curvedSpaceSim: A framework for simulating particles interacting along geodesics

Toler H. Webb, Daniel M. Sussman[*]

400 Dowman Dr., Emory University, Atlanta, GA, 30322

Abstract

A large number of powerful, high-quality, and open-source simulation packages exist to efficiently perform molecular dynamics simulations, and their prevalence has greatly accelerated discoveries across a wide range of scientific domains. These packages typically simulate particles in flat (Euclidean) space, with options to specify a variety of boundary conditions. While more exotic, many physical systems are constrained to and interact across curved surfaces, such as organisms moving across the landscape, colloids pinned at curved fluid-fluid interfaces, and layers of epithelial cells forming highly curved tissues. The calculation of distances and the updating of equations of motion in idealized geometries (namely, on surfaces of constant curvature) can be done analytically, but it is much more challenging to efficiently perform molecular-dynamics-like simulations on arbitrarily curved surfaces. This article discusses a simulation framework which combines tools from particle-based simulations with recent work in discrete differential geometry to model particles that interact via geodesic distances and move on an arbitrarily curved surface. We present computational cost estimates for a variety of surface complexities with and without various algorithmic specializations (e.g., restrictions to short-range interaction potentials, or multi-threaded parallelization). Our flexible and extensible framework is set up to easily handle both equilibrium and non-equilibrium dynamics, and will enable researchers to access time- and particle-number-scales previously inaccessible.

PROGRAM SUMMARY

[*]Corresponding author.
E-mail address: daniel.m.sussman@emory.edu

Program Title: curvedSpaceSim
CPC Library link to program files: (to be added by Technical Editor)
Developer's repository link: https://github.com/sussmanLab/curvedSpaceSim
Licensing provisions(please choose one): GPLv3
Programming language: C++
Supplementary material: Animations of energy minimizations described in the main text.
Nature of problem: Molecular-dynamics-like simulations of degrees of freedom evolving on a curved two-dimensional manifold according to standard equilibrium or non-equilibrium equations of motion and interacting via geodesics.
Solution method: We discretize both time and space, using modern tools from discrete differential geometry to efficiently find geodesic paths and distances. MPI parallelization is implemented to access large system sizes, and where appropriate (e.g., when dealing with short-ranged inter-particle potentials) we implement the ability to aggressively prune data structures, greatly decreasing the computational cost of our many-particle simulations.

---

1. Introduction

While most phenomena we observe take place in flat (Euclidean) space, efforts to understand the behavior of systems in curved space have a rich history across many disparate fields of research [1, 2]. These are perhaps most generally well-known in the context of the Thomson problem and its generalization – i.e., finding energy minima or maximally-separated arrangements of points on curved surfaces [3, 4, 5, 6, 7]. In some cases one can mathematically map physical problems in flat space to generalized packing problems in curved space, as in the connection between ground states of twisted fiber bundles and disk-packing on families of curved surfaces [8, 9]. Other, more direct, cases in which understanding the behavior of particles on curved surfaces is relevant can be found in settings ranging from the adsorption of particles on porous media or other highly curved substrates [10, 11], to colloids trapped at curved fluid-fluid interfaces [12], to the behavior of epithelial cells that form curved monolayers [13, 14, 15, 16, 17]. Indirect use cases can be found in which the curvature of space is used to tune relative levels of geometric frustration, as in explorations of disordered solids embedded in positively or negatively curved two- and three-dimensional spaces [18, 19].

Recent work in soft and living material systems has further highlighted the novel ways that curvature can affect collective dynamical and mechanical properties. For instance, theoretical and computational work has indicated the possibility for novel phenomena when considering non-equilibrium dynamics on curved surfaces [20, 21, 22], which may be related to recent observations of unusual velocity waves induced by curvature on the surface of cell spheroids [23]. Coupling between growth, deposition, and curvature has been proposed as a robust route to surface patterning in pollen grains, spores, and insect cuticles [24, 25]. Experimental work has shown that the modulation of both cell shape [26] and motility [27, 28] can be mediated by curved surfaces, and that cells can both sense and use these curvature cues to regulate remodeling and collectively migrate [29, 26, 30].

The above is an incomplete sampling of the many physical settings in which understanding the behavior of particles in curved space is important, but there are substantial challenges to studying the evolution of particles in curved spaces [1]. In flat space there are, of course, a wealth of well-developed and robust frameworks for performing molecular dynamics (or related particle-based) simulutions — to give just two examples, LAMMPS [31] and HOOMD-blue [32] are two remarkably successful packages that both perform "classic" molecular dynamics simulations in efficient, highly parallelizable ways. For *simple* curved spaces, such as spheres or cylinders, these packages can be extended straightforwardly; such surfaces allow analytic expressions for distance and derivatives thereof, which are the quantities needed to forward solve particle-based equations of motion. There are many examples of work being done on such simple surfaces with these or other simulation packages [20, 33, 34]. However, for more complex surfaces – even one as relatively simple as a torus – the absence of easy analytic expressions for distance or transport makes the straightforward extension of existing frameworks impossible.

Alternate approaches to studying dynamics on complex surfaces include considering instead the solution of continuum (e.g., phase field or hydrodynamic) models on curved surfaces, as in recent work on cylindrical [35] or other arbitrary surfaces [36, 37]. Yet another approach eschews the problem of calculating geodesic distances altogether, and considers particles that are constrained to manifolds but interact according to their Euclidean separation. This includes using sufficiently long-ranged repulsive interactions so energy ground states have particles pushed to the surface [38] (motivated by the so-called "poppy-seed bagel theorem" [39]), while typically constraining particles to the surface via projection operators [1, 40]. All of these approaches have been productive, but have left unfilled a need to directly study curved-space collective particle dynamics using geodesic distances and geodesic particle transport. In this work we address this need by describing a flexible, extensible, open-source computational framework for efficiently performing particle-based simulations on non-self-intersecting curved surfaces.

While our main focus is on closed surfaces with otherwise arbitrary geometry and topology (as in the tori considered in Sec. 5), we note that other surfaces are in principle straightforward to include. When considering surfaces with a boundary (as in the "silo-like" surfaces we consider in Sec. 6) one needs to also implement specific boundary conditions for particle positions and velocities, of which below we will consider only one flavor. Furthermore, while we have focused on a command-line interface to the underlying codebase, we have also integrated some of our framework with a graphical user interface. An example is shown in Fig. 1, and we hope that its inclusion helps with building intuition and guiding initial explorations of parameter space.

The rest of this article is organized as follows. In Sec. 2 we first review some of the core ideas of performing molecular-dynamics-like simulations, and in particular focus on a division of the problem into classes that perform the usual work of evaluating forces and integrating various equations of motion [42] and those that handle metric spaces and the core "shift" and "displacement" functions [43] that will differ from one curved surface to another. We next review the essential concepts and tools from discrete differential geometry that we use to construct these more specialized classes. In Sec. 3 we discuss some of our algorithmic implementations, and in Sec. 4 we show performance data, focusing on the scaling of computational cost with increasing particle number and on surfaces of increasing complexity (along with the strong scaling associated with parallelizing these calculations across multiple processing units). We demonstrate a simple application of our simulation framework (related to the generalized Thomson problem on the torus) in Sec. 5, and show results on a more complex open surface in Sec. 6. We close with a brief discussion of future directions in Sec. 7.

2. Background

2.1. Particle-based simulations

Molecular dynamics and related methods discretize time to solve for the evolution of large numbers of degrees of freedom interacting according to

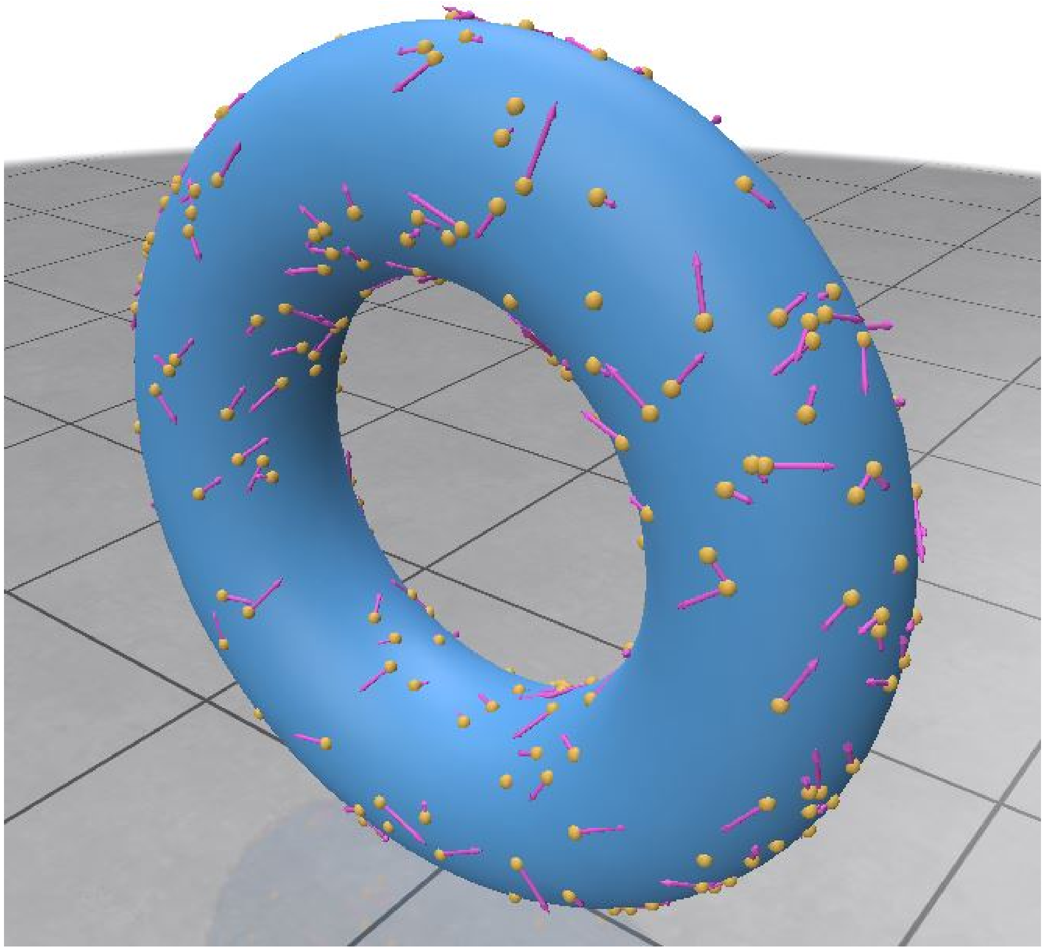

Figure 1: Snapshot from a real-time user interface demonstrating the NVE dynamics of 300 particles on the surface of a torus. In this example, particles interact according to a soft harmonic repulsion along geodesic curves, and velocity vectors are parallel transported on geodesics as the particles move. Visualization provided via the Polyscope package [41].

pairwise or many-body forces and updating according to one of any number of equations of motion [42]. One common choice is to integrate Newton's law of motion,

$$\boldsymbol{f}_i = m_i \boldsymbol{a}_i, \tag{1}$$

where $\boldsymbol{f}_i$ is the force on particle i, m_i is its mass, and $\boldsymbol{a}_i$ its acceleration. One can discretize time via, e.g., a velocity Verlet update scheme to simulate particles in the NVE ensemble, using any number of thermostats and barostats to specify alternate statistical ensembles. Particle motion can also be coupled to a strongly dissipative background, as in the case of discrete Brownian dynamics (a limiting form of Langevin dynamics),

$$\lambda \Delta r_{i,\alpha} = (\Delta t) f_{i,\alpha} + \sqrt{2 k_B T \lambda \Delta t}\, \eta_i, \tag{2}$$

where α denotes a Cartesian component, λ is a frictional term and the noise η_i is delta-correlated white noise with $\langle \eta \rangle = 0$ and $\langle |\eta|^2 \rangle = 1$. Active systems can be simulated by using any number of out-of-equilibrium dynamics, such as self-propelled particle dynamics [44], or via active fluctuations in mechanical stresses [45]. Non-physical dynamics can be imposed to measure transport coefficients by imposing artificial fluxes (e.g., of momentum) on the degrees of freedom [46]. Although we do not focus on this issue here, we note that careful analysis may be required to derive appropriate equations of motion in the context of particles constrained to curved surfaces, for which surprising and novel terms may arise [47].

Many object-oriented simulation packages have been developed to perform one or more of these flavors of particle-based simulations while easily swapping between different equations of motion and particle-particle interactions. All of these methods rely on a small number of algorithmic primitives: the ability to calculate forces (typically by considering gradients of a conservative potential), and the ability to update state information (updating the position or velocity of a particle, for instance). Computing forces is straightforward in principle, and in practice much historical algorithmic development has centered on efficiently computing forces for large numbers of interacting particles – Verlet lists, cell lists, and similar accelerating structures for short ranged potentials [48], Ewald summation for long-range potentials under periodic boundary conditions [49], and so on. Interesting variations sometimes include the addition of non-equilibrium aligning "forces" [50, 51, 52] or forces representing specialized interactions found in living materials [53, 54, 55], but this typically induces at most a modest increase in the difficulty of performing simulations. The second core algorithmic need – updating state information – is so trivial that typically little thought is given to it: particle positions and velocity updates simply amount to vector addition, potentially with some additional bookkeeping to deal with boundary conditions.

Concretely, consider a pairwise interparticle potential corresponding to a repulsive Gaussian,

$$U(l_{ij}) = Ae^{-\frac{l_{ij}^2}{2\sigma^2}}, \tag{3}$$

where A determines the stiffness of the interaction, σ is a measure of the length scale over which the repulsive force acts, and l_{ij} is the distance between particles i and j. The force corresponding to this potential is

$$f_{ij} = -\nabla U = \frac{l_{ij}}{\sigma^2} U(l_{ij}) \nabla l_{ij}. \tag{4}$$

In a typical particle-based simulation in Euclidean space, both the distance l_{ij} and its gradient are easily calculable. Using $r_{i,\alpha}$ to represent the Cartesian coordinate α of particle i, one needs the appropriate Euclidean metric for length,

$$l_{ij} = \sqrt{\sum_{\alpha} \left(r_{j,\alpha} - r_{i,\alpha}\right)^2}, \tag{5}$$

and the fact that the gradient is the unit vector tangent to the line between the two particle positions:

$$\nabla_\alpha l_{ij} = \frac{\left(r_{j,\alpha} - r_{i,\alpha}\right)}{\sqrt{\sum \left(r_{j,\alpha} - r_{i,\alpha}\right)^2}}. \tag{6}$$

From Eq. 4 we see that generalizing the calculation of a force from Euclidean to curved spaces corresponds to generalizing the calculation of *distances* and *gradients of distances*. We will refer to the joint task of computing these two quantities as the "`displacement`" algorithm, and will refer to the task of updating state information (either moving a particle position along a geodesic that points in some direction, or parallel transporting a particle's velocity vector along the same path) as the "`shift`" algorithm.

2.2. Discrete differential geometry

In this software package we focus our attention on simulating particles constrained to curved surfaces and interacting along geodesics – the so-called "curved-line-of-force" problem [1, 56]. Here we review the basic (discrete) differential geometry employed in our simulations, and refer interested readers to Refs. [57, 58] for more thorough treatments.

In curved space, the first quantity of interest is the equivalent of the distance l_{ij} between points i and j. The challenge is to find *geodesics* – paths which are straightest and locally shortest curves between two points on the surface. If a path on the surface – $\mathbf{r}(s)$, where s is an arc-length parameterization of the path – is already known, the distance along that path between points can be expressed as an integral,

$$d_{ij} = \int ds \sqrt{g_{\alpha\alpha}\left(\frac{dr_\alpha}{ds}\right)^2 + g_{\beta\beta}\left(\frac{dr_\beta}{ds}\right)^2 + 2g_{\alpha\beta}\frac{dr_\alpha}{ds}\frac{dr_\beta}{ds}}. \tag{7}$$

Here $r_\alpha(s)$ and $r_\beta(s)$ are the intrinsic coordinates of the path on the surface. The *metric tensor* $\mathbf{g}$ contains the information needed to define distances and angles in the tangent space of the surface at every point. The components of the metric tensor are given by

$$g_{\alpha\beta} = \partial_\alpha \boldsymbol{R} \cdot \partial_\beta \boldsymbol{R}, \tag{8}$$

where $R(s_\alpha, s_\beta)$ is a parameterization of the surface in Euclidean space. For example, given a surface defined by $\boldsymbol{R}(x, y) = (x, y, \sin x)$, we have $\partial_x \boldsymbol{R} = (1, 0, \cos x)$, $\partial_y \boldsymbol{R} = (0, 1, 0)$, and hence

$$\mathbf{g} = \begin{pmatrix} 1 + \cos^2 x & 0 \\ 0 & 1 \end{pmatrix}. \tag{9}$$

Finding the geodesic (and also the geodesic distance) between the two points corresponds to extremizing Eq. 7 over all possible paths between the two points. This is equivalent to solving the "geodesic equation," a second order ordinary differential equation given by

$$\ddot{r}^\gamma + \Gamma^\gamma_{\alpha\beta}\dot{r}^\alpha\dot{r}^\beta = 0. \tag{10}$$

Here Γ represents the Christoffel symbols, themselves functions of the metric tensor,

$$\Gamma^\gamma_{\alpha\beta} = \frac{1}{2}g^{\gamma\zeta}\left(\frac{\partial g_{\beta\zeta}}{\partial r^\alpha} + \frac{\partial g_{\alpha\zeta}}{\partial r^\beta} - \frac{\partial g_{\alpha\beta}}{\partial r^\zeta}\right), \tag{11}$$

which help describe vector transport. In the above expressions Greek indices correspond to intrinsic surface coordinates, subscripts and superscripts correspond to covariant and contravariant quantities, and we use the Einstein summation convention for repeated indices. While possible, the direct method of solving the geodesic equation for every pair of points in the simulation and integrating over the found geodesic paths to calculate distances is hopelessly slow – replacing a simple Euclidean norm of the difference of two vectors with the solution to a 2nd order ODE followed by a numerical integration will not permit simulations with large numbers of particles. Evidently, the direct approach is reasonable only when the geodesic equation can be analytically solved, but this is limited to extremely specialized geometries (i.e., spheres and the hyperbolic plane). Indeed, even for the relatively simple case of a torus the solutions to the geodesic equation are surprisingly complicated, and not all torus geodesics can be described analytically [59].

2.3. Simulation strategy and algorithmic challenges

To make progress, then, we formulate our simulations with a discretization of both time *and* space, allowing us to use the tools of discrete differential geometry [60]. We replace the smooth curved surface of interest with a triangulated mesh with V total vertices. Just as discretization in time leads to deviations from the true solution to the equations of motion – where the order in the error depends on the specific algorithm (e.g., $\mathcal{O}(\Delta t)$ for an Euler time-stepping method, or $\mathcal{O}(\Delta t^2)$ for a straightforward velocity Verlet integrator, etc.) – this discretization of space will lead to errors in the calculation of distances, forces, and shifts that are typically of order $\mathcal{O}(h^p)$, where h is a characteristic length scale of the triangles in the mesh and p is a number that will be sensitive to the precise discrete-differential-geometry algorithms used.

A large number of techniques for computing geodesic lengths in the discrete setting have been proposed; see Ref. [61] for a recent survey. Different techniques fall into either exact or approximate categories – exact methods possess guarantees about finding true geodesics *on the approximated surface* whereas approximate methods do not but are typically faster. This corresponds to effectively different values p for the order of error made by the discretization of the surface in various calculations. For the problem of finding geodesic distances, e.g., Ref. [62] suggested an $\mathcal{O}(h^1)$ scaling for the approximate "vector heat method" and $\mathcal{O}(h^2)$ for a competing exact method (described below).

Potential tradeoffs in these error scales go hand in hand with the computational cost of computing those geodesic distances. In the generic case finding the geodesic between arbitrary points on a surface requires information about the entire surface, and so while the error scales with (some power of) the typical mesh size the computational cost finding a geodesic distance also scales with (some power of) the number of faces in the mesh. In the context of a particle-based simulation, this should be compared with the cost of computing a generic pairwise distance in Euclidean space, which takes only a handful of arithmetic operations (c.f. Eq. 5). Given that reasonably smooth approximations of a given surface might involve discretizing into tens of thousands of faces, it is clear that efficiently simulating a large number of particles with geodesic interactions requires specializing the existing implementations of discrete geodesic solvers. Thus, in Sec. 3.1 we emphasize our efforts to incorporate "submeshing" operations in the common case that particle interactions are short ranged: this will let us perform geodesic calculations whose cost scales not with the number of faces of the entire mesh, but only with a more limited subset of the original mesh.

In considering a base algorithm a build our simulation around, we ultimately settled on a version of one of the exact algorithms, and in particular one which returns not only the geodesic distance between points but also the full discrete path. First proposed in Ref. [63], the "MMP" algorithm finds straightest, shortest paths between points by considering all possible local unfoldings of the mesh into a single plane and then looking for the shortest path between the two points of interest that stays within the unfolded mesh. Substantial improvements to the core algorithm – adding priority queues for promising unfolding orders, aggressively pruning away unfoldings which cannot contain the correct path, and managing memory [64, 65] – have made it quite performant, and a form of the latest Xin and Wang (XW) [66] version of MMP can be found in the Computational Geometry Algorithms Library (CGAL) [67]. MMP and its variants are "single-source-all-distance" algorithms – given a source point they consider all possible unfoldings and find for essentially the same amount of computational effort the distance from the source point to *all vertices of the mesh*. This makes them particularly appropriate for the task of constructing neighbor lists by finding, e.g., the distances to all neighbors of particle i simultaneously. We note, however, that the vector heat method mentioned above may be fast enough that using it with much finer meshes (to compensate for the $\sim \mathcal{O}(h)$ difference in discretization error) may ultimately be more efficient overall.

The implementation of the XW algorithm that we use *requires* that the input surface be a triangulated mesh. We exploit this requirement in our choice of internal data structures, since we can express not only particle positions but also ray-edge intersection tests in barycentric coordinate systems. This does imply, however, that if one wished to perform simulations on surfaces initially represented in other ways (e.g., with point clouds or meshes constructed from non-triangular polygons) one would first need to convert such representations to a triangulated mesh. We note that many tools exist to perform these conversions, and we largely take the preparation of input surfaces to be outside the scope of our code base. Finally, we note that because the XW algorithm considers flat unfoldings of the mesh into the two-dimensional plane, “sharp” edges (i.e., edges across which the surface normal has a large variation) are not *a priori* an issue. In contrast, we do not presently allow for particle motion — or, indeed, even distance calculations that return a finite value — between disconnected parts of a surface.

3. Methods and implementation

As noted above, many of the common components in frameworks for performing particle-based simulations are quite standard [42]. Our object-oriented C++ code uses a standard shared-pointer paradigm in which a governing `simulation` object connects a `model` containing state information (particle positions and velocities) with `updater`s that implement specific equations of motion and also with `force`s by which particles interact. This basic structure (plus associated utility classes) makes it straightforward to implement new equations of motion, new pairwise potentials, and other more exotic interactions between degrees of freedom. It also cleanly separates the standard components of a particle-based simulation from the more unusual `displacement` and `shift` algorithms we need to implement on discretized surfaces. We define `space` classes that implement these algorithms – either the trivial versions in Euclidean spaces or the discrete differential geometry analogs mentioned above in “Mesh” spaces – and then connect a `model` object to a `space` object its degrees of freedom live in. When particles live in a Euclidean space, positional information is stored directly as a location in that space; when particles instead live on a mesh, we overload the relevant data structure to contain an index for the face containing the particle and a triple of doubles indicating the position of the particle within that face in barycentric coordinates [60].

3.1. Calculating distances, gradients, and forces

The XW algorithm is a single-source-all-distance approach that requires a significant computational cost to create a *sequence tree* data structure (that stores information about the set of possible unfoldings of the mesh relative to the source). Once created, this structure can be queried to calculate both discrete geodesic distances to any other point in the mesh and also the full geodesic path corresponding to that distance. The sequence tree has a computational complexity that is in principle $\mathcal{O}(V^2 \log V)$ to build, where V is the number of vertices in the mesh. Interestingly, in practice it seems that this worst-case complexity rarely holds for most meshes [66], as is evident from our performance analysis in Sec. 4.

As forces are gradients of potential energy, we next consider the gradients in distance. Since the geodesics we want to consider are the shortest and straightest paths between points, the gradient of the geodesic distance with respect to one of the endpoints of the path must point in the direction locally tangent to the geodesic at that endpoint [58]. One can intuitively view this as a triangle inequality applied to our curved space calculations: adding an infinitesimal segment of fixed magnitude dr to a geodesic path can only maximally increase the distance between the path endpoints if the segment extends in the direction tangent to the original path at that endpoint. Fortunately, the XW algorithm is one which finds geodesic distances by actually computing the full geodesic path, and so this tangent vector information is trivial to retrieve.

This approach breaks down, however, near a "cut locus," where multiple geodesics with very different headings may connect the same two positions. On smooth surfaces, such multiple connecting geodesics arise (for instance) for antipodal points on a sphere or around opposite points on the outer ring of a torus. For discretized meshes, large deviations in geodesic direction can occur around much smaller features. For example, a geodesic that passes on one side of a surface vertex might change direction significantly because a small perturbation causes the geodesic to take a path on the opposite side of the vertex. This effect is a source of discretization error, and can be an especially sensitive issue when trying (for instance) to find force-balanced configurations to high degrees of accuracy. One can expect its frequency to increase with mesh complexity – due to the increasing number of mesh vertices – but the magnitude of the effect itself to decrease with mesh complexity as the variation in geodesic headings becomes smaller.

Indeed, since our code base ties together standard components of molecular-dynamics-like simulations (which we have validated in flat space) with new components for computing distances, gradients, and displacements in curved space, the fundamental validation of code involves quantifying the error made in our spatial discretization of the curved surface. In order to quantify this, we measured the relative error in several quantities on meshes with different levels of refinement (i.e., with an increasing number of increasingly small triangles): the geodesic distance, the gradient of the geodesic distance, and the error in final position when a particle is displaced by a fixed magnitude in a random direction. These results are shown in Fig. 2, where in order to have access to the correct answer we performed our tests on the sphere (for which, of course, geodesics, gradients, and parallel transport are all analytically calculable). Consistent with the coarse results seen in Ref. [62], we observe error in all of these quantities that scale as $\sim h^2$, where we take h to be the square root of the average area of a triangle on the surface.

Returning to our simulation framework, the practical calculation of forces reduces to the operation of the `displacement` algorithm. Given a source particle, we identify all potentially interacting particles as "targets," generate the sequence tree for the source particle, and then query the cached sequence tree to obtain the paths from the source to each of the targets. This gives the lengths l_{ij} and the tangent directions $\dot{\boldsymbol{r}}_{ij}$ relative to the source point's location, from which one can (e.g.) compute the total force on i via

$$F_i = \sum_{j=1}^{N} -\frac{dU\left(l_{ij}\right)}{dl_{ij}} \dot{\boldsymbol{r}}_{ij}. \tag{12}$$

where the magnitude $\frac{dU(l_{ij})}{dl_{ij}}$ is a pre-computed quantity written into a force class.

For particles interacting via short-ranged forces, this approach contains an obvious inefficiency. The XW algorithm scales with the *total* number of vertices of the discretized surface, but particles may only be interacting in a much smaller local patch of the surface. For short-range interactions, then, we perform a *submeshing* operation: for a potential that vanishes beyond some distance σ, we exploit the fact that the Euclidean distance between two points is a *lower bound* to the geodesic distance. For a given source particle, we consider the local patch of the surface which contains only those triangular faces (and target particles) whose minimum Euclidean distance is within σ of the source particle position, and then build the XW sequence tree

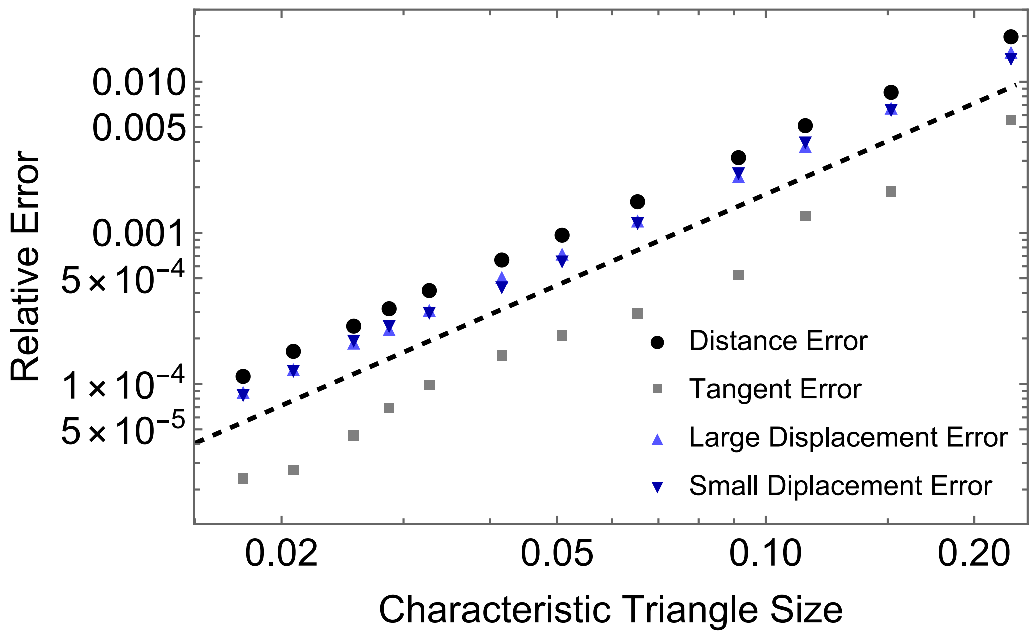

Figure 2: Relative error in the geodesic distance (black circles), gradient of the geodesic distance (gray squares), and location after random displacements with large (light blue upward triangle) and small (dark blue downward triangles) magnitdes on a spherical surface of unit radius. Errors are measured against same quantities calculated analytically on a smooth sphere, and we take large and small displacements to correspond to 0.5 and 0.05, respectively. Coarser meshes yield less accurate results, and the relative error of all quantities decreases roughly as h^2, where h is the square root of the mean triangle area of the corresponding mesh. The dashed line is a guide to the eye with slope 2, and all data points are averages over 200 measurements.

only for these local patches. Since the local patch might contain a number of vertices $V' \ll V$, this can speed up the computation of each time step by orders of magnitude. The overarching workflow of distance calculation is depicted in Fig. 3 as an algorithmic flowchart.

3.2. Updating state information

The simpler of the two main algorithmic tasks, in flat space and on curved surfaces, is the `shift` update of positions and velocities of particles given a displacement vector. Force and velocity vectors are *always* defined in the tangent plane of the particle's current face, and we require (a) that all requested displacements are likewise in the tangent plane and (b) that at all times the particles remain on the surface. One strategy for achieving this goal is to use a projection operator approach: for instance, a particle would be displaced by $\boldsymbol{d}$ via standard vector addition (which is, indeed, the entirety of the `shift` algorithm in flat space), and then projected back to the nearest point on the discretized surface. While straightforward, this leads to displacements that do not themselves follow geodesics – i.e., they are not the equivalent of a "straight" displacement.

Instead, we move across connected faces along a path whose total length is the same as the magnitude of the desired initial displacement. Every time an edge is crossed, the path is rotated around the axis defined by the cross product of the face normals, as shown in Fig. 4. This is equivalent to first defining a vector corresponding to the direction and magnitude of desired motion and then unfolding a sequence of faces so that the vector corresponds to a straight line that stays entirely within a connected set of faces – i.e., of shifting the particle along a geodesic. The only ambiguity occurs when a vector would pass through a spherical vertex —that is, a vertex for which the sum of edge-edge angles is less than 2π. Here the straightest path (which continues through the vertex) is no longer the shortest path on the surface (which does not). The physical choice in such a case is to displace particles along the straightest-but-not-shortest path, which corresponds to maintaining (e.g.) the direction of particle velocity [68].

Because it is not *a priori* clear given a point and a displacement vector how many faces will be crossed, in our `shift` algorithm we implement a while loop that (a) checks for intersections — in barycentric coordinates, to minimize finite-precision issues in these geometrical calculations — between a ray emanating from the particle's current location and any edge of its current face, (b) moves the particle along that vector to either the final

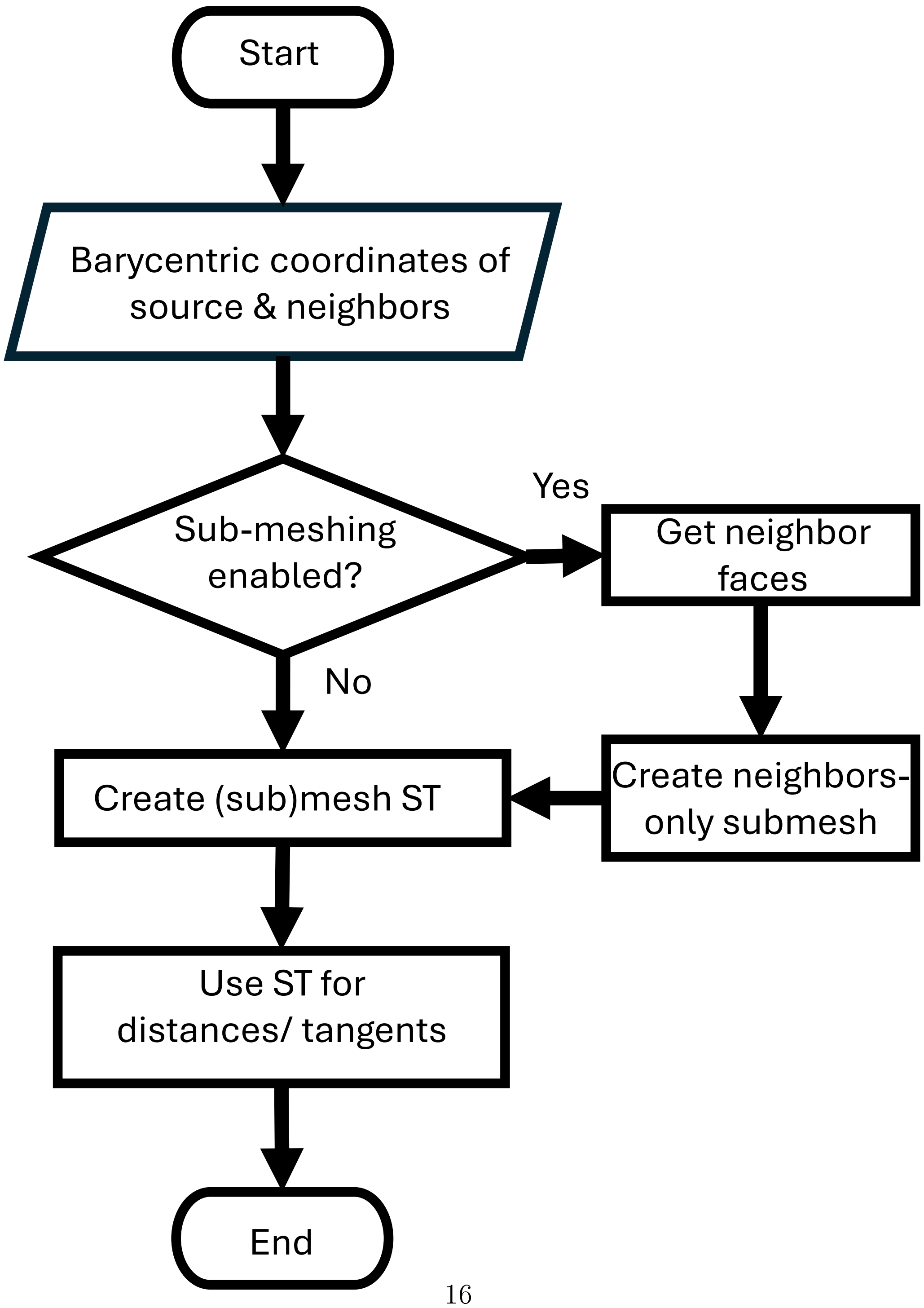
Start
Barycentric coordinates of source & neighbors
Sub-meshing enabled?
Yes
Get neighbor faces
No
Create (sub)mesh ST
Create neighbors-only submesh
Use ST for distances/ tangents
End

Figure 3: Depiction of the distance calculation process. The *sequence tree* (ST) is an object which stores information about mesh traversal used in the Xin and Wang algorithm.

location (if in the face) or the intersection point, (c) if necessary, updates the face associated with the particle to the face that shares the intersected edge with the original face, and (d) rotates the vector about the intersected edge into the plane of the new face. This procedure is repeated until a total displacement of the correct magnitude has been performed (with appropriate cases handling intersections with vertices rather than edges). This loop has the benefit of also easily handling parallel transport of vectors as well as shifts of particle positions: a velocity or force vector simply gets carried along with the position, rotating along with the rotation of the displacement vector as edges or vertices are crossed. Figure 5 depicts the `shift` routine as an algorithmic flowchart, and its accuracy as a function of mesh discretization is shown for spherical meshes in Fig. 2.

4. Performance Benchmarks

For all single-core simulations we benchmarked our simulation architecture on a Dell OptiPlex 7080 workstation with an Intel i9-10900K CPU clocked at 3.70 GHz. All benchmarking and example simulations were done using a single thread, except where noted otherwise; where used MPI benchmarks were conducted on the NCSA Delta CPU cluster using AMD EPYC 7763 processors.

We first test the performance of our algorithm on a fixed surface at fixed discretization scale (in this case, a torus with minor radius $r_1 = 1$ and major radius $r_2 = 3$, discretized into 4616 triangular faces), as a function of the number of particles, N. For the main test we considered a finite-range repulsive pairwise potential whose interaction range was set to $\sigma = 1$, and we computed the mean time to execute a timestep of velocity Verlet dynamics. For a standard (flat space) simulation *at fixed density* one would expect $\mathcal{O}(N \log N)$ scaling; in a domain of fixed spatial extent the number of interacting particle pairs itself grows with N, and we would anticipate roughly $\mathcal{O}(N^{1.5})$ scaling for particles that attempt to uniformly cover the curved surface. This is roughly consistent with the observed scaling shown in Fig. 6.

We also have implemented a basic MPI communication pattern using the openMPI framework [69] pattern to partition work across multiple processors – for our basic implementation we note that the cost of computing even submeshed sequence trees is substantially higher than the cost of the mere vector addition and subtraction needed to compute distances and gradients in flat space, and we expect large gains from parallelization by arbitrarily and evenly dividing the cost of computing forces on particles among the available processors. A future enhancement to this would involve further using spatial location of the particles to divide the computational workload, as is common in molecular dynamics in flat space. Nevertheless, we observe reasonable strong scaling behavior of our simple implementation at modest system sizes. We test our implementation by choosing a fixed mesh (a torus whose large radius is 3 in simulation units, small radius is 1, discretized into 4616 faces) and investigating the time to complete NVE timesteps relative to a single-core simulation for varying numbers of particles and CPU cores. The bottom panel of Fig. 6 shows the speedup (defined as the profiled time per timestep when running on a single MPI rank divided by profiled time per timestep when running on multiple ranks) for particles interacting via a soft harmonic potential of range $\sigma = 1/8$. For this naive implementation, we estimate that assigning we estimate that assigning each processor of order 2×10^3 particles almost completely hides the communication cost and results in ideal strong scaling. For particles interacting with greater range, we expect that a correspondingly smaller number of particles per processor would hide the communication cost.

The counterpart to these analyses is to consider simulating a fixed number of particles on meshes of increasing numbers of vertices – this corresponds to the spatial version of choosing a time-step size that sufficiently minimizes discretization error. In addition to the $\mathcal{O}(V^2 \log V)$ worst case scaling for building the sequence tree, there are also geometric prefactors (the length scale of curvature variations, the density and placement of saddle vertices, etc) that can strongly influence the cost of these calculations. Some benchmarks have even reported non-monotonic scaling in the sequence tree generation time as a function of vertex number in some regimes [70]. In Fig. 7 we report the mean time to complete a timestep as a function of V for a small number of representative surfaces: a sphere (for which all calculations can be compared with the analytical result), tori of two different aspect ratios (3 and 20), and an elephant (in this case, represented as a geometrically interesting genus-3 surface). In all of these tests we hold the mesh span and the range of interactions fixed while increasing the number of triangles used to discretize the surface with an incremental triangle-based isotropic remeshing algorithm [71]. We indeed find experimentally that there can be important differences when simulating particles on different meshes. Reflecting the local flattening that submeshing can accomplish, we find very roughly $\mathcal{O}(V^{1.5})$ scaling for finite-range simulations compared to roughly $\mathcal{O}(V^2)$ behavior for computing sequence trees across the entire mesh.

5. Crystallization on the surface of a torus

In this section and the next we present two examples in which we study ground-state configurations of particles on curved surfaces and compare with some results in the literature. Here, we first consider particles interacting via purely repulsive potentials on the surface of the torus. The torus is often taken as one of the simplest examples of a topologically non-trivial surface with gradients in the surface curvature [72], and crystallization on their surface has been studied both theoretically and numerically [38, 73]. We make some direct comparisons with those earlier numerical studies, which implemented with forces between particles according to Euclidean (rather than geodesic) distance and used a long-range Riesz potential ($U(l_{ij}) = l_{ij}^{-3}$) to ensure that in ground states particle positions were on the surface of the torus.

Ref. [38] predicts a variety of interesting toroidal crystals – these range from precise ground state configurations in the small-N regime, to patterns of crystalline defects, to limiting torus aspect ratios beyond which defect-free configurations should be the true ground state of the system. In Fig. 8 we show an in-surface Voronoi diagram (generated as a restricted Euclidean Voronoi diagram via SurfaceVoronoi [74]) of a typical low-energy configuration.

As gradient descent is particularly suitable for finding nearby *local* minima rather than global minima, in the tests below we primarily aim to find expected distributions of defects and qualitative geometric signatures, rather than true ground states. For example, given 32 particles on an aspect ratio 3 torus, the true ground state consists of 16 pairs of 5-7 defects – 16 particles with 5 neighbors and 16 particles with 7 neighbors – arranged on the outer and inner portions of the torus, respectively [38]. While gradient descent starting from a random point pattern does not find the true ground state, we indeed find a set of defect pairs distributed in the expected regions of the torus, as seen in Fig. 8.

As the number of particles on the surface of the torus increases, we expect to find *chains* of defects coexisting with a large number of particles which interact with six neighbors [38]. We performed an $N = 1000$ particle simulation which began from a random initial distribution of positions and relaxed via gradient descent. We observe a greater total number of defects than expected in the true ground state [38], but as shown in the top panel of Fig. 9 we indeed found the the qualitatively expected result that in this particle-number regime defects are primarily localized along chains. Leveraging the computational efficiency afforded by our simulation framework, we performed over 1000 separate energy minimizations of $N = 500$ particles to find the expected distribution of defect charge in these local energy minima. As seen in the bottom panel of Fig. 9, we find that (as expected) there tend to be excesses of 7-fold and 5-fold defects on the inner and outer portions of the torus, and where the Gaussian curvature vanishes, so too does the expected defect charge.

For tori of larger aspect ratio, the ground states correspond to lattices with a slow twist – particles forming helical chains around the surface – but fewer total defects. We initially seed 300 particles positions on an aspect-ratio-20 torus on a lattice with *incommensurate* twist (so that the initial configuration, while not disordered, is nevertheless far from the true ground state). We find that the particles relax to a structure with regions of correct crystalline order separated by grain boundaries. This is shown in Fig. 10. In detail, this configuration has 11 five-fold and seven-fold defects, many localized between different twisted lattice configurations (e.g., on the top left of the figure).

We have included supplemental videos showing the evolution of particle positions during minimization corresponding to the results shown in Figs. 8-10. We have shown that we can reproduce qualitatively expected signatures of ground state configurations using direct gradient descent. This includes signatures of the high-defect-number ground state for 32 particles in their relaxed distributions on an aspect ratio 3 torus, defect chains and expected distributions of defects over multiple quenches for many more particles on the same, and finally twisted regions within the aspect ratio 20 torus that reflect the expected helical configuration in the defectless ground state. Although not our goal in this section, we fully expect that implementing more sophisticated minimization protocols – simulated annealing, quench-and-perturb, etc. – would allow us to more fully reproduce the true ground-state behavior reported in the literature [38].

6. Geometric Configurations on a Filament Bundle-Analogous Surface

While the toroidal surfaces considered in the previous section are more complex than spherical geometries, we also highlight here our framework's ability to study more complex surfaces — in this case, by including non-trivial boundary conditions. To that end, we show numerical results that are complementary to the data on the ground-state configurations of twisted filament bundles in Ref. [8]. In that work, the filament bundles are characterized by a rate of helical rotation, Ω, about the z-axis. The centerline of a filament with position $\boldsymbol{x} = (x, y)$ at $z = 0$ is given by

$$\boldsymbol{r}(\boldsymbol{x}, z) = \boldsymbol{x}\cos(\Omega z) + (\hat{z} \times \boldsymbol{x})\sin(\Omega z) + z\hat{z}, \tag{13}$$

which is then conveniently rewritten in polar coordinates for which ρ is the radial coordinate and ϕ the angular coordinate. Bruss and Grason work out [8] the geometry of the vector of closest approach between two filaments as a function of the centerline position in the plane. In the limit of infinitesimal separation, this is

$$ds^2 = (\delta\rho)^2 + \Omega^{-2}\sin^2\theta(\rho)(\delta\phi)^2, \tag{14}$$

where

$$\theta(\rho) = \arctan(\Omega\rho) \tag{15}$$

is the local tilt angle of a filament.

Bruss and Grason note that the metric properties of bundle packing can then be understood by actually constructing surfaces that have the metric of Eq. 14, embedding those surfaces in 3D space, and studying disk packings on them. The authors describe these surfaces as *silo-like*, with a hemispherical cap smoothly transitioning to a cylindrical regime. An example of such a surface is depicted in Fig. 11.

Previous work in Ref. [8] includes both asymptotic close packing analyses of twisted filament bundles and augmented steepest-descent energy minimizations to attempt to find ground state configurations. In the latter, filaments interact via a Lennard-Jones potential; to deal with the exponential increase in local energy minima, the authors studied increasingly large ensembles as the bundle grew. Configurations were analyzed by first conformally mapping the filament coordinates to the 2D plane and performing a standard Delaunay triangulation. Notable numerical findings include that (a) for finite-sized bundles of radius R the total topological charge of disclinations, Q (expressed not fractionally but rather as the deviation of total coordination from $6N$), was a function only of ΩR, and (b) Q monotonically increased from 0 for weakly twisted ($\Omega R \ll 1$) bundles to a maximum topological charge of $Q = 6$ in highly twisted bundles.

We used curvedSpaceSim to study disk packings on these silo-like surfaces not via asymptotic close packing algorithms that start from a hexagonal lattice [8], or by an augmented steepest-descent-plus-tapping algorithm, but rather by simpler minimization protocols starting from many independently initialized particle configurations. Because curvedSpaceSim takes fixed meshes as its input, varying particle number in our simulations at fixed area fraction allows us to investigate different characteristic lengthscales of the bundle width to the particle size, R/d, without having to use many copies of meshes of equal Ω but varying extent.

We performed our energy minimizations using soft harmonic repulsions, in which the geodesic particle diameter d was set to have approximately equal area fraction $f = 0.93$ for different particle number N, $d = 2\sqrt{f\frac{A}{N\pi}}$. Initial particle positions were randomized, and we then performed a sequence of minimizations interleaved with high-temperature melting performed in the NVT ensemble by coupling the system to a Nosé-Hoover thermostat. We set a large enough temperature such that the result was functionally equivalent to starting with a new random configuration, but we wanted to document additional functionality of our simulation framework. Minimizations were done using the FIRE algorithm [75], with the configuration recorded after each minimization step. 200 minimizations were performed on each mesh for each of $N = 32$, 82, 124, and 256 particles. We cataloged ΩR for each configuration, where R is the geodesic cluster radius. R was measured as the average geodesic arc length from the center of the mesh cap to particle centers at the cluster edge.

We make two comments on our use of surfaces with a boundary rather than the tori of the previous section. The first is that when using such surfaces we must make a choice of boundary conditions for particle displacements and velocities near the boundary. In this case, we chose to implement boundary conditions that (a) prevent any particle motion beyond the edge of the mesh, (b) set the component of the velocity normal to the boundary to zero for any particle that contacts the boundary, and (c) freely permits particles to slide tangentially along the boundary.

The second is that these boundary conditions correspond to in-principle unbounded fictitious external forces that we allow to act on the particles (corresponding to the effectively hard constraint keeping particles on the surface). Because of this, while we can still search for force-balanced configurations via energy minimization, we choose not to use an energetic criterion to characterize the quality of the minima we find. Instead, we evaluate the geodesic Voronoi diagram for all of our minimized configurations, count the total topological charge for each diagram, and report charge statistics for all of these configurations (rather than just the best local minima found). We further explicitly note that in counting topological charge, we exclude particles on the boundary or their neighbors.

Our results on silos of varying Ω are shown in Figs. 12 and 13. In the statistics of the average charge over all of our configurations, Fig. 12, we unambiguously recover the previously reported monotonic increase of Q with ΩR. This is true for all values of R/d, but it is not clear that in these relatively-highly-defective local states of force balance the mean charge is in fact independent of R/d as suggested in Ref. [8]. The charges of the minimally-defective state in each of our 200 minimizations are shown in Fig. 13. We note that with our protocol (i.e., melting the configuration rather than gently tapping the boundary particles) it is much harder to find true crystalline ground states, even for relatively modest particle number. We also note that our counting method for Q — excluding boundary and boundary-neighbor particles — is especially problematic at low N, and can lead to specific configurations with very low or even spuriously negative counts of topological charge.

In addition to the difficulty in finding *minimally* defective ground states (i.e., states with $Q = 0$ at low twist), we surprisingly find *fewer* than expected defects for moderately large filament bundles. Although this may be related to the difficulty in properly accounting for defect charge of boundary particles — and hence may actually be reflective of states with much higher total charge — this was nevertheless surprising. As noted above, this boundary counting issue is almost certainly at play for small particle simulations (for which the boundary and boundary-neighbor particles make up a majority of the system), suggesting that our method may only recover the correct behavior for large N. While our *mean* charge counts seem to appropriately approach the limiting value of $Q = 6$ expected of ground states at high ΩR, we nevertheless find that even for minimizations of $N = 256$ particles we typically find "optimal" ground states with smaller than expected Q. We note that although the scale of spatial discretization we have chosen for these studies is reasonably small, we have not systematically checked the robustness of our defect counts to progressively refining the mesh.

7. Discussion and future developments

We have a presented a new framework for simulating particles on curved surfaces. We build around existing exact geodesic solvers on discretized surfaces [66], and implement new interfaces and data structures to efficiently submesh the surface when dealing with particles that interact via finite-range potentials. We integrate this with a standard, easily extensible object-oriented framework written in C++ to allow researchers to easily adapt our code with specialized particle-particle interactions or dynamical update rules appropriate to their system. We have provided standard reports for the scaling of our implementation – at fixed or varying particle number, on meshes of fixed or varying complexity, and when parallelizing across multiple processors – so that other researchers may easily estimate whether simulations of a particular use-case may be achieved.

We have chosen a particular, MMP-like geodesic distance solver, but we note that there are interesting differences in the computational complexity of alternate choices – such as the vector heat method [62] – and the way exact vs approximate solvers lead to different discretization errors. We are currently exploring the implementation of other approaches, something our object-oriented design makes straightforward. Given the potential importance of parallelizing the geodesic calculations, we are particularly interested in exploring the possibility of algorithms that will permit some of the geodesic calculations to be done on a GPU.

Beyond this, natural extensions of our code base include the addition of vector-alignment interactions – as are common in models of flocking [50] – to supplement the positional particle potentials already implemented. While we have so far focused on closed surfaces, it is well known that some interesting phenomena can only manifest themselves on surfaces with boundaries [76], for which we have looked at just one test case. Expansions of our work that explore and perhaps improve upon the existing boundary-handling routines — including by implementing periodic boundary conditions — may prove fruitful in the investigation of new problems.

The most up-to-date version of our codebase, including branches for our graphical user interface, other experimental future features, and a dedicated branch which allows the figures in this paper to be easily reproduced, can be found at `https://github.com/sussmanLab/curvedSpaceSim`.

8. Acknowledgements

We would like to thank Helen Ansell, Haicen Yue, and Tomilola Obadiya for fruitful discussions and critical comments on this manuscript. This material is based upon work supported by the National Science Foundation under Grant No. DMR-2143815. This research used the Delta advanced computing and data resource which is supported by the National Science Foundation (award OAC 2005572) and the State of Illinois. Delta is a joint effort of the University of Illinois Urbana-Champaign and its National Center for Supercomputing Applications.

References

[1] G. Tarjus, F. Sausset, P. Viot, Statistical mechanics of liquids and fluids in curved space, Advances in Chemical Physics 148 (2011) 251–310.

[2] B. Schamberger, R. Ziege, K. Anselme, M. Ben Amar, M. Bykowski, A. P. Castro, A. Cipitria, R. A. Coles, R. Dimova, M. Eder, et al., Curvature in biological systems: its quantification, emergence, and implications across the scales, advanced materials 35 (13) (2023) 2206110.

[3] J. J. Thomson, Xxiv. on the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 7 (39) (1904) 237–265.

[4] P. M. L. Tammes, On the origin of number and arrangement of the places of exit on the surface of pollen-grains, Recueil des travaux botaniques néerlandais 27 (1) (1930) 1–84.

[5] M. Bowick, A. Cacciuto, D. R. Nelson, A. Travesset, Crystalline order on a sphere and the generalized thomson problem, Physical Review Letters 89 (18) (2002) 185502.

[6] S. Agarwal, S. Hilgenfeldt, Simple, general criterion for onset of disclination disorder on curved surfaces, Physical review letters 125 (7) (2020) 078003.

[7] S. Agarwal, S. Hilgenfeldt, Predicting the characteristics of defect transitions on curved surfaces, Soft Matter 17 (15) (2021) 4059–4068.

[8] I. R. Bruss, G. M. Grason, Non-euclidean geometry of twisted filament bundle packing, Proceedings of the National Academy of Sciences 109 (27) (2012) 10781–10786.

[9] I. R. Bruss, G. M. Grason, Topological defects, surface geometry and cohesive energy of twisted filament bundles, Soft Matter 9 (34) (2013) 8327–8345.

[10] P. Pincus, C. Sandroff, T. Witten, Polymer adsorption on colloidal particles, Journal de Physique 45 (4) (1984) 725–729.

[11] A. Hanke, S. Dietrich, Critical adsorption on curved objects, Physical Review E 59 (5) (1999) 5081.

[12] I. B. Liu, N. Sharifi-Mood, K. J. Stebe, Curvature-driven assembly in soft matter, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374 (2072) (2016) 20150133.

[13] K. Goodwin, S. Mao, T. Guyomar, E. Miller, D. C. Radisky, A. Košmrlj, C. M. Nelson, Smooth muscle differentiation shapes domain branches during mouse lung development, Development 146 (22) (2019) dev181172.

[14] S.-M. Yu, B. Li, F. Amblard, S. Granick, Y.-K. Cho, Adaptive architecture and mechanoresponse of epithelial cells on a torus, Biomaterials 265 (2021) 120420.

[15] Y.-W. Chang, R. Cruz-Acuña, M. Tennenbaum, A. A. Fragkopoulos, A. J. García, A. Fernández-Nieves, Quantifying epithelial cell proliferation on curved surfaces, Frontiers in Physics 10 (2022) 1055393.

[16] A. Marín-Llauradó, S. Kale, A. Ouzeri, T. Golde, R. Sunyer, A. Torres-Sánchez, E. Latorre, M. Gómez-González, P. Roca-Cusachs, M. Arroyo, et al., Mapping mechanical stress in curved epithelia of designed size and shape, Nature communications 14 (1) (2023) 4014.

[17] M. Luciano, M. Versaevel, Y. Kalukula, S. Gabriele, Mechanoresponse of curved epithelial monolayers lining bowl-shaped 3d microwells, Advanced Healthcare Materials 13 (4) (2024) 2203377.

[18] F. Sausset, G. Tarjus, P. Viot, Tuning the fragility of a glass-forming liquid by curving space, Physical review letters 101 (15) (2008) 155701.

[19] F. Turci, G. Tarjus, C. P. Royall, From glass formation to icosahedral ordering by curving three-dimensional space, Physical Review Letters 118 (21) (2017) 215501.

[20] R. Sknepnek, S. Henkes, Active swarms on a sphere, Physical Review E 91 (022306) (2015).

[21] Y. Fily, A. Baskaran, M. F. Hagan, Active particles on curved surfaces, arXiv:1601.00324 [cond-mat.soft] (2016).

[22] S. Shankar, M. J. Bowick, M. C. Marchetti, Topological sound and flocking on curved surfaces, Physical Review X 7 (031039) (2017).

[23] T. Brandstätter, D. B. Brückner, Y. L. Han, R. Alert, M. Guo, C. P. Broedersz, Curvature induces active velocity waves in rotating spherical tissues, Nature Communications 14 (1) (2023) 1643.

[24] M. O. Lavrentovich, E. M. Horsley, A. Radja, A. M. Sweeney, R. D. Kamien, First-order patterning transitions on a sphere as a route to cell morphology, Proceedings of the National Academy of Sciences 113 (19) (2016) 5189–5194.

[25] A. Radja, E. M. Horsley, M. O. Lavrentovich, A. M. Sweeney, Pollen cell wall patterns form from modulated phases, Cell 176 (4) (2019) 856–868.

[26] M. Luciano, S.-L. Xue, W. H. De Vos, L. Redondo-Morata, M. Surin, F. Lafont, E. Hannezo, S. Gabriele, Cell monolayers sense curvature by exploiting active mechanics and nuclear mechanoadaptation, Nature Physics 17 (12) (2021) 1382–1390.

[27] L. Pieuchot, J. Marteau, A. Guignandon, T. D. Santos, I. Brigaud, P. Chauvy, T. Cloatre, A. Ponche, T. Petithory, P. Rougerie, M. Vassaux, J. Milan, N. T. Wakhloo, A. Spangenberg, M. Bigerelle, K. Anselme, Curvotaxis directs cell migration through cell-scale curvature landscapes, Nature Communications 9 (3995) (2018).

[28] E. W. Gehrels, B. Chakrabortty, M.-E. Perrin, M. Merkel, T. Lecuit, Curvature gradient drives polarized tissue flow in the drosophila embryo, Proceedings of the National Academy of Sciences 120 (6) (2023) e2214205120.

[29] F. A. Maechler, C. Allier, A. Roux, C. Tomba, Curvature-dependent constraints drive remodeling of epithelia, Journal of cell science 132 (4) (2019) jcs222372.

[30] W. Tang, A. Das, A. F. Pegoraro, Y. L. Han, J. Huang, D. A. Roberts, H. Yang, J. J. Fredberg, D. N. Kotton, D. Bi, et al., Collective curvature sensing and fluidity in three-dimensional multicellular systems, Nature Physics 18 (11) (2022) 1371–1378.

[31] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. In't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, et al., Lammps-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Computer Physics Communications 271 (2022) 108171.

[32] J. A. Anderson, J. Glaser, S. C. Glotzer, Hoomd-blue: A python package for high-performance molecular dynamics and hard particle monte carlo simulations, Computational Materials Science 173 (2020) 109363.

[33] D. M. Sussman, Interplay of curvature and rigidity in shape-based models of confluent tissue, Physical Review Research 2 (023417) (2020).

[34] E. C. Thomas, S. Hopyan, Shape-driven confluent rigidity transition in curved biological tissues, Biophysical Journal 122 (21) (2023) 4264–4273.

[35] L. Happel, D. Wenzel, A. Voigt, Effects of curvature on epithelial tissue-coordinated rotational movement and other spatiotemporal arrangements, Europhysics Letters (2022).

[36] M. Rank, A. Voigt, Active flows on curved surfaces, Physics of Fluids 33 (072110) (2021).

[37] C. L. Hueschen, A. R. Dunn, R. Phillips, Wildebeest herds on rolling hills: Flocking on arbitrary curved surfaces, Phys. Rev. E 108 (2023) 024610. `doi:10.1103/PhysRevE.108.024610`.
URL `https://link.aps.org/doi/10.1103/PhysRevE.108.024610`

[38] L. Giomi, M. J. Bowick, Elastic theory of defects in toroidal crystals, The European Physical Journal E 27 (2008) 275–296.

[39] D. P. Hardin, E. B. Saff, Minimal riesz energy point configurations for rectifiable d-dimensional manifolds, Advances in Mathematics 193 (1) (2005) 174–204.

[40] P. W. Schönhöfer, S. C. Glotzer, Curvature-controlled geometrical lensing behavior in self-propelled colloidal particle systems, Soft Matter 18 (45) (2022) 8561–8571.

[41] N. Sharp, et al., Polyscope, www.polyscope.run (2019).

[42] D. Frenkel, B. Smit, Understanding molecular simulation: from algorithms to applications, Elsevier, 2023.

[43] S. S. Schoenholz, E. D. Cubuk, Jax, md a framework for differentiable physics, Journal of Statistical Mechanics: Theory and Experiment 2021 (12) (2021) 124016.

[44] S. Mishra, A. Baskaran, M. C. Marchetti, Fluctuations and pattern formation in self-propelled particles, Physical Review E 81 (6) (2010) 061916.

[45] T. Yamamoto, D. M. Sussman, T. Shibata, M. L. Manning, Non-monotonic fluidization generated by fluctuating edge tensions in confluent tissues, Soft Matter 18 (11) (2022) 2168–2175.

[46] F. Müller-Plathe, A simple nonequilibrium molecular dynamics method for calculating the thermal conductivity, The Journal of chemical physics 106 (14) (1997) 6082–6085.

[47] B. Németh, R. Adhikari, Intrinsic langevin dynamics of rigid inclusions on curved surfaces, arXiv preprint arXiv:2405.07539 (2024).

[48] M. P. Howard, J. A. Anderson, A. Nikoubashman, S. C. Glotzer, A. Z. Panagiotopoulos, Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units, Computer Physics Communications 203 (2016) 45–52.

[49] A. Y. Toukmaji, J. A. Board Jr, Ewald summation techniques in perspective: a survey, Computer physics communications 95 (2-3) (1996) 73–92.

[50] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, O. Shochet, Novel type of phase transition in a system of self-driven particles, Physical review letters 75 (6) (1995) 1226.

[51] G. Grégoire, H. Chaté, Onset of collective and cohesive motion, Physical review letters 92 (2) (2004) 025702.

[52] A. Cavagna, L. Del Castello, I. Giardina, T. Grigera, A. Jelic, S. Melillo, T. Mora, L. Parisi, E. Silvestri, M. Viale, et al., Flocking and turning: a new model for self-organized collective motion, Journal of Statistical Physics 158 (2015) 601–627.

[53] C. Dombrowski, L. Cisneros, S. Chatkaew, R. E. Goldstein, J. O. Kessler, Self-concentration and large-scale coherence in bacterial dynamics, Physical review letters 93 (9) (2004) 098103.

[54] D. M. Sussman, cellgpu: Massively parallel simulations of dynamic vertex models, Computer Physics Communications 219 (2017) 400–406.

[55] H. D. Vuijk, H. Merlitz, M. Lang, A. Sharma, J.-U. Sommer, Chemotaxis of cargo-carrying self-propelled particles, Physical Review Letters 126 (20) (2021) 208102.

[56] A. J. Post, E. D. Glandt, Statistical thermodynamics of particles adsorbed onto a spherical surface. i. canonical ensemble, The Journal of chemical physics 85 (12) (1986) 7349–7358.

[57] R. D. Kamien, The geometry of soft materials: a primer, Reviews of Modern physics 74 (4) (2002) 953.

[58] T. Needham, Visual differential geometry and forms: a mathematical drama in five acts, Princeton University Press, 2021.

[59] R. T. Jantzen, Geodesics on the torus and other surfaces of revolution clarified using undergraduate physics tricks with bonus: nonrelativistic and relativistic kepler problems, arXiv preprint arXiv:1212.6206 (2012).

[60] K. Crane, Discrete differential geometry: An applied introduction, Notices of the AMS, Communication 1153 (2018).

[61] K. Crane, M. Livesu, E. Puppo, Y. Qin, A survey of algorithms for geodesic paths and distances, arXiv preprint arXiv:2007.10430 (2020).

[62] N. Sharp, Y. Soliman, K. Crane, The vector heat method, ACM Transactions on Graphics (TOG) 38 (3) (2019) 1–19.

[63] J. S. Mitchell, D. M. Mount, C. H. Papadimitriou, The discrete geodesic problem, SIAM Journal on Computing 16 (4) (1987) 647–668.

[64] J. Chen, Y. Han, Shortest paths on a polyhedron, in: Proceedings of the sixth annual symposium on Computational geometry, 1990, pp. 360–369.

[65] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, H. Hoppe, Fast exact and approximate geodesics on meshes, ACM transactions on graphics (TOG) 24 (3) (2005) 553–560.

[66] S.-Q. Xin, G.-J. Wang, Improving chen and han's algorithm on the discrete geodesic problem, ACM Transactions on Graphics (TOG) 28 (4) (2009) 1–8.

[67] The CGAL Project, CGAL User and Reference Manual, 5.6 Edition, CGAL Editorial Board, 2023.
URL `https://doc.cgal.org/5.6/Manual/packages.html`

[68] K. Polthier, M. Schmies, Straightest geodesics on polyhedral surfaces, in: ACM SIGGRAPH 2006 Courses, Association for Computing Machinery, 2006, pp. 30–38.

[69] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, et al., Open mpi: Goals, concept, and design of a next generation mpi implementation, in: Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users' Group Meeting Budapest, Hungary, September 19-22, 2004. Proceedings 11, Springer, 2004, pp. 97–104.

[70] S. Kiazyk, S. Loriot, É. C. de Verdière, Triangulated surface mesh shortest paths, in: CGAL User and Reference Manual, 5.6 Edition, CGAL

Editorial Board, 2023.
URL `https://doc.cgal.org/5.6/Manual/packages.html#PkgSurfaceMeshShortestPath`

[71] M. Botsch, L. Kobbelt, A remeshing approach to multiresolution modeling, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, 2004, pp. 185–192.

[72] M. J. Bowick, L. Giomi, Two-dimensional matter: order, curvature and defects, Advances in Physics 58 (5) (2009) 449–563.

[73] L. Giomi, M. J. Bowick, Defective ground states of toroidal crystals, Physical Review E 78 (1) (2008) 010601.

[74] S. Xin, P. Wang, R. Xu, D. Yan, S. Chen, W. Wang, C. Zhang, C. Tu, Surfacevoronoi: Efficiently computing voronoi diagrams over mesh surfaces with arbitrary distance solvers, ACM Transactions on Graphics (TOG) 41 (6) (2022) 1–12.

[75] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, P. Gumbsch, Structural relaxation made simple, Physical review letters 97 (17) (2006) 170201.

[76] L. Giomi, M. Bowick, Crystalline order on riemannian manifolds with variable gaussian curvature and boundary, Physical Review B 76 (5) (2007) 054106.

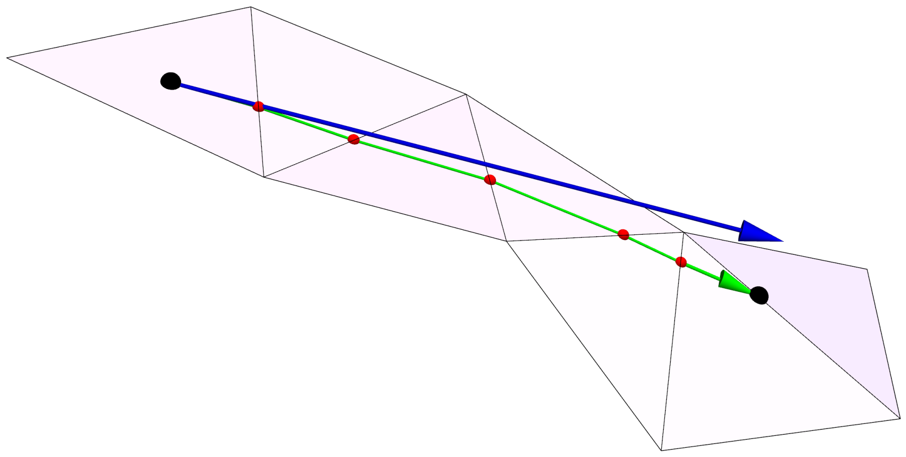

Figure 4: Operation of the `shift` algorithm: an initial position, marked by the left-most large black point, is moved by a target displacement of direction and magnitude indicated by the blue arrow. The in-surface green arrow represents the progressive rotation of the displacement vector around intersected edges in order to follow a discrete geodesic, which terminates at the right-most large right-most black point. Intersections between the path and the edges are marked by smaller red points. In-face displacements and rotations around edges continue until the total target displacement has been achieved.

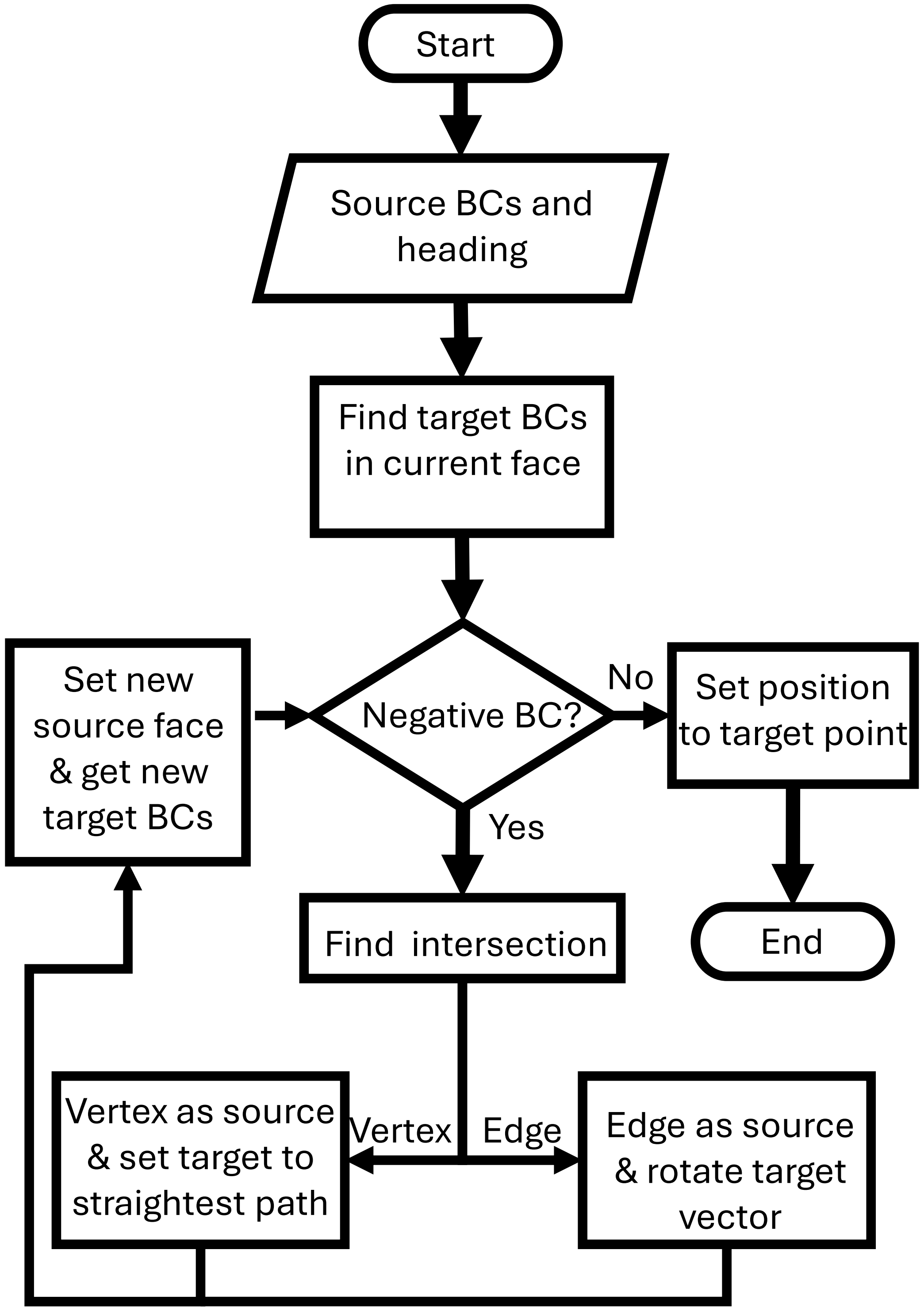
Start
Source BCs and heading
Find target BCs in current face
Negative BC?
No
Set position to target point
End
Yes
Find intersection
Set new source face & get new target BCs
Vertex
Edge
Vertex as source & set target to straightest path
Edge as source & rotate target vector

Figure 5: Depiction of the shift process, where BC stands for "barycentric coordinates." As described in the main text, how target or parallel transported vectors are rotated depends on whether the intersection is with an edge or a vertex.

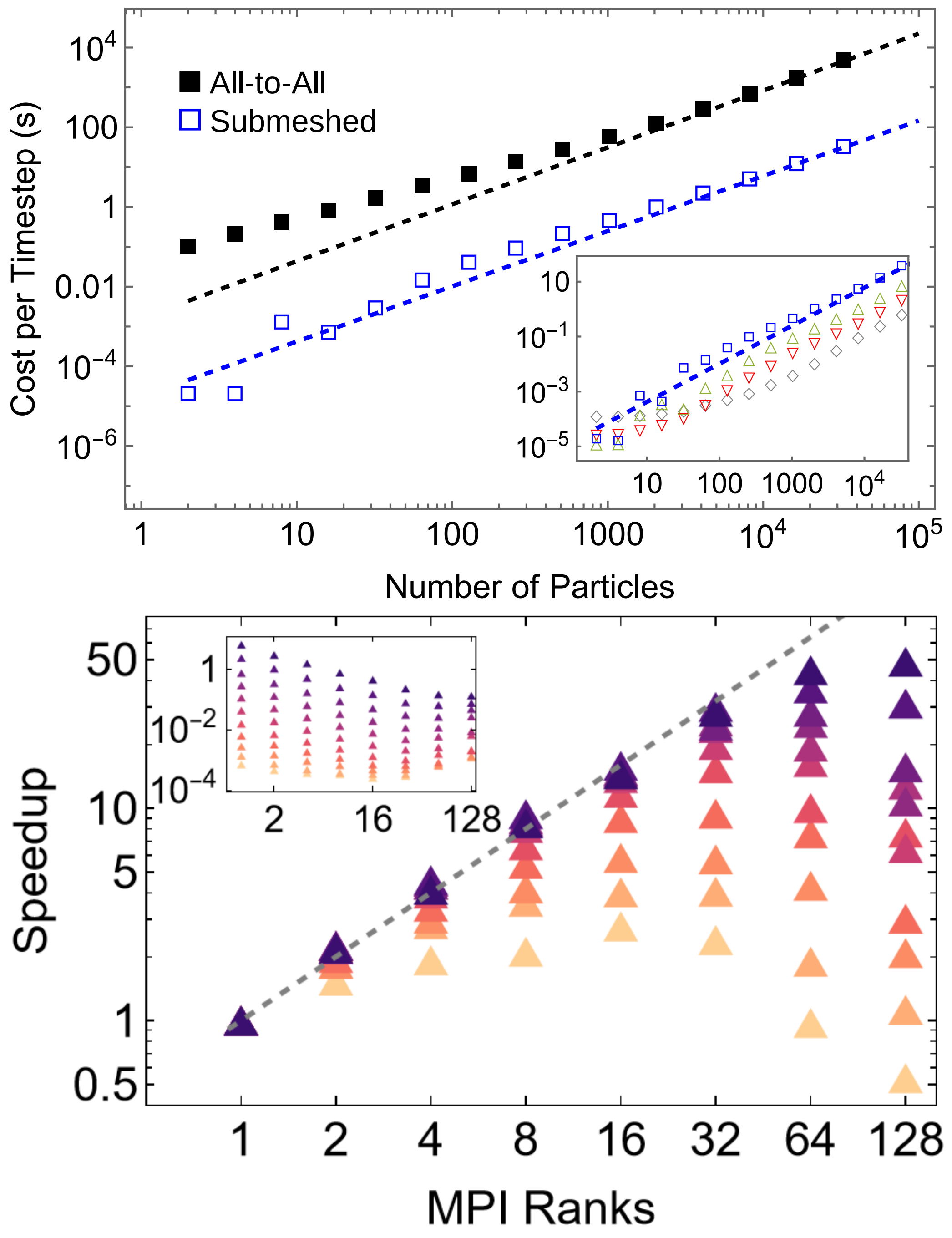

Figure 6: Simulation cost on a fixed (toroidal) mesh. (Top) The main figure shows the time per time step for a velocity Verlet timestep as a function of the number of particles on the surface. The filled black squares correspond to calculating sequence trees for the entire mesh, whereas the open blue squares correspond to using a submesh scale of $\sigma = 1$. Lines reflect fits to the asymptotic behavior of the two strategies, which both behave as power laws with scaling exponent approximately 1.4. The inset shows the simulation cost as a function of varying σ. From top to bottom the symbols correspond to $\sigma = 2^{-n}$ for $n = 0, 1, 2, 3$, with the dashed line from the main image reproduced for convenient comparison. (Bottom) Strong scaling analysis of the default MPI implementation, showing the speedup relative to single-core performance as a function of the number of MPI ranks. From light to dark, the total number of simulated particles ranges from $N = 128$ to $N = 131072$ in powers of 2. This test corresponds to implementing NVE equations of

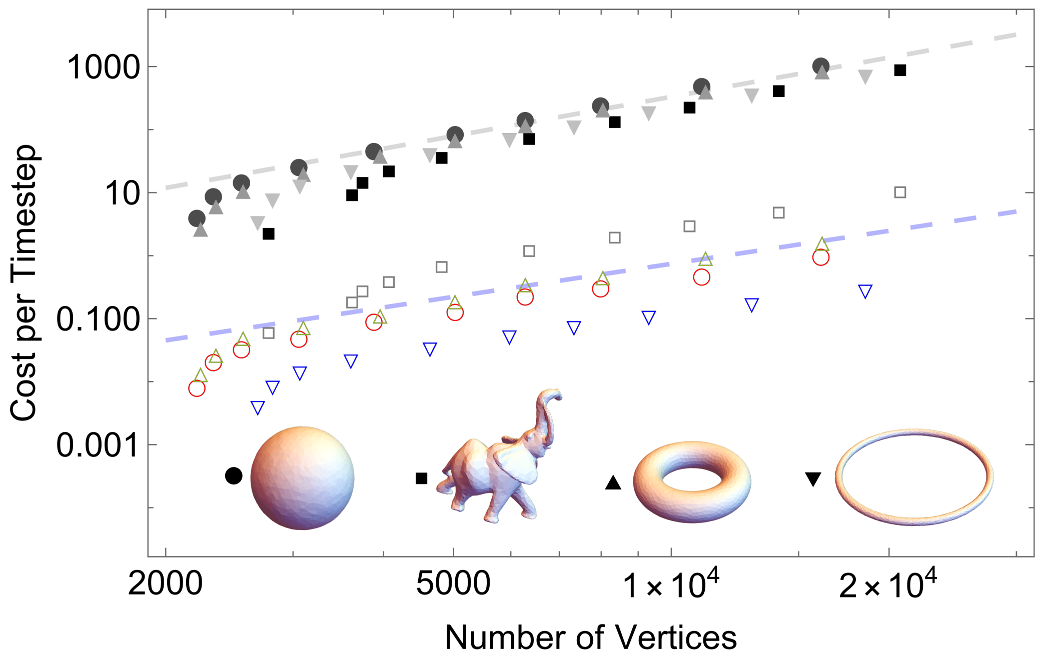

Figure 7: Mean time per NVE timestep for $N = 100$ particles on four surfaces upon isotropically increasing the number of vertices in each mesh. Solid symbols correspond to calculating sequence trees over the entire mesh for every particle, and open symbols correspond to using our submeshing routines for interactions of fixed range. For all surfaces, we chose an interaction range corresponding to one-tenth of the total span of the mesh in order to fairly compare across very different surface geometries. The upper dashed line has slope $m \approx 2.07$, while the lower line has slope $m \approx 1.73$; each is an approximate fit to the the aspect-ratio-3 torus data.

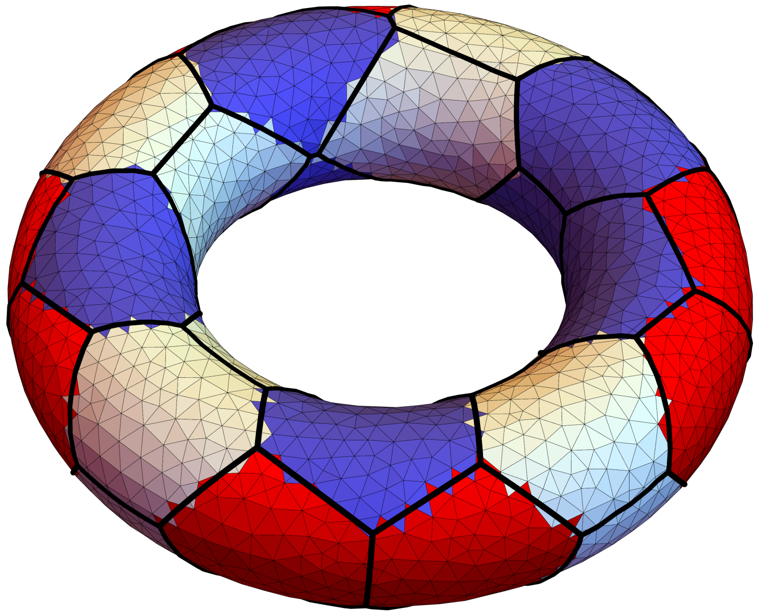

Figure 8: The Voronoi diagram of a 32-particle local energy minima on the surface of the torus found via gradient descent. The boundaries of the Voronoi regions are shown with thick black lines, and the triangles of the underlying mesh are colored according to the connectivity of the particle nearest to their centroid in the surface Voronoi diagram: red regions correspond to particles with five neighbors, blue to particles with seven, and white to particles with six. The ground state found here has 10 five-seven defect pairs, whereas the ground state configuration predicted in Ref. [38] has 16 five-seven defect pairs.

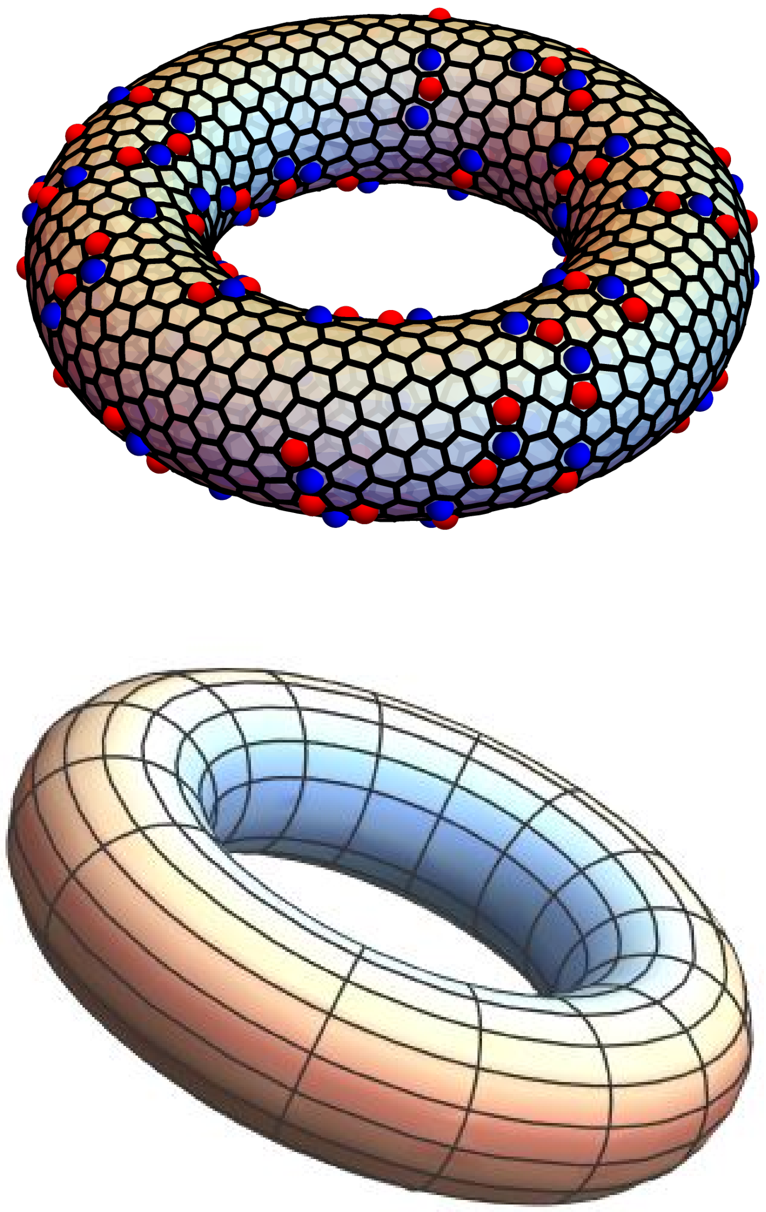

Figure 9: (Top) A relaxed configuration of 1000 particles on the surface of a torus. Here, rather than a strict localization of five-fold and seven-fold defects to the outer and inner equators of the torus as expected in the true ground state, we observe chains of defects populating the surface. This is expected from the elastic theory (described in more detail in [38]), from which we expect that larger particle numbers cause the amount of curvature required to screen defects from each other to be so large that defect chains, or “scars,” develop. (Bottom) A heat map indicating the average net defect charge across the surface (with red corresponding to an excess of five-fold defects and blue to an excess of seven-fold defects) over 1000 quenches of an $N = 500$ particle configuration. Again, the correlation between the spatial distribution of Gaussian curvature and defect distribution is as expected.

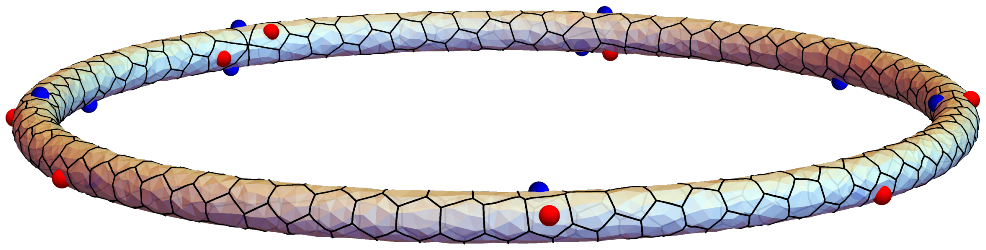

Figure 10: A low-energy configuration of $N = 300$ particles on a narrow (aspect ratio 20) torus. While gradient descent does not easily find the defectless ground state, this local minimum has relatively low defect number (11 five-fold and seven-fold defects, denoted with red and blue points, respectively).

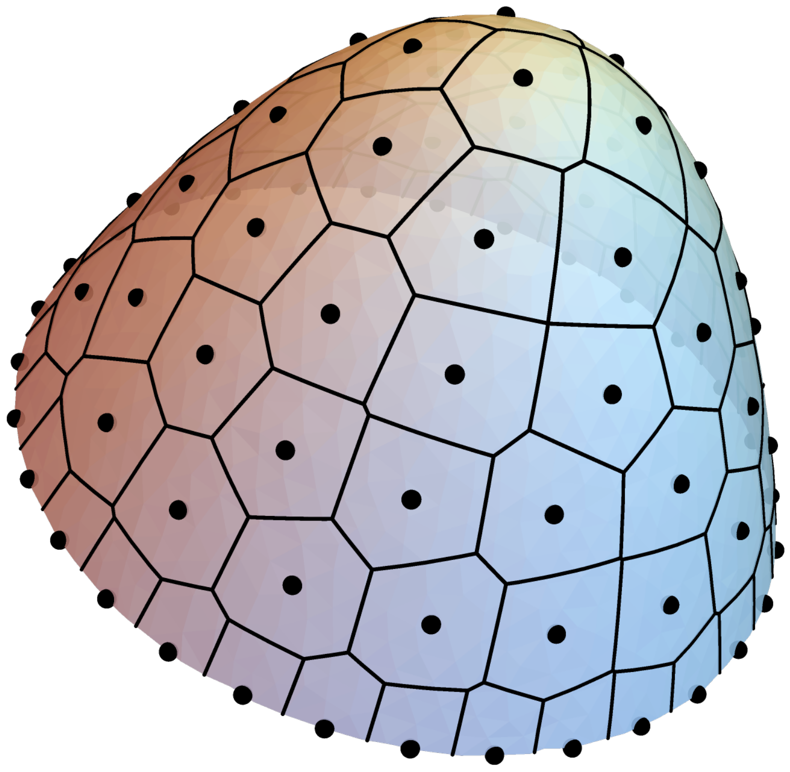

Figure 11: A representation of the surface implied by Eq. 14, here with $\Omega = 0.028$. A minimized configuration of particles interacting via soft harmonic repulsion is shown, together with the geodesic Voronoi diagram corresponding to that configuration.

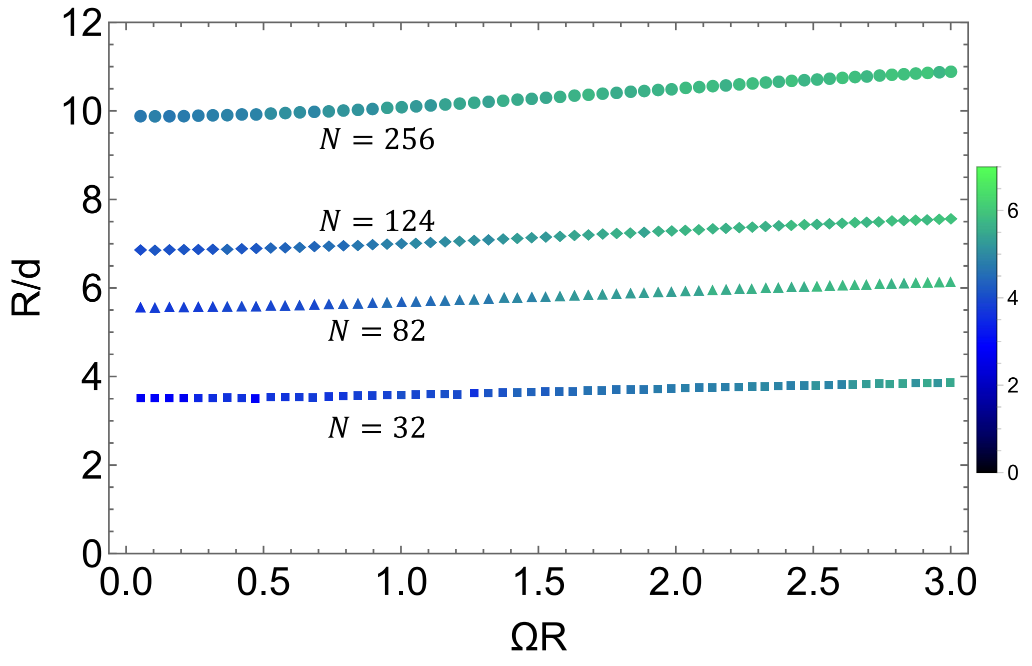

Figure 12: Mean topological charges Q found for different R/d vs. ΩR represented by a continuous color scale. As ΩR increases the mean topological charge increases, as does the charge at fixed ΩR with increasing R/d.

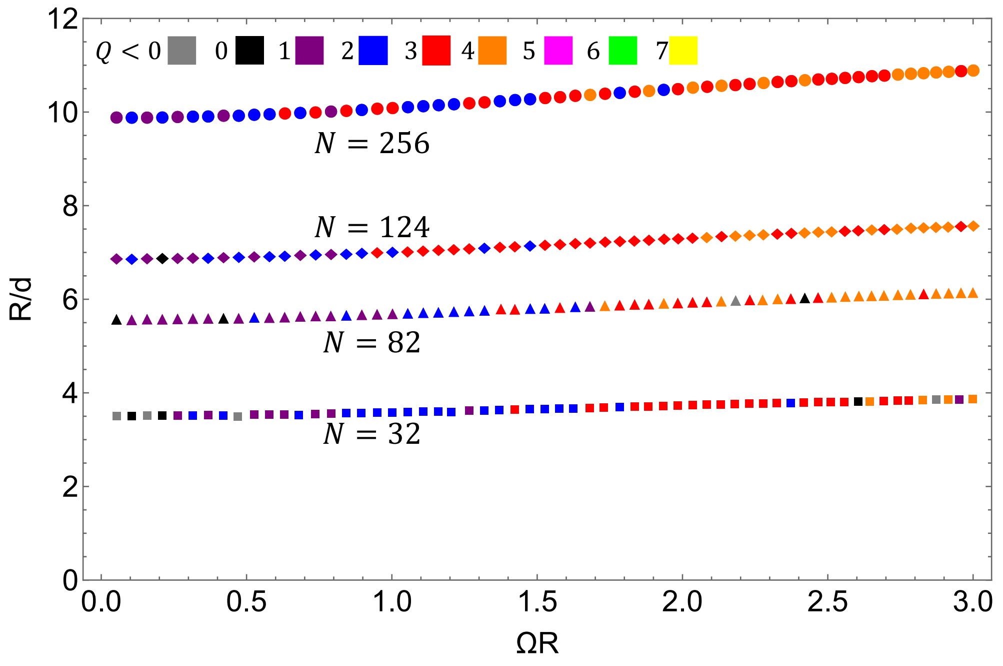

Figure 13: Minimum topological charge Q found for different R/d vs. ΩR represented by a discrete color scale. While we vary ΩR over a similar range as in Ref. [8], we note that we only infrequently observe $Q = 0$ (black markers), even in the low-twist regime.