The sorted cell pair list is then partitioned into contiguous chunks with approximately equal estimated cost, and these chunks are assigned to MPI ranks. This strategy provides good load balance even for disordered systems.

A modest performance loss with increasing node count is unavoidable because serial component reduction overheads become more visible as the parallel portion shortens. The degradation is slightly more pronounced for disordered systems, reflecting a residual imbalance that is difficult to eliminate fully with a simple contiguous partitioning of the cell pair list. Nevertheless, the penalty remains below 5% over the tested range.

For workloads consisting of many trajectory frames but only moderate system sizes, throughput is better improved through ensemble parallelism: using fewer nodes per frame and processing multiple frames concurrently.

4.4. Performance comparison

Cross-code comparisons are necessarily approximate because implementations differ in numerical choices, execution models and hardware targets. As a representative reference, we compared *AES-Debye* with *DebyeCalculator* (Johansen *et al.*, 2024), a recent brute-force implementation designed primarily for GPU execution. All benchmarks were performed on a single NVIDIA A40 GPU and a single AMD EPYC 7713 (Milan, Zen 3) CPU.

Table 2
Throughput in million pair distances per second (MPD s^{-1}) for a Pd nanoparticle with 32 000 atoms and Gaussian displacements of standard deviation σ = 0.01 Å.

Because *DebyeCalculator* does not support substantially large systems, the comparison is performed near the upper end of the problem sizes reported by Johansen *et al.* (2024).

Implementation	CPU	GPU
DebyeCalculator	0.14 (128 threads)	6.5
AES-Debye	371 (8 threads)	1531

As summarized in Table 2, *AES-Debye* achieves substantially higher throughput than *DebyeCalculator* on this benchmark. On the A40 GPU, *AES-Debye* reaches 1531 MPD s^{-1}, in contrast to 6.5 MPD s^{-1} for *DebyeCalculator*, corresponding to a speedup of approximately 235×. On the CPU, *AES-Debye* reaches 371 MPD s^{-1} using eight threads, whereas *DebyeCalculator* reports 0.14 MPD s^{-1} with 128 threads. Although these CPU runs use different thread counts and therefore should not be interpreted as a strictly controlled scaling comparison, the absolute performance gap remains very large. Overall, *AES-Debye* outperforms *DebyeCalculator* by a wide margin on both CPUs and GPUs for this problem size.

5. Applications

To complement the synthetic benchmarks, we present four representative applications that showcase the scalability and efficiency of the *AES-Debye* implementation. Nanocube superlattices highlight the value of the Debye equation for hierarchical self-assembly by resolving how superlattice ordering modifies the wide-angle atomic scattering signal (Toso *et al.*, 2019). Polycrystalline copper aggregates probe both size scaling and computational performance in realistic atomistic microstructures generated from simulations. Halloysite nanotubes test the method on non-periodic curved geometries with multi-component chemistry. Colloidal nanoparticle assemblies demonstrate the length-scale independence of the approach by extending its application beyond the atomic scale to simulate small-angle scattering during crystallization. Across all four cases, *AES-Debye* yields high-resolution powder profiles, and partial PDFs where relevant, with high numerical fidelity, demonstrating how its computational advantage translates into practical capabilities for materials systems of direct experimental interest.

5.1. Nanocube superlattices

To illustrate the ability of *AES-Debye* to probe hierarchical self-assembly, we computed diffraction profiles for finite nanocube superlattices. Such assemblies can require specialized treatment for each case, whereas the DSE naturally captures coherence across multiple length scales.

We constructed truncated nanocubes with a 5 nm edge length (∼20% truncation) and arranged them on a simple cubic lattice with a surface-to-surface separation of 0.5 nm to

mimic a self-assembled superlattice [Fig. 7(*a*)] The diffraction profile of the full assembly differs markedly from that of an isolated nanocube [Fig. 7(*b*)]. Because the particles are small and closely packed, interparticle correlations affect not only the low-Q superlattice reflections but also the wide-angle scattering signal. The high-frequency serrations in the background are expected finite-sized features of the idealized nanocubes and are enhanced by the absence of size dispersity.

Fig. 7(*b*) also shows that increasing the size of the primary nanocrystals reduces the influence of the assembly on the wide-angle profile. Introducing positional and rotational disorder into the superlattice similarly suppresses these assembly-induced features, bringing the profile closer to that of the isolated particle. Together, these results show that *AES-Debye* can resolve how hierarchical ordering and disorder shape total scattering signatures in finite nanoparticle assemblies.

Table 3
Corresponding runtimes for varying numbers of grains and atoms.

No. of grains	500	1000	2000
No. of atoms (million)	22.4	44.5	90.1
Time (s)	197	766	3159

5.2. Polycrystalline copper sample

To demonstrate the performance of the final implementation on realistic large-scale microstructures, we computed diffraction profiles for a series of polycrystalline Cu models. Each structure consisted of multiple randomly oriented grains generated with the constrained modified Voronoi tessellation (CMVT) method (Leonardi *et al.*, 2013) and subsequently relaxed by molecular dynamics [Fig. 8(*a*)] (Leonardi & Bish, 2017). Time-averaged atomic positions were used as input. Calculations were performed on 64 CPU nodes of the Fritz cluster (4608 cores total) using the local PDF mode and 25 cells per spatial direction.

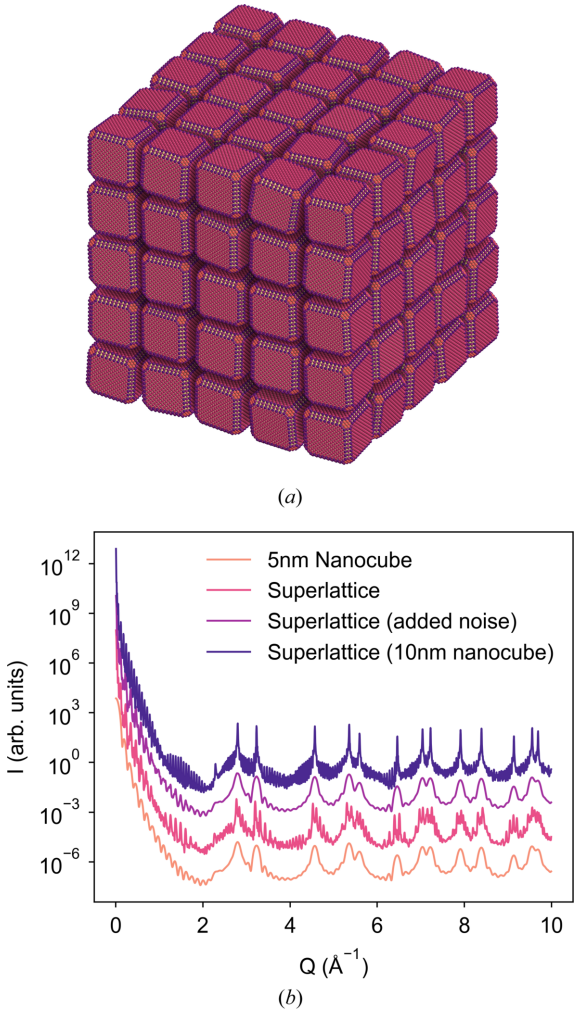


Figure 7
(*a*) Pd nanocubes (5 nm) assembled into a 5 × 5 × 5 simple cubic superlattice with positional noise (σ = 0.05 nm). (*b*) Comparison of four diffraction profiles for nanocube systems. From bottom to top: an isolated single particle, an ideal simple cubic assembly of 5 nm truncated nanocubes, a similar assembly with translational and rotational noise, which closely matches the single-particle profile, and an ideal superlattice assembly of ∼10 nm truncated nanocube building blocks.

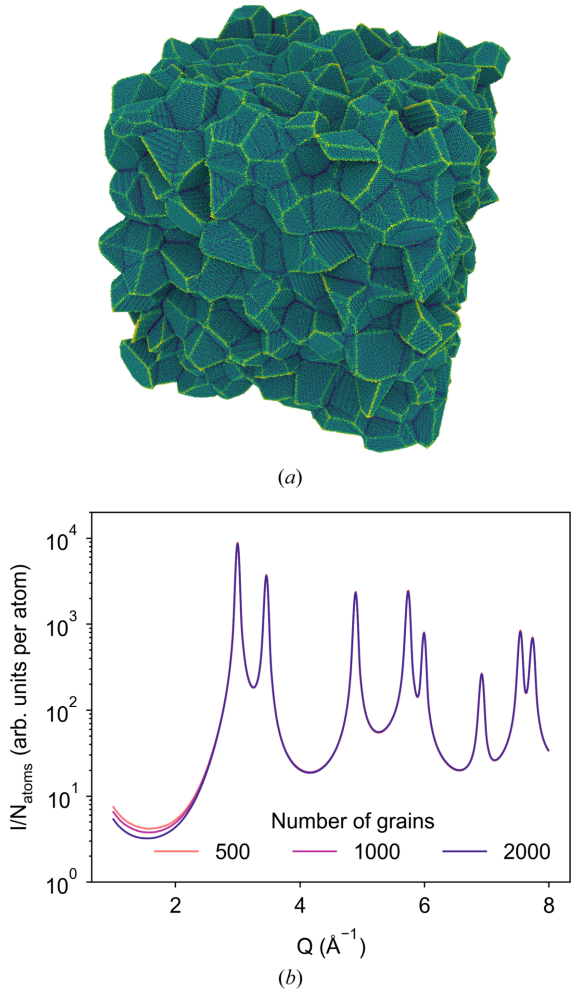


Figure 8
(*a*) Polycrystalline Cu (f.c.c.) sample with 1000 grains. Atoms are colored by coordination number. Atomic positions were averaged over multiple molecular dynamics frames. (*b*) Powder diffraction profiles for polycrystalline samples with different numbers of grains, normalized by atom count. The largest differences appear in the low-angle region and in the relative peak intensities.

Fig. 8(*b*) shows the resulting powder diffraction profiles for samples with different numbers of grains, normalized by atom count. Clear differences appear in the background and in relative peak intensities, indicating that grain statistics influence the averaged profile. A more systematic analysis of the number of grains required to reproduce a representative polycrystalline diffraction pattern is left for future work.

Table 3 details the corresponding runtimes. As expected, the cost follows approximately quadratic scaling, consistent with the $O(N^2)$ pair distance computation. The largest sample was completed in approximately 52 min, with a sustained performance of about 280 MPD s^{-1} per core across the benchmark set.

For comparison, we estimate that *Rose-X* would require roughly twenty times longer for the largest case, even under idealized linear scaling assumptions. In practice, such scaling is difficult to achieve at high core counts, so the advantage of the present implementation is expected to be larger.

5.3. Halloysite nanotubes

Halloysite is a naturally occurring aluminosilicate clay mineral with a tubular microstructure formed by rolling kaolinite layers. Halloysite nanotubes (HNTs) typically have lengths of 1–15 µm and diameters of 10–100 nm, making them attractive for applications in drug delivery, catalysis and nanotechnology (Rawtani & Agrawal, 2012; Fizir *et al.*, 2018; Yang *et al.*, 2023). Their hollow interiors enable encapsulation, while their crystalline framework remains accessible to X-ray diffraction analysis. Molecular dynamics simulations are also widely used to investigate their mechanical and thermal behavior (Heidari Pebdani, 2023; Prishchenko *et al.*, 2018; Gianni *et al.*, 2023).

The curved non-periodic geometry of HNTs is challenging for reciprocal-space methods, making the DSE a natural alternative. We therefore use *AES-Debye* to examine how structural parameters, particularly tube diameter, influence the powder diffraction profile.

Atomistic models were generated by rolling kaolinite sheets into tubes of varying diameters, as illustrated in Fig. 9(*a*). Larger tubes can contain hundreds of millions of atoms and therefore require an optimized implementation to remain computationally tractable.

Fig. 9(*b*) shows the resulting diffraction intensity as a function of Q and tube diameter. Clear shifts in peak positions and peak shapes demonstrate the sensitivity of the scattering pattern to tube geometry and illustrate how such variations can appear in experimental data.

This example also highlights support for multi-component systems. Each halloysite model contains three atomic species (O, Si and Al), giving rise to six distinct partial PDFs: O–O, O–Si, O–Al, Si–Si, Si–Al and Al–Al. These partial contributions are weighted by the corresponding atomic form factors and summed to obtain the final powder diffraction profile.

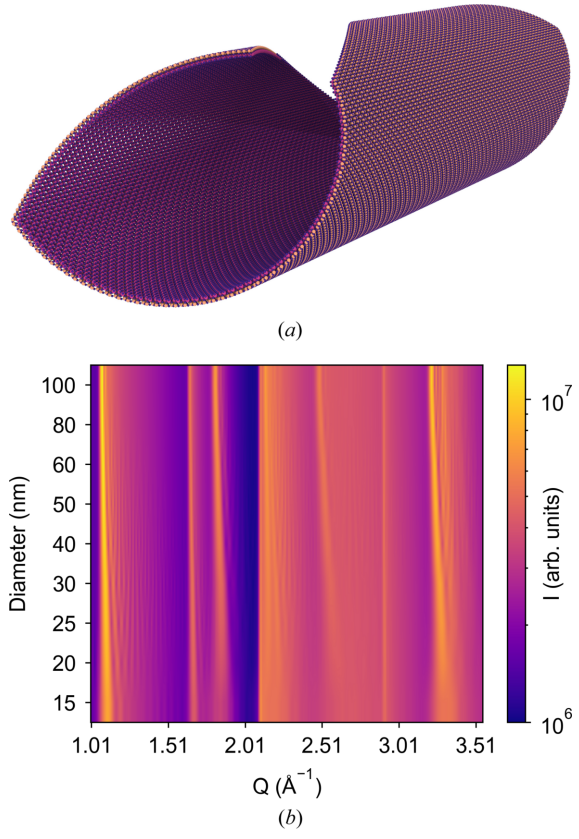


Figure 9
(*a*) Atomic model of a halloysite nanotube generated from a kaolinite sheet. (*b*) Two-dimensional visualization of diffraction intensity as a function of Q and tube diameter.

5.4. Colloidal nanoparticles

Since the Debye scattering equation is formulated strictly in terms of pairwise distances, it is fundamentally independent of the absolute length scale of the system. This allows the method to be readily adapted for studying self-assembly in colloidal crystals and simulating their corresponding small-angle scattering patterns. To demonstrate this capability, we modeled a macroscopic colloidal assembly comprising 200 000 nanoparticles. Rather than modeling atomistic details, each nanoparticle is represented as a uniform coarse-grained sphere with a diameter of 200 nm [Fig. 10(*a*)]. The ensemble was then compressed within a spherical confinement to induce crystallization, allowing us to track the structural evolution starting from an initial disordered state through subsequent highly ordered assembly stages.

We computed the scattering profiles for these large-scale configurations specifically in the small-angle region, where superlattice features typically emerge [Fig. 10(*b*)]. By multiplying the resulting interparticle structure factor by the single-particle form factor, we obtained the complete scattering profiles representing different stages of the crystallization process. As the system transitions into a crystalline assembly, distinct Bragg peaks begin to emerge and superimpose over the underlying small-angle form factor. These emerging peaks directly reflect the development of long-range periodic order during self-assembly. Ultimately, this application highlights

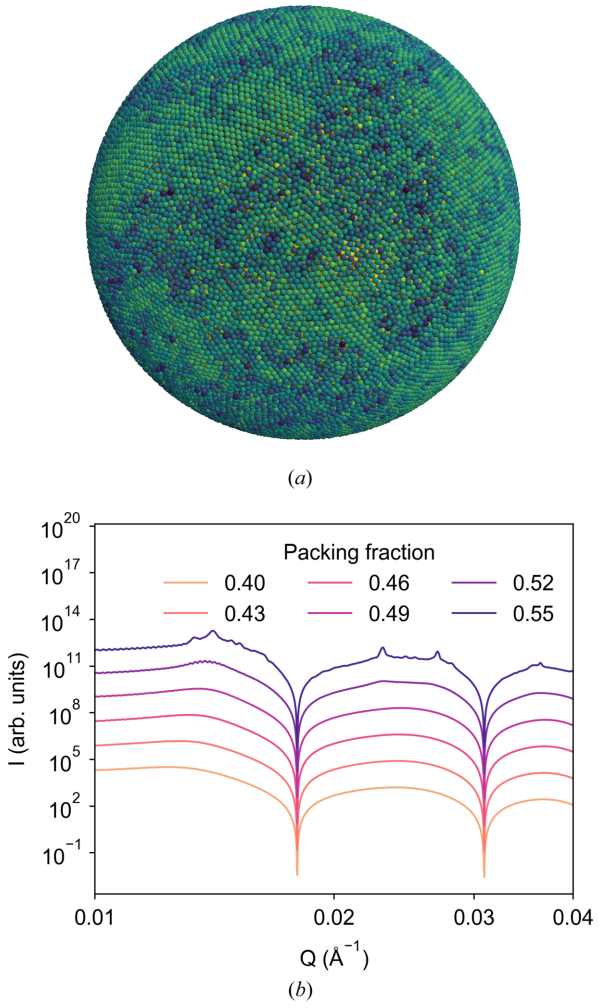


Figure 10
(*a*) Spherical nanoparticles (diameter 200 nm) crystallized in a spherical confinement at a packing faction of 0.55. The particles are colored using their coordination number to highlight the surface defects that form due to the spherical confinement. (*b*) Small-angle scattering profiles. As the packing fraction increases (bottom to top, shifted for clarity), we start to see the f.c.c. superlattice peaks on the sphere form factor in the background.

how *AES-Debye* can be seamlessly extended beyond the atomic scale, providing a powerful computational tool for investigating mesoscale and macroscopic soft-matter systems.

6. Conclusions and outlook

We have presented *AES-Debye*, a modern high-performance implementation for computing powder diffraction profiles from the Debye scattering equation. By combining a PDF-based formulation with bin center correction, precision-aware integer accumulation, cache-friendly data structures and a cell-list-based domain decomposition, the method improves memory locality and reduces latency compared with other implementations, particularly for large and disordered systems. A hybrid OpenMP/MPI/CUDA design enables efficient execution on CPUs and GPUs, while distributed memory parallelization sustains performance across nodes with only minimal communication overhead.

The method is numerically robust over a wide range of system sizes and disorder levels. In particular, bin center correction removes nonphysical negative intensities and improves the fidelity of the diffuse background in the high-Q region. Because PDFs are constructed internally, the same framework also provides high-resolution PDFs for downstream structural analysis, where fine binning and numerical accuracy are essential.

Applications to nanocube superlattices, polycrystalline copper, halloysite nanotubes and colloidal nanoparticles further demonstrate that *AES-Debye* can treat large compositionally complex systems with irregular geometries, enabling high-resolution total scattering calculations for realistic materials models. In the benchmarks presented here, *AES-Debye* achieves state-of-the-art throughput on current hardware and substantially outperforms existing tools, including *Rose-X* and *DebyeCalculator*, for comparable problem sizes while preserving numerical accuracy.

Despite these advances, evaluation of the DSE remains inherently $O(N^2)$, and extreme disorder can degrade memory locality. Several extensions remain of interest, including a HIP/ROCm backend for AMD GPUs, and adaptive or mixed-precision schemes with explicit error control. Optional instrument effects, such as resolution broadening, absorption, diffraction geometry or goniometer error, can help incorporate other experimental conditions into the profile. More broadly, refinement workflows such as PDF-based fitting can also exploit the high-resolution PDFs produced internally.

7. Code availability

The library can be found at https://gitlab.cs.fau.de/iq23adyz/debye.

The library is released as open source under the GNU General Public License (GPLv3) and provides the following:

(i) *Python interface.* A user-friendly API supports flexible data loading and seamless integration into existing workflows. Example scripts are included. The bindings are implemented with *pybind11* (Jakob *et al.*, 2017).

(ii) *CLI interface.* A command line tool enables rapid computations and accepts all structure formats supported by the *ASE* Python library (Larsen *et al.*, 2017).

(iii) *LAMMPS plugin.* A command style plugin for *LAMMPS* reads atomic positions during simulations and writes the computed profiles to user-specified locations.

Users may also link the C++ library directly; the C++ API closely mirrors the Python API (see the *LAMMPS* plugin documentation for details).

The library can be installed via the Python package manager *pip*. We employ *scikit-build-core* with *CMake* to automatically detect and enable MPI and GPU support when the requisite hardware and libraries are present. Additional component details and installation instructions are provided on the project's GitHub page.

APPENDIX *A*

Additional optimization strategies evaluated

We also assessed several alternative optimization strategies that were not adopted in the final implementation because they did not provide a favorable performance–accuracy trade-off.

(i) *SIMD vectorization.* We explored single-instruction multiple-data (SIMD) vectorization on CPUs to accelerate bin computations. In practice, the SIMD implementation performed worse than the scalar baseline, probably because the kernel is dominated by memory latency and because there is no native vectorized integer square root for `int64`; emulation or type conversion eliminated all potential gains.

(ii) *Reduced PDF resolution.* We tested whether decreasing the number of bins could improve performance. Although this produced modest speedups for crystalline systems, it introduced noticeable errors in the diffraction profiles and did not improve runtimes for disordered systems. Because the accuracy loss outweighed the limited benefits, this strategy was not pursued further.

(iii) *Cell lists on the GPU.* We also evalulated cell-list-based decomposition on GPUs. This did not improve performance: in the absence of an L3 cache and given the already high memory bandwidth of the device, the added indirection and bookkeeping outweighed any gain from improved locality. On smaller GPUs, lightweight sorting of atoms by cell index produced only minor benefits and may be useful only in more constrained settings.

These results further motivated the emphasis on domain decomposition, cache locality optimizations and architecture-specific parallelization in the final *AES-Debye* implementation.

Acknowledgements

Author contributions: implementation of the code, development of optimization strategies, performance of testing, curation of data and drafting of the manuscript, NP; support of the development of the CPU-based high-performance computing implementation, MW; support of the GPU-based implementation, SK; conception of the method and contribution of the earlier *Rose-X* software, AL; supervision of the project, securing of funding and revision of the manuscript, ME and AL; editing and approval of the final version, all authors. Open access funding enabled and organized by Projekt DEAL.

Conflict of interest

The authors declare no conflict of interest.

Funding information

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, grant No. LE4543/2-1) and by the Competence Network for Scientific High Performance Computing in Bavaria (KONWIHR, grant No. Z.4-M7635.1/23/15). We gratefully acknowledge computational resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) under NHR project No. CRC1411D04.

References

Anderson, J. A., Glaser, J. & Glotzer, S. C. (2020). *Comput. Mater. Sci.* **173**, 109363.
Cernuto, G., Masciocchi, N., Cervellino, A., Colonna, G. M. & Guagliardi, A. (2011). *J. Am. Chem. Soc.* **133**, 3114–3119.
Cervellino, A. & Frison, R. (2020). *Acta Cryst.* A**76**, 302–317.
Debye, P. (1915). *Nachricht. König. Ges. Wissen. Göttingen Math.-Phys. Klass.* pp. 70–60.
Fizir, M., Dramou, P., Dahiru, N. S., Ruya, W., Huang, T. & He, H. (2018). *Microchim. Acta* **185**, 389.
Gelisio, L., Azanza Ricardo, C. L., Leoni, M. & Scardi, P. (2010). *J. Appl. Cryst.* **43**, 647–653.
Gelisio, L. & Scardi, P. (2016). *Acta Cryst.* A**72**, 608–620.
Gianni, E., Pšenička, M., Macková, K., Scholtzová, E., Jankovič, L., Mareš, M., Papoulis, D. & Pospíšil, M. (2023). *J. Mol. Struct.* **1287**, 135639.
Grover, R. F. & McKenzie, D. R. (2001). *Acta Cryst.* A**57**, 739–740.
Hall, B. D. & Monot, R. (1991). *Comput. Phys.* **5**, 414–417.
Heidari Pebdani, M. (2023). *Comput. Mater. Sci.* **218**, 111948.
Jakob, W., Rhinelander, J. & Moldovan, D. (2017). *pybind11*, https://github.com/pybind/pybind11.
Johansen, F. L., Anker, A. S., Friis-Jensen, U., Dam, E. B., Jensen, K. M. Ø. & Selvan, R. (2024). *J. Open Source Software* **9**, 6024.
Kaduk, J. A., Billinge, S. J. L., Dinnebier, R. E., Henderson, N., Madsen, I., Černý, R., Leoni, M., Lutterotti, L., Thakral, S. & Chateigner, D. (2021). *Nat. Rev. Methods Primers* **1**, 77.
Kieffer, J. & Wright, J. P. (2013). *Powder Diffr.* **28**, S339–S350.
Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jennings, P. C., Bjerre Jensen, P., Kermode, J., Kitchin, J. R., Kolsbjerg, E. L., Kubal, J., Kaasbjerg, K., Lysgaard, S., Bergmann Maronsson, J., Maxson, T., Olsen, T., Pastewka, L., Peterson, A., Rostgaard, C., Schiøtz, J., Schütt, O., Strange, M., Thygesen, K. S., Vegge, T., Vilhelmsen, L., Walter, M., Zeng, Z. & Jacobsen, K. W. (2017). *J. Phys. Condens. Matter* **29**, 273002.
Leonardi, A. (2021). *IUCrJ* **8**, 257–269.
Leonardi, A. & Bish, D. L. (2016). *J. Appl. Cryst.* **49**, 1593–1608.
Leonardi, A. & Bish, D. L. (2017). *Acta Mater.* **133**, 380–392.
Leonardi, A., Leoni, M. & Scardi, P. (2013). *Comput. Mater. Sci.* **67**, 238–242.
Neverov, V. S. (2017). *SoftwareX* **6**, 63–68.
Prishchenko, D. A., Zenkov, E. V., Mazurenko, V. V., Fakhrullin, R. F., Lvov, Y. M. & Mazurenko, V. G. (2018). *Phys. Chem. Chem. Phys.* **20**, 5841–5849.
Rawtani, D. & Agrawal, Y. K. (2012). *Rev. Adv. Mater. Sci.* **30**, 282–295.
Reuter, K. & Köfinger, J. (2019). *Comput. Phys. Commun.* **236**, 274–284.
Rietveld, H. M. (1967). *Acta Cryst.* **22**, 151–152.
Ross, K. C., Petrus, J. A. & McDonald, A. M. (2014). *Powder Diffr.* **29**, 337–345.
Rudolph, M., Motylenko, M. & Rafaja, D. (2019). *IUCrJ* **6**, 116–127.
Scardi, P. & Gelisio, L. (2016). *Sci. Rep.* **6**, 22221.
Scardi, P. & Leoni, M. (2004). *Whole Powder Pattern Modelling: Theory and Applications*, pp. 51–91. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-06723-9_3.
Thomas, N. W. (2010). *Acta Cryst.* A**66**, 64–77.
Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., in 't Veld, P. J., Kohlmeyer, A., Moore,

S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C. & Plimpton, S. J. (2022). *Comput. Phys. Commun.* **271**, 108171.
Toso, S., Baranov, D., Giannini, C., Marras, S. & Manna, L. (2019). *ACS Mater. Lett.* **1**, 272–276.
Vaccari, M. & Fornasini, P. (2006). *J. Synchrotron Rad.* **13**, 321–325.
Warren, B. E. (1990). *X-ray Diffraction.* Dover Publications.
Watson, M. C. & Curtis, J. E. (2013). *J. Appl. Cryst.* **46**, 1171–1177.
Yang, L., Yang, X., Xia, F., Gong, Y., Li, F., Yu, J., Gao, T. & Li, Y. (2023). *Chem. Asia. J.* **18**, e202300473.
Zhang, S., Huang, Z., Wen, Z., Zhang, L., Jin, J., Shahbazian-Yassar, R. & Yang, J. (2017). *Nano Lett.* **17**, 3518–3526.
Zhang, Y., Wu, L., Ji, M., Wang, B., Kong, Y. & Xu, J. (2011). *Opt. Mater. Expr.* **2**, 92.