CHAPTER 7

freud: A Software Suite for High Throughput Analysis of Particle Simulation Data

This chapter is reproduced from Ramasubramani, V., Dice, B. D., Harper, E. S., Spellings, M. P., Anderson, J. A. & Glotzer, S. C. freud: A software suite for high throughput analysis of particle simulation data. *Computer Physics Communications* **254,** 107275. ISSN: 00104655 (Sept. 2020).

7.1 Introduction

Molecular simulation is a crucial pillar in the investigation of scientific phenomena. Increased computational resources, better algorithms, and new hardware architectures have made it possible to simulate complex systems over longer timescales than ever before [6, 27, 115, 116, 167]. The sheer volume of data necessitates computationally efficient analysis tools, while the diversity of data requires flexible tools that can be adapted for specific systems. Additionally, to support scientists with limited prior computing experience, tools must be usable without extensive knowledge of the underlying code.

Numerous software packages that satisfy these requirements have been developed in recent years. Tools such as MDTraj [179], MDAnalysis [180], LOOS [181], MMTK [182], and VMD [177] provide efficient implementations of various standard analysis methods. Although powerful, such tools are generally limited in scope to all-atom simulations, particularly biomolecular simulations. This focus is manifested not only through the features these tools provide, but also in their general design philosophies.

Perhaps the most pronounced characteristic of such tools is a strong emphasis on trajectory management, which includes parsing trajectory files and supporting extensive topology selection features to enable, for instance, selecting all residues or atoms in a protein backbone. Although such tools are crucial for working with topologies in atomistic

simulations, they are frequently cumbersome for working with coarse-grained simulation data where the trivial selection (all particles in the system) is the most common selection for various analyses. Moreover, such topology selection tools make assumptions that are inappropriate for non-atomistic systems: "bonding" in colloidal systems, for instance, is typically based on whether two particles are found to be in the same neighborhood by some distance-based metric, not by the presence of a true chemical bond. Since such determination of nearest neighbors is highly dynamic and parameter-dependent, it must be calculated on-the-fly and cannot be stored in a trajectory.

Another inconvenient but almost universal implementation choice is to directly tie analysis methods to trajectories by writing code that acts directly on some in-memory representation of a trajectory. This direct linkage is generally inflexible because it inhibits pre-processing of the data before running the analysis, which is often crucial to analyzing more specialized systems. More importantly, existing tools emphasize implementations of highly specific analyses involving, for instance, hydrogen bonding and protein secondary structure (using, e.g., DSSP [183]), which are far less useful for analyzing non-biomolecular systems. The predominant analyses of coarse-grained, colloidal-scale, or nanoparticle simulations usually involve measurements like numbers of nearest neighbors, diffraction patterns, or bond-orientational order parameters. These analyses bear little relation to the analyses performed for atomistic systems. These considerations suggest a need for a different type of analysis package that offers different methods than most existing tools.

In this paper we introduce `freud`, an open-source simulation analysis toolkit that addresses these needs. All inputs to and outputs from `freud` are numerical arrays of data, and the package makes no reference to predefined notions of atoms or molecules. As a result, `freud` can analyze particle-based data from both experiments and simulations regardless of the specific tools, methods, or software that were used to generate it. The package provides a Python API for accessing fast methods implemented in C++, and it implements numerous specific methods such as radial distribution functions and correlation functions that are common in the field of soft-matter physics (see fig. 7.1). Prior works have used `freud` for: determining spatial correlation functions and PMFTs in two dimensions [167]; calculating Steinhardt order parameters for identifying solid-like particles [184, 185]; computing spherical harmonics for machine learning on crystal structures [186]; optimizing pair potentials for designing complex crystals [187]; calculating strain fields by finding neighbors of particles against a uniform grid [188]; finding PMFTs in depletion-mediated self-assembly of hard cuboctahedra [54]; measuring rotational degrees of freedom in entropically ordered systems [189]; umbrella sampling of solid-solid phase transitions using Steinhardt order parameters [190]; evaluating PMFTs in analysis of two-dimensional

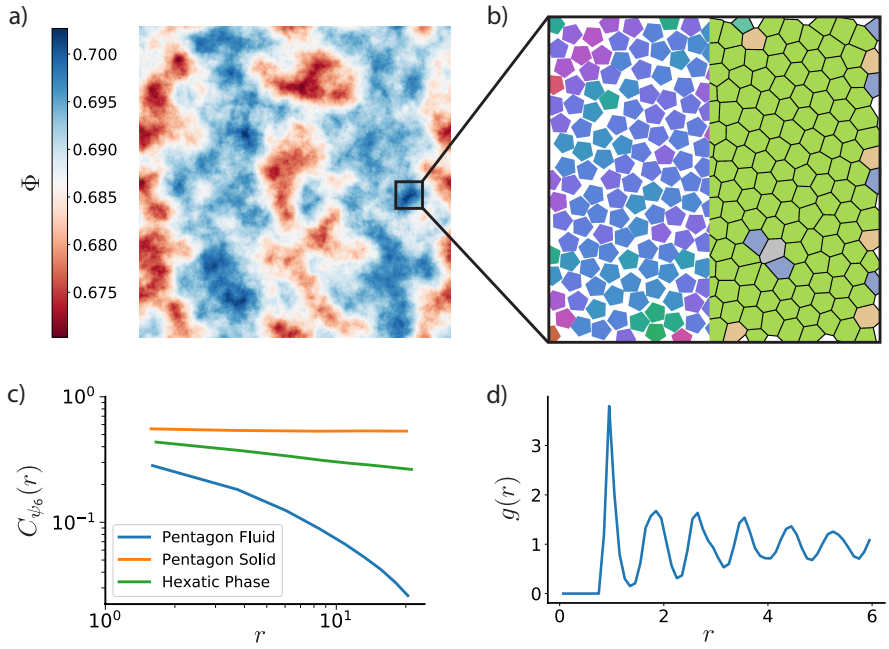


Figure 7.1 | The `freud` library is capable of computing a number of characteristics of a system of particles. Here, we demonstrate some of those features on a 2D Monte Carlo simulation of polygons that exhibits hexatic ordering [167]. a) Phase separation is clearly evident in this system of 512^2 pentagons colored by local density Φ; the system is divided into denser (blue) and less dense (red) regions. b) Zooming into a particularly dense region shows that the hexatic ordering (left half) is generally uniform across the region. The Voronoi diagram of the system (right half) is also largely defect-free, with just a few pentagons having more or fewer than 6 nearest neighbors. c) The spatial correlation of the hexatic order parameter $C_{\psi_6}(r)$ is nearly constant for a nearly perfect crystal of pentagons (orange), whereas it decays very quickly in a fluid (blue). For a comparable system of hexagons, however, we see a power-law decay (green) in the hexatic order parameter due to the presence of a hexatic phase between the solid and fluid phases. d) The radial distribution function $g(r)$ for the system of pentagons shows the expected sequence of neighbor shells as a function of distance.

shape allophiles [191]; and more. The `freud` library is designed to work well with coarse-grained particle models, such as those used in simulations of anisotropic nanoparticles, colloidal crystals, and polymers, and its methods are particularly useful for studies of phase transitions and critical phenomena in such systems. The package is likely to be of greatest interest to scientific communities in materials science, chemical engineering, and physics, though many of its analysis methods would be useful in generic particle systems. The `freud` library also integrates well into the scientific Python ecosystem, especially in data pipelines for machine learning and visualization [192].

The paper is organized as follows. We first address the core design principles that went into building `freud` in section 7.2. Section 7.3 focuses more specifically on the details of the code, including information on class structures. Section 7.4 describes the various analysis methods in `freud` and details their uses. Finally, in section 7.5 we provide some

example code demonstrating the usage of `freud`[1]. The figures in this paper are rendered using Matplotlib [85] unless otherwise noted.

7.2 Design

Many of the best known tools for analyzing molecular simulations are built into either simulation toolkits (such as LAMMPS [175], GROMACS [176], or the cpptraj [193] plugin to Amber [194]) or visualization toolkits (such as VMD [177], PyMOL [195], or OVITO [63]). Although most of these have introduced varying degrees of scripting support over the years, the analyses built into simulation toolkits are primarily focused on performing one-shot analyses on trajectory files directly from the command line. The visualization toolkits tend to have more full-featured scripting interfaces, but they are frequently difficult (if not impossible) to use outside their own sandboxed environments, complicating or even prohibiting integration with other tools. More recently, many newer tools such as MDTraj [179], MDAnalysis [180], LOOS [181], and Pteros [196] have aimed to decouple analysis from simulation and visualization, making scriptability a primary focus to increase flexibility. Among such tools, Python is the most common language of choice due to its ease of use and the fact that it can be naturally extended with high performance languages like C, C++, and FORTRAN.

`freud` follows in the footsteps of these tools, providing a full-featured Python API to access all of its routines. However, while most other tools focus on calculating properties involving molecular topologies, `freud` is fundamentally designed for analyzing the local neighborhoods of particles, particularly where such local analyses provide global insight about the system. Such analyses are typically far more varied and system-dependent than the standard analyses of molecular topologies and therefore require more flexible tools. To meet this need, `freud` eschews any form of trajectory object encoding system topology and is instead designed such that each analysis method is an independent Python class that performs computations directly on NumPy arrays [137] of data.

This design makes it possible to use a far wider range of data with `freud` than is possible with tools that are tied to simulation trajectories. For instance, calculating Voronoi diagrams and computing spatial correlation functions with `freud` is possible for essentially arbitrary spatial data, not just the result of a molecular simulation. Another major benefit is that NumPy arrays (the de facto standard for numerical data in Python) can be easily passed between multiple tools, making `freud` equally easy to use for one-off analyses or as part

[1]The code for these examples and many others is available at https://github.com/glotzerlab/freud-examples and in our online documentation.

of a larger analysis pipeline involving many steps and using various software packages. As a result, `freud` is a much more flexible choice both for analyzing disparate sources of data and for incorporating into Python workflows. For example, most of `freud`'s analyses can be used within the OVITO visualization environment for real-time visualization with almost no noticeable performance cost.

Producing such array data from simulation trajectories for input to `freud` is straightforward because high quality file parsers with Python APIs already exist for all common trajectory file formats. Through integration with tools like MDAnalysis, GSD [197], and garnett [198], `freud` can be used with data from over 25 different file formats, including common formats like DCD, XTC, and TRJ files. `freud` integrates with the trajectory objects produced by many of these tools, but if necessary, users can read trajectories into arrays and modify them before passing the data to `freud` for analysis. By using data read by other tools, `freud`'s analyses can be made aware of molecular topology if needed, but only when the analysis method requires such information. Similarly, since the outputs of `freud`'s analyses are also NumPy arrays, they can be passed to almost any tool in the scientific Python software stack. For example, constructing a Pandas [83] `DataFrame` from the outputs of any `freud` analysis requires just one line of code, and it immediately enables writing the output to text, CSV, or HDF5, or saving into an SQL database.

Beyond differences in trajectory and data handling, the most significant design choice in `freud` stems from the most common pattern followed by its many analysis methods. Since the first task in characterizing local neighborhoods is often the identification of neighboring particles, `freud` provides efficient methods for finding neighbors in arbitrary system geometries. The nearest-neighbor finding routines are designed to be as fast and flexible as possible, supporting various algorithms optimized for different system configurations and offering different criteria for neighbor selection. In general, queries can be based on either a cutoff distance or a desired number of nearest neighbors. These tools are optimized to provide cheap access to neighbors even in highly performance-critical loops in C++ analysis routines, but the package's system box representation and neighbor finding tools also have Python APIs, so users can implement custom analyses directly in Python (for an example, see section 7.5.4).

The analysis methods in `freud` are essentially independent tools that make use of these objects to efficiently perform various calculations. These features are all presented with a common API, easing the transition between the different types of analyses needed for different simulations. All methods in `freud` are accelerated through extensive parallelization.

7.3 Implementation

The `freud` package is entirely object-oriented, with two core C++ classes: the `Box` class, which encapsulates all logic associated with periodicity in arbitrary triclinic boxes (boxes with 3 linearly independent basis vectors); and the `NeighborQuery` class, which facilitates efficiently finding, storing, and iterating over nearest neighbors. In keeping with the Python ethos, box objects in `freud` may be constructed from a variety of inputs. Any method in `freud` that accepts a box object also accepts a number of objects that can be interpreted as a box, such as a 3×3 NumPy array of box vectors or a list of three numbers representing the edge lengths of an orthorhombic box. There are two subclasses of `NeighborQuery` in `freud` that each implement different neighbor search algorithms: one implements a bounding volume hierarchy (BVH) [79], while the other implements a cell list [114]. The `NeighborList` class is a lightweight storage mechanism for `NeighborQuery` results that accelerates performing multiple analyses on the same set of neighbor pairs.

The analysis methods in `freud` are encapsulated by *Compute classes*, which are loosely defined as classes providing a `compute` method that populates class attributes after performing some computation. Compute classes, such as the `density` module's `RDF` class, are usually configured with constructor arguments, after which they can be used multiple times to perform distinct calculations. Some classes in `freud` (e.g. the RDF, PMFT, or bond-orientational order diagram) represent histogram-like quantities, and therefore allow the user to specify `reset=False` as an argument to `compute` in order to accumulate and average data over many calls.

Compute classes can be divided into two groups, those that depend on finding neighbors and those that do not. A majority of calculations in `freud` require neighbors, and the `compute` methods of such classes all share two arguments, `system` and `neighbors` (in addition to analysis-specific arguments like particle orientations for PMFTs; such arguments are also typically NumPy arrays). The `system` parameter accepts a `NeighborQuery` or any object that can be interpreted as a tuple `(box, points)`, where the `box` is any valid box-like object (as described above) and the `points` argument is anything that can be interpreted as an $N \times 3$ NumPy array of positions. Valid systems include simulation frame objects from tools such as MDAnalysis, GSD, garnett, OVITO, or the particle simulation engine HOOMD-blue [77–79].

When performance is critical, providing a `NeighborQuery` object is advantageous because many `compute` methods can reuse these neighbor search data structures. For all other `system` inputs, `freud` internally constructs a `NeighborQuery` if the `compute`

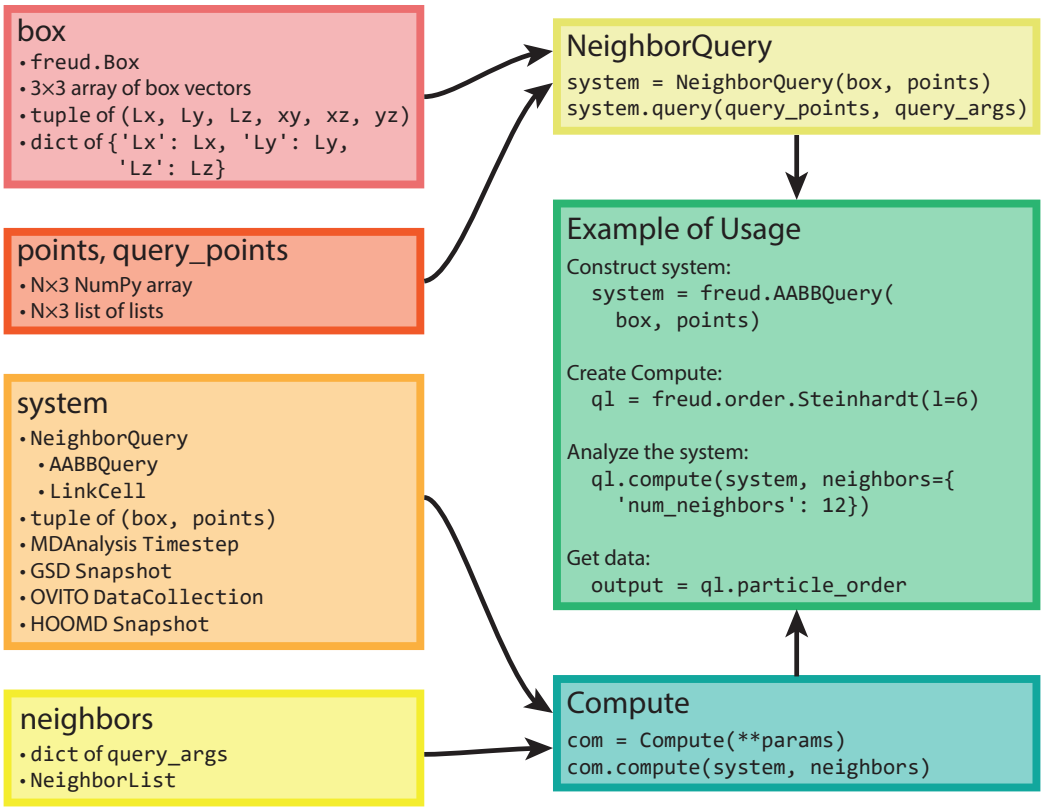


Figure 7.2 | Here we show the flow of various types of inputs into `freud`. Boxes can be constructed based on a variety of inputs, all of which can also directly be provided anywhere a `box` object is required. Similarly, any object that can be interpreted as an $N \times 3$ array can be provided where particle positions are required. Any valid pair of box and points can be used to construct a `NeighborQuery` object, which is one of the types of `systems` that `freud` accepts. In addition to a `NeighborQuery`, `freud` can also interpret raw tuples of boxes and points as `system` objects, or use simulation frames from numerous external tools (a subset are shown in the figure). Any computation that involves finding nearest neighbors also requires a specification of `neighbors`, which can be a `NeighborList` or a dictionary of query arguments. The Example of Usage on the right shows a typical use case of `freud` that combines these concepts.

method requires neighbor pairs. The `neighbors` argument is a dictionary of query arguments, such as `dict(num_neighbors=12)` or `dict(r_max=3.0)` (the complete specification for `freud`'s Query API is provided in the documentation). Alternatively, users may precompute a `NeighborList` and provide it as the `neighbors`. In this case, whether `system` is a `NeighborQuery` or not has no impact on performance because the calculation will be carried out directly on the provided set of neighbor pairs and no additional spatial searches are required. Figure 7.2 shows a flowchart demonstrating how these classes and data structures are used.

Some methods in `freud` do not operate on neighboring pairs of particles. For those that still depend on particle positions (such as `GaussianDensity`), the first argument is still any valid `system`, but no `neighbors` are provided. Some methods do not depend

on positions at all; for instance, the `Nematic` order parameter only requires particle orientations. In such cases, the user can simply pass that quantity alone to the calculation. This mode of operation is particularly useful when performing high-throughput analysis of large files; using file formats like GSD that permit reading only certain properties of the trajectory, users can minimize I/O operations by only reading the required arrays from memory.

All Compute classes use efficient, thread-parallel C++ implementations for performance-critical components. The Python bindings for these C++ classes are generated using Cython [199], and the C++ methods are mirrored in Python using thin Cython classes that dispatch calls to the underlying C++ class instances. The Cython classes have limited responsibilities: managing the memory of the underlying C++ instances, sanitizing inputs when necessary, and providing transparent access via memory views on C++ arrays.

The main exception to this design is the `msd` module, which is implemented in pure Python in `freud`. The MSD is a measure of, on average, how far particles move in a given window of time. In a simulation trajectory of N_f frames, the MSD of particle i over a window of length m frames is given by:

$$MSD(i,m) = \frac{1}{N_f - m} \sum_{k=0}^{N_f-m-1} \|(\vec{r}_i(k+m) - \vec{r}_i(k))\|^2 \tag{7.1}$$

Therefore, the total MSD is given by:

$$MSD(m) = \frac{1}{N_p} \sum_{i=1}^{N_p} MSD(i,m) \tag{7.2}$$

Direct computation of the MSD is an $\mathcal{O}(N_p N_f^2)$ operation, but by using a FFT this cost can be reduced to $\mathcal{O}(N_p N_f \log(N_f))$ [200]. When using this approach, the FFT is responsible for most of the computation time, and since packages like NumPy [137] and SciPy [201] already expose fast C and FORTRAN FFT routines to Python, `freud` simply leverages them directly and implements the rest of the MSD in pure Python.

Calculations in `freud` are generally parallelized over *particles* (e.g. the `Nematic` order parameter class) or over *pairs of particles* (e.g. computing inter-particle distances for an RDF with the `RDF` class). Both the number of particles and the number of particle-particle pairs increase with system size, ensuring that the work is load-balanced well among threads because the number of threads is much less than the number of particles or pairs. Parallelism in `freud` is accomplished using TBB [202]. Analysis routines are written as lambda functions operating on a particle or a pair of particles; `freud` provides wrappers that

then automatically parallelize these functions appropriately using TBB. Modern compilers aggressively inline such lambda functions, thereby optimizing away any additional cost that could arise from the extra function calls. `freud` uses thread-local storage extensively to avoid any parallel writes to data containers. For histograms that accumulate over many frames of simulation data, `freud` performs reduction over thread-local containers lazily.

Currently, `freud` is at version 2.1.0 and supports Python versions 3.5.0 or greater. The package is distributed through PyPI and the `conda-forge` channel of the Anaconda package manager [203], making it easy to install on any Unix-based operating system (e.g. Linux or macOS). Builds for the Windows operating system are also available on `conda-forge`. `freud` depends on NumPy and TBB libraries, which are automatically installed with `freud`. The library can also be compiled from source using a C++11 compliant compiler. Compilation requires NumPy and TBB headers as well as a Cython installation. Code documentation is written using Google-style docstrings rendered using Sphinx and hosted on ReadTheDocs. The `freud` library is released open source under the BSD 3-Clause License, and the source code is available in a GitHub repository [204]. Continuous integration testing is performed using CircleCI.

7.4 Features

7.4.1 General Utilities

The general utilities in `freud` are contained in two modules: `box` and `locality`. The `box` module contains the core `Box` class. The `locality` module contains the `NeighborQuery` abstract class, which defines the standardized query API. `NeighborQuery` results (neighboring particle pairs) can be obtained dynamically or stored in the `NeighborList` class provided by the `locality` module.

Box periodicity is built in at the lowest level of the `NeighborQuery` subclasses, which are highly optimized for this use case. The `AABBQuery` subclass implements a tree data structure of Axis-Aligned Bounding Boxes (AABBs), a type of BVH which greatly accelerates the process of finding particles' neighbors [79, 185]. A second approach is implemented in the `LinkCell` subclass [114], which uses linked cell lists to find particle neighbors. Both of these classes can find neighboring particle pairs based on a distance cutoff or a desired number of neighbors, and both were adapted from HOOMD-blue.

As a performance benchmark, we compare `freud`'s `AABBQuery` class with the `cKDTree` implementation in SciPy [201]. As part of SciPy, this implementation is the most readily available alternative to `AABBQuery`. Figure 7.3 shows that `freud`'s `AABBQuery`

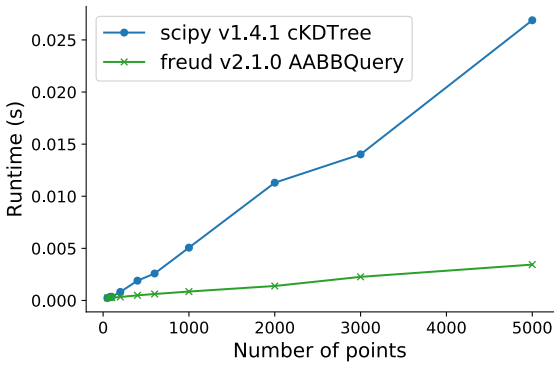


Figure 7.3 | Here we benchmark the `AABBQuery` implementation in `freud` against the `cKDTree` implementation in SciPy. We construct randomly generated sets of points such that each particle would have, on average, 12 neighbors within a distance of 1. We then measure the performance of finding all neighbors within this distance using both SciPy's `cKDTree` and `freud`'s `AABBQuery`. The benchmarks were performed on a system with an Intel® Xeon® CPU E5-2680 v2 @ 2.80GHz. The `AABBQuery` implementation in `freud` scales much better than SciPy's `cKDTree` for larger system sizes. We do not report error bars due to the extremely low variance in the data. The exact details are available at https://github.com/glotzerlab/freud-examples.

routines clearly outperform the `cKDTree` as system sizes increase to thousands of points. Moreover, we note that while `freud` supports general triclinic boxes, SciPy's `cKDTree` only supports periodic orthorhombic boxes (i.e. a cuboid, a rectangular prism where all angles are right angles).

In addition to these performance gains, the `NeighborQuery` objects in `freud` are designed to interoperate seamlessly with analysis routines. Since analyses in `freud` are written in C++, using a Python API to find neighbors and then pass them into other C++ routines would waste time in unnecessary memory transfers. Furthermore, while a Python API should make certain promises, such as sorting the order of the resulting neighbors, the analyses using neighbors simply loop over all pairs and therefore do not require any such extra work. To avoid this cost, the features of the `NeighborQuery` classes are directly accessible in C++ in the form of iterators that lazily produce neighbors. In practice, using the `NeighborQuery` classes in this manner speeds up computations by a factor of two or more depending on the system size. To make use of these iterators, developers implement analysis methods as lambda functions that are passed as arguments to `freud`'s internal TBB wrappers that apply these functions to neighbor pairs in parallel.

The final feature of the `locality` module is the `Voronoi` class, which uses the voro++ library [205] to generate Voronoi diagrams for systems of particles. Voronoi diagrams are a standard method for characterizing the local geometric arrangements in the system, and they also provide a parameter-free method for defining nearest-neighbor relationships [206]. The `Voronoi` class produces a `NeighborList` object that can then

be used as the `neighbors` argument for other compute classes.

7.4.2 Analysis Modules

The remaining modules in `freud` are independent of one another and contain groups of classes that implement related features. While some of `freud`'s features are unique, many others are standard techniques. However, implementations of these methods commonly lack support for periodicity. For example, the SciPy library [201] has functions for computing Voronoi diagrams and correlation functions, but these are restricted to aperiodic systems.

The `cluster` module of `freud` can be used to find clusters of particles—where cluster membership is defined by neighbor bonds—and then compute properties of these clusters such as gyration tensors. The `density` module contains features for calculating radial distribution functions as well as spatial correlation functions of arbitrary quantities. Additionally, the `density` module can estimate local particle density and interpolate particle density onto a regular grid suitable for, e.g., computing discrete Fourier transforms. The `interface` module provides a quick tool for identifying interfaces between two mutually exclusive sets of points (e.g. a solid and a liquid phase). The `msd` module enables the calculation of mean squared displacements of particles over the course of a trajectory.

The `order` module is the most extensive one in `freud`, containing a large number of different order parameters commonly used to measure ordering and identify phase transitions in crystalline systems. Of particular note are the bond-orientational order parameters Q_l and W_l [208] and the cubatic order parameter [209] (see fig. 7.4). The module also contains the nematic order parameter for identifying orientationally ordered, translationally disordered phases, as well as a solid-liquid order parameter for identifying generic ordered phases [210].

The other features of `freud` are analysis methods developed by researchers in our group and not yet implemented anywhere else. In particular, the `pmft` and `environment` modules implement features unique to `freud` that we now discuss in greater detail.

7.4.3 Potentials of Mean Force and Torque

The potential of mean force and torque (PMFT) is a generalization of the classical PMF that was recently developed to quantify directional entropic forces that emerge in crowded systems [56, 80]. Given the canonical partition function as a function of particle positions $\{q\}$ and particle orientations $\{Q\}$, ref. [56] derives the PMFT by separating out a component

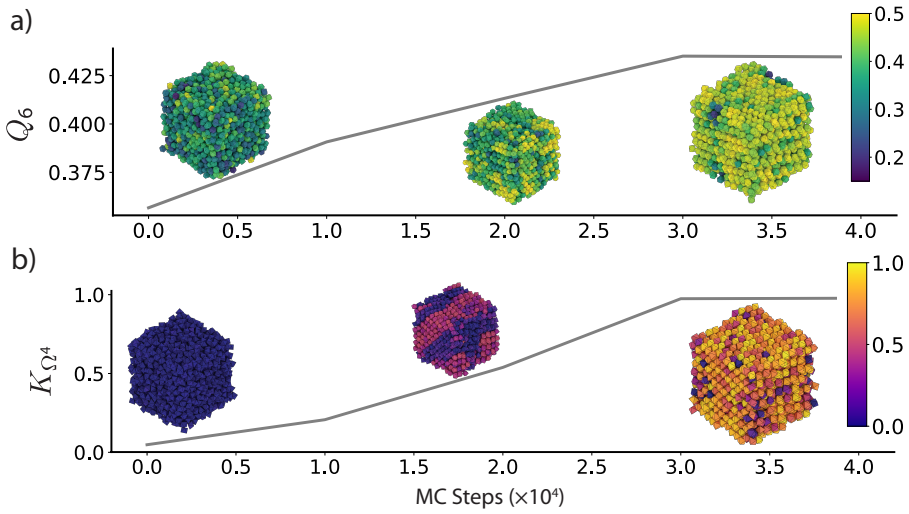


Figure 7.4 | Various order parameters can be used to characterize the degree of ordering in a system. The per-particle order parameter values eventually converge to a uniform global value as the system becomes globally well-ordered. These plots show the evolution of two order parameters over the course of a Monte Carlo simulation of hard particles, which over time rearrange into an ordered phase under compression. Simulation snapshots are colored by the per-particle order parameter and rendered with `fresnel` [207]. a) The Steinhardt Q_6 order parameter is an appropriate scalar descriptor for systems forming a BCC ($cI2$-W) structure. Systems of cuboctahedra in the fluid phase show a distinctly different characteristic value of the order parameter than in the solid phase. b) The cubatic order parameter K_{Ω^4} is useful for characterizing ordering in these systems of octahedra.

corresponding to the relative coordinates of a pair of particles $\Delta\xi_{12}$:

$$Z = \int d\Delta\xi_{12} J(\Delta\xi_{12}) e^{-\beta U(\Delta\xi_{12})} \int [d\tilde{q}][d\tilde{Q}] e^{-\beta U(\{\tilde{q}\},\{\tilde{Q}\},\Delta\xi_{12})} \tag{7.3}$$

$$= \int d\Delta\xi_{12} J(\Delta\xi_{12}) e^{-\beta U(\Delta\xi_{12})} \; e^{-\beta \tilde{F}_{12}(\Delta\xi_{12})} \tag{7.4}$$

where J is the Jacobian transforming to the local coordinate system and $\tilde{F}_{12}$ is the free energy of the other particles, which have been integrated over in eq. (7.4). The PMFT F_{12} is defined by the relation

$$Z \equiv \int d\Delta\xi_{12} e^{-\beta F_{12}(\Delta\xi_{12})} \tag{7.5}$$

Combining eqs. (7.4) and (7.5) gives an expression for the PMFT

$$\beta F_{12}(\Delta\xi_{12}) = \beta U(\Delta\xi_{12}) - \log J(\Delta\xi_{12}) + \beta \tilde{F}_{12}(\Delta\xi_{12}) \tag{7.6}$$

In hard particle systems governed exclusively by excluded volume interactions, the potential energy term becomes an infinite Heaviside function H and the PMFT can be simplified to

$$F_{12}(\Delta\xi_{12}) = -k_B T \log(H(d(\Delta\xi_{12}))J(\Delta\xi_{12})) + \tilde{F}_{12}(\Delta\xi_{12}) \tag{7.7}$$

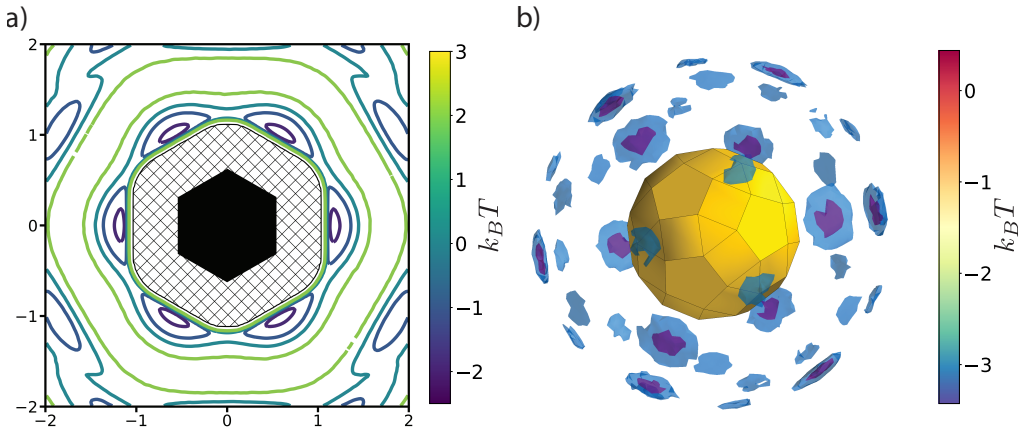


Figure 7.5 | The PMFT is related to the probability of finding particles at a given position and orientation relative to one another. a) The PMFT of an ordered system of hexagons [191], where the locations of the wells indicate that particles are much more likely to sit next to the edges of their neighbors than the corners. In two-dimensional systems, the full PMFT is 3-dimensional, since it also must account for the orientation of the second particle relative to the first; for clarity, in this figure we have integrated out that degree of freedom. b) A PMFT computed from a system of rhombicosidodecahedra shows distributions of neighboring particles in three dimensions (figure rendered using Mayavi [211]). There are six degrees of freedom in 3D systems, three translational and three rotational. This PMFT only shows the three translational degrees of freedom. The wells representing the deepest energy isosurfaces of the PMFT align with the largest (pentagonal) facets of the polyhedron.

To contextualize the PMFT, we note that if in eq. (7.3) we redefine $\Delta\xi_{12}$ to only include the center-to-center distance of the pair of particles and otherwise follow the same steps, the resulting potential F_{12} reduces to the classical PMF with the usual RDF relation $g(r) = e^{-\beta F_{12}(r)}$. This suggests that although the PMFT is a function of all degrees of freedom required to characterize the relative configuration of a pair of particles, examining a more limited coordinate system can still be informative. Figure 7.5 shows two examples of PMFTs that contain more information than a PMF without containing all available degrees of freedom. In the 2D PMFT in the left panel, the orientation of the second particle is ignored, but its angular position relative to the reference particle is sufficient to illustrate the clear preference for facet-to-facet alignment. Similarly, the right panel ignores the orientation of the second particle (which encodes three degrees of freedom in 3D, as represented by e.g. Euler angles), but once again the preference for facet-to-facet alignment is clear. For an example of a case where analyzing the full, high-dimensional PMFT is necessary, see ref.

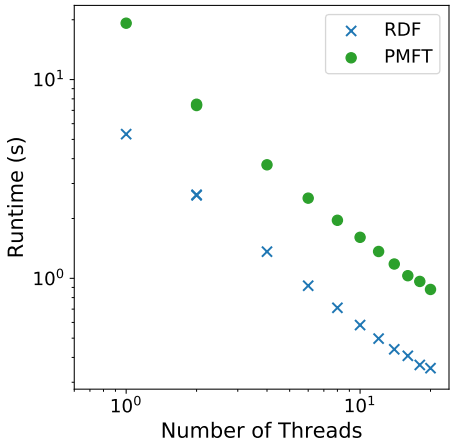


Figure 7.6 | This benchmark compares the performance of the 2D PMFT to that of an RDF on the same two-dimensional system of hexagons used in fig. 7.5. This computation was performed with a randomly generated trajectory of consisting of 10 frames of 20000 particles. Particle positions were constructed such that each particle would have, on average, 12 neighbors within a distance of 1. The benchmarks were performed on a system with an Intel® Xeon® CPU E5-2680 v2 @ 2.80GHz. Both methods have essentially the same performance characteristics, with the PMFT approximately three times slower than the RDF. We do not report error bars due to the extremely low variance in the data.

[212].

`freud` calculates the PMFT by accumulating a histogram of the configurations of all other particles and then taking the negative logarithm of the counts. PMFTs may be accumulated over many frames to generate smoother energy surfaces. This method of computing the PMFT is very similar to that of computing an RDF, so we compare their scaling behavior in a two-dimensional system in fig. 7.6. The calculations scale almost identically to many threads, with a constant scaling factor between them. There are two components contributing to the absolute difference in their performance: 1) the extra operations required to compute the orientation of the local coordinate system in the PMFT, and 2) the extra cost of binning in multiple dimensions.

We also tested performance as a function of the parameters of these two methods, namely the number of bins and the maximum interparticle distance. In the latter case, both methods show the expected quadratic growth, since the number of particles included in the calculation increases as the square of this cutoff distance. The behavior with respect to the number of bins is more interesting: this parameter has no effect on performance until it becomes sufficiently large, at which point performance begins to degrade. This degradation can be understood as the result of two things: 1) poor cache performance as the histograms become too large to fit in memory, and 2) increasing costs of reduction, which can eventually affect performance. Since the PMFT shown is a two-dimensional histogram, the number of bins that can be used along each dimension before experiencing this performance drop is

commensurately smaller than can be used with the RDF; this effect would be even more pronounced for a three-dimensional PMFT.

7.4.4 Local Environments

The `environment` module provides methods for characterizing the local environments of particles that we now illustrate in greater detail.

7.4.4.1 Bond-Orientational Order Diagrams

The `BondOrder` class enables the calculation of bond-orientational order diagrams (BOODs) [14, 213–216]. Inspired by the bond-orientational order parameters defined by Steinhardt et al. [208], BOODs characterize the local ordering of systems by calculating the vectors between all neighboring particles in a system and then projecting these vectors onto a sphere. One example of how BOODs can be used is to identify n-fold ordering in a system; in simple crystal structures with n-fold coordination, the BOOD will show n peaks corresponding to the average location of nearest neighbors.

In addition to the standard BOOD calculation, the class offers some additional modes of operation that can be useful in specific cases. One mode involves finding the positions of nearest neighbors in the local coordinate system of a given particle rather than the global coordinate system, which can prevent misidentifying systems with multiple grains [14]. Another mode modifies the BOOD to help identify plastic crystals, which appear crystalline due to having translational order but lack orientational ordering. In this mode, the positions of the nearest neighbors of each particle are modified by the relative orientations of the neighboring particles, creating a BOOD in which positional ordering will no longer appear except when orientational ordering is also present.

7.4.4.2 Spherical Harmonic Descriptors

The BOOD is closely related to the Steinhardt order parameters Q_l and W_l, which measure rotational order in a system using spherical harmonics [208]. While the BOOD is essentially a histogram of nearest-neighbor bonds, the Steinhardt order parameters take this one step further, measuring l-fold order by constructing scalar quantities from rotationally invariant combinations of spherical harmonics of degree l calculated from the locations of nearest-neighbor bonds. However, spherical harmonic representations can also be used in a variety of different ways. For example, distinguishing different grains of the same crystal structure could be done using descriptors that are not rotationally invariant. Alternatively, we can often obtain rotationally-invariant descriptions of local environments for crystal structure

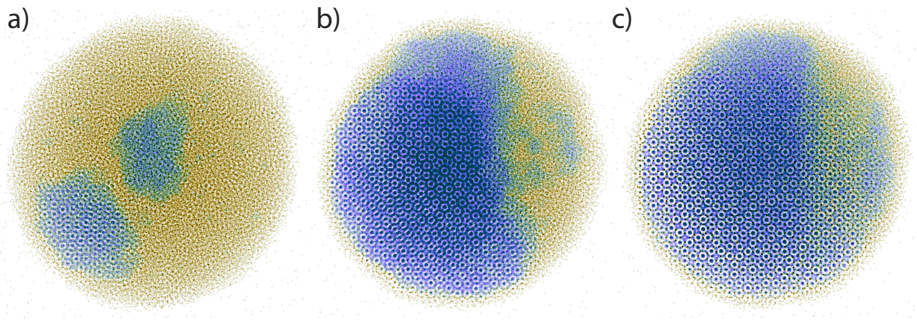


Figure 7.7 | Spherical harmonic descriptors can be used to identify the nucleation and growth of $tP30$-CrFe (Frank-Kasper σ phase). a–c) As time progresses, crystallites nucleate and grow. Solid-like particles (blue) are identified via a feedforward artificial neural network using spherical harmonic descriptors (described in more detail in [186]).

identification via the principal axes of the moment of inertia tensor of the environments, or by using particle orientations of anisotropic particles [186, 209, 217]. To support such spherical harmonic analyses, the `LocalDescriptors` class in `freud` computes spherical harmonics characterizing particle neighborhoods. These harmonics can then be combined in arbitrary ways to generate custom descriptors of local particle environments. Such descriptors have proven useful in identifying multiple complex crystals (see fig. 7.7). One method for identifying these structures is to use the information contained in this array of spherical harmonics as a set of per-particle features in an artificial neural network (ANN) for structure classification [186].

7.4.4.3 Environment Matching

Methods like the spherical harmonic descriptors and the BOOD characterize ordering in systems by calculating system-averaged quantities from neighbor bonds. The `EnvironmentCluster` class takes a different approach by defining environments according to the nearest neighbors of each particle and performing point set registration to identify and cluster similar environments [76]. This type of analysis is particularly useful because it emphasizes local information for each particle. As a result, it can be used for tasks such as identifying different Wyckoff positions in a crystal. The complementary `EnvironmentMotifMatch` class can be used to match clusters to specific motifs, allowing deeper analysis of a given structural motif.

Since this method performs a direct pairwise comparison of all motifs, it is substantially more expensive than common methods used for structure identification. For instance, the performance is at least an order of magnitude slower than the implementation of Polyhedral Template Matching in OVITO and Common Neighbor Analysis. Unlike these methods,

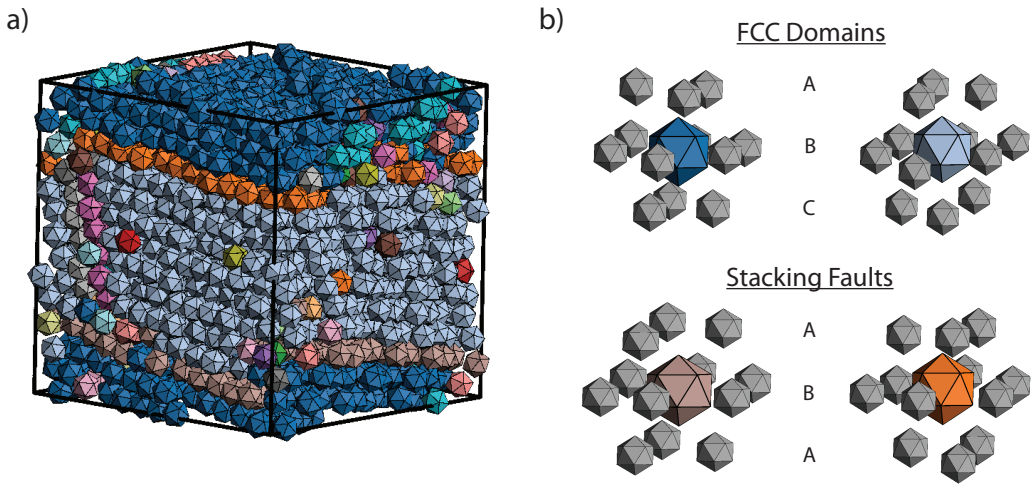


Figure 7.8 | Environment matching allows us to detect variations in the local environments of particles. a) The presence of grain boundaries in this system (rendered with `fresnel`) is clearly visible due to the different coloring according to local environments. b) The two distinct domains (blue and grey particles) are clustered separately, but we can see that they both exhibit FCC-like ($cF4$-Cu) ordering in their stacking pattern (upper-right). The environment matching method can also detect the different environments of the stacking faults themselves (mauve and orange), which exhibit an ABA stacking pattern instead of the expected ABC pattern (lower-right).

however, the environment matching algorithm does not depend on previous knowledge of possible structures, and instead infers possible structures entirely from the local motifs present in a system. Moreover, it can be tuned much more finely than the other methods, allowing not only the identification of crystal structures present, but also the precise identification of stacking faults like those found in fig. 7.8. As a result, it is a good complement to existing methods for identifying crystal structures in various systems.

7.4.4.4 Angular Separation

The `AngularSeparation` classes provide a way to characterize typical particle orientations in a system. The `AngularSeparationGlobal` class allows comparison of particle orientations to a set of reference orientations, which can be used to characterize orientational order relative to the reference input. This metric can be used as an order parameter for measuring orientational disorder in plastic crystals, which exhibit translational order and orientational disorder [218]. Alternatively, the `AngularSeparationNeighbor` computes minimum separation angles between neighboring particles, allowing a more fine-grained analysis of the orientational ordering in local motifs. Both of these methods account for symmetry by accepting an array of equivalent quaternions corresponding to all symmetry-preserving transformations of the particle (i.e. the particle's point group).

7.4.5 Data Generation and Plotting

For the purposes of teaching via code examples and testing `freud`'s analyses, `freud` includes the `freud.data` module. It includes the `UnitCell` class for representing arbitrary unit cells with user-provided box vectors and basis positions. The `UnitCell` class includes class methods that generate common crystal structures like face-centered cubic, body-centered cubic, and simple cubic. The `data` module also includes a method for generating a random system with uniformly distributed points in a periodic box.

Many analysis modules in `freud` implement a `plot()` method, which can be used for rapid data visualization. The Compute classes (e.g. instances of `freud.density.RDF`) also define a `_repr_png_()` method that allows their data to be automatically plotted in IPython environments (such as Jupyter notebooks) using Matplotlib [85], when the last line in a code cell returns that analysis object.

7.5 Examples

In this section, we demonstrate the use of `freud` in conjunction with the broader scientific software ecosystem. The code for these examples and many others is available at https://github.com/glotzerlab/freud-examples.

7.5.1 RDF and MSD from LAMMPS simulation

Here, we consider the problem of calculating the RDF and the MSD of a system simulated using LAMMPS (version 5 Jun 2019) [175]. LAMMPS is a standard tool for particle simulation used in many fields, and it supports multiple output formats, including those used by other simulation codes (e.g., the DCD format from CHARMM [219] and the XTC format from GROMACS [176]). In this case, we demonstrate the case of using the output of a custom dump format in LAMMPS, which allows users to dump selected quantities into a text file. Although the default XYZ file format lacks sufficient information to calculate an MSD, the necessary particle image information can be included as shown.

```
import numpy as np
import freud

# For the MSD we also need images, which can be dumped
# using the LAMMPS dump custom command as follows:
# dump 2 all custom 100 output_custom.xyz x y z ix iy iz
```

```
# We read the number of particles, the system box, and the
# particle positions into 3 separate arrays.
N = int(np.genfromtxt(
    'output_custom.xyz', skip_header=3, max_rows=1))
box_data = np.genfromtxt(
    'output_custom.xyz', skip_header=5, max_rows=3)
data = np.genfromtxt(
    'output_custom.xyz', skip_header=9,
    invalid_raise=False)

# Remove the unwanted text rows
data = data[~np.isnan(data).all(axis=1)].reshape(-1, N, 6)

box = freud.box.Box.from_box(
    box_data[:, 1] - box_data[:, 0])

# We shift the system by half the box lengths to match the
# freud coordinate system, which is centered at the origin.
# Since all methods support periodicity, this shift is
# simply for consistency but does not affect any analyses.
data[..., :3] -= box.L/2
rdf = freud.density.RDF(bins=100, r_max=4, r_min=1)
for frame in data:
    rdf.compute(system=(box, frame[:, :3]), reset=False)

msd = freud.msd.MSD(box)
msd.compute(
    positions=data[:, :, :3], images=data[:, :, 3:])

# The object contains all the data we need to plot the RDF.
from matplotlib import pyplot as plt
plt.plot(rdf.bin_centers, rdf.rdf)
plt.title('Radial Distribution Function')
plt.xlabel('$r$')
plt.ylabel('$g(r)$')
plt.show()
```

If our trajectory was stored in a DCD file, we could modify our code above to read the input data using MDAnalysis (version 0.20.1):

```
reader = MDAnalysis.coordinates.DCD.DCDReader(
    'output.dcd')
rdf = freud.density.RDF(bins=100, r_max=4, r_min=1)
for frame in reader:
```

```
    rdf.compute(system=frame, reset=False)
...
```

7.5.2 On-the-fly analysis with HOOMD-blue

A major strength of `freud` is that it can also be used for on-the-fly analysis. For example, `freud` can be used to terminate a simulation based on some additional condition, or to log a quantity at a higher frequency than we want to save the full system trajectory. In our previous example, we demonstrated the calculation of an RDF using `freud`. An RDF can be noisy when calculated with limited data, so we would like to average it over a large number of simulation frames; however, storing many frames can lead to unreasonably large simulation trajectory files. Using the simulation engine HOOMD-blue (v2.9.0), we can accumulate RDF data during a simulation without storing the entire output. Additionally, we show that we can log an order parameter over the course of the simulation:

```
import hoomd
from hoomd import hpmc
import freud
import numpy as np

hoomd.context.initialize()
system = hoomd.init.create_lattice(
    hoomd.lattice.sc(a=1), n=10)
mc = hpmc.integrate.sphere(seed=42, d=0.1, a=0.1)
mc.shape_param.set('A', diameter=0.5)

rdf = freud.density.RDF(bins=50, r_max=4)
q6 = freud.order.Steinhardt(l=6)

def calc_rdf(timestep):
    snap = system.take_snapshot()
    rdf.compute(system=snap, reset=False)

def calc_Q6(timestep):
    snap = system.take_snapshot()
    q6.compute(system=snap,
               neighbors={'num_neighbors': 12})
    return np.mean(q6.particle_order)

# Equilibrate the system before accumulating the RDF.
hoomd.run(1e4)
```

```
hoomd.analyze.callback(calc_rdf, period=100)

logger = hoomd.analyze.log(filename='output.log',
                           quantities=['q6'],
                           period=100,
                           header_prefix='#',
                           overwrite=True)

logger.register_callback('q6', calc_Q6)

hoomd.run(1e4)

# Store the computed RDF in a file.
np.savetxt('rdf.csv',
           np.vstack((rdf.bin_centers, rdf.rdf)).T,
           delimiter=',', header='r, g(r)')
```

7.5.3 Analyzing Atomistic Trajectories from GROMACS

As discussed in sections 7.1 and 7.2, `freud`'s design focus differs from that of many similar tools in the lack of focus on trajectory management. The example below is based on a simulation trajectory of water molecules in a box generated using GROMACS (version 2020) [176]. We use MDTraj (version 1.9.3) [179] to read in an XTC trajectory file and then compute an RDF of the oxygen atoms in the water molecules using `freud`. In the process, we demonstrate how the sophisticated subsetting functionality offered by tools like MDTraj can be replicated with Python code, which is very useful when such subsets must be computed from coarse-grained trajectories with highly customized topology definitions that standard trajectory management tools cannot handle.

```
import mdtraj
import freud
import numpy as np

traj = mdtraj.load_xtc(
    'output/prd.xtc', top='output/prd.gro')
bins = 300
r_max = 1
r_min = 0.01

# Expression selection, a common feature of analysis tools
```

```
# for atomistic systems, can be used to identify all
# oxygen atoms.
oxygen_pairs = traj.top.select_pairs('name O', 'name O')

# We can directly use the above selection in freud.
oxygen_indices = traj.top.select('name O')

# Alternatively, we can subset directly using Python logic.
# Such selectors require the user to define the nature of
# the selection, but can be more precisely tailored to a
# specific system.
oxygen_indices = [atom.index for atom in traj.top.atoms
                  if atom.name == 'O']

freud_rdf = freud.density.RDF(
    bins=bins, r_min=r_min, r_max=r_max)
for system in zip(np.asarray(traj.unitcell_vectors),
                  traj.xyz[:, oxygen_indices, :]):
    freud_rdf.compute(system, reset=False)

# We can plot these RDFs to verify that they are equivalent.
from matplotlib import pyplot as plt
fig, ax = plt.subplots()
ax.plot(freud_rdf.bin_centers, freud_rdf.rdf, 'o',
        label='freud', alpha=0.5)
ax.plot(*mdtraj_rdf, 'x', label='mdtraj', alpha=0.5)
ax.set_xlabel('$r$')
ax.set_ylabel('$g(r)$')
ax.set_title('Radial Distribution Function')
ax.legend()
```

7.5.4 Common Neighbor Analysis

Common Neighbor Analysis (CNA) [220] is a standard technique for analyzing the local neighborhoods of particles in a crystal. The method involves a classification of local neighborhoods based on a number of features. Using `freud`'s `NeighborList`, however, the method is straightforward to implement in Python.

We first consider the simpler problem of identifying all common neighbors between any pair of points. This is equivalent to searching for the **second-nearest** neighbor pairs, which can be done using `freud` as follows (note that this code is primarily written for clarity and could easily be optimized):

```
import freud
import numpy as np
from collections import defaultdict

# Use a face-centered cubic (fcc) system.
box, points = freud.data.UnitCell.fcc().generate_system(4)
aq = freud.AABBQuery(box, points)
query = aq.query(
    points, {'num_neighbors': 12, 'exclude_ii': True})
nl = query.toNeighborList()

# Get all sets of common neighbors.
common_neighbors = defaultdict(list)
for i, p in enumerate(points):
    selection1 = nl.query_point_indices == i
    for j in nl.point_indices[selection1]:
        selection2 = nl.query_point_indices == j
        for k in nl.point_indices[selection2]:
            if i != k:
                common_neighbors[(i, k)].append(j)
```

Our dictionary `common_neighbors` now contains lists of common neighbors `j` for every pair of points `(i, k)`. This information could itself be useful for performing some analysis on the system. If we are interested in actually implementing CNA, then we need to use this information to build local graphs, which we can do with the `networkx` (version 2.4) [221] Python package. Combined with the code above, the CNA algorithm can be implemented as follows:

```
import networkx as nx
from collections import Counter

diagrams = defaultdict(list)
particle_counts = defaultdict(Counter)

for (a, b), neighbors in common_neighbors.items():
    # Build up the graph of connections between the
    # common neighbors of a and b.
    g = nx.Graph()
    for i in neighbors:
        for j in set(nl.point_indices[
                nl.query_point_indices == i]
                ).intersection(neighbors):
```

```
            g.add_edge(i, j)

    # Define the four identifiers for a CNA diagram:
    #     1. 1 if particles are bonded, 0 if not.
    #     2. Number of shared neighbors.
    #     3. Number of bonds between shared neighbors.
    #     4. Index guaranteeing diagram uniqueness.
    diagram_type = 2 - int(
        b in nl.point_indices[nl.query_point_indices == a])
    key = (diagram_type, len(neighbors),
           g.number_of_edges())
    # If we've seen any neighborhood graphs with this
    # signature, we explicitly check if the two graphs are
    # identical to determine whether to save this one.
    # Otherwise, we always add the new graph.
    if key in diagrams:
        isomorphs = [
            nx.is_isomorphic(g, h) for h in diagrams[key]]
        if any(isomorphs):
            idx = isomorphs.index(True)
        else:
            diagrams[key].append(g)
            idx = diagrams[key].index(g)
    else:
        diagrams[key].append(g)
        idx = diagrams[key].index(g)
    cna_signature = key + (idx,)
    particle_counts[a].update([cna_signature])
```

In this code, we are looping over all pairs of previously identified second neighbor shells, and finding bonds between the common neighbors of these pairs. The graph of these bonds then uniquely identifies a new environment.

7.6 Conclusion

`freud` is a high-performance Python library for analyzing particle simulations. Among simulation analysis packages, `freud` is unique due to its emphasis on coarse-grained simulations and its flexibility. Its high-performance C++ back-end makes `freud` a suitable solution for large-scale, high-throughput simulation analysis, while its simple, compact API is highly amenable to integration with other tools for, e.g., machine learning applications. The package's API also promotes the prototyping of new analyses directly in Python, and

the intuitive design of `freud`'s internals ensures that translating such analyses into C++ is a relatively painless process.

The package's design is general enough to work with any particle-based system. However, `freud` is primarily targeted at communities of materials scientists, chemical engineers, and physicists analyzing molecular dynamics and Monte Carlo for which existing tools are too specialized to be convenient. Since it makes no assumptions about the types of its input data or the system topology, `freud` can be used with arbitrary simulation outputs based on topologies defined by the user. As a result, `freud` can find wide use in these areas to simplify workflows that require consideration of periodic systems without the complexity associated with specific atomistic features. Contributions to this open-source toolkit are highly encouraged as new methods are developed in future research applications.

CHAPTER 8

Conclusions and Future Work

8.1 Summary

In review, this dissertation considered the best ways to represent shape in various types of colloidal and nanoparticle systems. By covering a range of systems in both experiment and simulation, we demonstrated that hard particle approaches can indeed be a suitable means for efficient and accurate study of certain classes of particles. We also demonstrated ways to move beyond the limitations of the hard particle model, including the use of machine learning methods to inform more complex compound models based on many interactions as well as the development of anisotropic pair potentials consistent with classical approaches to such simulations. These results provided valuable information for future studies where simple hard particle models may not be appropriate, but where such models may be adapted to generate suitable high-fidelity minimal representations of the relevant physics.

In chapter 2, we discussed how the hard particle model could be used in concert with well-established physical models of the depletion interaction to capture the self-assembly behavior of systems of hard ATTs. We placed these results in the context of various recent computational studies that study ATTs in various contexts, focusing on the effects of the depletion force and particle rounding, and we showed how these results can be applied in this experimental system that provides a model for their study. Additionally, we demonstrated some cases where the particular idiosyncrasies of this system lead to results that differ from those of previously well-studied systems.

In chapter 3, we moved on to using a hard nonconvex polyhedron with embedded point charges to represent a supercharged protein. We discussed how this model provides useful stability results but fails to capture the physics of the system well enough to model its the assembly of these proteins into a protomeric structure. In this light, we discussed how machine learning approaches may be used to augment molecular simulations to predict protein structures from the vast array of data available in the PDB. We proposed a deep

learning approach to elucidate the protein-crystal structure relationship, discussing the challenges and benefits to its implementation and some results.

In chapter 4 we take an alternate approach: instead of building directly on top of hard shapes to create more complex models, we considered a generalization of classical isotropic pair potentials that allows shape anisotropy to be encoded within a pair potential function. We advanced a theory that systematically generalizes isotropic pair potentials via a mean-field approximation to an anisotropic field, and we developed a high-performance implementation within HOOMD-blue. We provided performance comparisons of this method to current best-in-class solutions for specific subclasses of shapes, demonstrating that our implementation provides substantial accelerations over existing methods. Finally, we validated the thermodynamic behavior of this model for the purpose of studying assembled morphologies.

Chapter 5 used this new potential to rigorously evaluate the effect of shape on particle dynamics, exploiting the analytical shape representation to perform dynamic simulations that would previously have been prohibitively expensive. Focusing on a model system of 2D regular polygons, we showed that particle anisotropy introduces additional modes into time correlation functions like the MSD that govern intermediate-time behavior in between the classical ballistic and diffusive regimes. Although increasing the numbers of sides of a regular polygon leads to monotonic changes in its moment of inertia, we found that other characteristic behaviors in fact do not change monotonically in this context, and we developed a collision-theory framework to explain this phenomenon. Ultimately, we found that nontrivial translation-rotation coupling is responsible for modifying particle dynamics in ways that cannot be observed from isotropic systems.

Chapters 6 and 7 described software development done to facilitate the research in chapters 2 to 5. Chapter 6 described a heuristic for optimizing research output via proper software development, an approach used not only in the development of the `freud` tool discussed in chapter 7, but also in the development of a number of other packages I have developed or contributed to in various ways. These chapters highlighted the critical role of not only writing code, but writing code well, in conducting computational research today.

8.2 Outlook and Future Work

8.2.1 Deep Learning of Protein Features

The work in chapter 3 offers an effective strategy for handling the large quantities of data that must be streamed through any machine learning model for proteins that does not attempt

an *a priori* feature reduction. Since the development of a sufficiently efficient strategy was a roadblock to actually building out the machine learning model, the current versions of the model are functional but provide highly inaccurate predictions. One significant challenge with these models is that space groups are a complex property to predict that may not have a simple relationship to the provided inputs. As a result, future work in this area should attempt to predict a simpler property of the assembly. Such models would have a greater chance of at least partial success, permitting iterative refinement that would also provide greater insight into the relative importance of protein shape, electrostatics, hydrophobicity, and perhaps other descriptors as well.

One natural candidate is binding sites for protein dimers, which recent ML models have had success in predicting [105]. These other models have employed alternate strategies for deep learning models of protein properties that exploit different features of the data for efficiency and embed shape in less explicit ways, suggesting that direct comparison with these approaches may also provide an additional means of evaluating the role of shape [105]. Adapting the existing models to predict dimerization sites would be relatively simple and would likely help to refine the model. Once the model has been shown to successfully predict a known feature, the resulting refined model could then be reapplied to the space group question to see what additional changes must be made to successfully predict the space group from a protein surface.

8.2.2 Evaluating Methodologies for Simulating Anisotropic Particles

As discussed in chapter 4, the dominant means of simulating anisotropic particles to date has been HPMC simulation, which provides information only about equilibrium (or more precisely, sufficiently long-lived) states in a simulation and includes no dynamic information. The advent of the ALJ and other related potentials provides a route to evaluate what information could be missing from MC simulations and whether studies of kinetic pathways using MC are reliable indicators of the true intermediate states in structure formation. Such studies would also permit rigorous investigation of exactly what features are necessary for two pathways to be considered "the same". More precisely, such studies can answer whether MC and MD methods can be used interchangeably for a study depending on a specific set of physical properties.

APPENDIX A

Supplementary Information for Chapter 2

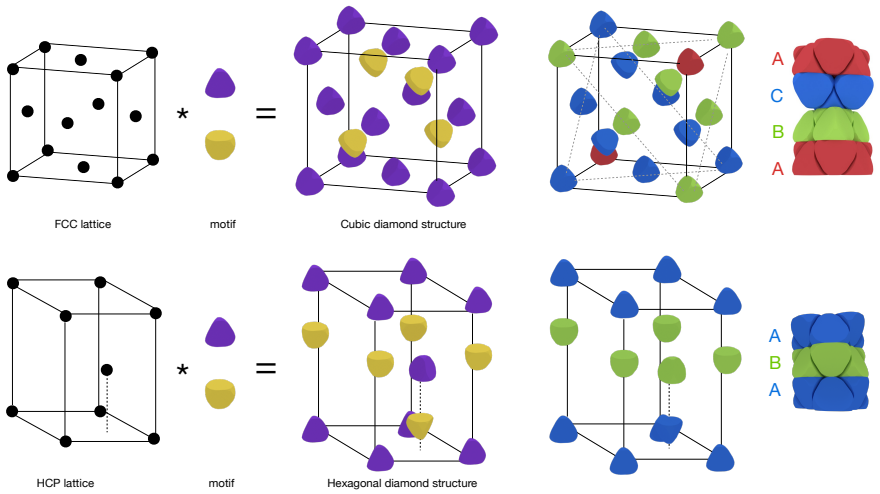


Figure A.1 | **Cubic diamond and hexagonal diamond structures.** Top: Cubic diamond structure assembled from truncated tetrahedra. The cubic diamond structure is obtained by the convolution of a face-centered cubic (FCC) lattice with a motif of oppositely-oriented tetrahedra. Particles form layers that stack in an ABCA... sequence. Bottom: Hexagonal diamond structure assembled from truncated tetrahedra. The hexagonal diamond structure is obtained by the convolution of a hexagonal close-packed (HCP) lattice with a motif of oppositely-oriented tetrahedra. Particles form layers that stack in an ABA... sequence.

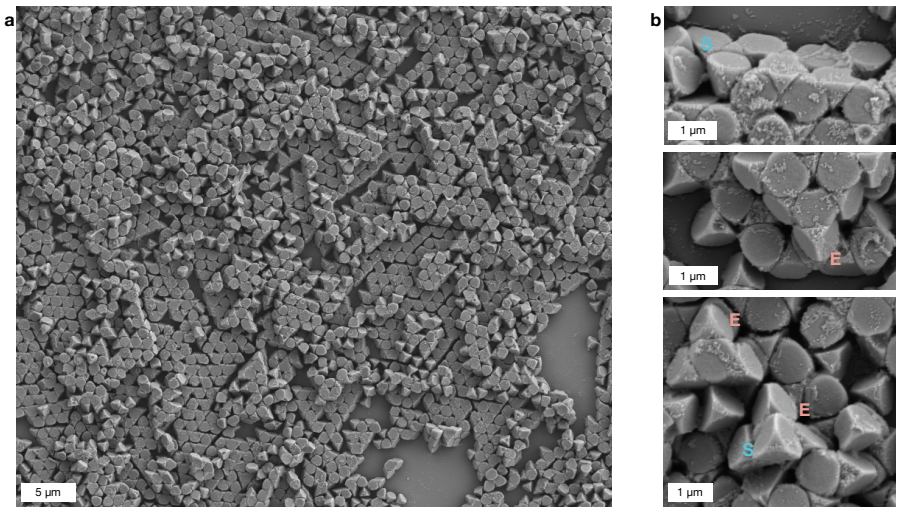


Figure A.2 | **Eclipsed and staggered inter-layer conformations in depletion-driven assembly.** (a) SEM image of a double-layered random hexagonal close-packed structure assembled in a capillary. Scale bar: $5\,\mu\mathrm{m}$. (b) Zoomed-in SEM images of the random hexagonal close-packed structure showing co-existence of two types of inter-layer stackings: staggered (S) and eclipsed (E). Scale bars: $1\,\mu\mathrm{m}$.

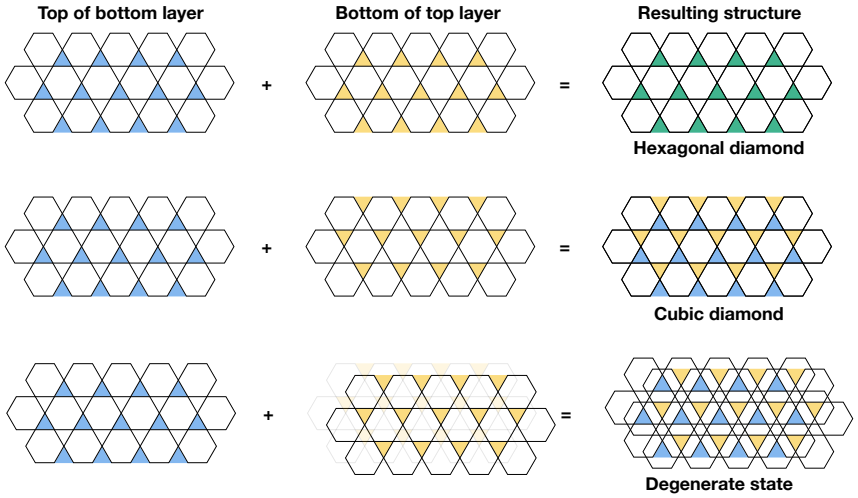


Figure A.3 | **Electrostatic contribution to the self-assembly.** Differences in electrostatic potential energy between different stackings follow from differences in contact area between the two layers. Holes in the top surface of the bottom layer are colored in blue. Holes in the bottom surface of the top layer are colored in yellow. In eclipsed stacking, holes are aligned. The resulting hexagonal diamond has higher contact area (in white) between layers than the cubic diamond. Since it is possible to translate the top layer keeping the same contact area as cubic diamond, it is a degenerate energy state.

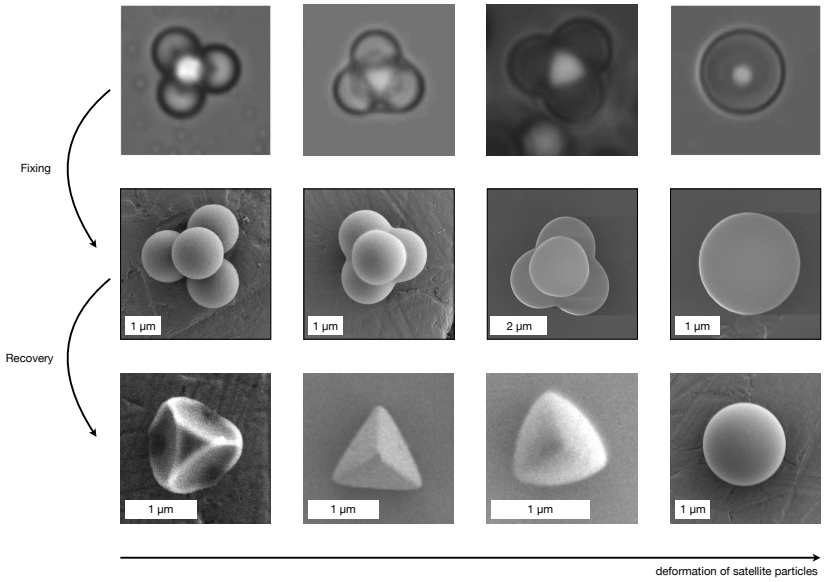


Figure A.4 | **From tetrahedral cluster to core-shell particle.** As the latex satellites are increasingly deformed, the core particles change shape from a concave truncated tetrahedra to a convex tetrahedra with sharp tips, to a sphere forming the core of a fully liquid core-shell particle. First row shows typical optical microscope images of the tetrahedral clusters along the deformation. Second row shows typical scanning electron images of the tetrahedral clusters acquired after fixing the core via radical polymerization. Third rows shows typical scanning electron images of the inner core particle recovered by dissolving the latex satellites.

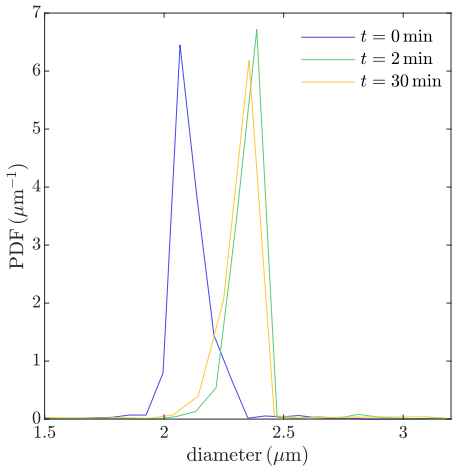


Figure A.5 | **Fast swelling of polystyrene colloids in THF.** Probability distribution functions of the diameter of polystyrene colloids $0\,\mathrm{min}$, $2\,\mathrm{min}$, and $30\,\mathrm{min}$ after dispersion in $20\,\mathrm{v}\%$ THF. Changes in particle volume is fast and happen in less than $2\,\mathrm{min}$. Consequently, satellites properties (viscosity, surface tensions) can be considered constant over the course of the tetrahedral cluster deformation. Size measurements are performed with xSights from Spheryx, Inc. by holographic characterization.

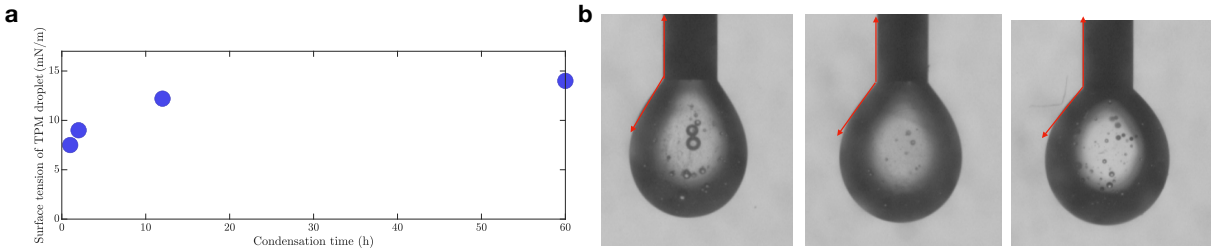


Figure A.6 | **Surface tension of TPM with change in cross-linking.** (a) Surface tension of TPM for increasing condensation time. (b) Pictures of the TPM droplets in 20 wt% THF water used to estimate the surface tension. Measurements were done with the commercial apparatus Attension Theta Optical Tensiometer, Biolin Scientific. The volume of the droplet is ∼3.3 µL.

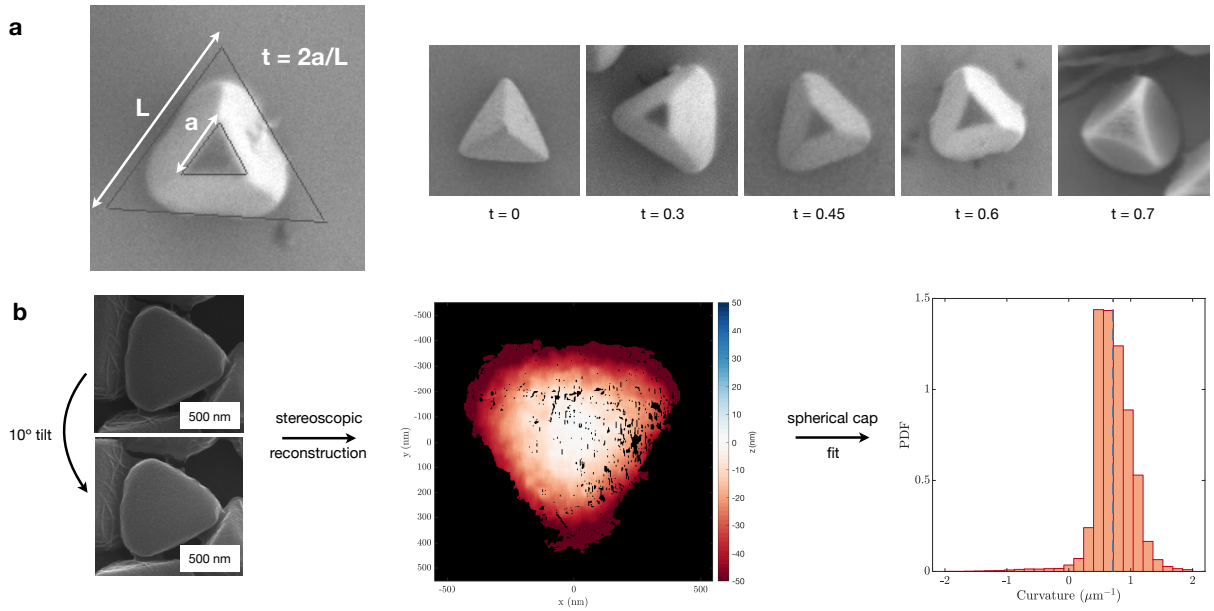


Figure A.7 | **Quantitative characterization of the morphology of colloidal tetrahedra.** (a) Measurements of the truncation t are performed on scanning electron images of tetrahedra pointing up. (b) Estimation of the face curvature is performed by steresoscopic reconstruction using the commercial software MountainsMap by Digital Surf and custom Matlab codes. Two scanning electron microscope images of the same particle are taken to a 10° angle. The elevation map of the surface is reconstructed from the correlations of the two images. We approximate the face by a spherical cap for which each point of the elevation map gives an estimate of the curvature. The most probable value gives a characteristic value of the face curvature.

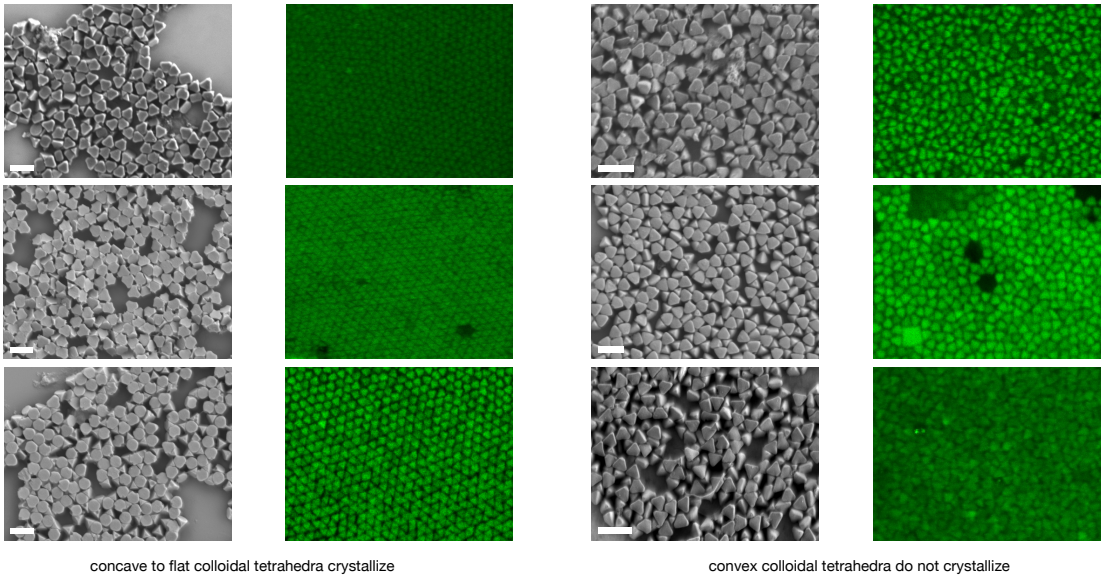


Figure A.8 | **Sedimentation of tetrahedra with various face curvatures.** Scanning electron images (black and white images) of various colloidal tetrahedra and corresponding confocal horizontal slices of their sediment (green images). Left: colloidal tetrahedra with concave to flat faces crystallize via sedimentation. Right: colloidal tetrahedra with convex faces do not crystallize via sedimentation. Scale bars: $2\,\mu\mathrm{m}$.

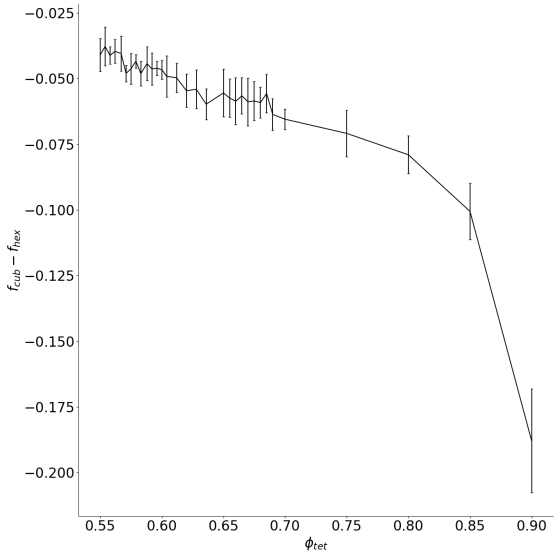
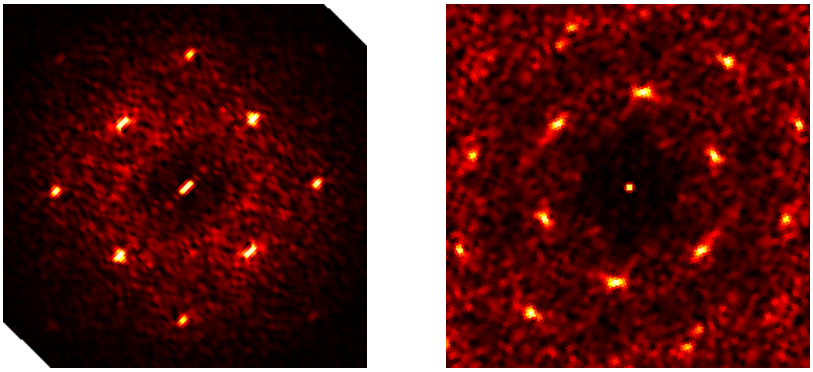


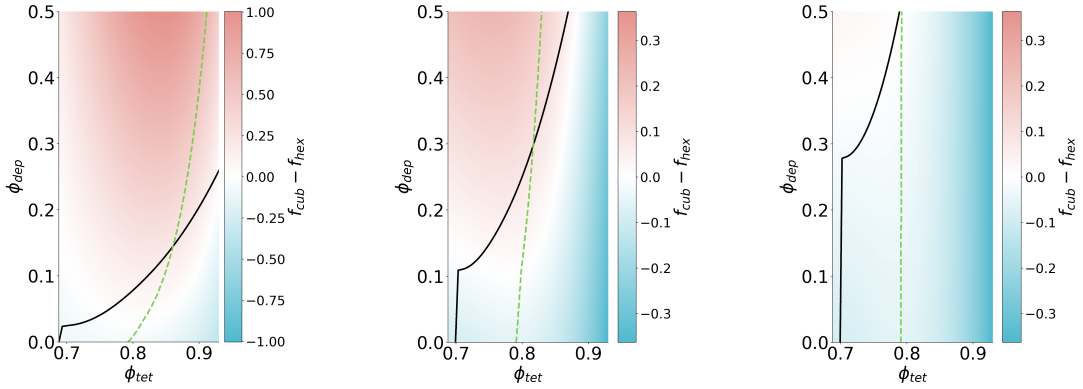
Figure A.9 | **Free energy differences without depletion.** A negative value on the y axis indicates a lower, and therefore thermodynamically preferable, free energy for cubic diamond.



(a) The diffraction pattern projected onto the (100) plane, which exhibits 4-fold symmetry.

(b) The diffraction pattern projected onto the (111) plane, which exhibits 6-fold symmetry.

Figure A.10 | **Diffraction patterns of sedimented structure from simulation.** (a) The 4-fold diffraction pattern is commonly associated with the diamond structure. (b) Since the sedimentation-driven assembly leads to growth in the (111) direction, we also include the diffraction pattern projected along this axis, which is a 6-fold pattern corresponding to the hexagonal ordering observed.



(a) The difference in free energies between CD and HD for a depletant 10% the size of the tetrahedra.

(b) The difference in free energies between CD and HD for a depletant 15% the size of the tetrahedra.

(c) The difference in free energies between CD and HD for a depletant 20% the size of the tetrahedra.

Figure A.11 | **Free energy differences in the presence of depletants.** For a given ϕ^r_{dep}, smaller depletants have a much larger n^r_{dep}, resulting in a much stronger effect on the assembly behavior as predicted by FVT. As shown in panel (**c**), sufficiently large depletants have a negligible effect ($\leq 0.1 k_B T$) except at very high packing fractions well beyond the assembly regime. The solid black line indicates the coexistence boundary, while the dashed green line indicates the boundary of the region where α-arsenic is the thermodynamically preferred structure. Note that panel (**a**) is identical to Fig. 4f in the main text, but is reproduced here for comparison.

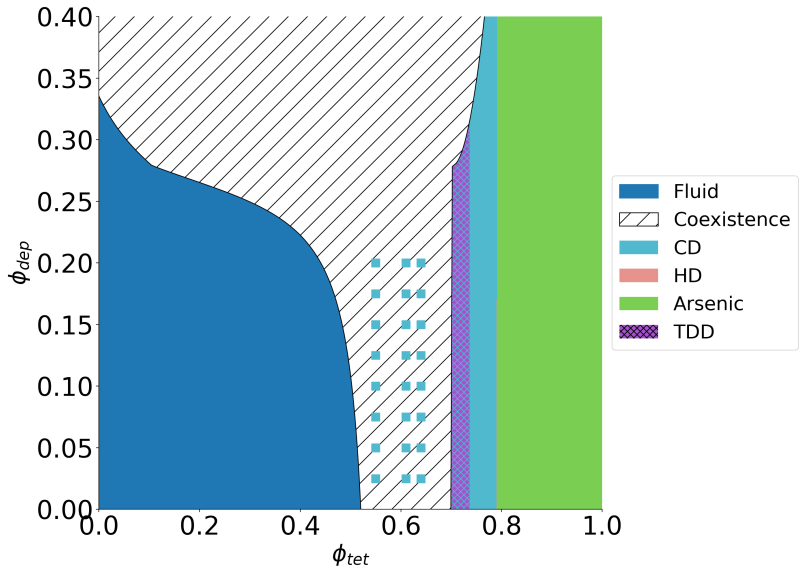


Figure A.12 | **Phase diagram for depletants with $q = 0.2$, larger than those shown in the main text.** Depletants of this size have a very minimal effect on the free energies of different structures as shown in fig. A.11. As a result, the phase diagram shows that phase boundaries between different crystal structures are effectively unaffected by ϕ^r_{dep}. Some widening of the coexistence regime is observed at higher ϕ^r_{dep}, but the effect is muted relatively to that observed for smaller depletants.

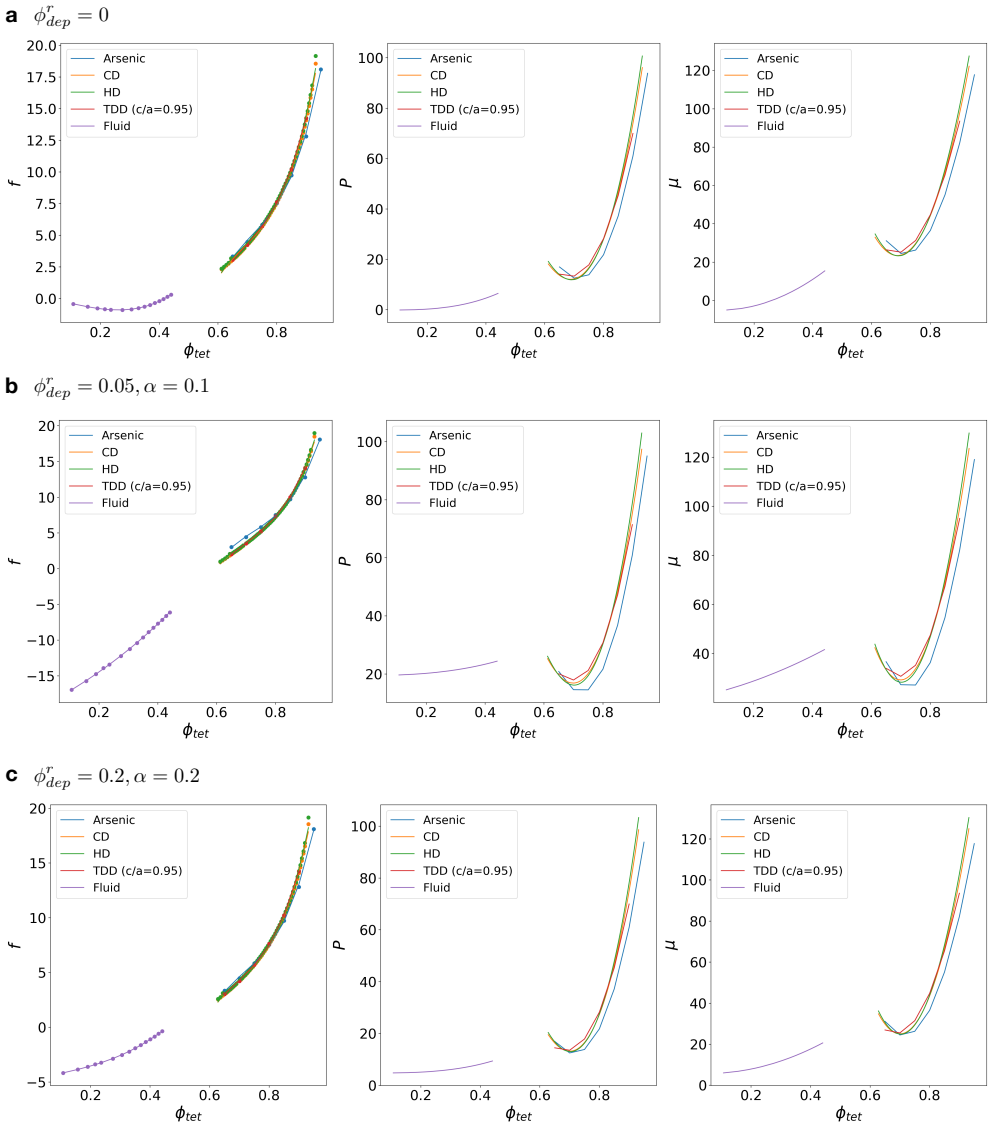


Figure A.13 | **Examples of common tangent construction application.** To find regions of phase coexistence, for a given depletant size (given by $\alpha = r_{dep}/r_{tet}$) and packing fraction we compute the free energy density for the fluid and a range of structures (left). We then compute derivatives to get the pressure (middle) and chemical potential (right). The coexistence condition for a fluid and a particular crystal structure is then given by a pair of packing fractions where both the pressure and the chemical potential are equal; equivalently, by the pair of packing fractions where a common tangent exists between the free energy density curves. The phase boundaries in the phase diagrams in the main text can be determined by repeating this process for a range of depletant packing fractions. **(a)** The construction in the absence of depletants. **(b)** The results for a small depletant representative of the ones in experiment; even at the low packing fraction shown, the free energy curves are clearly dramatically shifted. **(c)** The results for larger depletants; despite having a depletant packing fraction four times higher than in (b), the depletant effect is muted because the depletants are too large.

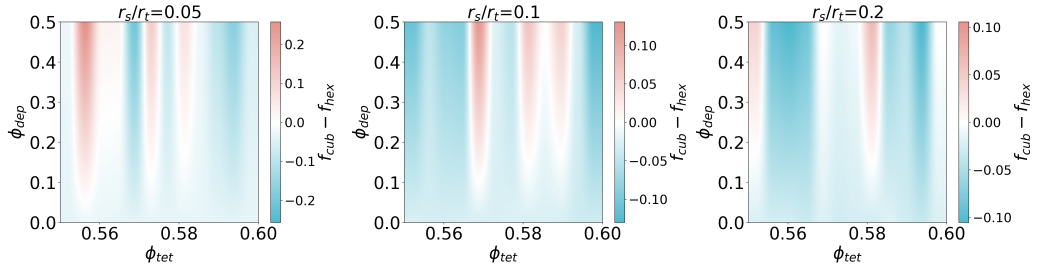


Figure A.14 | **Effect of particle rounding on cubic diamond (CD)/hexagonal diamond (HD) competition.** We consider the relative free energy densities of CD and HD for varied levels of tetrahedron rounding within a narrow range of packing fractions where systems are dense enough to easily facilitate nucleation without being so dense as to introduce kinetic traps. Using a depletant size representative of that in experiment ($q = 0.05$), we find that the primary effect of rounding is to decrease the magnitude of the free energy difference between CD and HD while otherwise retaining most of the same qualitative features in the Δf map. This result is consistent with the fact that for spheres (the $q \to \infty$ limit) FCC and HCP have nearly identical free energies, and CD and HD are, respectively, the decorations of these lattices with two particles. Therefore, at high rounding we expect CD and HD to become nearly indistinguishable from a thermodynamic perspective.

APPENDIX B

Supplementary Information for Chapter 5

B.1 Derivation of Anisotropic Langevin Equation

Here, we present a derivation of our proposed Langevin equation, specialized for hard anisotropic particles. We start from the generalized Langevin equation as derived using standard projection formalism:

$$\frac{\partial}{\partial t}\mathbf{A}\left(t\right) - i\mathbf{\Omega}\cdot\mathbf{A} + \int_0^t \mathbf{M}\left(t-s\right)\cdot\mathbf{A}\left(s\right)ds = \mathbf{f}\left(t\right) \tag{B.1}$$

where $\mathbf{A}$ is a multivariate vector containing various properties of interest (position, velocity, angular momentum etc...), $\mathbf{\Omega}$ is the frequency matrix, t is time, and $\mathbf{M}$ and $\mathbf{f}$ are the corresponding memory kernel and random force of the parameters in $\mathbf{A}$, respectively. For hard, anisotropic particles, the relevant parameters describing their dynamics are the position ($\mathbf{r}$), velocity ($\mathbf{v}$), and angular velocity ($\mathbf{l}$). We then define $\mathbf{A}$ as $\mathbf{A} = [\mathbf{r}, \mathbf{v}, \mathbf{l}_p]$, where $\mathbf{l}_p$ is the projected form of $\mathbf{l}$ defined as followed to ensure orthogonality: $\mathbf{l}_p = \mathbf{l} - (\mathbf{r}, \mathbf{l})(\mathbf{r}, \mathbf{r})^{-1}\mathbf{r}$.

Assuming a Maxwell-Boltzmann distribution for both linear and angular velocities, we now evaluate each term in Eq. B.1 based on our definition of $\mathbf{A}$. While tedious, it is straightforward to evaluate each term within the correlation matrix, frequency matrix, memory kernel, and fluctuating force. We, therefore, report only the final result for each. For ease of notation, we will introduce the subscript o to indicate the zero-time limit: $\mathbf{A}_o \equiv \mathbf{A}(0)$

$$\left(\mathbf{A}_o, \mathbf{A}_o\right) = \begin{bmatrix} \mathbf{R}_o & 0 & 0 \\ 0 & \frac{F}{2}\frac{NkT}{m} & 0 \\ 0 & 0 & \frac{F}{2}\frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o} \end{bmatrix} \tag{B.2}$$

where F is the degrees of freedom in the system, $\mathbf{R}_o$ is defined as $(\mathbf{r}_o, \mathbf{r}_o)$, and $\mathbf{R}_{l,o}^2$ represents $(\mathbf{r}_o, \mathbf{l}_o)^2 (\mathbf{r}_o, \mathbf{r}_o)^{-1}$. Using Eq. B.2, we can evaluate the frequency matrix $\mathbf{\Omega} = \left(\mathbf{A}_o, \dot{\mathbf{A}}_o\right)\left(\mathbf{A}, \mathbf{A}\right)^{-1}$.

$$\mathbf{\Omega} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{F}{2}\frac{NkT}{m}\frac{1}{\mathbf{R}_o} & 0 & -\left[\frac{F}{2}\frac{NkT}{m}\frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s\right]\left[\frac{F}{2}\frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o}\right]^{-1} \\ 0 & \frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s\left[\frac{F}{2}\frac{NkT}{m}\right]^{-1} & 0 \end{bmatrix} \tag{B.3}$$

where we defined $\mathbf{R}_s = \frac{F}{2}\frac{NkT}{m}\frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \left(\frac{\partial \mathbf{v}_o}{\partial t}, \mathbf{l}_o\right)$ for ease of notation. Eq. B.3 can now be used to evaluate the fluctuating force matrix $\mathbf{f}$ defined as $\mathbf{f} = \dot{\mathbf{A}} - i\mathbf{\Omega}\mathbf{A}$, giving

$$\mathbf{f} = \begin{bmatrix} 0 \\ \frac{\partial \mathbf{v}}{\partial t} - i\frac{F}{2}\frac{NkT}{m}\frac{\mathbf{r}}{\mathbf{R}_o} + i\left[\frac{F}{2}\frac{NkT}{m}\frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s\right]\left[\frac{F}{2}\frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o}\right]^{-1}\mathbf{l}_p \\ \frac{\partial \mathbf{l}}{\partial t} - i\left[\frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s\left(\frac{F}{2}\frac{NkT}{m}\right)^{-1}\right]\mathbf{v} \end{bmatrix} \tag{B.4}$$

Lastly, the memory kernel $\mathbf{M}$ is defined as $\mathbf{M} = (\mathbf{f}_o, \mathbf{f}(t))(\mathbf{A}_o, \mathbf{A}_o)^{-1}$, plugging in Eq. B.2 and B.4 gives

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{f}_{22}\left[\frac{F}{2}\frac{NkT}{m}\right]^{-1} & \mathbf{f}_{23}\left[\frac{F}{2}\frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o}\right]^{-1} \\ 0 & \mathbf{f}_{32}\left[\frac{F}{2}\frac{NkT}{m}\right]^{-1} & \mathbf{f}_{32}\left[\frac{F}{2}\frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o}\right]^{-1} \end{bmatrix} \tag{B.5}$$

where the elements $\mathbf{f}_{ij}$ defines the ij^{th} element of the inner product $(\mathbf{f}_o, \mathbf{f}(t))$. Analogous to the original Lagevin equation, we are specifically interested in the velocity component of Eq. B.1, giving us the following differential equation

$$\frac{\partial}{\partial t}\mathbf{v}(t) - i\left[\mathbf{\Omega}_{21}\mathbf{r}(t) + \mathbf{\Omega}_{23}\mathbf{l}_p(t)\right] + \int \mathbf{M}_{22}(t)\mathbf{v}(t-s)ds + \int \mathbf{M}_{23}(t)\mathbf{l}_\mathbf{p}(t-s)ds = \mathbf{f}_2(t) \tag{B.6}$$

Eq. B.6 is a very general result that readily comes out of evaluating different terms of the generalized Langevin equation (Eq. B.1). Since we are interested in the hard shape limits, we now make some simplifying assumptions inspired by the seminal Langevin equation developed for Brownian motion of hard spheres. Firstly, we select for only the real component. This allows us to drop terms in $\mathbf{\Omega}$. Secondly, similar to hard spheres, we assume no coupling between the memory kernel and the dynamical parameters, enabling us to drop the convolutional integration terms. Combining both assumptions gives

$$\frac{\partial}{\partial t}\mathbf{v}(t) + \mathbf{M}_{22}(t)\mathbf{v}(t) + \mathbf{M}_{23}(t)\mathbf{l}(t) = \mathbf{f}_2(t) \tag{B.7}$$

Our first approximation involves the term $\mathbf{l}(t)$ via treating an anisotropic shape as a first order perturbation about an isotropic particle. Namely, we wish to interchange angular and translation velocities via the relation $v = r_c l$, where r_c is an radius of the effective sphere with which we wish perturb about. In this limit, consider the case of a polygon whose area is equal to that of a circle of size r_c. We can write $\pi r_c^2 = na^2 \tan(\pi/n)$, where a is the polygon's apothem. Rearranging gives

$$\left(\frac{r_c}{a}\right)^2 = \frac{n \tan(\pi/n)}{\pi} \tag{B.8}$$

While this is specific for regular polygons, we can generalize our approximation to all shapes via recasting the right hand side of Eq. B.8 into a more convenient moment of inertia shape metric. For a regular polygon, the moment of inertia has the following scaling relation $I \sim s^2$, where s is the side length. Additionally $s = 2a \tan(\pi/n)$, resulting in $I^{1/2} \sim 2a \tan(\pi/n)$. Plugging into Eq. B.8 gives

$$\left(\frac{r_c}{a}\right)^2 = \frac{n}{\pi}\frac{I^{1/2}}{2a} \rightarrow \frac{r_c}{a} \sim \left(\frac{I}{\alpha}\right)^{1/4} \tag{B.9}$$

where we have collected terms so that $\alpha = (2 * \pi a/n)^2$. Eq. B.9 suggests that recasting the anisotropic problem into a perturbation relative to an isotropic counterpart overestimates the core shape by a ratio of $(I/\alpha)^{1/4}$. To correct for this excess, we write $v \sim (I/\alpha)^{-1/4} l$, which rearranges into $l \sim (\alpha/I)^{-1/4} v$. Plugging into Eq. B.7 gives

$$\frac{\partial}{\partial t}\mathbf{v}(t) + \mathbf{M}_{22}(t)\mathbf{v}(t) + \mathbf{M}_{23}(t)(\alpha/I)^{-1/4}\mathbf{v}(t) = \mathbf{f}_2(t) \tag{B.10}$$

Our next step now involves providing approximate forms for $\mathbf{M}_{22}$ and $\mathbf{M}_{23}$. As indicated by Eq. B.5, we need to explicitly evaluate the inner product defined by $(\mathbf{f}_o, \mathbf{f}(t))$ – specifically $\mathbf{f}_{22}$ and $\mathbf{f}_{23}$. By inspection, $\mathbf{f}_{22}$ has two components: traditional translational term and a new rotational term

$$\mathbf{f}_{22} = \frac{\partial \mathbf{v}}{\partial t} - i\frac{F}{2}\frac{NkT}{m}\frac{\mathbf{r}}{\mathbf{R}_o} + i\left[\frac{F}{2}\frac{NkT}{m}\frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s\right]\left[\frac{F}{2}\frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o}\right]^{-1}\mathbf{l}_p = \mathbf{f}_{22,T} + \mathbf{f}_{22,R} \tag{B.11}$$

where $\mathbf{f}_{22,T}$ defines the first two terms of Eq. B.11 and $\mathbf{f}_{22,R}$ describes the term in $\mathbf{l_p}$. The inner product then expands to four terms:

$$(\mathbf{f}_{22_o}, \mathbf{f}_{22}(t)) = (\mathbf{f}_{22,T_o}, \mathbf{f}_{22,T}(t)) + (\mathbf{f}_{22,T_o}, \mathbf{f}_{22,R}(t)) + (\mathbf{f}_{22,R_o}, \mathbf{f}_{22,T}(t)) + (\mathbf{f}_{22,R_o}, \mathbf{f}_{22,R}(t)) \tag{B.12}$$

$(\mathbf{f}_{22,T_o}, \mathbf{f}_{22,R}(t))$ and $(\mathbf{f}_{22,R_o}, \mathbf{f}_{22,T}(t))$ are imaginary, allowing us to drop those terms using the same assumption employ for $\mathbf{\Omega}$. $(\mathbf{f}_{22,T_o}, \mathbf{f}_{22,T}(t))$ is the traditional translational inner product commonly approximated via a drag coefficient ξ_{22}. The only term remaining of interest is $(\mathbf{f}_{22,R_o}, \mathbf{f}_{22,R}(t))$. Taking a similar conversion used to obtain Eq. B.10, we can write

$$(\mathbf{f}_{22,R_o}, \mathbf{f}_{22,R}(t)) \sim \gamma_{22} \left(\alpha/I\right)^{-1/2} \mathbf{v}(t) \tag{B.13}$$

where γ_{22} is defined as $\gamma_{22} = \left(\left[\frac{F}{2} \frac{NkT}{m} \frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s \right] \left[\frac{F}{2} \frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o} \right]^{-1} \right)^2$. Inspection of this term indicates that it is a coupling between the angular and translational terms. Therefore, we interpret γ_{22} as a coupling constant between the angular and translation components of anisotropic dynamics. Like the iner product corresponding to ξ_{22}, γ_c depends on quantities like $\frac{\partial \mathbf{v}_o}{\partial t}$ and $\mathbf{l}_o$ that depend on properties of the system; therefore, analogous to ξ we must estimate it via a

$\mathbf{R}_s = \frac{F}{2} \frac{NkT}{m} \frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \left(\frac{\partial \mathbf{v}_o}{\partial t}, \mathbf{l}_o \right)$

Combining these results, we can then approximate $\mathbf{M}_{22}$ as $\mathbf{M}_{22} = \xi_{22} + \gamma_{22} \left(\alpha/I\right)^{-1/2} \mathbf{v}(t)$, where the additional constant $\left[\frac{F}{2} \frac{NkT}{m} \right]^{-1}$ is grouped into ξ_{22} and γ_{22}. An analogous analysis for $\mathbf{M}_{23}$ yields $\mathbf{M}_{23} = \xi_{23} + \gamma_{23} \left(\alpha/I\right)^{-1/4} \mathbf{v}(t)$. Plugging $\mathbf{M}_{22}$ and $\mathbf{M}_{23}$ into Eq. B.10 gives

$$\frac{\partial}{\partial t} \mathbf{v}(t) + \xi \mathbf{v}(t) + \gamma_c (\alpha/I)^{-1/2} \mathbf{v}^2(t) = \mathbf{f}_2(t) \tag{B.14}$$

where $\xi = \xi_{22} + \xi_{23}$ and $\gamma_c = \gamma_{22} + \gamma_{23}$. Eq. B.14 is the differential equation employed in the main text.

B.1.1 Frequency Correction

Within the main text, we discussed a first-order extension of Eq. B.14 to include the effect of the frequency term $i\mathbf{\Omega} \cdot \mathbf{A}$ in Eq. B.1. The relevant term from the frequency matrix is

$$- \left[\frac{F}{2} \frac{NkT}{m} \frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s \right] \left[\frac{F}{2} \frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o} \right]^{-1} \mathbf{l}_p \tag{B.15}$$

Adding to Eq. B.14, performing a similar transformation for $\mathbf{l}_p$, and grouping gives

$$\frac{\partial}{\partial t}\mathbf{v}(t) + \left[\xi + \eta\left(\alpha/I\right)^{-1/4}\right]\mathbf{v}(t) + \gamma_c(\alpha/I)^{-1/2}\mathbf{v}^2(t) = \mathbf{f}_2(t) \tag{B.16}$$

where we have defined

$$\eta = \left[\frac{F}{2}\frac{NkT}{m}\frac{\mathbf{R}_{l,o}}{\mathbf{R}_o} + \mathbf{R}_s\right]\left[\frac{F}{2}\frac{NkT}{m} - \frac{\mathbf{R}_{l,o}^2}{\mathbf{R}_o}\right]^{-1} \tag{B.17}$$

We can then use Eq. B.15 – B.17 and their analytical solution to approximate the relaxation times employed in the main text.

B.2 General Solution of Anisotropic Langevin Equation

We now present our approximate solution to our derived anisotropic Langevin equation. Starting from the the presented equation in the main text

$$m\frac{\partial}{\partial t}\mathbf{v}(t) + m\xi\mathbf{v}(t) + m\gamma_c\left(\alpha/I_n\right)^{-1/2}\mathbf{v}^2(t) = \mathbf{f}(t) \tag{B.18}$$

Similar to the solution approach for the traditional Langevin equation, we first multiply Eq. B.18 by $\mathbf{r}(t)$ and average to give

$$\frac{\partial^2\langle\mathbf{r}^2(t)\rangle}{\partial t^2} + \xi\frac{\partial\langle\mathbf{r}^2(t)\rangle}{\partial t} + \gamma\left(\frac{1}{\langle\mathbf{r}^2(t)\rangle}\right)^{1/2}\left(\frac{\partial\langle\mathbf{r}^2(t)\rangle}{\partial t}\right)^2 = \frac{\langle m\mathbf{v}^2(t)\rangle}{m} \tag{B.19}$$

where, similar to the main text, we define $\gamma = \gamma_c\left(\alpha/I\right)^{-1/2}$ for ease of notation. Eq. B.19 is an inhomogeneous, second order, nonlinear ordinary differential equation. Using standard approaches for solving such ODEs, we first solve the homogeneous ODE of the form

$$\frac{\partial^2\langle\mathbf{r}^2(t)\rangle}{\partial t^2} + \xi\frac{\partial\langle\mathbf{r}^2(t)\rangle}{\partial t} + \gamma\left(\frac{1}{\langle\mathbf{r}^2(t)\rangle}\right)^{1/2}\left(\frac{\partial\langle\mathbf{r}^2(t)\rangle}{\partial t}\right)^2 = 0 \tag{B.20}$$

Eq. B.20 has the known general solution

$$\langle\mathbf{r}^2(t)\rangle = \left(4\gamma^2\right)^{-1}\left\{1 + \mathcal{W}\left(\frac{1}{\xi}\left[\frac{C_1\gamma^2}{e} + \exp\left(-\xi\left(t + C_2\right) - 1\right)\right]\right)\right\}^2 \tag{B.21}$$

where C_1 and C_2 are integration constants that we will later define and $\mathcal{W}$ is the Lambert W function. Recognizing that the long time behavior of $\langle\mathbf{r}^2(t)\rangle$ is linear in time, we guess

the particular solution to be $\langle \mathbf{r}_p^2(t) \rangle = ht + k$. Solving for h and k with boundary conditions $\langle \mathbf{r}^2(0) \rangle = 0$ and $\partial \langle \mathbf{r}^2(0) \rangle / \partial t = 0$ results in

$$\langle \mathbf{r}^2(t) \rangle = \left(4\gamma^2\right)^{-1} \left\{ 1 + \mathcal{W}\left(\frac{1}{\xi} \left[\frac{C_1 \gamma^2}{e} + e^{-\xi(t+C_2)-1} \right] \right) \right\}^2 \tag{B.22}$$

$$+ \frac{1}{\xi} \left(\frac{\langle m\mathbf{v}^2(t) \rangle}{m} \right) t + \frac{1}{2} \frac{\gamma^2}{\xi^4} \left(\frac{\langle m\mathbf{v}^2(t) \rangle}{m} \right)^2 \tag{B.23}$$

The last step involves solving for the constants C_1 and C_2 in Eq. B.23. For convenience, we define the functional inside the Lambert W function as $q(t) = \frac{1}{\xi} \left[\frac{C_1 \gamma^2}{e} + e^{-\xi(t+C_2)-1} \right]$. By inspection, $q(t)$ contains the exponential functional that we know exists in the limit of spherical Brownian motion. In the isotropic limit, the characteristic behavior of the exponential term goes to 1 as $t \to 0$. In order to enforce a similar limit for Eq. B.23, we must have $\mathcal{W}(q(t)) \to 0$. This happens when $q(t) = 0$, making it a natural point for Taylor expansion of the Lambert W function in order to simplify Eq. B.23. Performing the expansion

$$\mathcal{W}\left(q(t)\right) = \mathcal{W}\left(q(t_r)\right) + \int \frac{\partial \mathcal{W}(q(t_r))}{\partial q(t)} \left[q(t) - q(t_r)\right] dt \tag{B.24}$$

$$+ \frac{1}{2!} \int \int \frac{\partial^2 \mathcal{W}(q(t_r))}{\partial q(t) \partial q(t')} \left[q(t) - q(t_r)\right] \left[q(t') - q(t'_r)\right] dt dt' \tag{B.25}$$

$$+ \ \ldots \tag{B.26}$$

where t_r indicates the value of t where $q(t) = 0$. Noting that the derivative terms are constants, we define $\psi_1 = \mathcal{W}\left(q(t_r)\right)$, $\psi_2 = \frac{\partial \mathcal{W}(q(t_r))}{\partial q(t)}$, and $\psi_3 = \frac{\partial^2 \mathcal{W}(q(t_r))}{\partial q(t) \partial q(t')}$, giving

$$\mathcal{W}\left(q(t)\right) = \psi_1 + \psi_2 \int \left[q(t) - q(t_r)\right] dt + \frac{\psi_3}{2} \int \int \left[q(t) - q(t_r)\right] \left[q(t') - q(t'_r)\right] dt dt' + \ \ldots \tag{B.27}$$

Evaluation of the integrals in Eq. B.27 is straightforward, albeit tedious. The final result of evaluating all terms in Eq. B.27 is

$$\mathcal{W}\left(q(t)\right) \sim B_1 + B_2 e^{-\xi t} \left[e^{-\xi t} + \left(\frac{1}{\xi e} \right)^2 t + \left(\frac{\sqrt{2}}{\xi} \right)^2 \right] \tag{B.28}$$

where we have grouped the constants C_1 and C_2 into a composite set of constants B_1 and B_2 incorporating the various ψ terms that arise from expansion of the Lambert W functional.

Plugging Eq. B.28 into Eq. B.23 gives

$$\langle \mathbf{r}^2(t) \rangle = \left(4\gamma^2\right)^{-1} \left\{ B_1 + B_2 e^{-\xi t} \left[e^{-\xi t} \right.\right. \tag{B.29}$$

$$\left.\left. + \left(\frac{1}{\xi e}\right)^2 t + \left(\frac{\sqrt{2}}{\xi}\right)^2 \right] \right\}^2 \tag{B.30}$$

$$+ \frac{1}{\xi}\left(\frac{\langle m\mathbf{v}^2(t)\rangle}{m}\right) t + \frac{1}{2}\frac{\gamma^2}{\xi^4}\left(\frac{\langle m\mathbf{v}^2(t)\rangle}{m}\right)^2 \tag{B.31}$$

Using boundary conditions $\langle \mathbf{r}^2(0) \rangle = 0$ and $\partial \langle \mathbf{r}^2(0) \rangle / \partial t = 0$, the constants B_1 and B_2 is can then be evaluated as

$$B_1 = 4\left[\frac{B_3^2\gamma^4}{\xi^4}\right]^2 + B_2\left[1 + \frac{2}{\gamma^2}\right], \;\; B_2 = -\frac{1}{2}\left[\frac{\xi^7}{B_3^3\gamma^6}\right]\left[2\xi + \frac{2}{\xi} - \frac{1}{\xi^2 e^2}\right]^{-1} \text{with } B_3 = \frac{\langle mv^2(t)\rangle}{m} \tag{B.32}$$

Eq. B.31 and B.32 are the major results reported in the main text.
In order to obtain the relaxation times, we expand the first exponential in Eq. B.31 and drop terms of $O\left(t^2\right)$ and higher, leaving two terms linear in t and one exponential in t. Balancing the long-time diffusive term – B_3/ξ_p with the exponential term in t results τ_{trans}. Balancing the remaining linear t term with the exponential term gives τ_{rot}.

$$\tau_{trans} = \xi^{-1}\mathcal{W}\left(\frac{B_1 B_2^{-1} - 1 + \left(\xi^{-1} - \xi^{-2}\right)\left(e^{-2} + 2\right)}{\gamma B_2^2 B_3 \xi^{-2}}\right) \tag{B.33}$$

$$\tau_{rot} = \xi^{-1}\mathcal{W}\left(\frac{B_1 B_2^{-1} - 1 + \left(\xi^{-1} - \xi^{-2}\right)\left(e^{-2} + 2\right)}{\left(e^{-2} - 2\xi\right)\left(2\xi^{-1} - B_1 B_2^{-1}\right)\xi^{-3}}\right) \tag{B.34}$$

Since I_n changes monotonically as a function of the number of sides for regular polygons, computing the relevant timescales from Eq. B.33 – B.34 will not capture the observed crossover in relaxation ratios from simulations, as shown in Fig. B.1.

B.2.1 Frequency Correction

Solutions for the frequency extension can be obtained via an analogous set of derivation where we employ a transformation of variables for ξ to ξ_p, where $\xi_p = \xi + \eta\left(\alpha/I\right)^{-1/4}$. The functional form of the analytical solution remains identical to Eq. B.31, but now the relevant constants are:

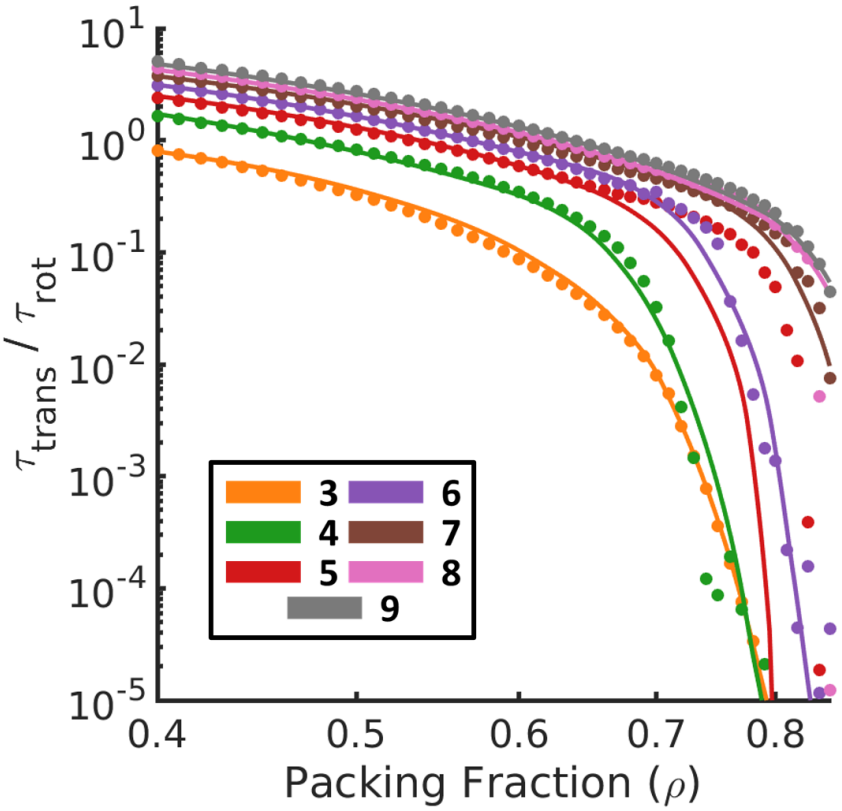


Figure B.1 | Relaxation ratio using Eq. B.33 – B.34.

$$B_1 = 4\left[\frac{B_3^2\gamma^4}{\xi_p^4}\right]^2 + B_2\left[1+\frac{2}{\gamma^2}\right], \quad B_2 = -\frac{1}{2}\left[\frac{\xi_p^7}{B_3^3\gamma^6}\right]\left[2\xi_p + \frac{2}{\xi_p} - \frac{1}{\xi_p^2 e^2}\right]^{-1} \text{ with } B_3 = \frac{\langle mv^2(t)\rangle}{m} \tag{B.35}$$

Similarly, the relaxation times are

$$\tau_{trans} = \xi_p^{-1}\mathcal{W}\left(\frac{B_1B_2^{-1} - 1 + \left(\xi_p^{-1} - \xi_p^{-2}\right)\left(e^{-2}+2\right)}{\gamma B_2^2 B_3 \xi_p^{-2}}\right) \tag{B.36}$$

$$\tau_{rot} = \xi_p^{-1}\mathcal{W}\left(\frac{B_1B_2^{-1} - 1 + \left(\xi_p^{-1} - \xi_p^{-2}\right)\left(e^{-2}+2\right)}{\left(e^{-2} - 2\xi_p\right)\left(2\xi^{-1} - B_1B_2^{-1}\right)\xi_p^{-3}}\right) \tag{B.37}$$

Bibliography

1. Alder, B. J. & Wainwright, T. E. *Phase transition for a hard sphere system* Nov. 1957. http://aip.scitation.org/doi/10.1063/1.1743957.
2. Fermi, E., Pasta, P., Ulam, S. & Tsingou, M. *Studies of the nonlinear problems* tech. rep. (Los Alamos Scientific Lab., N. Mex., 1955).
3. Levitt, M. & Warshel, A. Computer simulation of protein folding. *Nature* **253,** 694–698. ISSN: 00280836 (Feb. 1975).
4. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* **21,** 1087–1092 (1953).
5. Arkhipov, A., Freddolino, P. L. & Schulten, K. Stability and Dynamics of Virus Capsids Described by Coarse-Grained Modeling. *Structure* **14,** 1767–1777. ISSN: 0969-2126. http://www.sciencedirect.com/science/article/pii/S0969212606004023 (2006).
6. Niethammer, C. *et al.* Ls1 mardyn: The massively parallel molecular dynamics code for large systems. *Journal of Chemical Theory and Computation* **10,** 4455–4464. ISSN: 15499626 (Oct. 2014).
7. Noid, W. G. Perspective: Coarse-grained models for biomolecular systems. *Journal of Chemical Physics* **139,** 090901. ISSN: 00219606 (Sept. 2013).
8. Shi, Q., Izvekov, S. & Voth, G. A. Mixed atomistic and coarse-grained molecular dynamics: Simulation of a membrane-bound ion channel. *Journal of Physical Chemistry B* **110,** 15045–15048. ISSN: 15206106 (Aug. 2006).
9. Tozzini, V. *Coarse-grained models for proteins* Apr. 2005.
10. Chen, C. *et al.* A comparison of united atom, explicit atom, and coarse-grained simulation models for poly(ethylene oxide). *Journal of Chemical Physics* **124,** 234901. ISSN: 00219606. http://aip.scitation.org/doi/10.1063/1.2204035 (June 2006).
11. Smallenburg, F., Filion, L., Marechal, M. & Dijkstra, M. Vacancy-stabilized crystalline order in hard cubes. *Proceedings of the National Academy of Sciences of the United States of America* **109,** 17886–17890. ISSN: 00278424 (Oct. 2012).
12. Rossi, L. *et al.* Shape-sensitive crystallization in colloidal superball fluids. *Proceedings of the National Academy of Sciences of the United States of America* **112,** 5286–5290. ISSN: 10916490 (Apr. 2015).
13. Henzie, J., Grünwald, M., Widmer-Cooper, A., Geissler, P. L. & Yang, P. Self-assembly of uniform polyhedral silver nanocrystals into densest packings and exotic superlattices. *Nature Materials* **11,** 131–137. ISSN: 1476-1122 (2012).
14. Damasceno, P. F., Engel, M. & Glotzer, S. C. Predictive Self-Assembly of Polyhedra into Complex Structures. *Science* **337,** 453–457. ISSN: 0036-8075 (July 2012).

15. Damasceno, P. F., Engel, M. & Glotzer, S. C. Crystalline assemblies and densest packings of a family of truncated tetrahedra and the role of directional entropic forces. *Acs Nano* **6,** 609–614 (2012).

16. Lawrence, M. C. & Colman, P. M. Shape Complementarity at Protein/Protein Interfaces. *Journal of Molecular Biology* **234,** 946–950. ISSN: 0022-2836. http://www.sciencedirect.com/science/article/pii/S0022283683716487 (1993).

17. Chandler, D., Weeks, J. D. & Andersen, H. C. Van der waals picture of liquids, solids, and phase transformations. *Science* **220,** 787–794. ISSN: 00368075. https://science.sciencemag.org/content/220/4599/787%20https://science.sciencemag.org/content/220/4599/787.abstract (May 1983).

18. Onsager, L. The effects of shape on the interaction of colloidal particles. *Annals of the New York Academy of Sciences* **51,** 627–659. ISSN: 00778923. eprint: arXiv:1011.1669v3 (May 1949).

19. Glotzer, S. C. & Solomon, M. J. Anisotropy of building blocks and their assembly into complex structures. *Nature Materials* **6,** 557–562. ISSN: 14764660 (2007).

20. Tao, A., Sinsermsuksakul, P. & Yang, P. Polyhedral Silver Nanocrystals with Distinct Scattering Signatures. *Angewandte Chemie International Edition* **45,** 4597–4601. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.200601277. https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.200601277 (2006).

21. Agarwal, U. & Escobedo, F. A. Mesophase behaviour of polyhedral particles. *Nature Materials* **10,** 230–235. https://doi.org/10.1038/nmat2959 (2011).

22. Sarangapani, P. S., Hudson, S. D., Jones, R. L., Douglas, J. F. & Pathak, J. A. Critical Examination of the Colloidal Particle Model of Globular Proteins. *Biophysical Journal* **108,** 724–737 (Feb. 2015).

23. Zhang, Z. & Glotzer, S. C. Self-Assembly of Patchy Particles. *Nano Letters* **4.** PMID: 29048902, 1407–1413. eprint: https://doi.org/10.1021/nl0493500 (2004).

24. Fusco, D. & Charbonneau, P. Soft matter perspective on protein crystal assembly. *Colloids and Surfaces B: Biointerfaces* **137.** Biocolloids and Colloids in Biology, 22–31. ISSN: 0927-7765. http://www.sciencedirect.com/science/article/pii/S0927776515300576 (2016).

25. Fusco, D., Headd, J. J., De Simone, A., Wang, J. & Charbonneau, P. Characterizing protein crystal contacts and their role in crystallization: Rubredoxin as a case study. *Soft Matter* **10,** 290–302. ISSN: 17446848. www.rsc.org/softmatter (Jan. 2014).

26. Peltzer, R. M., Kolli, H. B., Stocker, A. & Cascella, M. Self-Assembly of α-Tocopherol Transfer Protein Nanoparticles: A Patchy Protein Model. *Journal of Physical Chemistry B* **122,** 7066–7072. ISSN: 15205207. https://pubs.acs.org/sharingguidelines (July 2018).

27. Simon, A. J. *et al.* Supercharging enables organized assembly of synthetic biomolecules. *Nature Chemistry* **11,** 204–212. ISSN: 17554349 (Mar. 2019).

28. Stradner, A. & Schurtenberger, P. Potential and limits of a colloid approach to protein solutions. *Soft Matter* **16,** 307–323. http://dx.doi.org/10.1039/C9SM01953G (2 2020).

29. Durumeric, A. E. P. & Voth, G. A. Adversarial-residual-coarse-graining: Applying machine learning theory to systematic molecular coarse-graining. *The Journal of Chemical Physics* **151,** 124110. eprint: https://doi.org/10.1063/1.5097559 (2019).

30. Grigo, C. & Koutsourelakis, P.-S. A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the Small Data regime. *Journal of Computational Physics* **397,** 108842. ISSN: 0021-9991 (2019).

31. Wang, J. *et al.* Machine Learning of Coarse-Grained Molecular Dynamics Force Fields. *ACS Central Science* **5,** 755–767. eprint: https://doi.org/10.1021/acscentsci.8b00913 (2019).

32. Anderson, J. A., Glaser, J. & Glotzer, S. C. HOOMD-blue: A Python package for high-performance molecular dynamics and hard particle Monte Carlo simulations. *Computational Materials Science* **173,** 109363. ISSN: 09270256 (Feb. 2020).

33. Ramasubramani, V. *et al.* freud: A software suite for high throughput analysis of particle simulation data. *Computer Physics Communications* **254,** 107275. ISSN: 00104655 (Sept. 2020).

34. Ramasubramani, V., Vo, T., Anderson, J. A. & Glotzer, S. C. A mean-field approach to simulating anisotropic particles. *The Journal of Chemical Physics* **153,** 084106 (Aug. 2020).

35. Adorf*, C. S., Ramasubramani*, V., Anderson, J. A. & Glotzer, S. C. How to Professionally Develop Reusable Scientific Software—And When Not To. *Computing in Science Engineering* **21.** (*these authors contributed equally to this work), 66–79. ISSN: 1521-9615 (Mar. 2019).

36. Gong*, Z. *et al.* Crystallization of colloidal truncated tetrahedra driven by entropic forces. *In Preparation.* (*these authors contributed equally to this work).

37. Frenkel, D. Order through entropy. *Nature Materials* **14,** 9 (2014).

38. Sciortino, F. Entropy in self-assembly. *La rivista del nuovo cimento* **42,** 511–548 (2019).

39. Asakura, S. & Oosawa, F. On interaction between two bodies immersed in a solution of macromolecules. *The Journal of Chemical Physics* **22,** 1255–1256 (1954).

40. Sacanna, S. & Pine, D. J. Shape-anisotropic colloids: Building blocks for complex assemblies. *Current Opinion in Colloid & Interface Science* **16,** 96–105. ISSN: 1359-0294 (2011).

41. Lee, K. J., Yoon, J. & Lahann, J. Recent advances with anisotropic particles. *Current Opinion in Colloid & Interface Science* **16,** 195–202. ISSN: 1359-0294 (2011).

42. Liu, B., Wu, Y. & Zhao, S. Anisotropic Colloids: From Non-Templated to Patchy Templated Synthesis. *Chemistry – A European Journal* **24,** 10562–10570 (2018).

43. Miszta, K. *et al.* Hierarchical self-assembly of suspended branched colloidal nanocrystals into superlattice structures. *Nature Materials* **10,** 872–876. ISSN: 1476-4660 (Nov. 2011).

44. Gong, J. *et al.* Shape-dependent ordering of gold nanocrystals into large-scale superlattices. *Nature Communications* **8.** ISSN: 20411723 (Jan. 2017).

45. Nagaoka, Y. *et al.* Superstructures generated from truncated tetrahedral quantum dots. *Nature* **561,** 378–382 (2018).

46. Yethiraj, A. & van Blaaderen, A. A colloidal model system with an interaction tunable from hard sphere to soft and dipolar. *Nature* **421,** 513–517. ISSN: 1476-4687 (Jan. 2003).

47. Meijer, J.-M. *et al.* Observation of solid–solid transitions in 3D crystals of colloidal superballs. *Nature Communications* **8,** 14352 (2017).

48. Wood, W. W. & Jacobson, J. Preliminary results from a recalculation of the Monte Carlo equation of state of hard spheres. *The Journal of Chemical Physics* **27,** 1207–1208 (1957).

49. Pusey, P. N. & Van Megen, W. Phase behaviour of concentrated suspensions of nearly hard colloidal spheres. *Nature* **320,** 340 (1986).

50. Engel, M. *et al.* Hard-disk equation of state: First-order liquid-hexatic transition in two dimensions with three simulation methods. *Physical Review E* **87,** 042134 (2013).

51. Thorneywork, A. L., Abbott, J. L., Aarts, D. G. & Dullens, R. P. Two-dimensional melting of colloidal hard spheres. *Physical Review Letters* **118,** 158001 (2017).

52. Vrij, A. Polymers at interfaces and the interactions in colloidal dispersions. *Pure and Applied Chemistry* **48,** 471–483 (1976).

53. Damasceno, P. F., Engel, M. & Glotzer, S. C. Predictive self-assembly of polyhedra into complex structures. *Science* **337,** 453–457. ISSN: 10959203 (July 2012).

54. Karas, A. S., Glaser, J. & Glotzer, S. C. Using depletion to control colloidal crystal assemblies of hard cuboctahedra. *Soft Matter* **12,** 5199–5204. ISSN: 17446848 (June 2016).

55. Damasceno, P. F., Karas, A. S., Schultz, B. A., Engel, M. & Glotzer, S. C. Controlling Chirality of Entropic Crystals. *Phys. Rev. Lett.* **115,** 158303 (15 Oct. 2015).

56. Van Anders, G., Klotsa, D., Ahmed, N. K., Engel, M. & Glotzer, S. C. Understanding shape entropy through local dense packing. *Proceedings of the National Academy of Sciences of the United States of America* **111,** E4812–21. ISSN: 1091-6490. arXiv: 1309.1187. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4234574%7B%5C&%7Dtool=pmcentrez%7B%5C&%7Drendertype=abstract (2014).

57. Gong, Z., Hueckel, T., Yi, G.-R. & Sacanna, S. Patchy particles made by colloidal fusion. *Nature* **550,** 234 (2017).

58. Escobedo, F. A. Engineering entropy in soft matter: The bad, the ugly and the good. *Soft Matter* **10,** 8388–8400 (2014).

59. Wang, Y., Jenkins, I. C., McGinley, J. T., Sinno, T. & Crocker, J. C. Colloidal crystals with diamond symmetry at optical lengthscales. *Nature Communications* **8,** 14173 (2017).

60. Ducrot, É., He, M., Yi, G.-R. & Pine, D. J. Colloidal alloys with preassembled clusters and spheres. *Nature materials* **16,** 652 (2017).

61. He, M. *et al.* Colloidal diamond. *Nature* **585,** 524–529 (Sept. 2020).

62. Schade, N. B. *et al.* Tetrahedral colloidal clusters from random parking of bidisperse spheres. *Physical Review Letters* **110,** 148303 (2013).

63. Stukowski, A. Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool. *Modelling and Simulation in Materials Science and Engineering* **18,** 015012. ISSN: 09650393 (2010).

64. Maras, E., Trushin, O., Stukowski, A., Ala-Nissila, T. & Jónsson, H. Global transition path search for dislocation formation in Ge on Si(001). *Computer Physics Communications* **205,** 13–21 (Aug. 2016).

65. Cromwell, P. R. *Polyhedra* (Cambridge University Press, 1999).

66. Sainis, S. K., Germain, V., Mejean, C. O. & Dufresne, E. R. Electrostatic interactions of colloidal particles in nonpolar solvents: Role of surface chemistry and charge control agents. *Langmuir* **24,** 1160–1164 (2008).

67. Hodeau, J. L. *et al.* High-pressure transformations of C_{60} to diamond and sp^3 phases at room temperature and to sp^2 phases at high temperature. *Phys. Rev. B* **50,** 10311–10314 (14 Oct. 1994).

68. Romano, F. & Sciortino, F. Patterning symmetry in the rational design of colloidal crystals. *Nature Communications* **3** (Jan. 2012).

69. Zygmunt, W., Teich, E. G., van Anders, G. & Glotzer, S. C. Topological order in densely packed anisotropic colloids. *Phys. Rev. E* **100,** 032608. https://link.aps.org/doi/10.1103/PhysRevE.100.032608 (3 Sept. 2019).

70. Glaser, J., Karas, A. S. & Glotzer, S. C. A parallel algorithm for implicit depletant simulations. *The Journal of Chemical Physics* **143,** 184110 (Nov. 2015).

71. Lekkerkerker, H. N. W. & Tuinier, R. *Colloids and the Depletion Interaction* (Springer Netherlands, Dordrecht, 2011).

72. Dijkstra, M. Phase diagram of highly asymmetric binary hard-sphere mixtures. *Physical Review E* **59,** 5744–5771 (1999).

73. Cersonsky, R. K. Pressure-tunable photonic band gaps in an entropic colloidal crystal. *Physical Review Materials* **2** (2018).

74. Bolhuis, P. G. Influence of Polymer-Excluded Volume on the Phase-Behavior of Colloid-Polymer Mixtures. *Physical Review Letters* **89** (2002).

75. Zhang, Y., Lu, F., van der Lelie, D. & Gang, O. Continuous Phase Transformation in Nanocube Assemblies. *Phys. Rev. Lett.* **107,** 135701 (13 Sept. 2011).

76. Teich, E. G., van Anders, G. & Glotzer, S. C. Identity crisis in alchemical space drives the entropic colloidal glass transition. *Nature Communications* **10,** 64. ISSN: 2041-1723 (Dec. 2019).

77. Anderson, J. A., Lorenz, C. D. & Travesset, A. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics* **227,** 5342–5359. ISSN: 0021-9991 (May 2008).

78. Glaser, J. *et al.* Strong scaling of general-purpose molecular dynamics simulations on GPUs. *Computer Physics Communications* **192,** 97–107. ISSN: 00104655. http://arxiv.org/abs/1412.3387%20http://linkinghub.elsevier.com/retrieve/pii/S0010465515000867%20https://www.sciencedirect.com/science/article/pii/S0010465515000867 (July 2015).

79. Anderson, J. A., Eric Irrgang, M. & Glotzer, S. C. Scalable Metropolis Monte Carlo for simulation of hard shapes. *Computer Physics Communications* **204,** 21–30. ISSN: 00104655 (July 2016).

80. Van Anders, G., Ahmed, N. K., Smith, R., Engel, M. & Glotzer, S. C. Entropically patchy particles: Engineering valence through shape entropy. *ACS Nano* **8,** 931–940. ISSN: 19360851. arXiv: 1304.7545 (2014).

81. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585,** 357–362 (Sept. 2020).

82. Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* **17,** 261–272 (Feb. 2020).

83. McKinney, W. *Data Structures for Statistical Computing in Python* in *Proceedings of the 9th Python in Science Conference* (eds van der Walt, S. & Millman, J.) (2010), 56–61. https://conference.scipy.org/proceedings/scipy2010/mckinney.html.

84. Ramasubramani, V. & C. Glotzer, S. rowan: A Python package for working with quaternions. *Journal of Open Source Software* **3,** 787. https://doi.org/10.21105/joss.00787 (2018).

85. Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering* **9,** 99–104. ISSN: 15219615 (May 2007).

86. Adorf, C. S., Dodd, P. M., Ramasubramani, V. & Glotzer, S. C. Simple data and workflow management with the signac framework. *Computational Materials Science* **146,** 220–229. ISSN: 09270256. https://www.sciencedirect.com/science/article/pii/S0927025618300429 (Apr. 2018).

87. Ramasubramani, V., Adorf, C. S., Dodd, P. M., Dice, B. D. & Glotzer, S. C. *signac: A Python framework for data and workflow management* in *Proceedings of the 17th Python in Science Conference* (eds Akici, F., Lippa, D., Niederhut, D. & Pacer, M.) (2018), 152–159.

88. Frenkel, D. & Ladd, A. J. C. New Monte Carlo method to compute the free energy of arbitrary solids. Application to the fcc and hcp phases of hard spheres. *The Journal of Chemical Physics* **81,** 3188–3193 (Oct. 1984).

89. Frenkel, D. & Smit, B. *Understanding molecular simulation: from algorithms to applications* ISBN: 0-12-267351-4 (2002).

90. Haji-Akbari, A., Engel, M. & Glotzer, S. C. Phase diagram of hard tetrahedra. *The Journal of Chemical Physics* **135,** 194101 (2011).

91. Singh, D. B. & Tripathi, T. *Frontiers in protein structure, function, and dynamics* 2020.

92. Goodsell, D. S. & Olson, A. J. Structural symmetry and protein function. *Annual Review of Biophysics and Biomolecular Structure* **29,** 105–153. ISSN: 10568700 (2000).

93. André, I., Strauss, C. E. M., Kaplan, D. B., Bradley, P. & Baker, D. *Emergence of symmetry in homooligomeric biological assemblies* in *Proceedings of the National Academy of Sciences of the United States of America* **105** (2008), 16148–16152.

94. Senior, A. W. *et al.* Improved protein structure prediction using potentials from deep learning. *Nature* **577,** 706–710 (2020).

95. Kmiecik, S. *et al.* Coarse-Grained Protein Models and Their Applications. *Chemical Reviews* **116,** 7898–7936. ISSN: 15206890 (July 2016).

96. Krissinel, E. & Henrick, K. Inference of Macromolecular Assemblies from Crystalline State. *Journal of Molecular Biology* **372,** 774–797. ISSN: 0022-2836 (2007).

97. Robertson, M. J., Tirado-Rives, J. & Jorgensen, W. L. Improved Peptide and Protein Torsional Energetics with the OPLS-AA Force Field. *Journal of Chemical Theory and Computation* **11,** 3499–3509 (June 2015).

98. Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A* **32,** 922–923 (1976).

99. Sanner, M. F., Olson, A. J. & Spehner, J.-C. Reduced surface: An efficient way to compute molecular surfaces. *Biopolymers* **38,** 305–320 (1996).

100. Sinkovits, D. W., Barr, S. a. & Luijten, E. Rejection-free Monte Carlo scheme for anisotropic particles. *The Journal of Chemical Physics* **136,** 144111. ISSN: 1089-7690. http://www.ncbi.nlm.nih.gov/pubmed/22502505 (Apr. 2012).

101. Monticelli, L. *et al.* The MARTINI Coarse-Grained Force Field: Extension to Proteins. *Journal of Chemical Theory and Computation* **4,** 819–834 (Apr. 2008).

102. Wukovitz, S. W. & Yeates, T. O. Why protein crystals favour some space-groups over others. *Nature Structural & Molecular Biology* **2,** 1062–1067 (Dec. 1995).

103. Geng, Y., van Anders, G., Dodd, P. M., Dshemuchadse, J. & Glotzer, S. C. Engineering entropy for the inverse design of colloidal crystals from hard shapes. *Science Advances* **5** (2019).

104. Friedman, J., Hastie, T. & Tibshirani, R. *The elements of statistical learning* **10** (Springer series in statistics New York, 2001).

105. Gainza, P. *et al.* Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*. https://doi.org/10.1038%2Fs41592-019-0666-6 (Dec. 2019).

106. Harris, R. C. Comparing the Poisson-Boltzmann Equation to Alternative Electrostatic Theories and Improving Stochastic Techniques for Implicit Solvent Models (2012).

107. Sharp, K. A. & Honig, B. Calculating total electrostatic energies with the nonlinear Poisson-Boltzmann equation. *Journal of Physical Chemistry* **94,** 7684–7692 (1990).

108. Baptista, M., Schmitz, R. & Dünweg, B. Simple and robust solver for the Poisson-Boltzmann equation. *Phys. Rev. E* **80,** 016705 (1 July 2009).

109. Grycuk, T. Deficiency of the Coulomb-field approximation in the generalized Born model: An improved formula for Born radii evaluation. *The Journal of Chemical Physics* **119,** 4817–4826 (2003).

110. Cooper, C. D., Clementi, N. C., Forsyth, G. & Barba, L. A. PyGBe: Python, GPUs and Boundary elements for biomolecular electrostatics. *Journal of Open Source Software* **1,** 43. https://doi.org/10.21105/joss.00043 (2016).

111. Kyte, J. & Doolittle, R. F. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology* **157,** 105–132. ISSN: 0022-2836 (1982).

112. Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* 2015.

113. Sergeev, A. & Balso, M. D. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799* (2018).

114. Allen, M. P. & Tildesley, D. J. *Computer simulation of liquids* 385. ISBN: 9780198556459 (Clarendon Press, 1987).

115. Freddolino, P. L., Liu, F., Gruebele, M. & Schulten, K. Ten-microsecond molecular dynamics simulation of a fast-folding WW domain. *Biophysical Journal* **94,** L75–L77. ISSN: 15420086 (May 2008).

116. Shaw, D. E. *et al. Millisecond-scale molecular dynamics simulations on Anton* in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis - SC '09* (ACM Press, New York, New York, USA, 2009), 1.

117. Warshel, A. Bicycle-pedal model for the first step in the vision process. *Nature* **260,** 679–683. ISSN: 00280836 (1976).

118. Karplus, M. & McCammon, J. A. Molecular dynamics simulations of biomolecules. *Nature Structural Biology* **9,** 646–652. ISSN: 10728368 (2002).

119. Wei, Y. *et al.* The nature of strength enhancement and weakening by pentagon-heptagon defects in graphene. *Nature Materials* **11,** 759–763. ISSN: 14764660 (July 2012).

120. Hadley, K. R. & McCabe, C. Coarse-grained molecular models of water: a review. *Molecular Simulation* **38,** 671–681. ISSN: 0892-7022. http://www.tandfonline.com/doi/abs/10.1080/08927022.2012.671942 (July 2012).

121. Zimm, B. H. Dynamics of polymer molecules in dilute solution: Viscoelasticity, flow birefringence and dielectric loss. *The Journal of Chemical Physics* **24,** 269–278. ISSN: 00219606. http://aip.scitation.org/doi/10.1063/1.1742462 (Feb. 1956).

122. Rouse, P. E. A theory of the linear viscoelastic properties of dilute solutions of coiling polymers. *The Journal of Chemical Physics* **21,** 1272–1280. ISSN: 00219606. http://aip.scitation.org/doi/10.1063/1.1699180 (July 1953).

123. Ingólfsson, H. I. *et al.* The power of coarse graining in biomolecular simulations. *Wiley Interdisciplinary Reviews: Computational Molecular Science* **4,** 225–248. ISSN: 17590884 (2014).

124. Allen, M. P. & Germano, G. Expressions for forces and torques in molecular simulations using rigid bodies. *Molecular Physics* **104,** 3225–3235. ISSN: 00268976 (Oct. 2006).

125. Horsch, M. A., Zhang, Z. & Glotzer, S. C. Self-assembly of end-tethered nanorods in a neat system and role of block fractions and aspect ratio. *Soft Matter* **6,** 945–954. ISSN: 1744683X (Feb. 2010).

126. Heine, D. R., Petersen, M. K. & Grest, G. S. Effect of particle shape and charge on bulk rheology of nanoparticle suspensions. *Journal of Chemical Physics* **132,** 184509. ISSN: 00219606. arXiv: 1004.2411. http://aip.scitation.org/doi/10.1063/1.3419071 (May 2010).

127. Nguyen, T. D., Phillips, C. L., Anderson, J. A. & Glotzer, S. C. Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units. *Computer Physics Communications* **182,** 2307–2313. ISSN: 00104655 (Nov. 2011).

128. Gay, J. G. & Berne, B. J. Modification of the overlap potential to mimic a linear site-site potential. *The Journal of Chemical Physics* **74,** 3316–3319. ISSN: 00219606. http://aip.scitation.org/doi/10.1063/1.441483 (Mar. 1981).

129. Berardi, R., Fava, C. & Zannoni, C. A Gay-Berne potential for dissimilar biaxial particles. *Chemical Physics Letters* **297,** 8–14. ISSN: 00092614 (Nov. 1998).

130. Cleaver, D. J., Care, C. M., Allen, M. P. & Neal, M. P. Extension and generalization of the gay-berne potential. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* **54,** 559–567. ISSN: 1063651X (July 1996).

131. Lubachevsky, B. D. How to simulate billiards and similar systems. *Journal of Computational Physics* **94,** 255–283. ISSN: 10902716 (June 1991).

132. Smith, S. W., Hall, C. K. & Freeman, B. D. Molecular dynamics for polymeric fluids using discontinuous potentials. *Journal of Computational Physics* **134,** 16–30. ISSN: 00219991 (June 1997).

133. Alder, B. J. & Wainwright, T. E. Studies in molecular dynamics. I. General method. *The Journal of Chemical Physics* **31,** 459–466. ISSN: 00219606. http://aip.scitation.org/doi/10.1063/1.1730376 (Aug. 1959).

134. Nguyen, H. D. & Hall, C. K. Molecular dynamics simulations of spontaneous fibril formation by random-coil peptides. *Proceedings of the National Academy of Sciences* **101,** 16180–16185. ISSN: 00278424 (Nov. 2004).

135. Cundall, P. A. & Strack, O. D. L. A discrete numerical model for granular assemblies. *Géotechnique* **29,** 47–65. ISSN: 0016-8505. http://www.icevirtuallibrary.com/doi/10.1680/geot.1979.29.1.47 (Mar. 1979).

136. Spellings, M., Marson, R. L., Anderson, J. A. & Glotzer, S. C. GPU accelerated Discrete Element Method (DEM) molecular dynamics for conservative, faceted particle simulations. *Journal of Computational Physics* **334,** 460–467. ISSN: 10902716 (Apr. 2017).

137. Oliphant, T. E. *A guide to NumPy* (Trelgol Publishing, 2006).

138. Vo, T. & Glotzer, S. C. Principle of corresponding states for hard polyhedron fluids. *Molecular Physics* **117,** 3518–3526. ISSN: 13623028 (Dec. 2019).

139. Lu, F. *et al.* Unusual packing of soft-shelled nanocubes. *Science Advances* **5,** eaaw2399. ISSN: 23752548 (May 2019).

140. Den Bergen, G. V. A Fast and Robust GJK Implementation for Collision Detection of Convex Objects. *Journal of Graphics Tools* **4,** 7–25. ISSN: 1086-7651 (Jan. 1999).

141. Van den Bergen, G. J. A. *Collison detection in interactive 3D environments* 277. ISBN: 9780080494234 (Elsevier, 2004).

142. Gilbert, E. G., Johnson, D. W. & Keerthi, S. S. A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space. *IEEE Journal on Robotics and Automation* **4,** 193–203. ISSN: 08824967 (1988).

143. Gilbert, E. G. & Foo, C. P. *Computing the Distance Between General Convex Objects in Three-Dimensional Space* 1990.

144. Montanari, M., Petrinic, N. & Barbieri, E. Improving the GJK Algorithm for Faster and More Reliable Distance Queries Between Convex Objects. *ACM Transactions on Graphics* **36,** 1. ISSN: 07300301 (June 2017).

145. Larsen, P. M., Schmidt, S. & Schiøtz, J. Robust structural identification via polyhedral template matching. *Modelling and Simulation in Materials Science and Engineering* **24,** 055007. ISSN: 1361651X (May 2016).

146. Gottschalk, S., Lin, M. C. & Manocha, D. *OBB tree: A hierarchical structure for rapid interference detection* in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996* (Association for Computing Machinery, Inc, Aug. 1996), 171–180. ISBN: 0897917464.

147. De Michele, C. Simulating hard rigid bodies. *Journal of Computational Physics* **229,** 3276–3294. ISSN: 10902716. http://arxiv.org/abs/0903.1608%20http://dx.doi.org/10.1016/j.jcp.2010.01.002 (May 2010).

148. Martyna, G. J., Tobias, D. J. & Klein, M. L. Constant pressure molecular dynamics algorithms. *The Journal of Chemical Physics* **101,** 4177–4189. ISSN: 00219606 (1994).

149. Ramasubramani, V., Vo, T., Anderson, J. A. & Glotzer, S. C. Brownian dynamics of anisotropic particles. *In Preparation.*

150. Einstein, A. Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der Physik* **322,** 549–560. ISSN: 00033804. http://doi.wiley.com/10.1002/andp.19053220806 (1905).

151. Von Smoluchowski, M. Zur kinetischen Theorie der Brownschen Molekularbewegung und der Suspensionen. *Annalen der Physik* **326,** 756–780. ISSN: 00033804. http://doi.wiley.com/10.1002/andp.19063261405 (1906).

152. Langevin, P. Sur la théorie du mouvement brownien. *C. R. Acad. Sci. (Paris)* **146,** 530–533 (1908).

153. Uhlenbeck, G. E. & Ornstein, L. S. On the Theory of the Brownian Motion. *Physical Review* **36,** 823–841. ISSN: 0031-899X. https://link.aps.org/doi/10.1103/PhysRev.36.823 (Sept. 1930).

154. Einstein, A. Zur Theorie der Brownschen Bewegung. *Annalen der Physik* **324,** 371–381. ISSN: 15213889. https://onlinelibrary.wiley.com/doi/full/10.1002/andp.19063240208%20https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19063240208%20https://onlinelibrary.wiley.com/doi/10.1002/andp.19063240208 (Jan. 1906).

155. Perrin, F. Mouvement brownien d'un ellipsoide-I. Dispersion diélectrique pour des molécules ellipsoidales. *J. Phys. Radium* **5,** pp. http://dx.doi.org/10.1051/jphysrad:01934005010049700 (1934).

156. Perrin, F. Mouvement Brownien d'un ellipsoide (II). Rotation libre et dépolarisation des fluorescences. Translation et diffusion de molécules ellipsoidales. *J. Phys. Radium* **7,** 1–11. https://hal.archives-ouvertes.fr/jpa-00233379 (1936).

157. Brenner, H. Coupling between the translational and rotational brownian motions of rigid particles of arbitrary shape I. Helicoidally isotropic particles. *Journal of Colloid Science* **20,** 104–122. ISSN: 00958522 (1965).

158. Brenner, H. Coupling between the translational and rotational brownian motions of rigid particles of arbitrary shape. II. General theory. *Journal of Colloid And Interface Science* **23,** 407–436. ISSN: 00219797 (1967).

159. Wegener, W. A. Diffusion coefficients for rigid macromolecules with irregular shapes that allow rotational-translational coupling. *Biopolymers* **20,** 303–326. ISSN: 0006-3525. http://doi.wiley.com/10.1002/bip.1981.360200205 (Feb. 1981).

160. Harvey, S. & Garcia de la Torre, J. Coordinate Systems for Modeling the Hydrodynamic Resistance and Diffusion Coefficients of Irregularly Shaped Rigid Macromolecules. *Macromolecules* **13,** 960–964. ISSN: 0024-9297. https://pubs.acs.org/doi/abs/10.1021/ma60076a037 (July 1980).

161. Dickinson, E., Allison, S. A. & McCammon, J. A. Brownian dynamics with rotation-translation coupling. *Journal of the Chemical Society, Faraday Transactions 2: Molecular and Chemical Physics* **81,** 591–601. ISSN: 03009238 (1985).

162. Kholodenko, A. L. & Douglas, J. F. Generalized Stokes-Einstein equation for spherical particle suspensions. *Physical Review E* **51,** 1081–1090. ISSN: 1063651X (1995).

163. Han, Y. *et al.* Brownian a of an ellipsoid. *Science* **314,** 626–630. ISSN: 00368075 (Oct. 2006).

164. Wojciechoski, K. W. & Frenkel, D. Tetratic Phase in the Planar Hard Square System? http://www.cmst.eu/articles/tetratic-phase-in-the-planar-hard-square-system (2004).

165. Walsh, L. & Menon, N. Ordering and dynamics of vibrated hard squares. *Journal of Statistical Mechanics: Theory and Experiment* **2016,** 083302. https://doi.org/10.1088%2F1742-5468%2F2016%2F08%2F083302 (Aug. 2016).

166. Gantapara, A. P., Qi, W. & Dijkstra, M. A novel chiral phase of achiral hard triangles and an entropy-driven demixing of enantiomers. *Soft Matter* **11,** 8684–8691. https://doi.org/10.1039%2Fc5sm01762a (2015).

167. Anderson, J. A., Antonaglia, J., Millan, J. A., Engel, M. & Glotzer, S. C. *Shape and symmetry determine two-dimensional melting transitions of hard regular polygons* Apr. 2017. arXiv: 1606.00687.

168. Wilson, G. *et al.* Best Practices for Scientific Computing. *PLoS Biology* **12.** ISSN: 1545-7885 (Electronic)\r1544-9173 (Linking) (2014).

169. Fowler, M., Beck, K., Brant, J., Opdyke, W. & Roberts, D. *Refactoring: improving the design of existing code* (Addison-Wesley Professional, 1999).

170. Fowler, M. & Highsmith, J. *Manifesto for Agile Software Development* http://agilemanifesto.org/ (2019).

171. Oliphant, T. E. Python for Scientific Computing. *Computing in Science & Engineering* **9,** 10–20. http://ieeexplore.ieee.org/document/4160250/ (2007).

172. Towns, J. *et al.* XSEDE: Accelerating Scientific Discovery. *Computing in Science & Engineering* **16,** 62–74 (2014).

173. Martin, R. *The Clean Architecture* 2012. https://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html.

174. Anderson, J. A., Jankowski, E., Grubb, T. L., Engel, M. & Glotzer, S. C. Massively parallel monte carlo for many-particle simulations on GPUs. *Journal of Computational Physics* **254,** 27–38. ISSN: 10902716. arXiv: 1211.1646 (Dec. 2013).

175. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* **117,** 1–19. ISSN: 0021-9991 (Mar. 1995).

176. Berendsen, H., van der Spoel, D. & van Drunen, R. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications* **91,** 43–56. ISSN: 0010-4655 (Sept. 1995).

177. Humphrey, W., Dalke, A. & Schulten, K. {VMD} – {V}isual {M}olecular {D}ynamics. *Journal of Molecular Graphics* **14,** 33–38 (1996).

178. Foster, I. Globus online: Accelerating and democratizing science through cloud-based services. *IEEE Internet Computing* **15,** 70–73. ISSN: 10897801 (2011).

179. McGibbon, R. T. *et al.* MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophysical Journal* **109,** 1528–1532. ISSN: 0006-3495 (Oct. 2015).

180. Michaud-Agrawal, N., Denning, E. J., Woolf, T. B. & Beckstein, O. MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. *Journal of Computational Chemistry* **32,** 2319–2327. ISSN: 01928651 (July 2011).

181. Romo, T. & Grossfield, A. *LOOS: An extensible platform for the structural analysis of simulations* in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (IEEE, Sept. 2009), 2332–2335.

182. Hinsen, K. The Molecular Modeling Toolkit: A New Approach to Molecular Simulations. *Journal of Computational Chemistry* **21,** 79–85. ISSN: 01928651 (2000).

183. Kabsch, W. & Sander, C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* **22,** 2577–2637. ISSN: 10970282 (Dec. 1983).

184. Reinhart, W. F. & Panagiotopoulos, A. Z. Crystal growth kinetics of triblock Janus colloids. *Journal of Chemical Physics* **148,** 124506. ISSN: 00219606 (Mar. 2018).

185. Howard, M. P. *et al.* Evaporation-induced assembly of colloidal crystals. *Journal of Chemical Physics* **149,** 094901. ISSN: 00219606 (Sept. 2018).

186. Spellings, M. & Glotzer, S. C. Machine learning for crystal identification and discovery. *AIChE Journal* **64,** 2198–2206. ISSN: 00011541 (June 2018).

187. Adorf, C. S., Antonaglia, J., Dshemuchadse, J. & Glotzer, S. C. Inverse design of simple pair potentials for the self-assembly of complex structures. *Journal of Chemical Physics* **149,** 204102. ISSN: 00219606 (Nov. 2018).

188. Vansaders, B., Dshemuchadse, J. & Glotzer, S. C. Strain fields in repulsive colloidal crystals. *Physical Review Materials* **2,** 063604. ISSN: 24759953 (June 2018).

189. Antonaglia, J. A., van Anders, G. & Glotzer, S. C. Mapping Disorder in Entropically Ordered Crystals. *arXiv preprint arXiv:1803.05936* (Mar. 2018).

190. Du, C. X., van Anders, G., Newman, R. S. & Glotzer, S. C. Shape Driven Solid–Solid Transitions in Colloids. *Proceedings of the National Academy of Sciences of the United States of America* **114,** E3892–E3899. ISSN: 1091-6490 (May 2016).

191. Harper, E. S., Marson, R. L., Anderson, J. A., Van Anders, G. & Glotzer, S. C. Shape allophiles improve entropic assembly. *Soft Matter* **11,** 7250–7256. ISSN: 17446848 (Sept. 2015).

192. Dice, B. *et al. Analyzing Particle Systems for Machine Learning and Data Visualization with freud* in *Proceedings of the 18th Python in Science Conference* (2019), 27–33.

193. Roe, D. R. & Cheatham, T. E. PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data. *Journal of Chemical Theory and Computation* **9,** 3084–3095. ISSN: 1549-9618 (July 2013).

194. Case, D. A. *et al.* The Amber biomolecular simulation programs. *Journal of Computational Chemistry* **26,** 1668–1688. ISSN: 0192-8651 (Dec. 2005).

195. Schrödinger, L. *The PyMOL Molecular Graphics System, Version 2.3* Nov. 2019.

196. Yesylevskyy, S. O. Pteros: Fast and easy to use open-source C++ library for molecular analysis. *Journal of Computational Chemistry* **33,** 1632–1636. ISSN: 01928651 (July 2012).

197. Lab, G. *GSD v2.0.0* 2020. https://github.com/glotzerlab/gsd.

198. Lab, G. *garnett v0.6.1* 2020. https://github.com/glotzerlab/garnett.

199. Behnel, S. *et al.* Cython: The Best of Both Worlds. *Computing in Science & Engineering* **13,** 31–39. ISSN: 1521-9615 (Mar. 2011).

200. Calandrini, V., Pellegrini, E., Calligari, P., Hinsen, K. & Kneller, G. nMoldyn - Interfacing spectroscopic experiments, molecular dynamics simulations and models for time correlation functions. *École thématique de la Société Française de la Neutronique* **12,** 201–232. ISSN: 2107-7223 (June 2011).

201. Jones, E., Oliphant, T., Peterson, P., *et al. SciPy: Open source scientific tools for Python* 2001. https://www.scipy.org/.

202. Intel. *Intel Threading Building Blocks* 2020. https://github.com/intel/tbb.

203. *Anaconda Software Distribution* 2020. https://anaconda.com.

204. Glotzer Lab. *freud Source Code Repository* Ann Arbor, MI, 2020. https://github.com/glotzerlab/freud.

205. Rycroft, C. *Voro++: a three-dimensional Voronoi cell library in C++* tech. rep. (Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, Jan. 2009).

206. Lazar, E. A., Han, J. & Srolovitz, D. J. Topological framework for local structure analysis in condensed matter. *Proceedings of the National Academy of Sciences of the United States of America* **112,** E5769–E5776. ISSN: 10916490 (Oct. 2015).

207. Glotzer Lab. *fresnel v0.11.0* 2020. https://github.com/glotzerlab/fresnel.

208. Steinhardt, P. J., Nelson, D. R. & Ronchetti, M. Bond-orientational order in liquids and glasses. *Physical Review B* **28,** 784–805 (1983).

209. Haji-Akbari, A. & Glotzer, S. C. Strong orientational coordinates and orientational order parameters for symmetric objects. *Journal of Physics A: Mathematical and Theoretical* **48,** 485201. ISSN: 17518121 (2015).

210. Rein ten Wolde, P., Ruiz-Montero, M. J. & Frenkel, D. Numerical calculation of the rate of crystal nucleation in a Lennard-Jones system at moderate undercooling. *The Journal of Chemical Physics* **104,** 9932–9947. ISSN: 0021-9606 (June 1996).

211. Ramachandran, P. & Varoquaux, G. Mayavi: 3D Visualization of Scientific Data. *Computing in Science & Engineering* **13,** 40–51 (Mar. 2011).

212. Harper, E. S., van Anders, G. & Glotzer, S. C. The entropic bond in colloidal crystals. *Proceedings of the National Academy of Sciences of the United States of America* **116,** 16703–16710. ISSN: 10916490 (Aug. 2019).

213. Dzugutov, M. Formation of a dodecagonal quasicrystalline phase in a simple monatomic liquid. *Physical Review Letters* **70,** 2924–2927. ISSN: 00319007 (May 1993).

214. Roth, J. W., Schilling, R. & Trebin, H. R. Nucleation of quasicrystals by rapid cooling of a binary melt: A molecular-dynamics study. *Physical Review B* **51,** 15833–15840. ISSN: 01631829 (June 1995).

215. Roth, J. & Denton, A. R. Solid-phase structures of the Dzugutov pair potential. *Physical Review E* **61,** 6845–6857. ISSN: 1063-651X (June 2000).

216. Engel, M., Damasceno, P. F., Phillips, C. L. & Glotzer, S. C. Computational self-assembly of a one-component icosahedral quasicrystal. *Nature Materials* **14,** 109–16. ISSN: 1476-1122 (Jan. 2015).

217. Keys, A. S., Iacovella, C. R. & Glotzer, S. C. Characterizing complex particle morphologies through shape matching: Descriptors, applications, and algorithms. *Journal of Computational Physics* **230,** 6438–6463. ISSN: 0021-9991 (July 2011).

218. Karas, A. S., Dshemuchadse, J., van Anders, G. & Glotzer, S. C. Phase behavior and design rules for plastic colloidal crystals of hard polyhedra via consideration of directional entropic forces. *Soft Matter.* ISSN: 1744-683X (2019).

219. Brooks, B. R. *et al.* CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry* **30,** 1545–1614. ISSN: 01928651 (July 2009).

220. Honeycutt, J. D. & Andersen, H. C. Molecular dynamics study of melting and freezing of small Lennard-Jones clusters. *The Journal of Physical Chemistry* **91,** 4950–4963. ISSN: 0022-3654 (Sept. 1987).

221. Hagberg, A. A., Schult, D. A. & Swart, P. J. *Exploring Network Structure, Dynamics, and Function using NetworkX* in *Proceedings of the 7th Python in Science Conference* (eds Varoquaux, G., Vaught, T. & Millman, J.) (Pasadena, CA USA, 2008), 11–15.