Repulsive Energies and their Applications

Christopher Yu

CMU-CS-21-123

July 2021

Computer Science Department
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Thesis Committee:
Keenan Crane, Chair
Jessica Hodgins
Jim McCann
Stelian Coros (ETH Zürich)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This research was sponsored by The David and Lucille Packard Foundation award 201868047; by National Science Foundation award: CCF1717320; and by a National Science Foundation Graduate Research Fellowships Program fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Computer graphics, optimization, geometry, computational design, visualization, self-intersection, hierarchical acceleration, Sobolev, gradient descent, repulsive energies

Abstract

Functionals that penalize bending or stretching of a surface play a key role in geometric and scientific computing, but to date have ignored a very basic requirement: in many situations, shapes must not pass through themselves or each other. This condition is critical, for instance, when shapes represent physical membranes (e.g. in biological simulation), physical products (e.g. in digital manufacturing), or certain mathematical objects (e.g. isotopy classes of embeddings). This thesis develops a numerical framework for the intersection-free optimization of curves and surfaces. The starting point is the *tangent-point energy*, a "repulsive energy" that effectively pushes apart pairs of points that are close in space but distant along the domain. We develop discretizations of this energy for curves and surfaces, and introduce a novel acceleration scheme based on a fractional Sobolev inner product. We further accelerate this scheme via hierarchical approximation, and describe how to incorporate a variety of constraints (lengths, areas, volumes, etc,). Finally, we explore how this machinery might be applied to problems in mathematical visualization, geometric modeling, and geometry processing.

Contents

Chapter 1

Introduction

A geometric functional assigns a real-valued score $\mathcal{E}(f)$ to each immersion $f : M \to \mathbb{R}^m$ of a surface M. Such functionals serve as regularizers in many geometric problems, helping to define a unique solution, or simply making the geometry "nicer" in some sense. For instance, in geometric modeling they are used to smoothly interpolate given boundary data [26], in mathematical visualization they can be used to endow an abstract surface with a concrete geometry [30], and in digital geometry processing they are used for e.g. hole filling [32] or denoising of measured data [43].

However, classic functionals ignore a basic requirement of many applications – namely, that surfaces should not exhibit (self-)intersections. Unintended clipping in 3D models can be visually distracting; self-intersections in mathematical visualizations such as knot embeddings may alter crucial invariants of the object; intersections in digital fabrication may render a deployable structure [88] unable to deploy correctly. With so many different application areas demanding solutions without self-intersection, it is therefore surprising that, to date, there has been little focus on interpenetration in variational geometric modeling.

In this thesis, we develop *repulsive energy functions* as an effective way to find intersection-free solutions to optimization-based curve and surface modeling problems. These energy functions are designed to approach infinity as any two elements come close to intersecting, thus presenting an infinite potential barrier to both external intersections and self-intersections. This approach bears similarities to *potential-based path planning* in robotics [49, 59], as well as short-range *penalty forces* in contact dynamics [22]; unlike the first, however, our approach does not assume a static scene and is aware of self-intersections; and unlike the second, our forces have global support and thus proactively prevent intersections rather than only reacting to dynamical collisions. In particular, we advocate for the use of the *tangent-point energy* [25], which has desirable mathematical properties and can also be efficiently evaluated; notably, the inclusion of tangent information in the energy enables it to distinguish between intersections and neighboring pairs, making it uniquely suitable for finding solutions that are free of *self*-intersections.

Conceptually, the core algorithm is quite simple: we define the tangent-point energy over all pairs of points on the domain, and run gradient descent on this objective function. A naïve implementation of this method, however, would be unusable for two reasons: (1) the objective function is an "all-pairs" energy that would be unacceptably slow to evaluate, and (2) a standard line search strategy will experience extremely slow convergence due to the presence of high-order spatial derivatives in the gradient flow equation. To address the first point, we will develop hierarchical acceleration techniques, including an approximate gradient evaluation technique based on the Barnes-Hut algorithm [9]. To address the second, we will develop a novel *fractional Sobolev* preconditioner, which enables large gradient steps to be taken by reducing the order of the partial differential equation underlying the gradient flow.

1.1 Contributions

The contributions presented in this thesis can be summarized as follows:

- We introduce the tangent-point energy of Buck and Orloff [25] as a tool for intersection-free optimization of manifold domains in geometry processing by giving discretizations for the cases of curves and surfaces in $\mathbb{R}^3$,
- We apply recent analysis by Blatt [14] and Strzelecki and von der Mosel [104] to develop a novel *fractional Sobolev-Slobodeckij inner product* that is highly effective at preconditioning gradient descent on the tangent-point energy, again with concrete discretizations for curves and surfaces,
- We develop and present a suite of numerical accelerations that achieve efficient performance in practice (e.g. Barnes-Hut, hierarchical matrices, multigrid methods) and
- We develop a host of applications of the tangent-point energy to many domains of computer graphics, including mathematical visualization, geometry processing, and computational design.

In the following sections, we discuss the scope of these contributions in more detail.

1.1.1 The Tangent-Point Energy as a Tool

The tangent-point energy is a tool that is readily applicable to intersection-free modeling and optimization of general curves and surfaces. As a simple objective function, it can be easily incorporated into a variational surface modeling context. It can either be optimized on its own to produce configurations that are "maximally" robust to intersections, or combined with other terms as a regularizer for ensuring intersection-free minimizers. This relative ease of use makes it an attractive option for many problems; in this thesis, we chiefly focus on geometry processing, mathematical visualization, and computational design, but we are certain that there are many more ways in which the tangent-point energy could prove useful.

Geometry processing. As a geometric energy, the tangent-point energy behaves similarly to the Willmore energy in that it provides some degree of curvature regularization, akin to a bending energy. It can thus be used in much the same way – for instance, to produce smoothing or surface fairing. The obvious advantage of the tangent-point energy, of course, is that it prevents intersections, and thus will not produce non-isotopic motions such as in Figure 4.3. As seen in Section 3.5 and Section 4.8, our optimization method is also highly consistent at reaching minimizers of the tangent-point energy on both curves and surfaces, and that more often than not, these minimizers appear to be global. Thus, not only does the tangent-point energy succeed at the objective of preventing intersections, but it does so while exhibiting other desirable properties, and one can be assured that adding the tangent-point energy to another optimization problem (e.g. as a regularization term) will not cause the problem to become unstable or otherwise detract from the quality of the solution.

Mathematical visualization. The ability of the tangent-point energy to be aware of intersections are a clear advantage when it comes to computing isotopies such as those in Section 3.6 and Section 4.9.1; any attempt to compute intersection-free transformations using non-repulsive energies would fail more often than not for obvious reasons. The minimizers themselves have uniform spacing and good readability (Section 3.6.3, Section 4.9.1), and we can further enhance readability by encouraging them to reside primarily in the 2D plane (Figure 4.14), making them suitable for use as illustrative mathematical diagrams. Considering our energy is the first to be able to reliably preserve the isotopy class of its input, we expect that there are many more "never-before-seen" examples that could now be produced – for instance, minimizers in isotopy classes that not been computed before (e.g. Figure 4.13), or animations that previously had only existed as hand-drawn series of still frames (Figure 4.15).

Computational design. A key property of the tangent-point energy is its global support; in other words, all pairs of points exert nonzero influence on each other regardless of distance. As such, the tangent-point energy can effectively express the global absence of self-contacts and intersections as a design objective. In conjunction with the varied set of constraints and potentials that we have explored on both curves and surfaces, the energy can thus be used to mimic natural phenomena, such as the tendency of some plants to maximize surface area within fixed volumes (Figure 4.22). Of course, this kind of generative modeling is not limited to natural phenomena, and many interesting shapes can be obtained by imposing growth or shrinkage constraints within various domains (Figure 3.19, Figure 3.20, Figure 4.19). Beyond gradient optimization, the tangent-point energy also shows potential in interactive editing contexts (Section 4.9.2). As a general-purpose objective that can be readily combined with others, the tangent-point energy can be used in many ways; as with many design tools, the possibilities are limited primarily by time and imagination.

1.1.2 Fractional Sobolev Gradient Descent

One way to improve the effectiveness of standard gradient descent is to compute a gradient direction with respect to an inner product other than the standard L^2 inner product. One common approach in particular is known as *Sobolev* gradient descent, wherein the inner product used is the H^1 (or *Sobolev*) inner product (whose Gram matrix is the Laplacian). Sobolev gradient descent algorithms have been used before in computer graphics for such tasks as minimal surface computation [92], surface fairing using Willmore flow [33], optimization of surface deformation energies [68, 120], and mesh improvement [28]. A feature that these prior works share in common is that the chosen inner product is either represented by the Laplacian (corresponding to H^1) or the bi-Laplacian (corresponding to H^2). Yu et al. [116] provides a thorough exposition of fractional Sobolev methods that clearly states the principle of matching the differential order of the inner product operator to that of the energy, even when this order is a fractional value.

As the tangent-point energy has fractional order, we must therefore use an inner product of fractional order to match. In this thesis, we demonstrate that using a fractional-order preconditioner is substantially more effective than even the closest integer-order preconditioner. We focus on the tangent-point energy, but we expect that fractional Sobolev methods could be useful for optimizing other energies of fractional order. In particular, using the fractional Laplacian Δ^p as a starting point, we derive a method by which a closely-related preconditioner of fractional order $2p$ can be obtained, which could be used to precondition other energies. It is worth noting that energies with fractional order tend to be "all-pairs" energies with highly non-local interaction kernels. As we will discuss next, fractional Sobolev methods therefore go hand-in-hand with our numerical acceleration techniques for these energies.

1.1.3 Optimization of All-Pairs Energies

While we only evaluate the numerical techniques developed throughout this thesis on the tangent-point energy, the same techniques could be readily adapted to other energies of global support, provided they also fall off with distance similarly. We demonstrate the utility of Barnes-Hut (originally developed for n-body simulation) for gradient-based energy minimization. Recent work on generalized winding numbers [8, 62] also evaluates a global-range function using an adaptation of the Barnes-Hut algorithm; we must also evaluate function gradients in such a way that they can be used for line search and subsequently gradient descent. By their very nature, all-pairs energy functions such as these are capable of taking into account the global configuration of a domain in a way that local energies such as area or curvature are not, and as such can express higher-level objectives beyond just local smoothness. We believe that our techniques open the door to considering and optimizing more members of this rich class of energy functions.

Barnes-Hut. Classically, the Barnes-Hut algorithm is best known as a technique for n-body simulation (e.g. in astrophysics or electrostatics); as just mentioned, more recent work has also

used the algorithm to evaluate winding numbers in free space. In both cases, only a single function is approximated: a single scalar function in the case of winding numbers, and vector-valued forces in the case of gravitation. In our case, we require both the scalar energy function and its vector-valued differential, with sufficient accuracy that line search in the approximate gradient direction will produce a sufficient decrease in the approximate energy to satisfy the Armijo condition. By carefully considering the set of terms to differentiate (Section 3.4.1, Section 4.4.2), we approximate derivatives in a way that is both accurate and efficient. Empirically, our coupled energy and gradient approximations are more than sufficient for line search purposes, and as shown by our numerical validation tests, the limiting factor on the accuracy of the energy is the quality of the normals (Section 4.8.1), rather than anything inherent to the approximation.

Hierarchical matrices. Hierarchical matrices [53] are a method for the blockwise low-rank approximation of dense matrices and the evaluation of matrix-vector products. In this thesis, we show the application of hierarchical matrices in a multi-iteration gradient optimization setting. We find that hierarchical matrix-vector products are able to maintain high accuracy and efficiency over the course of a gradient flow; in fact, in our case, per-iteration runtimes tend to decrease as the optimization continues, as near-contacts are eliminated and elements become more distant in space, enabling more aggressive hierarchical approximations. We have also shown the utility of hierarchical matrices in conjunction with other techniques for solving linear systems. In Section 3.4, we demonstrate the effectiveness of hierarchical matrix-vector products with multigrid solvers, with a new hierarchical matrix constructed at every level of the multigrid hierarchy; meanwhile, in Section 4.5.2, we show how the same hierarchical matrices can be used to produce a direct preconditioner for a GMRES solver. Importantly, these methods could be further combined; while constructing a multigrid hierarchy with a preconditioned GMRES smoother at every level would be a hefty engineering undertaking, such a system could potentially offer exceptionally rapid convergence on linear systems like ours.

1.1.4 Thesis Statement

Ultimately, this work serves to support the following statement: that the tangent-point energy, in conjunction with associated accelerations, is an efficient and useful tool for computing intersection-free curves and surfaces, with numerous applications to tasks such as modeling, design, and visualization.

In Chapter 2, we present the definition of the tangent-point energy, and then provide an overview of classical Sobolev gradient descent techniques, and present our novel fractional-order preconditioner. In Chapter 3, we present the concrete application of these techniques to the case of space curves embedded in $\mathbb{R}^3$, and in Chapter 4, we present the application of analogous techniques to surfaces embedded in $\mathbb{R}^3$.

Chapter 2

Optimization of Repulsive Energies

In this chapter, we will discuss several examples of repulsive energies, including the *tangent-point energy*, which will form the basis for the remainder of this work. We will then provide an overview of Sobolev gradient descent methods, and discuss their application to the tangent-point energy. For now, we will discuss only the continuous formulations of the energy and associated metrics; details about discretization will be deferred until Chapter 3. Portions of this chapter are based on material that previously appeared in Yu et al. [116].

Throughout this work, we will use single bars $|X|$ and brackets $\langle X, Y \rangle$ to denote the Euclidean inner product on vectors in $\mathbb{R}^n$, and reserve double bars $\|f\|$ and brackets $\langle\langle f, g \rangle\rangle$ for norms and inner products on functions. We also use $\cdot|_f$ to indicate that a quantity (e.g. an energy) is evaluated at a function f.

2.1 Background: Repulsive Energies

Broadly, on a manifold domain M with embedding $f : M \longrightarrow \mathbb{R}^n$, we can define a *repulsive energy* to be any real-valued scalar potential function $\mathcal{E}(f)$ whose value approaches ∞ "sufficiently quickly" as the embedding f approaches self-intersection. In particular, we will consider energy functions of the form

$$\mathcal{E}(f) = \int_M \int_M k(x, y)\ dx_f\ dy_f \tag{2.1}$$

where dx_f denotes the appropriate area form on f, and $k(x, y)$ is a pairwise potential function governing the interaction between any two points in the domain. We note that it is not merely sufficient for the value of $k(x, y)$ to approach ∞ pairwise; rather, the units of $\mathcal{E}(f)$ as a whole must be of the form meters^{-p} for some $p > 0$, as only then is the energy guaranteed to present an infinite potential barrier to intersections.

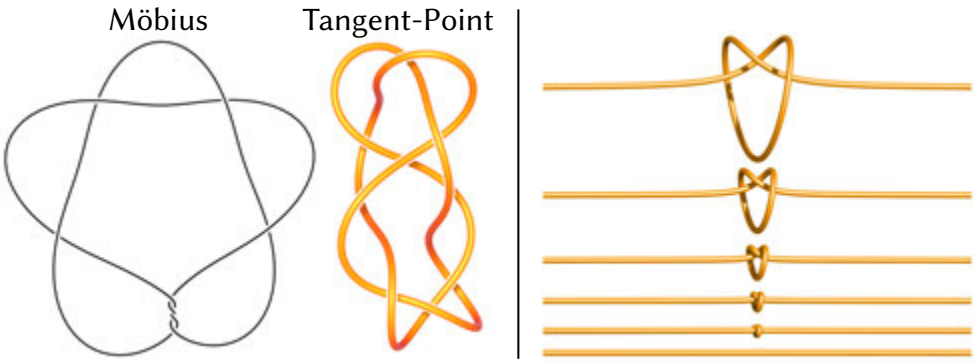


Figure 2.1: *Left:* Since the Möbius energy is scale-invariant, it allows "tight spots" where the curve nearly touches itself; such features are avoided by the tangent-point energy. *Right:* The Möbius energy can likewise artificially eliminate knots by pulling them tight at no energetic cost. (Leftmost image from Kusner and Sullivan [70].)

2.1.1 Coulomb Energy

One natural idea for defining k is to imagine that there is a uniformly positive electric charge distributed throughout the domain M. This would give rise to electrostatic repulsion governed by a Coulomb-like potential

$$k_{\text{Coulomb}}(x, y) := \frac{1}{|f(x) - f(y)|^p}. \tag{2.2}$$

The case of $p = 1$ corresponds to *Coulomb's law* in the real world. While this family of repulsive energies is well-defined for any p on a domain of isolated points, it cannot be evaluated on a continuous domain for any $p \geq 1$ due to the divergence of the resulting integrals. Intuitively, if we consider the simplest case of a one-dimensional domain M (corresponding to an embedded curve), then for any fixed point x, the inner integral of Equation 2.1 would include an ϵ-ball $B_\epsilon(x)$ for which $\int_{B_\epsilon(x)} \frac{1}{|f(x)-f(y)|^p}\, dy_f \approx \int_0^\epsilon \frac{1}{r^p}\, dr$, which diverges for $p \geq 1$.

One could attempt to repair this by restricting $p < 1$; however, the resulting energy $\mathcal{E}(f)$ would then fail to have units of inverse length for any domain M of more than zero dimensions. Considering again the simplest case of one dimension, Equation 2.1 would have units $(\text{meters}^2)(\text{meters}^{-p})$, the power of which would still be positive for any $p < 2$; in higher dimensions, the required power p only increases. Thus, there is no value of p for which a Coulomb-like energy is both well-defined and has the desired repulsive behavior. However, such energies provide a starting point for definitions of other repulsive energies, which attempt to repair these shortcomings in various ways.

2.1.2 Möbius Energy

One way to obtain a well-defined energy from a Coulomb-like starting point is to *regularize* the integrand in regions where x approaches y. One such regularization, proposed by O'Hara [85], is the *Möbius energy*, with kernel

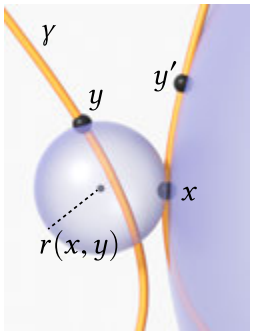
$$k_{\text{Möbius}}(x, y) := \frac{1}{|f(x) - f(y)|^2} - \frac{1}{d(x, y)^2} \tag{2.3}$$

where $d(x, y)$ denotes the shortest *geodesic* distance between x and y. Intuitively, if two points are both close in space and close intrinsically, then the geodesic distance $d(x, y)$ is roughly equal to the ambient distance $|f(x) - f(y)|$, and the contribution to the integral thus vanishes. If they are close in space but *distant* intrinsically, however, then the geodesic distance is much greater than the ambient distance, and the $\frac{1}{|f(x)-f(y)|^2}$ term therefore dominates. Assuming that f is indeed an embedding and thus free of self-intersections, then no singularity of the form $1/x$ is ever integrated, and the energy is always finite – though it will still grow arbitrarily large as self-intersections draw closer.

The Möbius energy was originally designed specifically for the case of one-dimensional curves, and the energy is known to be invariant to Möbius transformations [47]. This can be attractive in the context of knot theory, but it causes problems for computational design, since near-intersections may not be penalized in a natural way (Figure 2.1). One could attempt to remove this invariance, but this does nothing to address a second problem, which is that evaluating the energy requires the computation of geodesic distances. On a curve, this is fairly straightforward, but on surfaces and higher-dimensional domains, computation of all-pairs shortest geodesic distances quickly becomes a difficult and expensive task. While one could attempt to address this issue as well, we will instead propose an equally effective alternative that suffers from neither of these problems.

2.1.3 Tangent-Point Energy

The *tangent-point energy* was first proposed by Buck and Orloff [25]; it was originally defined only on curves, but generalizes easily to higher-dimensional manifolds. On a differentiable manifold M with embedding $f : M \longrightarrow \mathbb{R}^n$, the pairwise interaction kernel can be written as



$$k_f^p(x, y) := \frac{1}{r_f(x, y)^p} \tag{2.4}$$

where $r_f(x, y)$ is the radius of the smallest $(n - 1)$-sphere tangent to the domain at x and passing through y. This radius is called the *tangent-point radius*, and is what lends the energy its name. As with the Möbius energy, the inclusion of this radius acts to regularize the kernel in nearly intersecting regions. The radius shrinks to zero for points y that are close to x in space but far from x along the manifold itself, causing

the kernel to approach ∞. For points y' close to x *along* the curve, however, the radius is very large, causing the kernel to shrink to 0. (See inset for an illustration in the case of curves.) The power $p \in \mathbb{R}$ controls the strength of repulsion; in general, a higher p produces stronger repulsive forces.

We can write this energy more explicitly by noting that, up to a constant factor, the tangent-point radius can be written as

$$r(x, y) = \frac{|f(x) - f(y)|^2}{|\operatorname{proj}_\perp(f(x) - f(y), T_f(x))|} \tag{2.5}$$

where $T_f(x)$ denotes the local tangent space at x under the embedding f, and $\operatorname{proj}_\perp(v, T)$ denotes the projection of v onto the orthogonal complement of T. This expression agrees with the geometric intuition: for any point x, if y is sufficiently close to x along the manifold, then it nearly lies within the tangent space at x, and the denominator thus approaches 0.

Substituting Equation 4.1 into Equation 2.4 gives a concrete expression for the kernel:

$$k_f^p(x, y) := \frac{|\operatorname{proj}_\perp(f(x) - f(y), T_f(x))|^p}{|f(x) - f(y)|^{2p}}. \tag{2.6}$$

We call this kernel the *tangent-point kernel*. The resulting energy is known as the *generalized tangent-point energy* [15], and can be written as

$$\mathcal{E}^p(f) := \int_M \int_M k_f^p(f(x), f(y))\, dx_f\, dy_f.$$

In general, the energy does not necessarily need to have exponents $(p, 2p)$ in the numerator and denominator respectively; as shown by Blatt [14] it is well-defined for any exponents (α, β) satisfying $\alpha > 1$ and $\beta \in [\alpha + 2, 2\alpha + 1)$. Further, as long as $\beta - \alpha > 2$, the energy is not scale-invariant, and hence avoids the pull-tight phenomenon. For most purposes, however, enforcing the relationship $\beta = 2\alpha$ provides ample control over the strength of the repulsion. Throughout this work, $\mathcal{E}^p$ will refer to the energy with exponents $(p, 2p)$, while $\mathcal{E}_\beta^\alpha$ will refer to the energy with exponents (α, β).

The tangent-point energy is also attractive for design because it provides a natural regularization on the curvature of the domain, akin to a bending energy. This is because the integrand can vanish only for a straight line (where the radius r is infinite at every point). The power p also has an impact on this bending behavior: a higher p gives a more repulsive energy where curves are willing to bend more in order to avoid intersection (Figure 2.2).

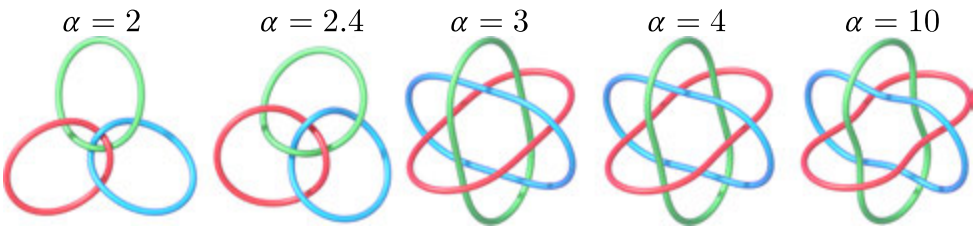


Figure 2.2: Local minimizers of the tangent-point energy $\mathcal{E}^\alpha$. When $\alpha = 2$ the tangent-point energy is scale-invariant and can exhibit "tight spots"; for larger values of α local interactions are penalized more than distant ones.

2.2 Background: Sobolev Gradient Descent

Consider an energy $\mathcal{E}$ that depends on a function f. A typical starting point for optimization is to integrate the gradient flow

$$\tfrac{d}{dt} f = -\operatorname{grad} \mathcal{E}(f), \tag{2.7}$$

i.e. to move in the direction of "steepest descent." The efficiency of this flow, however, depends critically on the *inner product* (or equivalently, the *metric*) used to define the gradient – in other words, there are many different notions of what it means to be "steepest." Recall in particular that the *differential* $d\mathcal{E}$ describes the change in $\mathcal{E}$ due to any small perturbation u of f:

$$d\mathcal{E}|_f(u) = \lim_{\varepsilon \to 0} \tfrac{1}{\varepsilon} \left(\mathcal{E}(f + \varepsilon u) - \mathcal{E}(f) \right).$$

The *gradient* of $\mathcal{E}$ is then the unique function $\operatorname{grad} \mathcal{E}$ whose inner product with any function u gives the differential in that direction:

$$\langle\!\langle \operatorname{grad} \mathcal{E}, u \rangle\!\rangle_V = d\mathcal{E}(u). \tag{2.8}$$

Traditionally, the inner product $\langle\!\langle \cdot, \cdot \rangle\!\rangle_V$ is just the L^2 inner product

$$\langle\!\langle u, v \rangle\!\rangle_{L^2} := \int_M \langle u(x), v(x) \rangle \, dx.$$

More generally, however, one can try to pick a so-called *Sobolev inner product* $\langle\!\langle u, v \rangle\!\rangle_{H^k}$ that yields an easier gradient flow equation. Examples include the H^1 and H^2 inner products, which for a domain without boundary can be written as

$$\langle\!\langle u, v \rangle\!\rangle_{H^1} := \langle\!\langle \operatorname{grad} u, \operatorname{grad} v \rangle\!\rangle_{L^2} = -\langle\!\langle \Delta u, v \rangle\!\rangle_{L^2}, \tag{2.9}$$

and

$$\langle\!\langle u, v \rangle\!\rangle_{H^2} := \langle\!\langle \Delta u, \Delta v \rangle\!\rangle_{L^2} = \langle\!\langle \Delta^2 u, v \rangle\!\rangle_{L^2}, \tag{2.10}$$

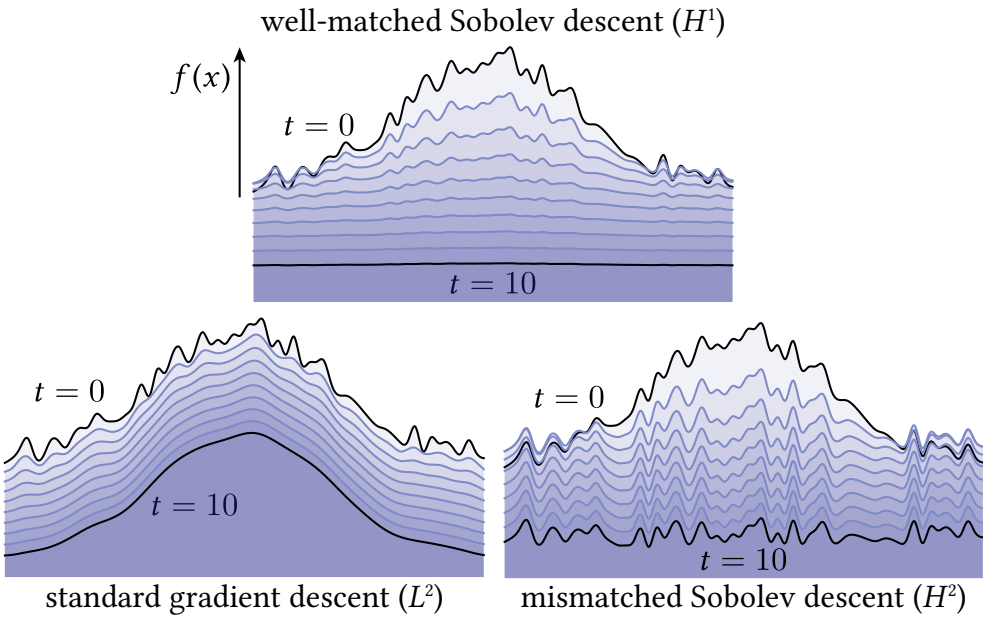


Figure 2.3: For Dirichlet energy, which penalizes variations in a function $f(x)$, standard L^2 gradient descent mostly smooths out local features *(bottom left)*, whereas an inner product that is too high-order has trouble removing high frequencies *(bottom right)*. A Sobolev descent that is well-matched to the order of the energy yields rapid progress toward a local minimizer *(top)*. We apply a similar strategy to quickly optimize the shape of curves.

which measure first and second derivatives (respectively) rather than function values. In general, if we write our inner product as $\langle\langle u, v \rangle\rangle_{H^k} = \langle\langle Au, v \rangle\rangle_{L^2}$ for some linear operator A, then we can express the new gradient direction g as the solution to

$$Ag = \operatorname{grad}_{L^2} \mathcal{E}. \tag{2.11}$$

This transformation is akin to the preconditioning provided by Newton's method, except that we replace the Hessian with an operator A that is always positive-definite, and often easier to invert. In particular, when A comes from a carefully-designed Sobolev inner product, it will eliminate spatial derivatives, avoiding the stringent time step restrictions typically associated with numerical integration of partial differential equations (Figure 2.4).

2.2.1 Tutorial: Dirichlet Energy

Since the application of Sobolev methods to the tangent-point energy is quite involved, we begin with a standard "toy" example that helps sketch out the main ideas of the approach. In particular, consider the *Dirichlet energy*

$$\mathcal{E}_D(f) := \tfrac{1}{2} \int_\Omega |\operatorname{grad} f(x)|^2 \, dx, \tag{2.12}$$

which penalizes variation in a function $f : \Omega \longrightarrow \mathbb{R}$. If the domain Ω has no boundary, then we can use integration by parts to write this energy as

$$\mathcal{E}_D(f) = \tfrac{1}{2}\langle\langle \operatorname{grad} f, \operatorname{grad} f \rangle\rangle_{L^2} = -\tfrac{1}{2}\langle\langle \Delta f, f \rangle\rangle_{L^2},$$

where Δ denotes the Laplace operator. The differential is then

$$d\mathcal{E}_D|_f(u) = -\langle\langle \Delta f, u \rangle\rangle_{L^2},$$

and from Equation 2.8, we see that the L^2 gradient of $\mathcal{E}_D$ is given by $\operatorname{grad}_{L^2} \mathcal{E}_D|_f = -\Delta f$. Hence, L^2 gradient descent yields the *heat flow*

$$\tfrac{d}{dt} f = \Delta f, \qquad (L^2 \text{ gradient flow})$$

which involves second-order derivatives in space [2, Section 1.2]. If we try to solve this equation using, say, explicit finite differences with grid spacing h, we will need a time step of size $O(h^2)$ to remain stable, which will significantly slow down computation as the grid is refined. To lift this time step restriction, we can use a different inner product to define the gradient. In particular, replacing $\langle\langle \cdot, \cdot \rangle\rangle_V$ with the H^1 inner product in Equation 2.8 yields

$$\langle\langle \Delta \operatorname{grad}_{H^1} \mathcal{E}_D, u \rangle\rangle_{L^2} = \langle\langle \Delta f, u \rangle\rangle_{L^2}. \tag{2.13}$$

This equation can be satisfied by letting $\operatorname{grad}_{H^1} \mathcal{E}_D := f$, in which case Equation 2.7 defines an H^1 gradient flow

$$\tfrac{d}{dt} f = -f. \qquad (H^1 \text{ gradient flow})$$

This flow involves no spatial derivatives, and hence comes with no time step restriction. In effect, rather than a PDE, we now have a system of independent ODEs, which is far easier to integrate numerically. As shown in Figure 2.3, the character of this flow is quite different: it makes progress by simultaneously flattening all spatial frequencies, rather than just performing local smoothing. While this approach is not appropriate for dynamical simulation, it is quite useful for finding local minima, as needed in geometric design. In general, of course, Sobolev descent is not as simple as just uniform scaling; instead, one must solve a linear PDE (Equation 2.11) for the new descent direction.

Note that we should not use an inner product with *too many* derivatives. For example, if we use the H^2 inner product (Equation 2.10) we get a gradient $\operatorname{grad}_{H^2} \mathcal{E}_D|_f = -\Delta^{-1} f$, and a flow

$$\tfrac{d}{dt} f = \Delta^{-1} f. \qquad (H^2 \text{ gradient flow})$$

This flow is again hard to integrate, and has trouble smoothing out high frequencies (Figure 2.3, *bottom-right*). In general, one cannot achieve good behavior by blindly picking a Sobolev inner product, but must instead *carefully match the inner product to the energy.*

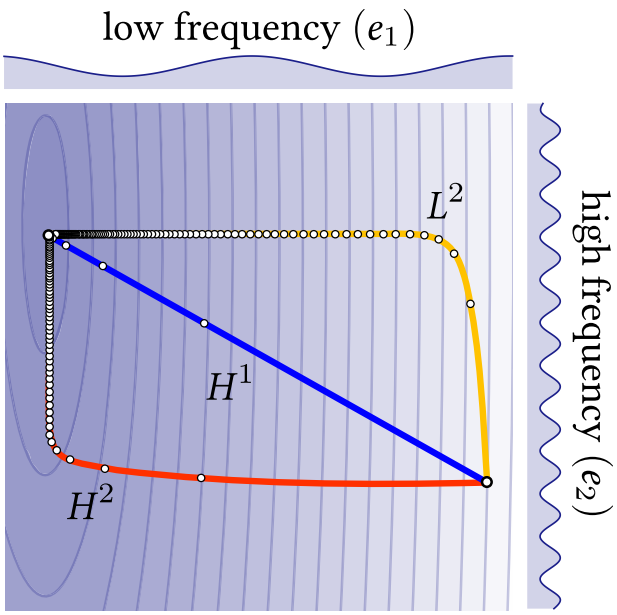


Figure 2.4: Gradient flows projected onto a low- and high-frequency mode e_1, e_2, respectively. Notice that poor preconditioning leads to slow convergence.

Low-Order Terms One remaining issue is that Equation 2.13 determines the H^1 gradient only up to functions in the null space of Δ. This situation is problematic, since it means we cannot obtain a gradient by solving Equation 2.11 directly (with $A = -\Delta$). Instead, we must include *low-order terms* that make the overall operator A invertible. For instance, we could let $A := -\Delta + \mathrm{id}$, where id denotes the identity. But if we uniformly scale the domain by a factor $c > 0$, the new operator becomes $-\frac{1}{c^2}\Delta + \mathrm{id}$, and the character of the flow changes substantially: when c is small it looks like the H^1 flow; when c is large, it looks more like the L^2 flow. Careful treatment of regularization and scaling will therefore be an important consideration in the application of Sobolev methods to the tangent-point energy.

2.3 Optimizing the Tangent-Point Energy

As with the Dirichlet energy, the gradient of the tangent-point energy $\mathcal{E}$ includes relatively high-order spatial derivatives, which hamper the convergence of standard L^2 gradient descent. Unlike the Dirichlet energy, which includes spatial derivatives of integer order 2, however, the tangent-point energy includes spatial derivatives of *fractional* order, due to its substantially non-local character. As such, a differential operator of *fractional* order is also necessary.

In general, suppose an energy $\mathcal{E}$ has a (Fréchet) differential $d\mathcal{E}$. To determine the highest-order derivatives, it is not necessary to derive an explicit expression for $d\mathcal{E}$ as we did for the Dirichlet energy (Section 2.2.1). Instead, we can reason about the associated function spaces: as long as we know the *order* of $d\mathcal{E}$, we can "cancel" spatial derivatives by constructing an inner product of the same order.

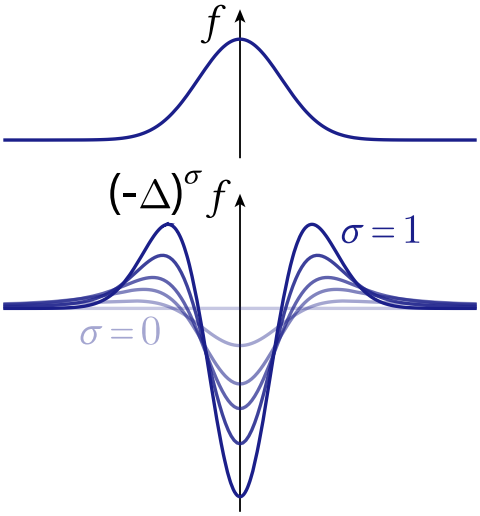


Figure 2.5: Fractional Laplacian of f for several values of σ.

For the tangent-point energy, existing analysis gives the maximum order of derivatives in $\mathcal{E}$ (Section 2.3.2), from which we deduce the fractional order of $d\mathcal{E}$ (Section 2.3.3). To build an inner product of matching order, we start with the *fractional Laplacian* (Section 2.3.1), and formulate an analogous operator for embedded manifolds. Taking further (integer) derivatives then yields an operator of the same order as $d\mathcal{E}$ (Section 4.2.2). From there, we use additional heuristics (inspired by numerical experiments) to choose a low-order term that makes this operator well-behaved and invertible (Section 2.3.4).

2.3.1 Fractional Analysis

We begin with a brief discussion of Sobolev spaces of *fractional* order $k \notin \mathbb{Z}$; for further background, see [37].

Fractional Differential Operators

Whereas standard differential operators L are purely *local* (i.e. the value of $(Lu)(x)$ depends only on an arbitrarily small neighborhood of $u(x)$), fractional differential operators are *nonlocal* (i.e. $(Lu)(x)$ can depend on the value of u at any point y). Since the tangent-point energy is nonlocal, it will also have nonlocal derivatives. Hence, finding an inner product well-matched to its gradient flow entails constructing an appropriate fractional differential operator—an important example in our setting is the *fractional Laplacian* $(-\Delta)^\sigma$ on $\mathbb{R}^n$, which is commonly defined by taking powers of the eigenvalues in the spectral expansion. For $\sigma \in (0, 1)$ and all sufficiently regular $u, v : \mathbb{R}^n \longrightarrow \mathbb{R}$, the operator can also be expressed via the integral

$$\langle\langle (-\Delta)^\sigma u, v \rangle\rangle = C \iint_{\mathbb{R}^n \times \mathbb{R}^n} \frac{u(x) - u(y)}{|x - y|^\sigma} \frac{v(x) - v(y)}{|x - y|^\sigma} \frac{dx\,dy}{|x - y|^n}, \tag{2.14}$$

where the constant $C \in \mathbb{R}$ depends only on n and σ [71]. The behavior of this operator is illustrated in Figure 2.5.

Fractional Sobolev Spaces

There are two common ways to understand Sobolev spaces of fractional order. One is to consider the Fourier transform of the Laplacian Δ, leading to the *Bessel potential spaces* $H^{s,p} := (-\Delta)^{-s/2}(L^p)$ [108, Section 2.2.2]. For us, however, this viewpoint helps only to understand the case $W^{s,2}$. The other, essential for studying the tangent-point energy, is via the *Sobolev-Slobodeckij spaces* $W^{k+\sigma,p}$. Functions u in these spaces look like functions in an ordinary Sobolev space, but with a nonlocal regularity condition on the highest-order derivative $u^{(k)}$. In particular, suppose we write $s = k+\sigma$ for $k \in \mathbb{Z}_{\geq 0}$ and $\sigma \in (0,1)$. Then, on an n-dimensional Riemannian manifold M, one defines

$$W^{k+\sigma,p}(M) := \left\{ u \in W^{k,p}(M) \,\middle|\, [u^{(k)}]_{W^{\sigma,p}} < \infty \right\}.$$

The expression in square brackets is the (Gagliardo) semi-norm

$$[u]_{W^{\sigma,p}} := \left(\iint_{M^2} \left| \frac{u(x)-u(y)}{d(x,y)^\sigma} \right|^p \frac{dx\, dy}{d(x,y)^n} \right)^{1/p},$$

where $d(x,y)$ is the shortest distance between x and y in M. Just as a Lipschitz function is more regular than an arbitrary continuous function without being differentiable, a function in $W^{k+\sigma,p}$ is more regular than one in $W^{k,p}$, without getting a whole additional derivative (i.e. $W^{k+1,p} \subsetneq W^{k+\sigma,p}$). Figure 2.6 shows an example.

Dual Space

Just as the dual of the classical Sobolev space $W^{k,p}$ is $W^{-k,q}$ (where $1/p + 1/q = 1$), the dual of the Sobolev-Slobodeckij space $W^{s,p}$ can be characterized as a space with "$-s$ derivatives", in the

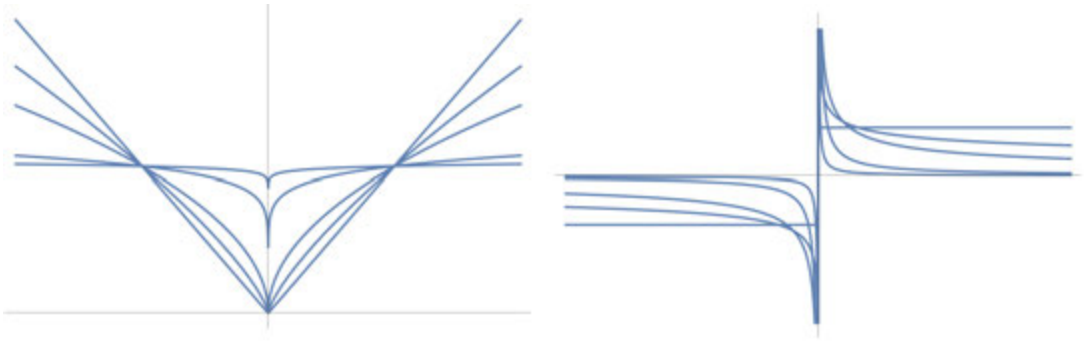


Figure 2.6: The curves $(x, |x|^\sigma)$ are examples of curves in $W^{\sigma,p}$ *(left)*. Their 1st derivatives are not L^p integrable *(right)*.

sense that the fractional Laplacian $(-\Delta)^s$ identifies $W^{s,p}$ with $W^{-s,q} := (W^{s,p})^*$ [37, Remark 2.5].

2.3.2 Energy Space

To determine the order of the tangent-point differential $d\mathcal{E}^p$, we must first know the biggest space of functions for which the energy $\mathcal{E}^p$ is well-defined, i.e. the domain of $\mathcal{E}^p$. Blatt [14, Theorem 1.1] gives the following condition on the differentiability of f (see also Blatt and Reiter [15]):

Lemma 2.3.1. *Let M be a compact, differentiable n-dimensional manifold with embedding $f : M \longrightarrow \mathbb{R}^3$, and let $p > 2n$. Then f has finite tangent-point energy $\mathcal{E}^p(\gamma)$ if and only if $f \in W^{s,p}$ where $s = 2 - \frac{n}{p}$.*

In other words, the tangent-point energy is well-defined only for embeddings that have an sth derivative, and for which the αth power of that derivative is integrable—for example, it will not be finite for a polygonal curve. Thus, we can consider the energy to be a function $\mathcal{E}^p : W^{s,\alpha} \longrightarrow \mathbb{R}$, which will enable us to deduce the order of its differential. The somewhat unusual situation is that s is not an integer: instead, it is a fractional value in the interval $(1, 2)$.

Note that, for energies $\mathcal{E}^\alpha_\beta$ where the relationship $\beta = 2\alpha$ is not enforced, following the same analysis of Blatt [14] results in the differentiability value $s = (\beta - n)/\alpha$.

2.3.3 Order of the Differential

In general, if an energy $\mathcal{E} : X \longrightarrow \mathbb{R}$ is defined for functions in a space X, then its differential $d\mathcal{E}$ will have the prototype $d\mathcal{E} : X \longrightarrow X^*$, where X^* is the dual space. For instance, the Dirichlet energy $\mathcal{E}_D$ operates only on functions $f \in H^1$. Hence, its differential is a map $d\mathcal{E}_D : H^1 \longrightarrow (H^1)^*$, which we saw explicitly in Section 2.2.1: given a function $f \in H^1$, evaluating its differential $d\mathcal{E}_D|_f$ at any f produces a linear map $\langle\langle -\Delta f, \cdot \rangle\rangle$ from functions in H^1 to real numbers, and this map is by definition a member of $(H^1)^*$.

In the case of the tangent-point energy, we similarly observe that $d\mathcal{E}$ is a map from $W^{s,p}$ to the dual space $(W^{s,p})^* = W^{-s,q}$ (Section 2.3.1). Hence, $d\mathcal{E}$ is a "differential operator" of order $2s$, i.e. it reduces the differentiability of its argument by $2s$. To get a well-behaved flow, we should therefore pick an inner product of the same order that (for computational purposes) is also reasonably easy to invert.

2.3.4 Fractional Inner Product

Just as one uses the Laplace operator Δ to define integer Sobolev inner products, we use a fractional operator to define a fractional Sobolev inner product. For a manifold M with embedding $f : M \longrightarrow \mathbb{R}^3$, one idea is to start with the fractional Laplacian $(-\Delta)^\sigma$. While one could attempt to build this fractional Laplacian using an explicit Fourier transform, this would be prohibitively expensive, requiring a full eigendecomposition of a discrete Laplace matrix. Instead, we can

construct or apply the fractional Laplacian using an integral expression such as Equation 4.5. While Equation 4.5 defines only the operator in Euclidean space $\mathbb{R}^n$, we can adapt it to an embedded manifold M by replacing the intrinsic distances $|x-y|$ in the denominators with extrinsic distances $|f(x) - f(y)|$, yielding an analogous integral expression

$$\langle\langle L_\sigma u, v \rangle\rangle := \iint_{M^2} \frac{u(x) - u(y)}{|f(x) - f(y)|^\sigma} \frac{v(x) - v(y)}{|f(x) - f(y)|^\sigma} \frac{dx_f\, dy_f}{|f(x) - f(y)|^n} \tag{2.15}$$

for all sufficiently regular scalar fields $u, v : M \longrightarrow \mathbb{R}$. For any $\sigma \in (0, 1)$, both $(-\Delta)^\sigma$ and L_σ are fractional operators of order 2σ. But the benefit of L_σ is that it requires only Euclidean distances, which in general are easier to evaluate than geodesic distances. In contrast, integral expressions like Equations 4.5 and 2.15 are straightforward (if still expensive) to evaluate exactly, and can also be accelerated using hierarchical techniques, as we will see in Chapter 3.

High-Order Term

While it would be simplest to use Equation 2.15 to construct an inner product for $d\mathcal{E}$ directly, this is not immediately possible. As per Kwaśnicki [71], Equation 4.5 (and subsequently Equation 2.15) only holds true for $\sigma \in (0, 1)$, yielding an operator of order $2\sigma \in (0, 2)$, whereas the order of the differential of the tangent-point energy is $2s \in (2, 4)$. However, we can achieve the desired order $2s$ by further composing L_σ with (integer) derivatives $\boldsymbol{D}$. In particular, if we let $\sigma = s - 1$, then we will have $\sigma \in (0, 1)$ (as is necessary to apply Equation 2.15), and the resulting operator order will be $2s - 2$. We then boost the order by an additional 2 by applying derivatives to the arguments of L_σ, resulting in the product

$$\langle\langle B_\sigma u, v \rangle\rangle := \langle\langle L_\sigma \boldsymbol{D} u, \boldsymbol{D} v \rangle\rangle$$

for all sufficiently regular $u, v : M \longrightarrow \mathbb{R}$. The resulting integral expression for the operator is

$$\langle\langle B_\sigma u, v \rangle\rangle := \iint_{M^2} \left\langle \frac{\boldsymbol{D}u(x) - \boldsymbol{D}u(y)}{|f(x) - f(y)|^\sigma}, \frac{\boldsymbol{D}v(x) - \boldsymbol{D}v(y)}{|f(x) - f(y)|^\sigma} \right\rangle \frac{dx_f\, dy_f}{|f(x) - f(y)|^n} \tag{2.16}$$

where again $\sigma = s - 1$. The resulting operator order is $2\sigma + 2 = (2s - 2) + 2 = 2s$, matching that of the differential of E^p exactly. We call this the "*high-order term*" to distinguish it from the other term of the full inner product that we will eventually use. Note that, like the Laplacian, B_σ is only *semidefinite*, since it vanishes for functions that are constant over each component of the domain M. Therefore, some regularization is necessary to make the full operator invertible, which we discuss in the next section.

Low-Order Term

As mentioned in Section 2.2.1, a common technique to handle semidefiniteness is to add some small $\epsilon > 0$ times the identity to the inner product. This, however, can yield undesirable behavior, because the inner product B_σ does not scale similarly to the identity. Instead, we carefully

choose an additional, low-order term B^0_σ that not only provides the right scaling behavior, but also enables us to steer the flow more quickly toward self-avoiding configurations. In particular, we add an operator $\langle\langle B^0_\sigma u, v\rangle\rangle$ defined by the integral expression

$$\langle\langle B^0_\sigma u, v\rangle\rangle := \iint_{M^2} k^2_f(x,y) \frac{u(x)-u(y)}{|f(x)-f(y)|^\sigma} \frac{v(x)-v(y)}{|f(x)-f(y)|^\sigma} \frac{dx_f\, dy_f}{|f(x)-f(y)|^n}, \tag{2.17}$$

where k^2_f is the tangent-point kernel given in Equation 2.6, with a power $p = 2$. We call this the "*low-order term*" to distinguish it from the "high-order term" presented in the previous section.

To see that B_σ and B^0_σ exhibit the same scaling behavior, consider a rescaling of the domain $f \mapsto cf$ by a factor $c > 0$, and omit the term $k^2_f(x,y)$ for the moment. The terms $1/|f(x)-f(y)|^\sigma$ and $dx_f\, dy_f/|f(x)-f(y)|^n$ scale identically in both integral expressions, so the only difference to consider is that between $u(x) - u(y)$ and $\mathcal{D}u(x) - \mathcal{D}u(y)$. The former does not scale with f, while the latter scales by a factor $1/c$. With two such factors, the high-order term scales by an additional $1/c^2$ factor compared to the low-order term. Thus, we must introduce a coefficient with $1/c^2$ scaling behavior to our regularization term to make up the difference.

We use $k^2_f(x,y)$ for two reasons: (1) it has the desired $1/c^2$ scaling behavior, and (2) adding a repulsive energy term to the inner product causes lengths in high-energy (and thus nearly self-colliding) regions to grow arbitrarily large. The latter behavior is desirable because, as the domain approaches self-intersection, the distance to the self-intersecting configuration will approach ∞ (as the energy itself also approaches ∞); thus, the continuous gradient flow will not be able to reach an actual self-intersection in a finite amount of time. In practice, this behavior manifests itself by "freezing" the domain in areas that are near self-contact, while allowing the rest of the domain to evolve until eventually the near-contact can be resolved.

Full Preconditioner

The continuous preconditioner is thus defined as the operator $\langle\langle (B_\sigma + B^0_\sigma)u, v\rangle\rangle$ with B_σ and B^0_σ as the previously-defined "high-order term" and "low-order term". Note that $B_\sigma + B^0_\sigma$ still has a kernel of *globally* constant functions, corresponding to global translations of the entire domain. This kernel can be easily removed by adding any constraint that fixes a single translation.

More importantly, B_σ and B^0_σ both have global support: any two hat functions $\hat{u}$ and $\hat{v}$ will produce a nonzero interaction, even if they are highly separated in space. A straightforward discretization of this inner product will thus result in a dense linear system, which is slow to solve. As such, efficient approximation of these operators (as well as the energy itself) will be a major focus throughout this work.

Chapter 3

Repulsive Curves

Space curves are a common tool in many aspects of computer graphics, including modeling, animation, and computational design. For many such tasks, it is essential to design *in context*, i.e. relative to the geometry of the surrounding environment. Hard boundary conditions (e.g. fixing the endpoints of a cable) provide a basic mechanism for providing context, but do not account for another fundamental requirement: physical objects cannot penetrate solid objects in the environment, nor can they intersect themselves. In some contexts, self-intersection can be avoided by detecting and resolving collisions at the moment of impact. However, forward simulation is not particularly effective at guiding shape optimization toward an intelligent design—for example, untangling a complicated knot via forward physical simulation is just as hard as trying to untangle it by hand. Here, we instead explore how a global variational approach to intersection-free modeling of curves using the tangent-point energy provides new opportunities for computational design.

Specifically, in this chapter, we develop:

- a principled discretization of the tangent-point energy on embedded curves in $\mathbb{R}^3$,
- a novel preconditioner based on the *Sobolev-Slobodeckij* inner product,
- a numerical solver that easily incorporates constraints needed for design, and
- a Barnes-Hut strategy and a hierarchical multigrid scheme for the tangent-point energy and associated preconditioner that greatly improve scalability.

We also explore a collection of constraints and potentials that enable us to apply this machinery to a broad range of applications in visualization and computational design (Section 3.6).

3.1 Related Work

We briefly review topics related to computational design of curves; Section 2.1 gives more detailed background on curve energies. At a high level, computational design of free-form curves

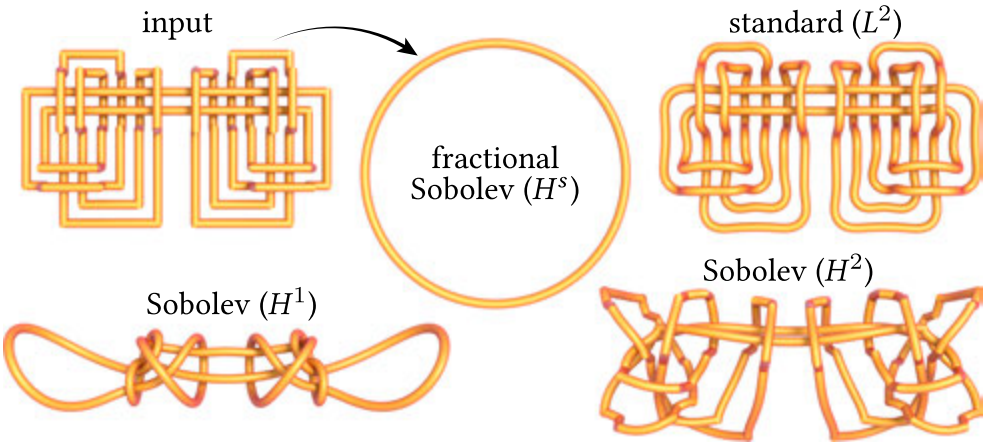


Figure 3.1: Untangling the *Freedman unknot (top left)* to the unit circle. For the same wall clock time, standard L^2 gradient descent makes almost no progress, whereas conventional Sobolev descent fails to smooth out low (H^1) or high (H^2) frequencies. By carefully matching the inner product to the energy, our fractional H^s descent quickly flows to the circle.

has generally focused on specific domains such as road networks [57, 80], telescoping structures [115], or rod assemblies [89, 118]; Moreton [82, Chapter 3] gives a history of traditional design via spline curves. Our goal is to develop tools that can be applied to a wide range of multi-objective design scenarios, as explored in Section 3.6.

3.1.1 Curve Simulation

One natural idea is to avoid collision via physics-based simulation of elastic rods [12]. However, the paradigm of collision detection and response is "too local": for computational design, one aims to globally optimize a variety of design criteria, rather than simulate the behavior of a given curve. *Sensitivity analysis*, which provides sophisticated *local* improvement of an initial design, has been successfully applied to several rod design problems [89, 90, 118]. This technique can be seen as complementary to global repulsion-based form-finding, helping to incorporate e.g. nonlinear mechanical phenomena into a final design. Curves also arise naturally as filaments or field lines in continuum phenomena like fluids, plasmas, and superfluids [3, 29, 35, 65, 87, 110]. However, using such phenomena for curve design is challenging since (i) initial conditions are hard to construct, and (ii) these systems naturally exhibit *reconnection events* where distinct pieces of a curve merge [79].

3.1.2 Knot Energies

Motivated by questions in mathematics, biology, and physics [27], there is a significant body of work on the *unknot problem*: can a closed loop be continuously deformed into a circle without passing through itself (i.e. via *isotopy*)? Solving this decision problem is not our goal—so far it is not clear it can even be done in polynomial time [72]. Still, knot untangling energies pro-

vide a valuable starting point for computational design. Numerically, simple ad-hoc methods that repel all pairs of vertices can yield inconsistent, unreliable behavior and slow convergence (Figure 3.10, *right*). Starting with more principled discretizations, *KnotPlot* [97] uses a simple relaxation scheme, and Kusner and Sullivan [70] apply a standard conjugate gradient method via *SurfaceEvolver* [20], both evaluating all $O(n^2)$ interactions between the n vertices. Other, adjacent methods have been developed for *tightening* a given knot [5, 91], simulating the knot tying process [24, 55, 69], or untangling knots without optimizing their shape [74]; more recent methods apply L^2 [109] or integer Sobolev (H^2) descent [10]. Octrees have been used to evaluate the ropelength of a static knot [4], but Barnes-Hut/multipole schemes have not yet been developed for energy minimization. Likewise, little has been said about fractional preconditioners, and treatment of general constraints.

Our approach builds on careful analysis of the fractional Sobolev spaces associated with the tangent point energy [13, 14, 15]. Whereas this prior work focuses on the existence of local minimizers and short-time existence of gradient flows in the smooth setting, we use it to develop numerical algorithms.

3.1.3 Geometric Optimization

Optimization of curve and surface energies can be greatly accelerated by "Sobolev-like" preconditioning. The idea is to replace the ordinary L^2 inner product with one that is better matched to the energy, yielding a gradient flow that is easier to integrate (as previously discussed in Section 2.2.1). Such flows make more rapid progress toward minimizers (Figure 3.1), since energy is reduced uniformly across all spatial frequencies. Crucially, Sobolev preconditioners are most effective when the *order of the preconditioner is perfectly matched to the order of spatial derivatives in the energy*. A preconditioner whose order is too high or too low can slow down convergence—see for instance Figure 2.3, *bottom-right*.

Sobolev-type preconditioners have seen some prior use in geometry processing and scientific computing. For example, the minimal surface algorithm of Pinkall and Polthier [92] effectively performs Sobolev descent [21, Section 16.10], but was not originally framed in these terms; Renka and Neuberger [95] give an algorithm directly formulated via a (variable) Sobolev inner product. Later work adopts Sobolev-like strategies for surface fairing and filtering [33, 36, 41, 78, 99]. More recently, Sobolev-like descent has become popular for minimizing elastic energies, such as those arising in surface parameterization or shape deformation [31, 68, 120]; see Section 3.5 for in-depth comparisons.

Importantly, previous work on shape optimization does not consider the challenging *fractional* case, which differs significantly from standard Sobolev preconditioning. From an analytical point of view, one must do work even to determine the order of derivatives arising in the differential by reasoning about the associated function spaces (see Section 2.3 for details). From a computational point of view, the machinery needed to apply a discretized fractional preconditioner is also different from ordinary Sobolev preconditioners: one cannot simply solve a sparse

linear system, but must instead construct an efficient hierarchical scheme for (approximately) inverting a dense nonlocal operator. None of these pieces appear in the previous optimization work discussed above, though the combination of Sobolev operators and multigrid methods has been studied in other contexts, such as finite element simulation [1] and multiphysics systems [6, 7]. Further, previously studied Sobolev preconditioners (such as those based on the Laplacian) and standard optimization strategies (such as Newton descent) are not as effective for our problem—as we show via extensive numerical experiments (Section 3.5).

3.2 Smooth Curve Optimization

In Chapter 2, we presented the tangent-point energy and an associated framework for optimizing it on general n-dimensional manifolds. In this section, we discuss the specialization of the energy and optimization method to the case of 1-dimensional curves embedded in $\mathbb{R}^3$.

3.2.1 Curve Tangent-Point Energy

Recall from Equation 2.6 the expression for the tangent-point kernel with exponent α on a general n-dimensional manifold:

$$k_f^\alpha(x, y) := \frac{|\operatorname{proj}_\perp(f(x) - f(y), T_f(x))|^\alpha}{|f(x) - f(y)|^{2\alpha}}.$$

Let M now be an embedded curve with embedding $\gamma : M \longrightarrow \mathbb{R}^3$. The tangent $T_\gamma(x)$ of a curve is uniquely defined, and length in the orthogonal complement can be computed using a cross product. Specifically, for any vector v, we have

$$|\operatorname{proj}_\perp(v, T_\gamma(x))| = |T_\gamma(x) \times v|.$$

This gives us the following expression for the tangent-point kernel on a curve:

$$k_\gamma^\alpha(x, y) := \frac{|T_\gamma(x) \times (f(x) - f(y))|^\alpha}{|f(x) - f(y)|^{2\alpha}}. \tag{3.1}$$

For convenience, it will be useful to define a version of this kernel that does not depend directly on the embedding γ, so we additionally define

$$k^\alpha(p, q, T) := \frac{|T \times (p - q)|^\alpha}{|p - q|^{2\alpha}} \tag{3.2}$$

where p, q are points in $\mathbb{R}^3$, and T is assumed to be a tangent vector to the curve at p. Then, the total energy of the curve γ can be written as

$$\mathcal{E}^\alpha(\gamma) = \int_M \int_M k_\gamma^\alpha(x, y)\, dx_\gamma\, dy_\gamma = \int_M \int_M k^\alpha(\gamma(x), \gamma(y), T_\gamma(x))\, dx_\gamma\, dy_\gamma \tag{3.3}$$

3.2.2 Curve Fractional Sobolev Preconditioner

The specialization of the fractional Sobolev inner product can be carried out in a similarly straightforward fashion. The two crucial elements of the inner product are the high-order term and the low-order term, which we discuss in sequence.

Curve High-Order Term

As discussed in Section 2.3.4, the high-order term for our fractional Sobolev preconditioner is defined on an n-dimensional manifold M with embedding f as

$$\langle\langle B_\sigma u, v\rangle\rangle := \iint_{M^2} \frac{\mathcal{D}u(x)-\mathcal{D}u(y)}{|f(x)-f(y)|^\sigma} \frac{\mathcal{D}v(x)-\mathcal{D}v(y)}{|f(x)-f(y)|^\sigma} \frac{dx_f\, dy_f}{|f(x)-f(y)|^n}$$

The only component that requires some care is the derivative operator $\mathcal{D}$, which can be defined on a curve with embedding γ as

$$\mathcal{D}u := du\, d\gamma^\top / |d\gamma|^2. \tag{3.4}$$

Intuitively, this operator takes the usual derivative of u along M and expresses it as a vector in $\mathbb{R}^3$ tangent to γ; the factor $1/|d\gamma|^2$ accounts for the fact that the curve is not in general arc-length parameterized. The remainder of the high-order term follows naturally, resulting in the integral expression

$$\langle\langle B_\sigma u, v\rangle\rangle := \iint_{M^2} \frac{\mathcal{D}u(x)-\mathcal{D}u(y)}{|\gamma(x)-\gamma(y)|^\sigma} \frac{\mathcal{D}v(x)-\mathcal{D}v(y)}{|\gamma(x)-\gamma(y)|^\sigma} \frac{dx_\gamma\, dy_\gamma}{|\gamma(x)-\gamma(y)|} \tag{3.5}$$

for all sufficiently regular $u, v : M \longrightarrow \mathbb{R}$, where $\sigma = s - 1$, and $s = 2 - \frac{1}{p}$ (following from Section 2.3.2 for dimension $n = 1$).

Curve Low-Order Term

Likewise, as discussed in Section 2.3.4, the low-order term on an n-dimensional manifold is defined as

$$\langle\langle B^0_\sigma u, v\rangle\rangle := \iint_{M^2} k_f^2(x,y) \frac{u(x)-u(y)}{|f(x)-f(y)|^\sigma} \frac{v(x)-v(y)}{|f(x)-f(y)|^\sigma} \frac{dx_f\, dy_f}{|f(x)-f(y)|^n}, \tag{3.6}$$

All terms within the expression can be specialized to curves in a straightforward way, yielding

$$\langle\langle B^0_\sigma u, v\rangle\rangle := \iint_{M^2} k^2(\gamma(x), \gamma(y), T_\gamma(x)) \frac{u(x)-u(y)}{|\gamma(x)-\gamma(y)|^\sigma} \frac{v(x)-v(y)}{|\gamma(x)-\gamma(y)|^\sigma} \frac{dx_\gamma\, dy_\gamma}{|\gamma(x)-\gamma(y)|^n}. \tag{3.7}$$

Smooth Gradient Flow

Following Equation 2.8, our final gradient $\operatorname{grad}_{H^s_\gamma}$ is defined via the fractional inner product:

$$\langle\!\langle \operatorname{grad}_{H^s_\gamma} \mathcal{E}^\alpha, X \rangle\!\rangle_{H^s_\gamma} = d\mathcal{E}^\alpha|_\gamma(X), \quad \text{for all } X : M \longrightarrow \mathbb{R}^3. \tag{3.8}$$

Since $\operatorname{grad}_{H^s_\gamma} \mathcal{E}$ and X are vector- rather than scalar-valued, we apply the inner product componentwise. In other words,

$$\operatorname{grad}_{H^s_\gamma} \mathcal{E}^\alpha = \bar{A}_\sigma^{-1} \operatorname{grad}_{L^2} \mathcal{E}^\alpha|_\gamma, \tag{3.9}$$

where $\bar{A}_\sigma$ denotes componentwise application of A_σ.

3.3 Discrete Curve Optimization

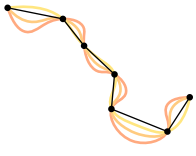
We now discretize the smooth inner product described in previous sections to obtain an efficient numerical scheme for minimizing the tangent-point energy. Our discretization operates on polygonal curves; while in principle splines could be used as in Bartels et al. [10], this makes little difference in practice due to the use of numerical quadrature in both cases. The description given here assumes a naïve implementation using dense matrices and an $O(n^2)$ evaluation of the energy and its differential; hierarchical acceleration is described in Section 3.4.

Notation. In the discrete setting, we will model any collection of curves and loops (including several curves meeting at a common point) as a graph $G = (V, E)$ with vertex coordinates $\gamma : V \longrightarrow \mathbb{R}^3$ (Figure 3.2); we use $|V|$ and $|E|$ to denote the number of vertices and edges, respectively. For each edge $I \in E$ with endpoints i_1, i_2, we use

$$\ell_I := |\gamma_{i_1} - \gamma_{i_2}|, \quad T_I := (\gamma_{i_2} - \gamma_{i_1})/\ell_I, \quad \text{and} \quad x_I := (\gamma_{i_1} + \gamma_{i_2})/2$$

to denote the edge length, unit tangent, and midpoint, respectively. For any quantity $u : V \longrightarrow \mathbb{R}$ on vertices we use $u_I := (u_{i_1} + u_{i_2})/2$ to denote the average value on edge $I = (i_1, i_2)$, and $\mathsf{u}[I] := [u_{i_1}\ u_{i_2}]^\top$ to denote the 2×1 column vector storing the values at its endpoints. Finally, we refer to any pair $(T, x) \in \mathbb{R}^6$ as a *tangent-point*.

3.3.1 Discrete Energy



Since the tangent-point energy is infinite for polygonal curves [105, Figure 2.2], we assume that γ is inscribed in some (unknown) smooth curve, and apply numerical quadrature to the smooth energy $\mathcal{E}$. The resulting discrete energy then approximates the energy of any sufficiently smooth curve passing through the vertices γ_i. We start by integrating k^α over all pairs of edges:

$$\sum_{I \in E} \sum_{J \in E} \int_{\bar{I}} \int_{\bar{J}} k^\alpha(\gamma(x), \gamma(y), T_I)\, dx_\gamma\, dy_\gamma. \tag{3.10}$$

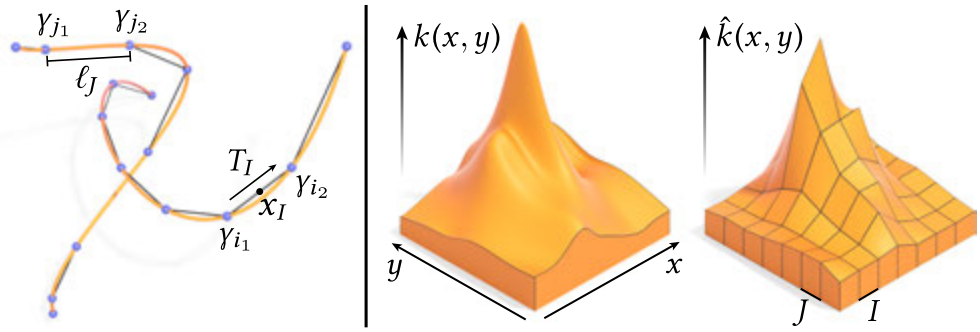


Figure 3.2: *Left:* notation used for discrete curves. *Right:* Our discrete energy is obtained by applying the trapezoidal rule to the smooth energy for each edge pair I, J.

Here $\bar{I}$ denotes the interval along edge I. As stated, this expression is ill-defined since any two edges with a common endpoint contribute infinite energy. One idea is to replace any such term with one proportional to the curvature of the circle passing through the three distinct endpoints. However, such terms would contribute nothing to the energy in the limit of regular refinement (Figure 3.3)—hence, we simply omit neighboring edge pairs. Applying midpoint quadrature to Equation 3.10 then yields a discrete energy

$$\hat{\mathcal{E}}(\gamma) = \sum_{I,J \in E, I \cap J = \emptyset} \sum k^\alpha(\gamma_I, \gamma_J, T_I)\ \ell_I\ \ell_J. \tag{3.11}$$

3.3.2 Discrete Energy Gradient

Mathematically, the discrete differential of the tangent-point energy consists simply of the partial derivatives of the discrete energy with respect to the coordinates of all the curve vertices:

$$d\hat{\mathcal{E}}|_\gamma = \begin{bmatrix} \partial\hat{\mathcal{E}}/\partial\gamma_1 & \cdots & \partial\hat{\mathcal{E}}/\partial\gamma_{|V|} \end{bmatrix} \in \mathbb{R}^{3|V|}.$$

This requires the discrete derivative of the tangent-point kernel $k^\alpha(\gamma_I, \gamma_J)$ with respect to a curve vertex position γ_i. Assuming for the moment that vertex i is adjacent to edge I, we can derive an analytic expression for this derivative by applying the chain rule

$$\frac{d}{d\gamma_i} k^\alpha(\gamma_I, \gamma_J, T_I) = \frac{d\gamma_I}{d\gamma_i} \frac{d}{d\gamma_I} k^\alpha(\gamma_I, \gamma_J, T_I)$$

Then, $\frac{d}{d\gamma_I} k^\alpha(\gamma_I, \gamma_J, T_I)$ can be evaluated via the quotient rule. Let $A = |T \times (\gamma_I - \gamma_J)|^\alpha$ and $B = |\gamma_I - \gamma_J|^{2\alpha}$ be the numerator and denominator of the tangent-point energy, respectively. Then,

we have

$$\begin{aligned}\frac{d}{d\gamma_I}A &= \frac{d}{d\gamma_I}|T_I \times (\gamma_I - \gamma_J)|^\alpha \\ &= \alpha|T_I \times (\gamma_I - \gamma_J)|^{\alpha-1}\frac{d}{d\gamma_I}|T_I \times (\gamma_I - \gamma_J)| \\ &= \underbrace{\alpha|T_I \times (\gamma_I - \gamma_J)|^{\alpha-1}\operatorname{sgn}(T_I \times (\gamma_I - \gamma_J))}_{(\ldots)}\frac{d}{d\gamma_I}(T_I \times (\gamma_I - \gamma_J)) \\ &= (\ldots)\left[(\gamma_I - \gamma_J)_\times(\frac{d}{d\gamma_I}T_I) + (T_I)_\times\right]\end{aligned}$$

where $v_\times$ denotes the skew-symmetric cross-product matrix for the vector v. Further,

$$\begin{aligned}\frac{d}{d\gamma_I}B &= \frac{d}{d\gamma_I}|\gamma_I - \gamma_J|^{2\alpha} \\ &= 2\alpha|\gamma_I - \gamma_J|^{2\alpha-1}\frac{d}{d\gamma_I}|\gamma_I - \gamma_J| \\ &= 2\alpha|\gamma_I - \gamma_J|^{2\alpha-1}\left(\frac{\gamma_I - \gamma_J}{|\gamma_I - \gamma_J|}\right)^\top\end{aligned}$$

Then, the quotient rule gives the full expression:

$$\frac{d}{d\gamma_I}k^\alpha(\gamma_I, \gamma_J, T_I) = \frac{\left(\frac{d}{d\gamma_I}A\right)B - \left(\frac{d}{d\gamma_I}B\right)A}{B^2}.$$

The expression for edge J follows analogously, though certain terms are negated due to the differentiation of $\gamma_I - \gamma_J$. Then, the partial derivatives of the full discrete energy $\partial\hat{\mathcal{E}}/\partial\gamma_i$ can be computed from derivatives of k^α and lengths l_I and l_J using the product rule. We note that only terms where vertex i is adjacent to either edge I or J produce nonzero contributions, and as such, only these terms need to be considered.

3.3.3 Discrete Inner Product

As in the smooth setting, we define our inner product matrix as a sum $\mathsf{A}_\sigma = \mathsf{B}_\sigma + \mathsf{B}^0_\sigma$ of high-order and low-order terms $\mathsf{B}, \mathsf{B}^0 \in \mathbb{R}^{|V|\times|V|}$ (as defined below). For $\mathbb{R}^3$-valued functions, we also define a corresponding $3|V| \times 3|V|$ matrix A_3, a block matrix constructed from A by expanding every scalar entry A_{ij} into a 3×3 block

$$A_{ij} \mapsto \begin{bmatrix} A_{ij} & 0 & 0 \\ 0 & A_{ij} & 0 \\ 0 & 0 & A_{ij} \end{bmatrix}$$

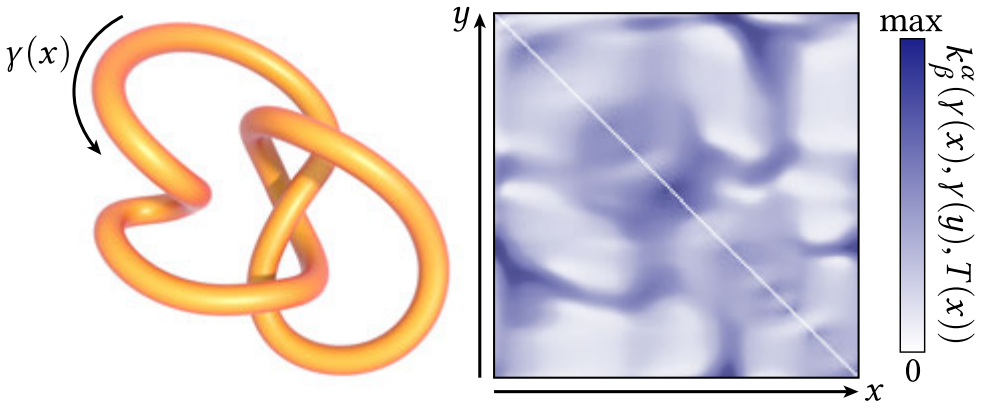


Figure 3.3: The tangent-point energy is a double integral of the kernel k^α *(right)* over the curve γ *(left)*. Since this kernel is only weakly singular, omitting diagonal terms has an insignificant effect on the overall energy.

by reduplicating the value along the diagonal. Mirroring Equation 2.11, the discrete (fractional) Sobolev gradient $\mathsf{g} \in \mathbb{R}^{3|V|}$ is then defined as the solution to the matrix equation

$$\mathsf{A}_3\, \mathsf{g} = d\hat{\mathcal{E}} \tag{3.12}$$

where the entries of g are ordered with x-, y-, and z-coordinates interleaved for each vertex.

Discrete Derivative Operator

For each edge $I \in E$ we approximate the derivative $\mathcal{D}u$ of a function $u : M \longrightarrow \mathbb{R}$ (Equation 3.4) via the finite difference formula $\frac{1}{\ell_I}(\mathsf{u}_{i_2} - \mathsf{u}_{i_1})T_I$, where u_i denotes the value of u sampled at vertex i. If needed, the corresponding derivative matrix $\mathsf{D}_\gamma \in \mathbb{R}^{3|E| \times |V|}$ can be assembled from local 3×2 matrices

$$(\mathsf{D}_\gamma)_I = \tfrac{1}{\ell_I}\begin{bmatrix} -T_I & T_I \end{bmatrix}$$

with rows corresponding to x-, y-, and z-coordinates for each edge I, and columns corresponding to the first and second vertices of the edge, respectively. Alternatively, the result $\mathsf{D}_\gamma \mathsf{u}$ can be computed directly using the aforementioned finite difference formula for each edge.

Discrete Fractional Laplacian

The fractional Laplacian can be discretized as a $|V| \times |V|$ matrix with entries obtained from the right-hand side of Equation 4.5. Let L_σ denote the discrete fractional Laplacian. A direct discretization of Equation 2.15 on a curve domain yields an expression for inner products between discrete vectors u and v:

$$\mathsf{u}^\top \mathsf{L}_\sigma \mathsf{v} = \sum_{\substack{I \in E \\ I \neq J}} \sum_{J \in E} \frac{(\mathsf{u}(x) - \mathsf{u}(y))(\mathsf{v}(x) - \mathsf{v}(y))}{|\gamma(x) - \gamma(y)|^{2\sigma+1}}\, \ell_I\, \ell_J \tag{3.13}$$

ALGORITHM 1: Exact assembly of the discrete fractional Laplacian on curves

initialize $L_\sigma \longleftarrow 0$
forall *distinct pairs of edges* I, J **do**
  forall *vertices* u *adjacent to* I *or* J **do**
    forall *vertices* v *adjacent to* I *or* J **do**
      $(L_\sigma)_{uv} \longleftarrow (L_\sigma)_{uv} + \frac{(\hat{u}(I)-\hat{u}(J))(\hat{v}(I)-\hat{v}(J))}{\|f(x)-f(y)\|^{2\sigma+1}}\ \ell_I\ \ell_J$
    end
  end
end
return A

By evaluating this expression for all pairs of unit basis functions $\hat{\mathsf{u}}$ and $\hat{\mathsf{v}}$ centered on pairs of vertices u and v, one can compute the entries of L_σ directly. This leads to a quadratic-time algorithm for assembling the exact discrete fractional Laplacian (Algorithm 1).

High-Order and Low-Order Term. The same algorithm can be used to assemble both the high-order term B_σ and low-order term B^0_σ that we will use for preconditioning the gradient flow. All that needs to be done is to replace the kernel in Algorithm 1 with the appropriate kernel for the desired inner product term. For the high-order term, this kernel is

$$\mathsf{u}^\top B_\sigma \mathsf{v} = \sum_{\substack{I\in E \\ I\neq J}} \sum_{J\in E} \frac{\langle \mathsf{D}_\gamma \mathsf{u}(x) - \mathsf{D}_\gamma \mathsf{u}(y), \mathsf{D}_\gamma \mathsf{v}(x) - \mathsf{D}_\gamma \mathsf{v}(y)\rangle}{|\gamma(x) - \gamma(y)|^{2\sigma+1}}\ \ell_I\ \ell_J \tag{3.14}$$

and for the low-order term, the kernel is

$$\mathsf{u}^\top B^0_\sigma \mathsf{v} = \sum_{\substack{I\in E \\ I\neq J}} \sum_{J\in E} k^2(\gamma_I, \gamma_J, T_I) \frac{(\mathsf{u}(x) - \mathsf{u}(y))(\mathsf{v}(x) - \mathsf{v}(y))}{|\gamma(x) - \gamma(y)|^{2\sigma+1}}\ \ell_I\ \ell_J. \tag{3.15}$$

The full inner product matrix we assemble is then $A_\sigma = B_\sigma + B^0_\sigma$, and can be constructed by assembling and adding the two terms.

3.3.4 Constraints

For design applications, we will need to impose a variety of scalar constraints $\Phi_i(\gamma) = 0$, $i = 1, \ldots, k$, which we encode as a single constraint function $\Phi : \mathbb{R}^{3|V|} \longrightarrow \mathbb{R}^k$. To enforce these constraints, we project the gradient onto a valid descent direction; after taking a step in this direction, we also project the result onto the constraint set (Section 3.3.4).

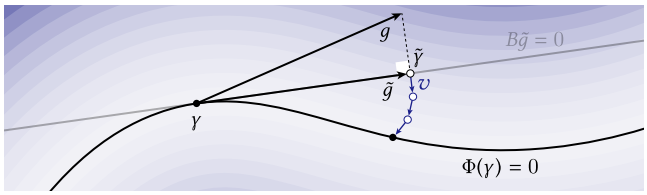


Figure 3.4: To enforce constraints $\Phi(\gamma) = 0$ on the curve, we both project the gradient g onto the tangent of the constraint set, and also apply an iterative procedure to project the curve itself back onto the constraint set. In both cases, the fractional Sobolev norm provides the definition of closeness.

Gradient Projection

Let $\mathsf{C} := d\Phi(\gamma)$ be the Jacobian matrix of the constraint, and let $\mathbf{g} := \operatorname{grad}_{H^s_\gamma} E \in \mathbb{R}^{3|V|}$ denote the unconstrained energy gradient. We seek the descent direction $\tilde{\mathbf{g}}$ that is closest to $\mathbf{g}$ with respect to the fractional Sobolev norm, but which is also tangent to the constraint set:

$$\min_{\tilde{\mathbf{g}}} \tfrac{1}{2}\|\tilde{\mathbf{g}} - \mathbf{g}\|^2_{H^s_\gamma} \quad \text{s.t.} \quad \mathsf{C}\tilde{\mathbf{g}} = 0.$$

Writing $\|v\|^2_{H^s_\gamma}$ as $v^\top \mathsf{A}_3 v$ (Section 3.3.3), we can apply the method of Lagrange multipliers to obtain the usual first-order optimality conditions, given by the saddle point system

$$\begin{bmatrix} \mathsf{A}_3 & \mathsf{C}^\top \\ \mathsf{C} & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{g}} \\ \lambda \end{bmatrix} = \begin{bmatrix} d\mathcal{E}|_\gamma^\top \\ 0 \end{bmatrix}, \tag{3.16}$$

where $\lambda \in \mathbb{R}^k$ are the Lagrange multipliers, and we have applied the identity $\mathsf{A}_3\mathbf{g} = d\mathcal{E}|_\gamma^\top$ (Equation 3.12). By solving this system, we obtain the constrained descent direction $\tilde{g}$.

Constraint Projection

Suppose that we take a small step of size τ along the projected gradient direction $\tilde{\mathbf{g}}$ to reach a new embedding $\tilde{\gamma} := \gamma - \tau\tilde{\mathbf{g}}$. In general, due to numerical drift, $\tilde{\gamma}$ will not satisfy $\Phi(\tilde{\gamma})$ exactly, and corrective projection is therefore necessary. To perform this projection, we will apply a "frozen" approximation of Newton's method. In particular, to find a displacement $\mathsf{x} \in \mathbb{R}^{3|V|}$ that takes us from $\tilde{\gamma}$ back toward the constraint set $\Phi(\mathbf{g}) = 0$, we solve the problem

$$\min_{\mathsf{x}} \tfrac{1}{2}\mathsf{x}^\top \mathsf{A}_3 \mathsf{x} \quad \text{s.t.} \quad \mathsf{C}\mathsf{x} = -\Phi(\tilde{\gamma}).$$

We then update our guess via $\tilde{\gamma} \leftarrow \tilde{\gamma} + \mathsf{x}$ and repeat until the constraint violation $\Phi(\tilde{\gamma})$ is numerically small (10^{-4} in our experiments). In practice, this process rarely takes more than three iterations. At each iteration, x is obtained by solving the saddle point problem

$$\begin{bmatrix} \mathsf{A}_3 & \mathsf{C}^\top \\ \mathsf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{x} \\ \mu \end{bmatrix} = \begin{bmatrix} 0 \\ -\Phi(\tilde{\gamma}) \end{bmatrix}, \tag{3.17}$$

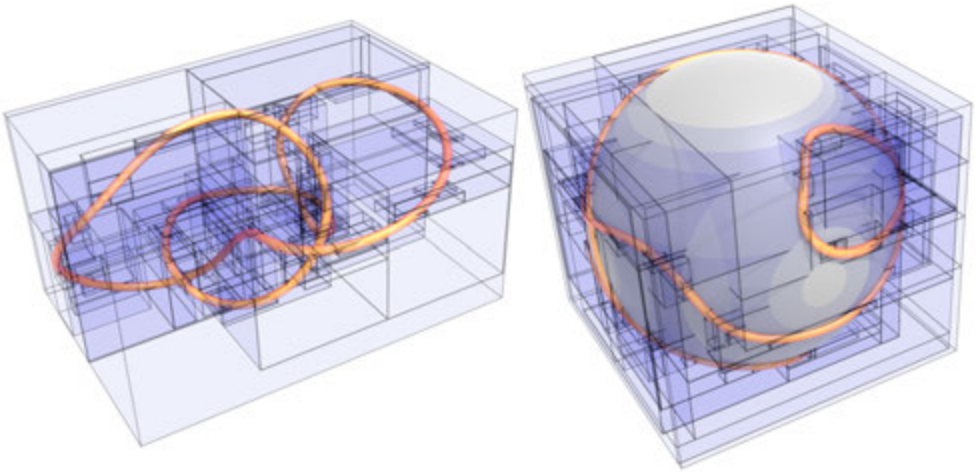


Figure 3.5: To accelerate evaluation of the tangent-point energy, we build a bounding volume hierarchy that partitions both positions *(left)* and tangent directions *(right)*, here drawn as a curve on the unit sphere.

where $\mu \in \mathbb{R}^k$ are Lagrange multipliers. We refer to this as a "frozen" approximation because the approximate Hessian (in this case, the saddle matrix) is not rebuilt after each intermediate projection, which has the advantage of allowing the same matrix (and, in the exact case, the same factorization) to be reused throughout.

3.3.5 Time Stepping

A judicious choice of time step can significantly improve the efficiency of the flow. One strategy is to use the first time step $\tau_{\max}$ at which a intersection occurs as the starting point for a line search, which guarantees that the curve remains in the same isotopy class. (Similar approaches have been used in e.g. *KnotPlot* [97] for knot untangling, and by Smith and Schaefer [101] for surface parameterization.) Computing this time step via standard techniques [94] costs about as much as a single energy evaluation, which is significantly less than the overall cost of a single time step. From here we apply standard backtracking line search [18, Algorithm 9.2]; as a heuristic, we start this search at $\frac{2}{3}\tau_{\max}$. We use this strategy throughout our runtime comparisons in Section 3.5.

An even simpler strategy that works well in practice (but comes with no intersection-preventing guarantees) is to just normalize the gradient and perform backtracking line search starting with $\tau = 1$, until both (i) the Armijo condition is satisfied and (ii) constraint projection succeeds (Section 3.3.4). We use this latter strategy for all application examples in Section 3.6. We stop when the L^2 norm of the fractional Sobolev gradient goes below a user-specified tolerance ε. In our examples we use $\varepsilon = 10^{-4}$, though of course for design applications one can also stop whenever the results are aesthetically pleasing.

3.4 Acceleration

While the naïve optimization strategy of Section 3.3 can be effective on very small examples, it rapidly becomes intractable on resolutions of only a few thousand vertices, since it involves not only an all-pairs energy (Section 3.3.1), but also a dense matrix inversion (Section 3.3.3). However, since the relevant kernels falls off rapidly in space, we can use hierarchical approximation to avoid a $\Omega(|V|^2)$ time and storage cost. Though our high-level approach is reasonably standard, careful consideration of the tangent-point energy is needed to develop a scheme that is efficient, easy to implement, and handles general nonlinear constraints. At a high level, our acceleration consists of three main parts: (1) an adaptation of Barnes-Hut to evaluate the energy and its gradient, (2) a hierarchical strategy to evaluate matrix-vector products with A_σ, and (3) a multigrid scheme for solving the saddle problem of Equation 3.17. Note that since we care only about finding a good descent direction—and not accurately simulating a dynamical trajectory—we are free to use low-order schemes, which still provide good preconditioning. Empirically, the overall strategy exhibits near-linear scaling in both time and memory (Figure 3.15).

3.4.1 Energy and Differential Evaluation

To accelerate evaluation of the energy $\hat{\mathcal{E}}$ and its differential, we apply the *Barnes-Hut algorithm* from N-body simulation [9]. The basic idea is to approximate distant energy contributions by aggregating values in a spatial hierarchy. In our case, this hierarchy must have six dimensions rather than three, since $\hat{\mathcal{E}}$ depends on both positions $\gamma \in \mathbb{R}^3$ and tangents $T \in \mathbb{R}^3$. In lieu of a standard octree we therefore use an axis-aligned *bounding volume hierarchy (BVH)*, for which the additional 3 dimensions do not incur significant cost.

Bounding Volume Hierarchy

To build the BVH, we first construct tangent-points $p_I := (T_I, \gamma_I) \in \mathbb{R}^6$ for each edge $I \in E$. We then cycle through all six coordinates, choosing a splitting plane that minimizes the sum of squared diameters of the two child bounding boxes. Splitting continues until all leaf nodes contain only a single edge each. In each node $\mathcal{N}$ we also store data needed for Barnes-Hut. Specifically,

$$L_{\mathcal{N}} := \sum_{I \in \mathcal{N}} \ell_I, \qquad \overline{\gamma}_{\mathcal{N}} := \sum_{I \in \mathcal{N}} \ell_I \gamma_I / L_{\mathcal{N}}, \qquad \overline{T}_{\mathcal{N}} := \sum_{I \in \mathcal{N}} \ell_I T_I / L_{\mathcal{N}},$$

give the total mass, center of mass, and (length-weighted) average tangent, respectively; we will use $\overline{p}_{\mathcal{N}} := (\overline{T}_{\mathcal{N}}, \overline{\gamma}_{\mathcal{N}})$ to denote the corresponding tangent-point. We also store the bounding box radii $r_x^{\mathcal{N}}$ and $r_T^{\mathcal{N}}$ with respect to spatial and tangential coordinates, respectively.

Approximate Energy

At the start of each iteration, we construct a bounding-volume hierarchy (BVH) on the edge set E as just described. The energy evaluation then becomes a sum

$$\widetilde{\hat{\mathcal{E}}(\gamma)} = \sum_{I \in E} \sum_{\mathcal{B} \in \mathrm{adm}(I)} k(x_I, \overline{\gamma}_{\mathcal{B}}, T_I)\ \ell_I\ L_{\mathcal{B}} \tag{3.18}$$

where $\mathrm{adm}(x_I)$ denotes the set of all *admissible* nodes with no admissible ancestors when viewed from the position x_I. In the context of Barnes-Hut, the admissibility condition quantifies the notion that a cluster is "far enough" away that it can be approximated using its center of mass. Leaf nodes are always considered admissible; for interior nodes, a Taylor series analysis of Equation 3.18 indicates that to keep approximation error below a user-specified threshold $\theta > 0$, it is sufficient to ensure that

$$r_x^{\mathcal{B}} / |\gamma_I - \overline{\gamma}_{\mathcal{B}}| \lesssim \theta \qquad \text{and} \qquad r_T^{\mathcal{B}} \lesssim \theta \tag{3.19}$$

where $r_x^{\mathcal{B}}$ denotes the bounding radius of the node in the 3 spatial dimensions, and $r_T^{\mathcal{B}}$ denotes the same in the 3 tangent directions. Intuitively, if $\mathcal{B}$ is far from the query point p_I relative to its size, *and* contains tangents that are close together, then the "lumped" energy is a good approximation of the total energy between edge I and the edges in $\mathcal{B}$.

Approximate Differential

For each vertex v of the curve, to approximate the partial derivative with respect to v, we evaluate the sum

$$\widetilde{\frac{\partial}{\partial \gamma_v}} \hat{\mathcal{E}}(\gamma) = \sum_{I \in E(v)} \sum_{\mathcal{B} \in \mathrm{adm}(\gamma_v)} h_\gamma(I, \mathcal{B}) \tag{3.20}$$

where $E(v)$ denotes the edges adjacent to v, and

$$h_\gamma(I, \mathcal{B}) = \frac{\partial}{\partial \gamma_v} \left[k^\alpha(\gamma_I, \overline{\gamma}_{\mathcal{B}}, T_I)\ \ell_I\ L_{\mathcal{B}} \right] + \frac{\partial}{\partial \gamma_v} \left[k^\alpha(\overline{\gamma}_{\mathcal{B}}, \gamma_I, \overline{T}_{\mathcal{B}})\ \ell_I\ L_{\mathcal{B}} \right].$$

In the exact case, for every edge $J \in \mathcal{B}$, differentiating by v would produce nonzero contributions for both the forward terms $k^\alpha(\gamma_I, \gamma_J, T_I)$ and the reverse terms $k^\alpha(\gamma_J, \gamma_I, T_J)$. All such forward terms are collectively approximated by the first term of h_γ, and likewise, all such reverse terms are collectively approximated by the second term.

3.4.2 Hierarchical Matrix-Vector Product

For optimization we need to solve linear systems involving so-called *kernel matrices*. Any such matrix $\mathsf{K} \in \mathbb{R}^{|E| \times |E|}$ has a special form

$$\mathsf{Q}_{IJ} = q(p_I, p_J)\ \ell_I\ \ell_J,$$

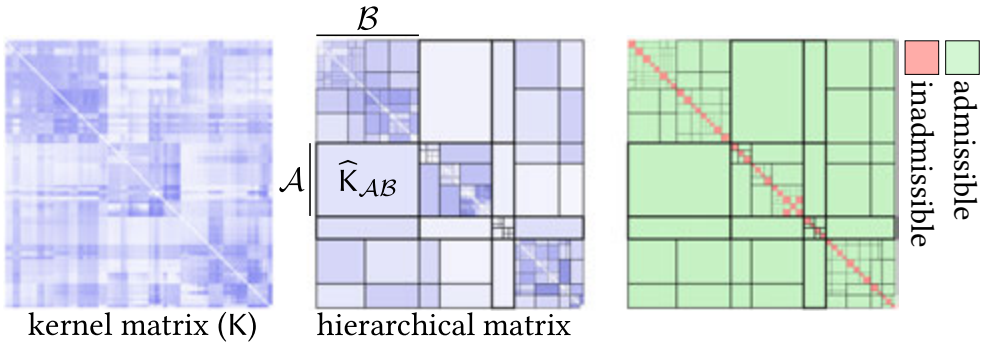


Figure 3.6: *Left:* A kernel matrix K encodes interactions between all pairs of edges. *Center:* To accelerate multiplication, this matrix is approximated by rank-1 blocks $\widehat{K}_{\mathcal{AB}}$, corresponding to pairs $(\mathcal{A}, \mathcal{B})$ of distant BVH nodes. *Right:* For pairs that are too close, this approximation is *inadmissible*, and we must use the original matrix entries.

where the kernel q maps a pair of tangent-points to a real value. If q is a sufficiently regular function, then K is well-approximated by a *hierarchical matrix* [53], which can be thought of as a blockwise low-rank approximation of the exact matrix. In particular, hierarchical matrices can be used to perform approximate matrix-vector products with Q without explicitly storing the entries as a matrix, which is crucial in our case, where Q is a dense matrix.

Block Cluster Tree

The key data structure behind a hierarchical matrix is known as a *block cluster tree* (BCT). The BCT can be constructed from a hierarchical partitioning of the domain such as a BVH; in our case, we can reuse the BVH that we previously constructed for Barnes-Hut energy evaluations. The nodes of the BCT consist of pairs of clusters $(\mathcal{A}, \mathcal{B})$ from the BVH, with rows indexed by $\mathcal{A}$, and columns indexed by $\mathcal{B}$ (Figure 3.6).

To construct the BCT, we initialize it with the node $(\mathcal{M}, \mathcal{M})$ as the root, where $\mathcal{M}$ is the root node of the BVH, containing all elements in the domain. We mark this initial node as "unresolved." We then iteratively loop over all currently "unresolved" BCT nodes. For each such node, corresponding to a pair of BVH nodes $(\mathcal{A}, \mathcal{B})$, we test if the pair of clusters is *admissible* or not. If the pair is admissible, we mark it as "admissible" and do not consider it further. Otherwise, if $|A| + |B| < S_{\min}$ for some minimum size threshold $S_{\min}$, we mark the pair as "inadmissible" and likewise leave it. If neither of these are true, then the node is discarded and replaced with the Cartesian product of the child sets of $\mathcal{A}$ and $\mathcal{B}$ in the BVH, all of which are then initially marked as "unresolved". Assuming a binary BVH, each such pair will be split into four children. The construction procedure thus concludes once all remaining BCT nodes are either admissible pairs, or inadmissible pairs below the fixed size threshold $S_{\min}$. In practice, the tree structure of the BCT is not important to maintain; all that is needed is the final list of admissible and inadmissible pairs.

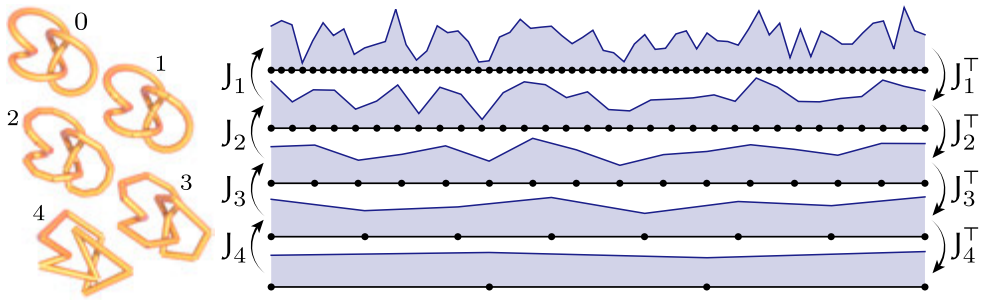


Figure 3.7: We accelerate linear solves using multigrid on a hierarchy of curves.

Cluster Admissibility. The admissibility test for a pair of clusters is similar to the Barnes-Hut test for a point and a cluster (Equation 3.19). A pair of clusters $(\mathcal{A}, \mathcal{B})$ is considered admissible if they satisfy the inequalities

$$\frac{\max(r_x^{\mathcal{A}}, r_x^{\mathcal{B}})}{|\overline{\gamma}_{\mathcal{A}} - \overline{\gamma}_{\mathcal{B}}|} \lesssim \theta \qquad \text{and} \qquad \max(r_T^{\mathcal{A}}, r_T^{\mathcal{B}}) \lesssim \theta \tag{3.21}$$

for the same accuracy parameter θ.

Fast Approximate Kernel Matrix Multiplication

As discussed, the block cluster tree enables the fast computation of approximate matrix-vector products with kernel matrices Q. Intuitively, every node of the BCT corresponds to a block of Q indexed by the clusters $\mathcal{A}$ and $\mathcal{B}$; the problem of multiplying by Q thus reduces to multiplying by every block of Q. By construction, many such blocks are admissible, representing the majority of the matrix entries, and these admissible blocks can be compressed into low-rank approximations that are efficient to multiply with vectors.

Concretely, to compute the approximate product $\tilde{\mathsf{y}} = \mathsf{Q}\mathsf{x}$, we first initialize $\tilde{\mathsf{y}} \leftarrow 0$. Then, for all nodes $(\mathcal{A}, \mathcal{B})$ of the BCT:

1. If $(\mathcal{A}, \mathcal{B})$ is inadmissible, then we multiply the block exactly by computing the entries of the indexed block $\mathsf{Q}_{\mathcal{A}\mathcal{B}}$ exactly, and multiplying:

$$\tilde{\mathsf{y}}_{\mathcal{A}} \leftarrow \tilde{\mathsf{y}}_{\mathcal{A}} + \mathsf{Q}_{\mathcal{A}\mathcal{B}}\mathsf{x}_{\mathcal{B}}.$$

2. If $(\mathcal{A}, \mathcal{B}$ is admissible, then we multiply by a rank-1 approximation of the block:

$$\tilde{\mathsf{y}}_{\mathcal{A}} \leftarrow \tilde{\mathsf{y}}_{\mathcal{A}} + \ell_{\mathcal{A}}\, q(\gamma_{\mathcal{A}}, T_{\mathcal{A}}; \gamma_{\mathcal{B}}, T_{\mathcal{B}})\, \ell_{\mathcal{B}}^\top\, \mathsf{x}_{\mathcal{B}}$$

where $\ell_{\mathcal{A}}$ for a cluster $\mathcal{A}$ denotes the vector of edge lengths within that cluster.

Applying the Fractional Laplacian via Kernel Matrices

The discrete fractional Laplacian L_σ (and associated operators derived from it) cannot itself be easily expressed as a kernel matrix; however, the result of the product $L_\sigma \mathsf{x}$ can be expressed in terms of products with kernel matrices. We show the full derivation in this section; see Equation 3.22 for the final expression.

To begin, consider the kernel

$$q_\gamma(I,J) := \begin{cases} \frac{\ell_I \ell_J}{\left\|\gamma_I - \gamma_J\right\|^{2\sigma+1}} & I \neq J \\ 0 & I = J \end{cases}$$

Rewriting Equation 3.13 with this kernel yields

$$\mathsf{u}^\top L^\sigma \mathsf{v} = \sum_{\substack{I \in E \\ I \neq J}} \sum_{J \in E} (\mathsf{u}(I) - \mathsf{u}(J))\,(\mathsf{v}(I) - \mathsf{v}(J))\, q_\gamma(I,J)$$

for two functions $\mathbf{u}$ and $\mathbf{v}$. Multiplying the product inside the sum gives

$$(\mathsf{u}(I)\mathsf{v}(I) + \mathsf{u}(J)\mathsf{v}(J) - \mathsf{u}(J)\mathsf{v}(I) - \mathsf{u}(I)\mathsf{v}(J))\, q_\gamma(S,T)$$

for the pair (I,J). Because $q_\gamma(I,J) = q_\gamma(J,I)$, we can move some terms between the summands for (I,J) and (J,I), thus reorganizing the sum into

$$\begin{aligned} \mathbf{u}^\top L^\sigma \mathbf{v} &= \sum_{\substack{I \in E \\ S \neq T}} \sum_{J \in E} (2\mathsf{u}(I)\,\mathsf{v}(I) - 2\mathsf{u}(I)\,\mathsf{v}(J))\, q_\gamma(S,T) \\ &= \sum_{\substack{I \in E \\ S \neq T}} \sum_{J \in E} \left(2\mathsf{u}(I)\, q_\gamma(S,T)\, \mathsf{v}(I) - 2\mathsf{u}(I)\, q_\gamma(S,T)\, \mathsf{v}(J)\right) \end{aligned}$$

Now, let u and v be vectors of length $|F|$ with the values of $\mathsf{u}(I)$ and $\mathsf{v}(I)$ respectively for all $I \in E$. Then, the sum is equivalent to

$$\begin{aligned} \mathbf{u}^\top L^\sigma \mathbf{v} &= 2\left(\mathsf{u}^\top \operatorname{diag}(Q_\gamma \mathbf{1})\mathsf{v}\right) - 2\left(\mathsf{u}^\top Q_\gamma \mathsf{v}\right) \\ &= 2\mathsf{u}^\top \left(\operatorname{diag}(Q_\gamma \mathbf{1})\mathsf{v} - Q_\gamma \mathsf{v}\right) \end{aligned}$$

where Q_γ is again the kernel matrix defined by q_γ. Now, let B_γ be the operator that averages vertex values onto edge midpoints; then we have $\mathsf{u} = B_\gamma \mathbf{u}$ and $\mathsf{v} = B_\gamma \mathbf{v}$, and

$$\begin{aligned} \mathbf{u}^\top L^\sigma \mathbf{v} &= 2\mathbf{u}^\top B_\gamma^\top \left(\operatorname{diag}(Q_\gamma \mathbf{1})(B_\gamma \mathbf{v}) - Q_\gamma (B_\gamma \mathbf{v})\right) \\ \langle \mathbf{u}, L^\sigma \mathbf{v} \rangle &= 2\left\langle \mathbf{u}, B_\gamma^\top \left(\operatorname{diag}(Q_\gamma \mathbf{1})(B_\gamma \mathbf{v}) - Q_\gamma (B_\gamma \mathbf{v})\right)\right\rangle \end{aligned}$$

$$L^\sigma \mathbf{v} = 2B_\gamma^\top \left(\operatorname{diag}(Q_\gamma \mathbf{1})(B_\gamma \mathbf{v}) - Q_\gamma (B_\gamma \mathbf{v})\right). \tag{3.22}$$

Thus, we can perform a matrix-vector product with L_σ using only products with the kernel

matrix Q_γ and inexpensive operations involving vectors or small matrices.

Applying High-Order and Low-Order Terms

The same derivation for the high- and low-order matrices B and B_0 proceeds analogously to that of the fractional Laplacian, with the substitution of the derivative operator D_γ for B_γ in the case of B. The resulting expression for the high-order term is

$$B_\sigma \mathbf{v} = 2\mathsf{D}_\gamma^\top \left(\operatorname{diag}(Q_\gamma \mathbf{1})(\mathsf{D}_\gamma \mathbf{v}) - Q_\gamma(\mathsf{D}_\gamma \mathbf{v})\right) \tag{3.23}$$

with the kernel

$$q(T_1, x_1; T_2, x_2) = \frac{1}{|x_1 - x_2|^{2\sigma+1}}.$$

Likewise, for the low order term, the expression is

$$B_\sigma^0 \mathbf{v} = 2B_\gamma^\top \left(\operatorname{diag}(Q_\gamma \mathbf{1})(B_\gamma \mathbf{v}) - Q_\gamma(B_\gamma \mathbf{v})\right) \tag{3.24}$$

with the kernel

$$q(T_1, x_1; T_2, x_2) = \frac{k^2(x_1, x_2, T_1) + k^2(x_2, x_1, T_2)}{2\,|x_1 - x_2|^{2\sigma+1}}.$$

where we symmetrize the kernel k^2 by averaging the forward and reverse terms.

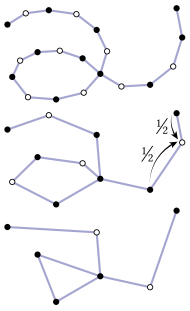
3.4.3 Multigrid Solver

Since the hierarchical matrix-vector multiply does not build an explicit matrix, we use an iterative method to solve our linear systems. Empirically, off-the-shelf methods such as GMRES and BiCGStab are not well-suited for our problem. Instead, we use geometric multigrid (Figure 3.7), since (i) it is straightforward to coarsen a curve network, and (ii) the low frequency modes of our Laplace-like operators are well-captured on a coarse mesh. In the Euclidean case, this type of approach has been used successfully by Ainsworth and Glusa [1].

In general, suppose we want to solve a linear equation $Ax = b$. The basic idea of geometric multigrid is to use a coarser mesh to reduce the residual of an equation on the finer mesh. Consider a simple two-level hierarchy—in particular, let $\mathsf{A}_0 \in \mathbb{R}^{|V_0|\times|V_0|}$ and $\mathsf{A}_1 \in \mathbb{R}^{|V_1|\times|V_1|}$ be discretizations of A on a fine and coarse mesh, respectively, and let b_0 be a discretization of the function b onto the finest mesh. Also let $\mathsf{J}_1 \in \mathbb{R}^{|V_0|\times|V_1|}$ be a so-called *prolongation operator*, which interpolates data from the coarse mesh onto the fine mesh. Starting with any initial guess $\mathsf{x}_0 \in \mathbb{R}^{|V_0|}$, we first apply a *smoothing procedure* S to the system $\mathsf{A}_0\mathsf{x}_0 = \mathsf{b}_0$, i.e. a fixed number of iterations of any iterative linear solver to get an improved guess $\tilde{\mathsf{x}}_0 \leftarrow S(\mathsf{A}_0, \mathsf{x}_0, \mathsf{b}_0)$. We then compute the residual $\mathsf{r}_0 \leftarrow \mathsf{A}_0\tilde{\mathsf{x}}_0 - \mathsf{b}_0$, and transfer it to the coarse mesh via $\mathsf{b}_1 \leftarrow \mathsf{J}_1^\top \mathsf{r}_0$. On the coarse mesh we solve the system $\mathsf{A}_1\mathsf{x}_1 = \mathsf{b}_1$ directly, and transfer the result back to the fine mesh via $\mathsf{y}_0 \leftarrow \mathsf{J}_1\mathsf{x}_1$. These values are used to update our guess via $\tilde{\mathsf{x}}_0 \leftarrow \tilde{\mathsf{x}}_0 + \mathsf{y}_0$, and smoothed again. If the residual is small enough, we stop; otherwise, we repeat another such *V-cycle* until

convergence. More generally, one can apply this two-level strategy to solve the linear system on the coarser level, yielding a multi-level strategy. The size of the coarsest level is chosen so that a direct solve at this level is more efficient than continuing to apply multigrid.

Curve Coarsening and Prolongation



To build a multigrid hierarchy on a general curve network, we apply a simple coarsening scheme. We mark alternating vertices as "black" and "white", and mark all endpoints and junctures where two or more curves meet as black. The next coarsest curve is obtained by removing white vertices, and we stop when we reach a target size or when there are no more white nodes. The prolongation operator J preserves values at black vertices, and at white vertices takes the average of the two neighboring black vertices. In our experience, using linear interpolation based on edge lengths made no appreciable difference in performance. Although coarsening can change the isotopy class of the curve network, it still provides useful preconditioning for the next level of the hierarchy.

Multigrid for Saddle Point Problems

Our constraint scheme entails solving *saddle point problems* of the form

$$\begin{bmatrix} \mathsf{A}_3 & \mathsf{C}^\top \\ \mathsf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathsf{a} \\ 0 \end{bmatrix}, \tag{3.25}$$

where A_3 is the inner product for vector-valued functions, and C is the constraint matrix (Section 3.3.4); the data $\mathsf{a} \in \mathbb{R}^{3|V|}$ depends on the problem being solved. We follow the approach of Braess and Sarazin [19], who note that for the structurally identical *Stokes' problem* (where A_3 and C are replaced by the Laplace and divergence operators, respectively, applying multigrid to the whole matrix does not work well. Instead, let $\mathsf{P} \in \mathbb{R}^{3|V| \times 3|V|}$ be a projection onto the null space of C, i.e. $\mathsf{CP} = 0$ and $\mathsf{P}^2 = \mathsf{P}$. Then by construction, any solution y to the equation

$$\mathsf{P}^\top \mathsf{A}_3 \mathsf{P} \mathsf{y} = \mathsf{P}^\top \mathsf{a} \tag{3.26}$$

yields a vector $\mathsf{x} = \mathsf{P}\mathsf{y}$ within the constraint space $\mathsf{C}\mathsf{x} = 0$ that satisfies our original equation. Equation 3.26 is therefore the system that we actually solve via multigrid. In particular, we use the projection $P := \mathsf{C}\mathsf{C}^\dagger$, where $\dagger$ denotes the (Moore-Penrose) pseudoinverse

$$\mathsf{C}^\dagger := (\mathsf{C}\mathsf{C}^\top)^{-1}\mathsf{C}^\top.$$

Since our constraints are typically sparse, we can factorize the inner term $\mathsf{C}\mathsf{C}^\top$ (once per time step) to further accelerate computation. Note that one must build a constraint matrix C_i and projection matrix P_i at each level i of the multigrid hierarchy.

Gradient Solve and Constraint Projection

With these pieces in place, we can apply multigrid to compute the constrained gradient (Equation 3.16), and perform constraint projection (Equation 3.17).

Initialization. We obtain an initial guess x_0 by first coarsening the fine right-hand side b_0 down to the coarsest mesh. We then perform a direct solve and prolong the solution all the way to the finest mesh, applying smoothing after each refinement. In practice, this strategy works much better than starting with the zero vector.

Smoother. We use a standard conjugate gradient smoother, with a target relative residual on the order of 10^{-3}. Making the residual smaller via further cycles (and a more accurate BCT) yields diminishing returns: we need only a reasonable intermediate descent direction. We observe that, typically, we only require 6 or fewer V-cycles to reach this residual value. Note that forward matrix-vector products are performed approximately at all levels of the hierarchy using the method outlined in Section 3.4.2; as such, we must build a block cluster tree at every level of the hierarchy. The total construction cost is only about twice the cost at the finest level, however, due to geometrically decreasing vertex counts on successive levels.

Gradient. To compute the gradient, recall that $\mathsf{A}_\sigma = B_\sigma + B_\sigma^0$; a matrix-vector product with A_σ can thus be implemented via products with its two terms. As discussed in Section 3.4.2, products with the high- and low-order terms can be implemented using products with kernel matrices (see Equation 3.23 and Equation 3.24). Finally, the product with A_3 can be implemented using three successive products with A_σ on the strided vectors consisting of each coordinate per vertex.

Constraint Projection To use our multigrid solver for constraint projection, we apply a simple transformation to Equation 3.17 that gives it the same form as Equation 3.25. In particular, we solve

$$\begin{bmatrix} \mathsf{A}_3 & \mathsf{C}^\top \\ \mathsf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{y} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathsf{A}_3\mathsf{z} \\ 0 \end{bmatrix},$$

where $\mathsf{z} := \mathsf{C}^\dagger \mathsf{b}$, and b is the lower block of the right-hand side of Equation 3.17. The final result is then given by

$$\mathsf{x} = \mathsf{z} - \mathsf{y}. \tag{3.27}$$

3.5 Evaluation and Comparisons

We performed an extensive evaluation and comparisons of our fractional Sobolev descent strategy relative to other methods. Here we give an overview of results; a detailed account of how these evaluations were performed can be found in the supplemental material of Yu et al. [116].

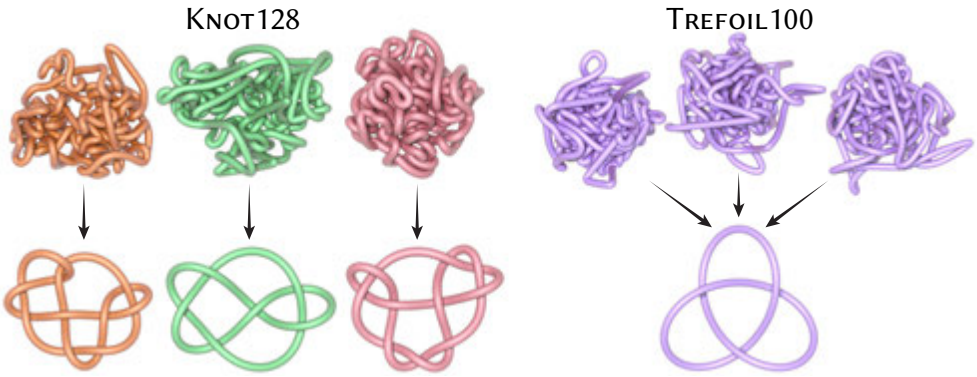


Figure 3.8: To evaluate performance, we built a "stress test" dataset of 128 random embeddings of different knot classes *(left)* and 100 random embeddings of the trefoil knot *(right)*. The tangent point energy drives these curves toward much simpler embeddings, as shown here.

3.5.1 Dataset

We created two datasets of difficult knot embeddings: KNOT128, which contains random embeddings of 128 distinct isotopy classes from *KnotPlot*'s "knot zoo," and TREFOIL100, which contains 100 random embeddings of the trefoil knot (Figure 3.8). We also used the *Freedman unknot* (Figure 3.1, *top left*), which is a standard "challenge problem" from the knot energy literature [97, Section 3.3]. To examine scaling under refinement, we performed regular refinement on knots from each of these sets.

3.5.2 Performance Comparisons

We compared our fractional Sobolev descent strategy to a variety of methods from optimization and geometry processing. Overall, methods that use our fractional preconditioner performed best, especially as problem size increases. We first ran all methods on several resolutions of a small set of test curves (Figure 3.13); we then took the fastest methods, and ran them on all 228 curves from our two datasets (Figure 3.14). For simplicity we did not use hierarchical acceleration in our method (and instead just solve dense systems), which gave a *significant* performance advantage to alternative methods (which are based on sparse solves). Even with this handicap, the fractional approach outperformed all other methods; as indicated in Figure 3.15, hierarchical acceleration would widen this gap even further. Importantly, previous methods also have a much higher failure rate at untangling difficult curves such as those in our dataset (Figure 3.14). Further, cases on which the fractional approach itself fails generally contain near-intersections in the initial configuration (Figure 3.9), which also lead to failures in most or all other methods.

Note that some previous methods do not directly handle hard nonlinear constraints; for these methods we perform an apples-to-apples comparison by replacing—in *all* methods—hard edge length constraints with a soft elastic penalty (see supplemental material for further details).

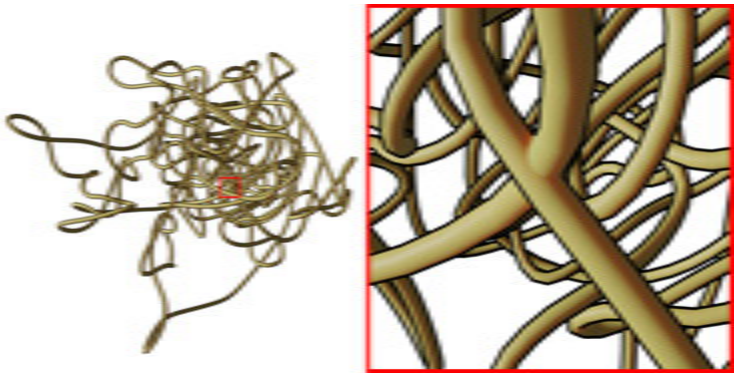


Figure 3.9: An example of a "failed" knot, which features a near-self-intersection in the initial configuration.

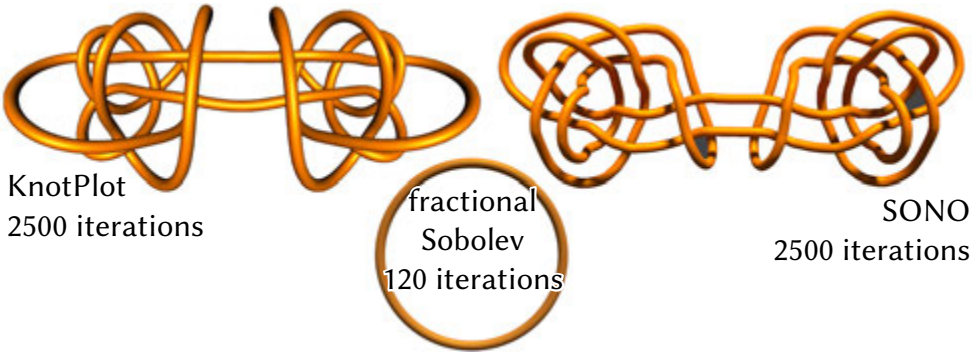


Figure 3.10: Our fractional Sobolev strategy is dramatically more efficient than previous methods for knot untangling—here we untangle the unknot from Figure 3.1. Neither KnotPlot nor SONO converged after several hours.

Knot untangling methods. We first compared to two well-known methods for knot untangling (Figure 3.10): *KnotPlot*, based on the so-called *symmetric energy*, and *shrink on no overlaps (SONO)* [91] which performs a local iterative projection in the spirit of contemporary *position-based dynamics* [84]. Both methods successfully untangle the Freedman knot, but only after tens of thousands of iterations [97, Figure 7.6]. The basic reason is that, like L^2 descent, such methods focus on reduction of local error, making global convergence quite slow.

1st-order methods Figure 3.11 indicates that basic 1st-order schemes like ordinary L^2 gradient descent, L-BFGS using 10, 30, or 100 vectors, and nonlinear conjugate gradients [44] exhibit poor performance relative to our fractional scheme in terms of both wall clock time and number of iterations. This example also indicates that for $1 < s < 2$, the next smallest or largest *integer* Sobolev preconditioners (H^1 and H^2) underperform the fractional H^s preconditioner, whether using explicit (forward) or implicit (backward) Euler. We solve the backward Euler update equation using Newton's method, either by updating the Hessian for each Newton step ("Newton"), or "freezing" the Hessian at the beginning of the time step ("frozen"). If Newton's

method fails to converge within a few (10) iterations, the step size is halved and the solve is reattempted. We also tried stochastic gradient descent (SGD) with respect to the L^2 inner product, implemented by subsampling a fixed proportion (25% in our trials) of edge pairs (u, v) for each edge u in each iteration for energy and gradient evaluation. This method did far worse than any other scheme we tried. SGD with respect to H^s works better, but the speedup from stochastic evaluation does not compensate for the poor quality of the descent direction.

2nd-order methods Second-order schemes like Newton's method can be adapted to non-convex problems by projecting the Hessian onto a nearby positive-semidefinite matrix. Since a global projection is prohibitively expensive, a heuristic sometimes used in geometric optimization is to project and sum up the Hessians of each local energy term [107]; in our case we can decompose the energy into the edge-edge terms from Equation 4.10. Though this heuristic can work well for, e.g. elastic energies, it does not appear to work very well for the tangent-point energy, and for larger examples had among the slowest run times of any scheme we tried (Figure 3.13).

Quasi-Newton methods Several recent methods from geometry processing apply Sobolev-like preconditioning to elastic energies, such as those used for shape deformation or surface parameterization [31, 68, 120]. Since the highest-order term in such problems often looks like a Dirichlet energy, H^1 preconditioning via the Laplacian Δ can be an effective starting point for optimization (as discussed in Section 2.2.1). However, such preconditioners do not perform as well as our fractional preconditioner, since they are not as well-matched to the order of the differential $d\mathcal{E}$. For instance, as seen in Figure 3.13, the AQP strategy of Kovalsky et al. [68] significantly underperforms our preconditioner when the Laplacian is used as the quadratic proxy; using our fractional operator as the quadratic proxy improves performance—but of course requires the machinery introduced in this paper. Another possibility is to use Laplacian-initialized L-BFGS (in the spirit of Zhu et al. [120]); we found this strategy works a bit better than AQP, but again not as well as the fractional preconditioner. We also considered several variants of these strategies, such as applying Nesterov acceleration, and combining nonlinear conjugate gradients (NCG) (as in Polak and Ribiere [93]) or L-BFGS with our fractional preconditioner. For hard constraints we advocate the use of our fractional (H^s) projected gradient scheme (as detailed in Section 3.3); if soft constraint enforcement is acceptable, then L-BFGS or H^s-preconditioned NCG are both good options: the former converges faster near minima; the latter gets stuck less often.

3.5.3 Local minimizers

As seen in Figure 3.12, the local minimizers found via our fractional descent strategy generally appear to be the same as with other schemes, up to rigid motions. Hundreds more such examples can be found in the supplemental material. Very rarely, two different methods produced local minimizers that were identical up to a *reflection*; such *amphichiral* pairs exist in some knot classes [76], but of course have the same energy.

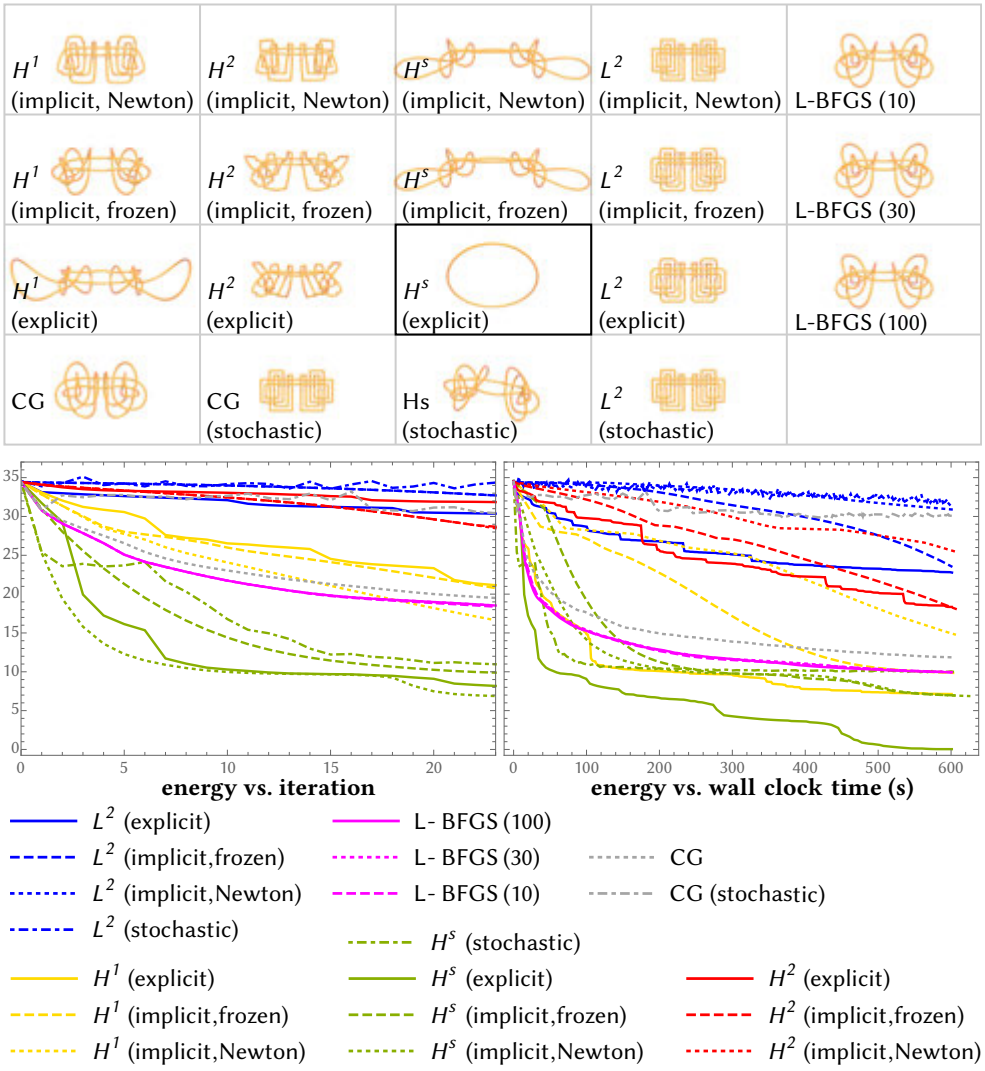


Figure 3.11: Across a wide variety of descent methods and inner products, our fractional Sobolev approach does significantly better both in terms of energy reduction per iteration *(middle left)* and real-world run time *(middle right)*. At top we show results for an equal amount of compute time.

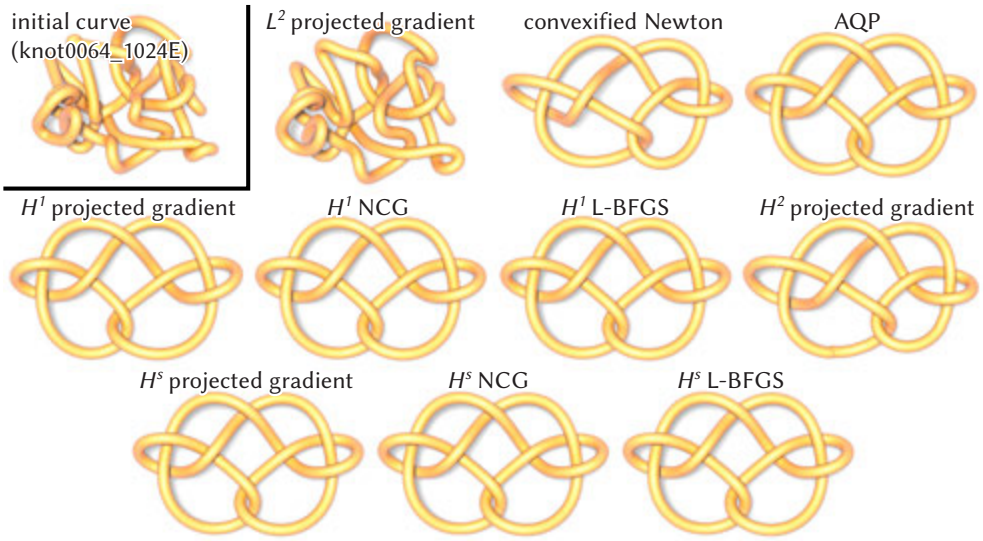


Figure 3.12: The tangent point energy appears to have relatively few local minima; hence, different descent strategies tend to find the same local minimizers (though some, like L^2, do not find solutions in a reasonable amount of time). See supplemental material for several hundred more examples.

3.5.4 Scaling behavior

We compared per-iteration costs of the unaccelerated scheme, a scheme using only Barnes-Hut (Section 3.4.1), and the full acceleration scheme described in Section 3.4—see Figure 3.15. With full acceleration we observe near-linear scaling, whereas schemes that directly solve the dense system exhibit super-quadratic scaling and quickly run out of memory. Note that constraint projection with direct solvers comes nearly for free, since a factorization of Equation 3.17 can be reused to solve Equation 3.16. In contrast, no reuse is possible in the fully accelerated scheme, making constraint projection relatively expensive. Disabling this step further speeds up the accelerated scheme, but leads to constraint drift over time. Alternative methods for constraint enforcement (such as soft penalties, as noted above) might hence provide further improvement.

3.6 Results and Applications

Given how ubiquitous plane and space curves are in areas like geometry, graphics, robotics, and visualization—and how natural it is to want to avoid intersection of such curves—our method provides a useful computational framework for a wide variety of tasks. Here we explore some preliminary applications that we hope will inspire future work. All other examples in this section completed within a few minutes, except for the 3D curve packing example where we allowed curves to grow longer for several hours as a stress test. We first describe constraints and potentials used for these examples.

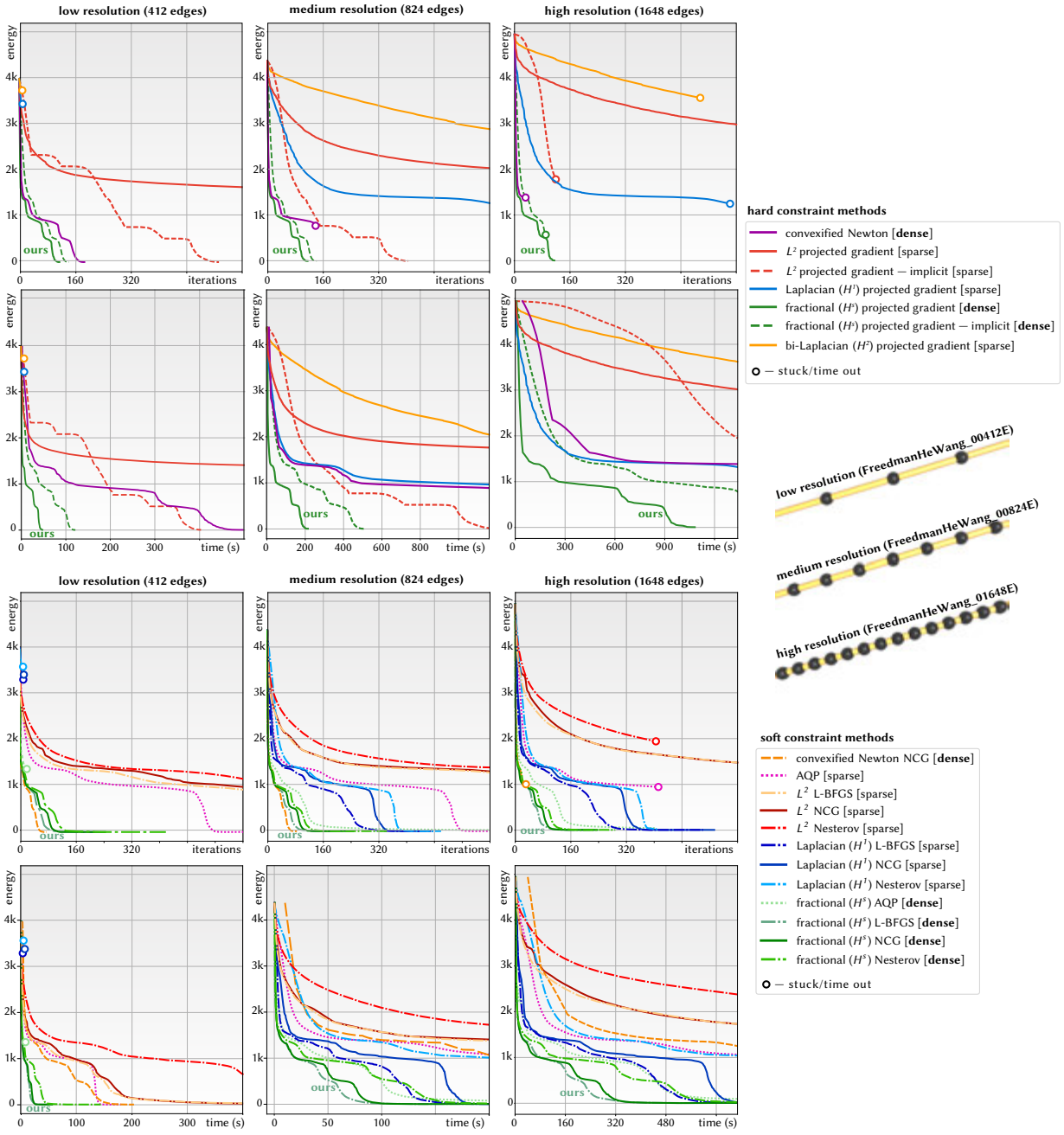


Figure 3.13: We compared our descent strategy to a variety of 1st-order, 2nd-order, and quasi-Newton strategies, using both hard constraints (top) and a soft penalty (bottom) to preserve length. Here we show energy versus both time and iteration count for several resolutions of the initial curve from Figure 3.1; tests on additional curves yield very similar results (see supplemental material). Note that we achieve the best real-world clock time—even though we compare a dense implementation of our method (without hierarchical acceleration) to sparse versions of other schemes.

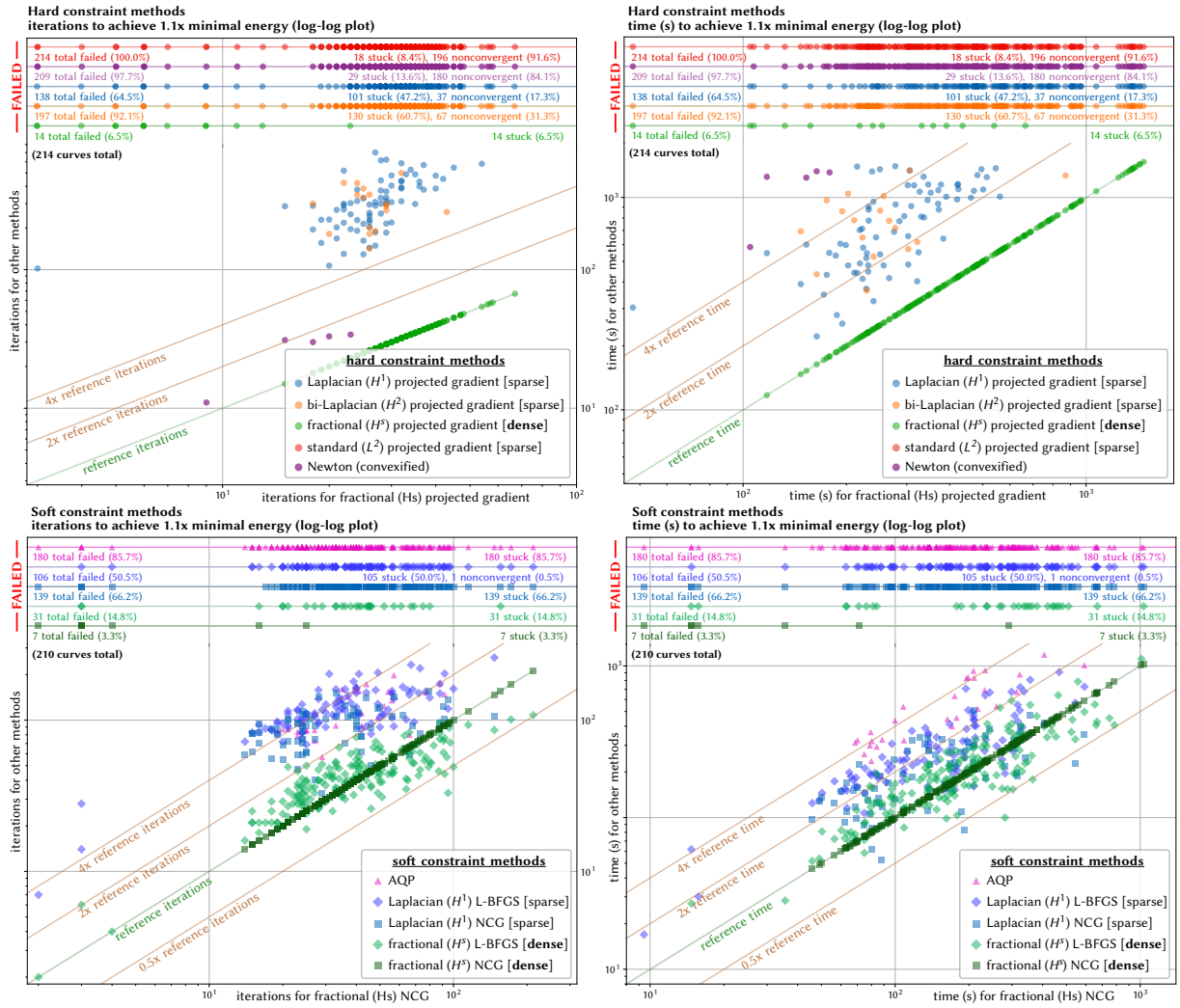


Figure 3.14: We used a dataset of about two hundred difficult knot embeddings to evaluate the performance of our strategy compared to the next most competitive methods. Even without hierarchical acceleration, our fractional strategy was significantly faster—and succeeded at untangling a much larger fraction of knots. Here we plot the time it took for each method to get within 1.1x of the reference energy, against the time taken by our fractional strategy. Results have been split into hard/soft constraint enforcement (top/bottom rows), and iteration count/wall clock time (left/right columns). At the top of each plot we show the number of failures after 24 minutes of compute time—*stuck* indicates a failure of line search to make progress due to intersections; *nonconvergent* means the method failed to get below 1.1x of the reference energy.

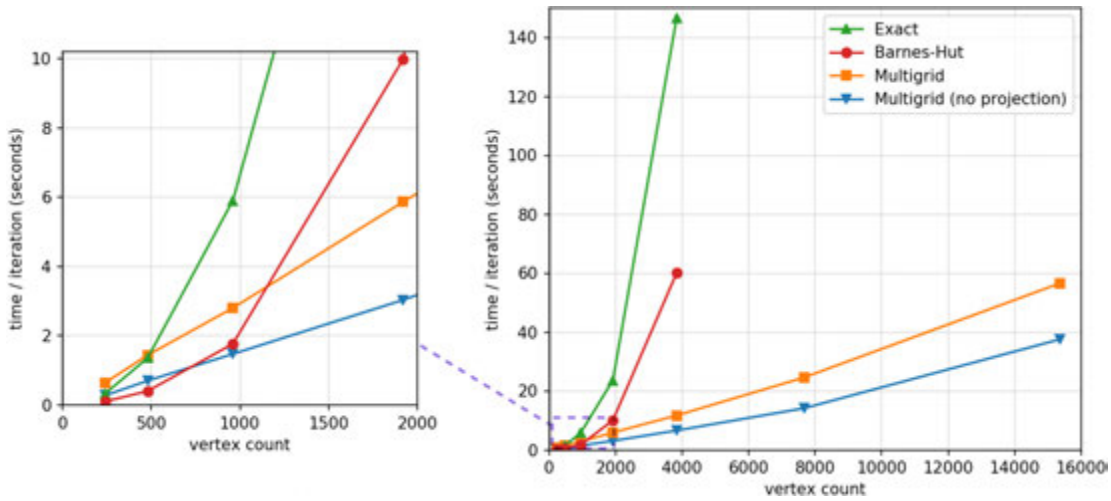


Figure 3.15: A comparison of runtime per iteration on samplings of the same curve with increasing resolution. "Exact" indicates no acceleration, "Barnes-Hut" indicates accelerated gradients only, and "Multigrid" indicates all accelerations enabled, with and without constraint projection. Reported numbers are averages over up to 500 iterations or until convergence.

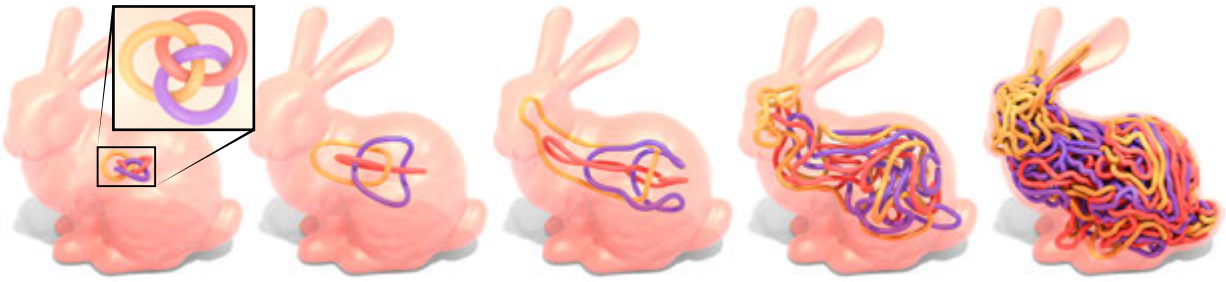


Figure 3.16: Interwoven curves of increasing length are confined inside a fixed domain, resulting in an intricate "curve packing."

3.6.1 Constraints and Potentials

A key feature of our optimization framework is that it not only efficiently minimizes knot energies, but that it can do so in conjunction with fairly arbitrary user-defined constraints and penalties (Section 3.3.4). This opens the door to a rich variety of computational design applications beyond the basic "knot untangling" that has been the focus of previous work. For the applications that will be explored in Section 3.6, we consider the following constraints:

- **Barycenter.** This fixes the barycenter of the curve to a point x_0 via $\Phi_{\text{barycenter}}(\gamma) := \sum_{I \in E} \ell_I (x_I - x_0)$. In the absence of other constraints, this eliminates the null space of globally constant functions discussed in Section 4.2.2.

- **Length.** The repulsive curve energy naturally wants to make the curve longer and longer. A simple way to counteract this is via a total length constraint $\Phi_{\text{length}}(\gamma) := L^0 - \sum_{I \in E} \ell_I$,

where L^0 is the target length.

- **Edge Length.** We can also constrain the lengths of each individual edge, allowing only isometric motions. This entails a constraint $\Phi_{\text{length},I}(\gamma) := \ell_I^0 - \ell_I$ for each edge I, where ℓ_I^0 is the target edge length.
- **Point Constraint.** To fix the position of a vertex i to the point $x_i \in \mathbb{R}^3$, we can add the constraint $\Phi_{\text{point},i}(\gamma) := \gamma_i - x_i$.
- **Surface Constraint.** To keep a point of the curve constrained to an implicit surface $f(x) = 0$, we can add the constraint $\Phi_{\text{surface},i}(\gamma) := f(\gamma_i)$.
- **Tangent Constraint.** We can force the tangent T_I of an edge I to match a unit vector $X \in \mathbb{R}^3$ via the constraint $\Phi_{\text{tangent},I}(\gamma) := T_I - X$.

In several applications, we progressively increase or decrease the target length values L_0 or l_I^0; the next constraint projection step then enforces the new length. We also consider the following penalties:

- **Total length.** A simple energy is the total curve length, which provides a "soft" version of the total length constraint. Discretely, this energy is given by $\hat{\mathcal{E}}_{\text{length}}(\gamma) := \sum_{I \in E} \ell_I$.
- **Length difference.** This energy penalizes differences in adjacent edge lengths, given by $\hat{\mathcal{E}}_{\text{diff}}(\gamma) = \sum_{v \in V_{\text{int}}} (\ell_{I_v} - \ell_{J_v})^2$, where V_{int} denotes the set of "interior" vertices with degree 2, and I_v and J_v are the indicent edges to v.
- **Surface potential.** Given a surface $M \subset \mathbb{R}^3$, we use the energy $\mathcal{E}_M(\gamma) := \int_\gamma \int_M 1/|x_M - \gamma(x_\gamma)|^{\beta-\alpha} dx_M \; dy_\gamma$ to avoid intersections. This is effectively a Coulomb potential of the same order as $\mathcal{E}$ on M. In the discrete setting, M is a triangulated surface, and we use a BVH on M to accelerate the evaluation of $\mathcal{E}_M$ and its differential, in a similar fashion to $\mathcal{E}$.
- **Field potential.** Given a fixed unit vector field X on $\mathbb{R}^3$, the energy $\mathcal{E}_X(\gamma) := \int_0^L |T(x) \times X(\gamma(x))|^2 \; dx_\gamma$ encourages γ to run parallel (or anti-parallel) to X. We discretize this as $\hat{\mathcal{E}}_X(\gamma) := \sum_{I \in E} \ell_I |T_I \times X(\gamma_I)|^2$.

Note that the energies considered here involve lower-order derivatives than those in $\mathcal{E}$, and do not therefore have a major effect on the stiffness of the overall system. Hence, the fractional Sobolev inner product will continue to provide efficient convergence on these composite energies.

3.6.2 Curve Packing

Packing problems (such as *bin packing*) appear throughout geometry and computer graphics, playing an important role in, e.g. 2D layouts for manufacturing or UV atlas generation. An adjacent problem is generation of regular sampling patterns, e.g. blue noise sampling via Poisson

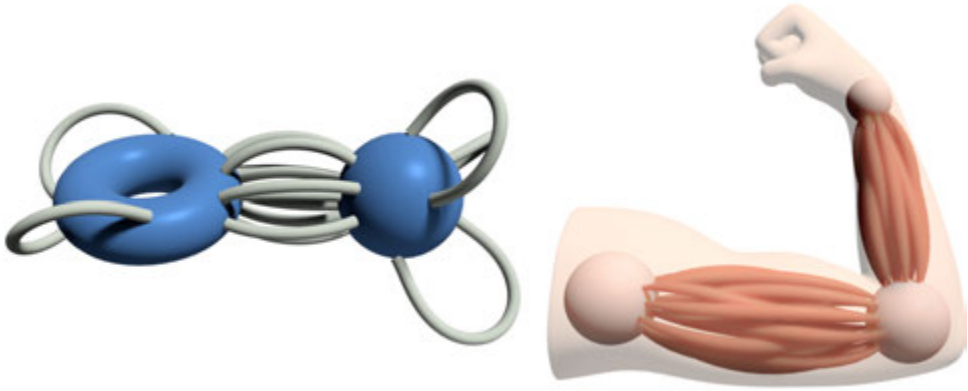


Figure 3.17: Allowing curves to slide freely over constraint surfaces *(left)* enables design tasks like arranging networks of muscles or muscle fibers *(right)*.

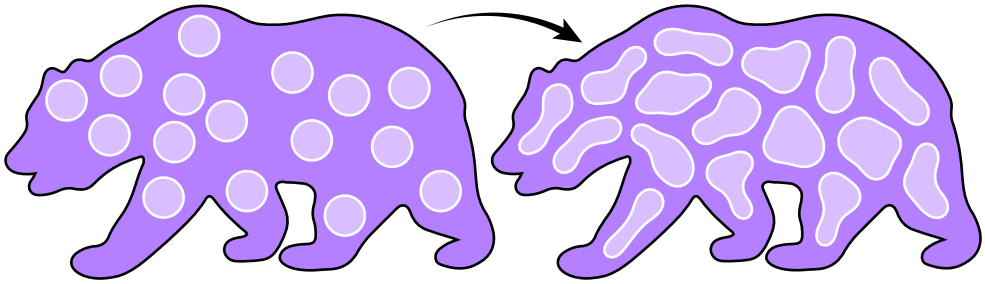


Figure 3.18: Just as repulsive potentials are commonly used to find equally-distributed points, we can compute collections of equally-spaced curves (here constrained to a region via a fixed curve potential).

disk rejection. The ability to optimize large families of repulsive curves enables us to solve analogous "curve packing" problems—for instance, in Figure 3.18, we use a fixed boundary curve to pack disks of increasing length; likewise, in Figure 3.16 and Figure 3.19, we use a surface penalty to pack increasingly long curves into a target region. Figure 3.20 likewise packs increasingly long curves on a surface. Going the opposite direction, we can also *decrease* length while encouraging repulsion to generate clean illustrations that are difficult to draw by hand (Figure 3.21). Finally, by constraining only parts of curves to lie on surfaces, we can design biologically-inspired curve networks such as muscle fibers (Figure 3.17), which are attached to objects at their endpoints but are otherwise free.

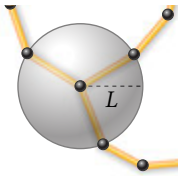
3.6.3 Graph Drawing

A basic problem in data visualization is drawing *graphs*; a typical approach is to use a force-based layout that seeks to avoid intersections between nodes, or over/under-extension of edges [48]. Our framework makes it easy to optimize the geometry of the edges themselves, opening the door to graph layouts that are both more compact and more legible (Figure 3.22). We can also use

Figure 3.19: By penalizing proximity to a fixed surface, we can pack curves into any domain. Progressively increasing edge length forces curves to maintain a balance between surface avoidance and self-avoidance. (Here we render curves with a non-circular cross section, which is not modeled by the energy.)

this machinery to obtain legible drawings of nonplanar graphs, by perturbing a planar embedding (Figure 3.23); here, the ability to preserve lengths conveys information about edge weights. A particularly interesting graph embedding problem is the design of synthetic hydrogel *vascular networks* [51]; Figure 3.24 shows a simple example where we optimize a multivascular network (starting from subgraphs of a tet mesh and its dual).

Note that at junctures between more than two edges, the tangent-point energy will always be large (since three or more edges cannot be collinear), rapidly forcing vertices away from each other. This can be counteracted by constraining their edge lengths, forcing the vertices to lie on spheres of constant radii around the junctures.



3.6.4 Self-Avoiding Splines

Beyond standard Bézier input, sophisticated tools have been developed for drawing spline curves—but do not consider the basic constraint of ensuring that curves do not cross themselves (which is often desirable for physical or aesthetic reasons). For instance, Figure 3.26 (*center*) shows the interpolation of a set of control points by *k-curves* [113], which underpin one of the basic drawing tools in Adobe Illustrator (the *Curvature Tool*). By simply applying point constraints at the control points, and letting the length increase under our repulsive flow, we obtain a nice interpolating curve without self-intersection (Figure 3.26, *right*). In this context we can also use our tangent constraint to control the behavior of such a curve at open endpoints (Figure 3.25).

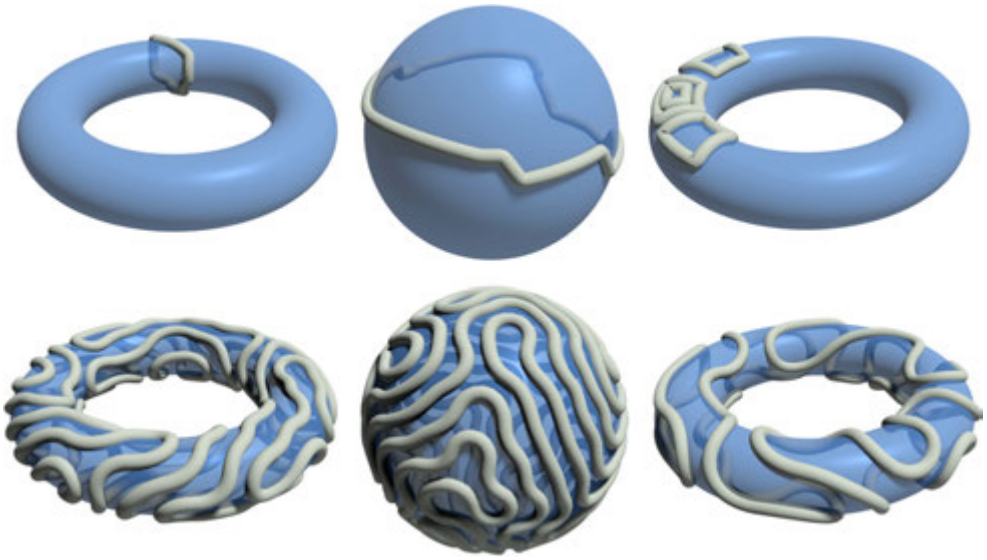


Figure 3.20: Patterns obtained by constraining a collection of repulsive curves to a surface and increasing their lengths (initial states shown above their final configurations).

3.6.5 Multi-agent Path Planning

In robotics, numerous algorithms have been developed for the problem of multi-agent path planning [34], wherein multiple agents must travel from fixed start to end locations without colliding with the environment or each other. Many algorithms operate on a discrete grid or graph [117], which quantizes the solution space and does not penalize near-intersections; such trajectories may therefore not be robust to sensing or control error. By treating path planning as a space-time optimization of continuous curves with fixed endpoints, we can use curve repulsion to find (or refine) trajectories that maximize collision avoidance, making them more resilient to error (Figure 3.27). Finding such trajectories in n dimensions is equivalent to optimizing a braid in $n+1$ dimensions; since neither the size of the curve nor the cost of a BVH/BCT depends strongly on dimension, this strategy easily generalizes to three (or more) dimensions.

3.6.6 Streamline Visualization

A common way to visualize vector fields is by tracing integral curves or *streamlines*; significant effort has gone into algorithms that provide uniform spacing e.g. by incrementally constructing a Delaunay triangulation [81]), though such methods can be difficult to generalize to 3D volumes or vector fields on surfaces. We can generate nicely-spaced streamlines by adding a field alignment potential to the tangent-point energy—for instance, in Figure 3.28 we start with a set of random curve segments, which automatically coalesce into streamlines.

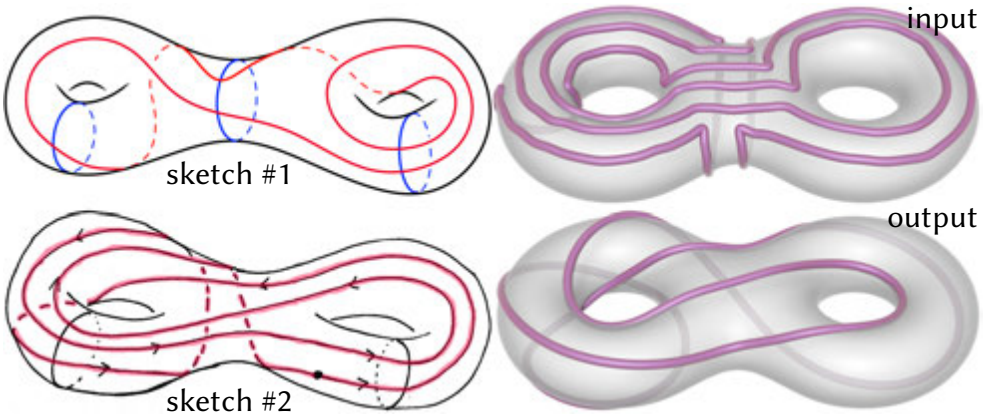


Figure 3.21: Loops arising in topology can be difficult to draw by hand—the sketches at left were done by Nathan Dunfield to illustrate *Dehn-Thurston* coordinates. At right we generate an equispaced version of this curve by flowing a rough sketch, subject to an implicit surface constraint.

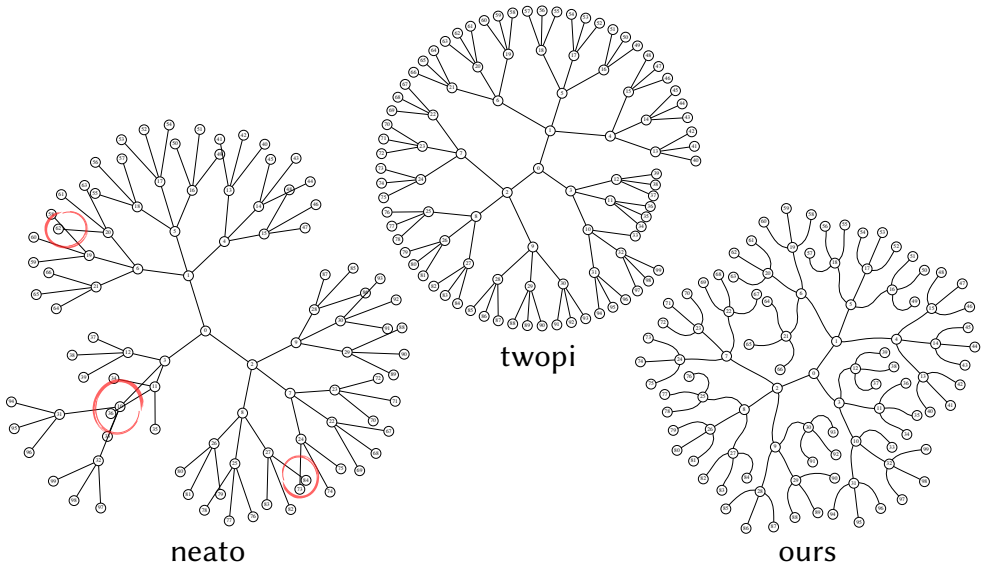


Figure 3.22: Traditional 2D graph drawing algorithms based on nodal proximity may cause edges to cross *(left)* or position nodes extremely close together *(center)*; these layouts were produced by the popular *Graphviz* library [42]. By treating edges as repulsive curves, we can obtain graph drawings that are both more compact and more legible *(right)*.

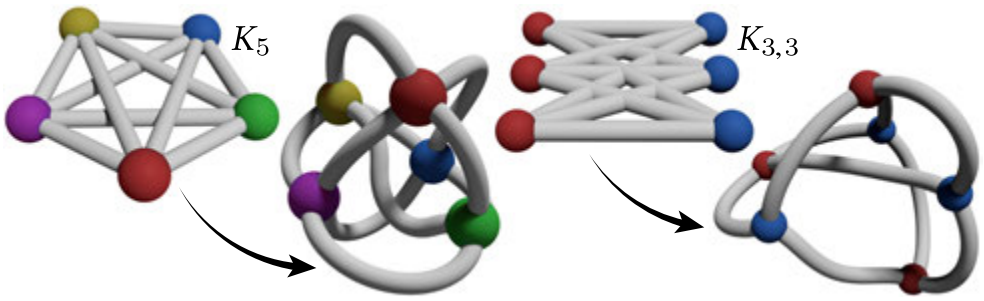


Figure 3.23: Isometric embedding: by jittering 2D drawings of non-planar graphs (which necessarily have crossings), curve repulsion with length constraints yields nicely spaced embeddings in R^3 with prescribed edge lengths.

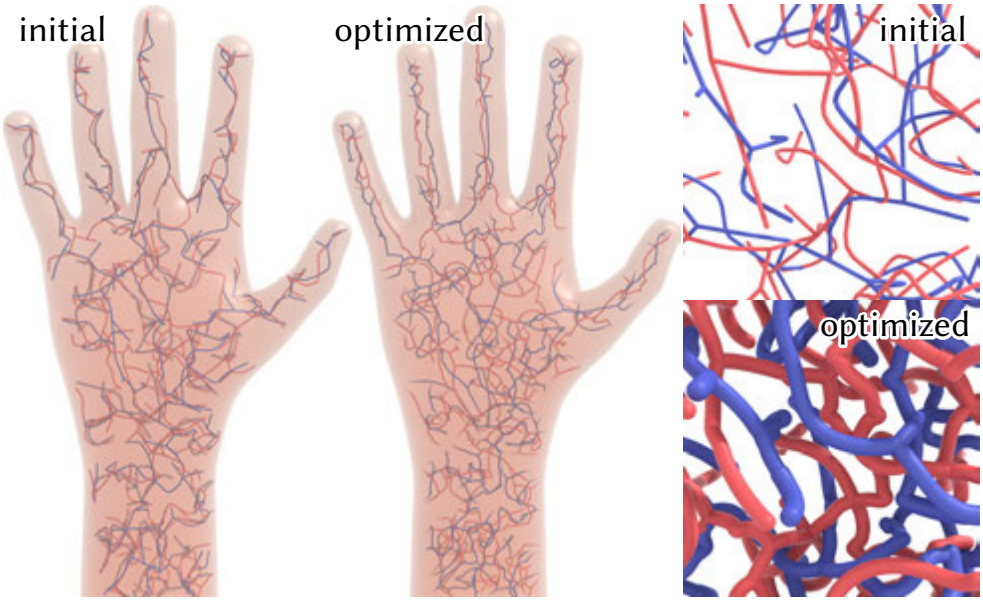


Figure 3.24: *Left:* a crude initial topology for a synthetic vascular network *(left)* is optimized to achieve more uniform delivery of nutrients throughout a volume. *Right:* plotting the maximum intersection-free thickness helps to visualize the improvement in uniformity.

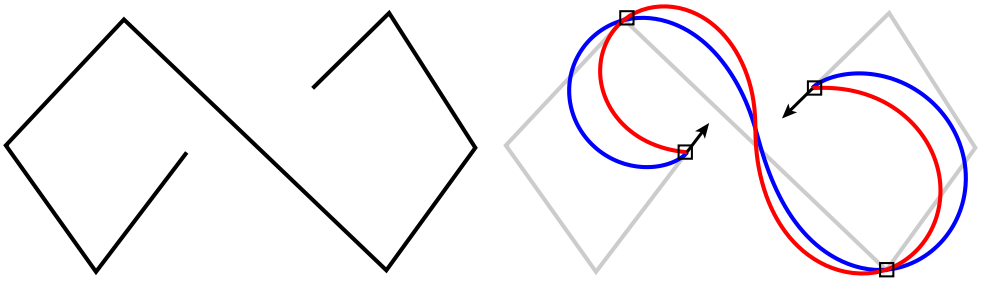


Figure 3.25: As with Bézier curves, we can also control curve tangents at both interior and endpoints. Here we flow a polygonal curve *(left)*, to a smooth interpolant with fixed points *(red)*, and fixed points and tangents *(blue)*.

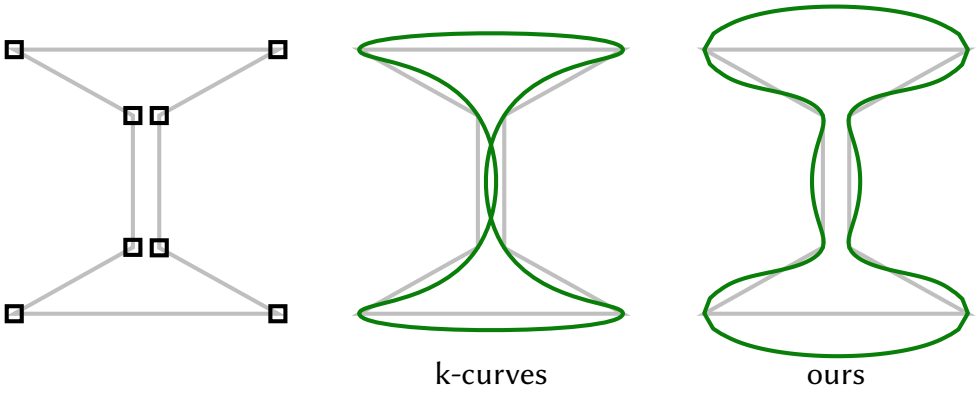


Figure 3.26: Standard curve interpolation methods in 2D drawing programs can cause curves to self-intersect *(center)*, even when the control polygon *(left)* does not. By starting from the control polygon and constraining the control points, we obtain a smooth, non-intersecting interpolant *(right)*.

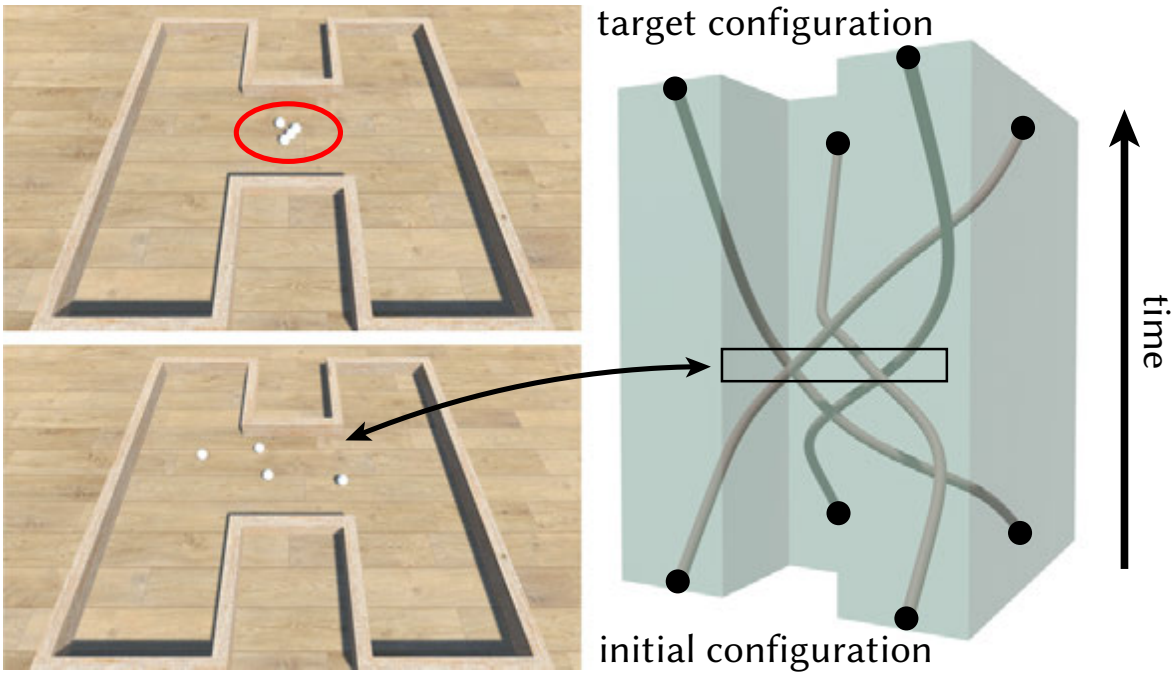


Figure 3.27: *Top-left:* In this path planning scenario, an initial trajectory brings the four agents dangerously close together. *Bottom-left:* By treating trajectories as curves in space-time, our system provides solutions that maximally avoid intersections, making them more robust to control errors. *Right:* Finding 2D trajectories is equivalent to optimizing a 3D braid with fixed endpoints constrained to an extrusion of the given environment.

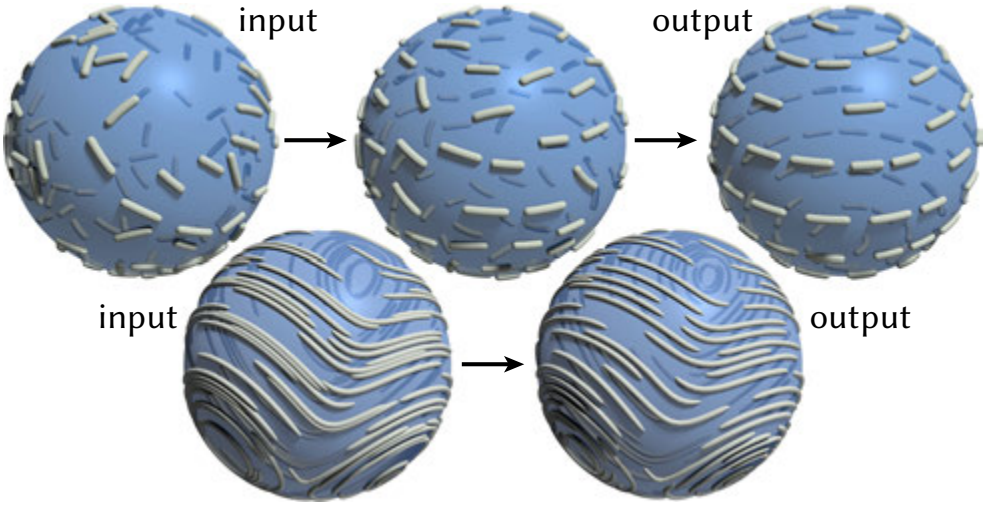


Figure 3.28: Encouraging curve tangents to align with a given vector field improve the quality of streamline visualization. Here, a random set of curve segments *(top)* aligns itself with a rotational vector field; we can also optimize randomly sampled streamlines *(bottom)* to improve their spacing.

Chapter 4

Repulsive Surfaces

When working with surfaces, intersections and self-intersections are just as much (if not more) of a consideration as they are on curves. Intersections can have serious implications when the surfaces in question represent physical membranes (e.g. in biological simulation), boundaries of solid objects (e.g. for digital manufacturing), or certain mathematical objects (e.g. isotopy classes of embeddings). It is therefore surprising that, to date, there is still no known general-purpose algorithm for preventing non-local intersections on embedded surfaces. In this chapter, we aim to fill this void by presenting an algorithm for intersection-free optimization of surfaces based on the tangent-point energy, which can be easily incorporated into variational surface modeling frameworks. In particular, we develop

- the first discretization of tangent-point energy for surfaces,
- a novel preconditioner that avoids a multigrid hierarchy,
- a hierarchical solver that scales to large meshes, and
- a framework for handling auxiliary constraints and penalties.

We also perform a preliminary investigation of applications in geometric modeling, mathematical visualization, and geometry processing. Notably, although one can prove that minimizers of the tangent-point energy exist [67, Theorem 2], these proofs are non-constructive. Since we provide the first discretization and optimization procedure for the tangent-point energy on surfaces, we obtain the first glimpse (experimentally) at what some of these surfaces might actually look like.

After a brief review of related work (Section 4.1), we will begin by defining our problem in the smooth setting (Section 4.2), followed by a novel discretization of the tangent-point energy and a basic numerical strategy for minimizing it subject to constraints (Section 4.3). We then significantly accelerate this strategy in two distinct ways. First, we choose an inner product in the smooth setting that vastly improves the convergence of the gradient flow (Section 4.2.4). Second, in the discrete setting, we propose a preconditioner that dramatically reduces the cost

of solving for the descent step (Section 4.5). We also accelerate evaluation of the energy and its derivatives, as well as dense matrix-vector products, using hierarchical acceleration (Section 4.4, Section 4.5). We then consider dynamic remeshing (Section 4.6) and auxiliary penalties and constraints (Section 4.7), which enable a variety of potential applications (Section 4.9); Section 4.8 provides numerical validation.

4.1 Related Work

The tangent-point energy is an example of a *geometric functional*, which assigns a real-valued score $\mathcal{E}(f)$ to each immersion $f : M \longrightarrow \mathbb{R}^n$ of a surface M. Such functionals serve as regularizers in many geometric problems, helping to define a unique solution, or simply making the geometry "nicer" in some sense. For instance, in geometric modeling they are used to smoothly interpolate given boundary data [26], in mathematical visualization they can be used to endow an abstract surface with a concrete geometry [30], and in digital geometry processing they are used for tasks such as hole filling [32] or denoising of measured data [43].

4.1.1 Curvature Functionals

A basic functional for surfaces is total surface area; gradient descent on total area leads to *mean curvature flow*, which has been used for surface denoising [36] but can develop non-smooth singularities or pinch-off artifacts. Though efforts have been made to desingularize this flow [64], sharp peaks and cusps are ultimately impossible to detect from area alone. For this reason, functionals used in geometric modeling typically incorporate curvature information—most prominently the *Willmore energy* $\mathcal{E}_W(f) := \int_M (H^2 - K)\, dA$, where H and K are the mean and Gaussian curvatures, respectively. Significant work has focused on numerical optimization of Willmore energy [16, 33, 38, 102], but since this energy is Möbius invariant, it effectively provides a notion of regularity for surfaces in the 3-sphere S^3, rather than Euclidean $\mathbb{R}^3$. In the context of geometric modeling, this means that even minimizers of Willmore energy can have poor distributions of curvature—see for example Figure 4.3, *bottom left*. Though further energies have been developed to address such issues [63, 83], none of these energies avoid intersections.

4.1.2 Repulsive Forces

Collision response forces from physical simulation [23] and contact mechanics [112] can be used to locally *resolve* contact, but do not help to guide shape optimization toward a state that is far from interpenetration. Moreover, whereas level set representations of geometry ensure (by construction) that surfaces have no self-intersections, the *raison d'être* of such methods is to allow the surface topology to *change*, rather than to preserve it [86]. We instead consider

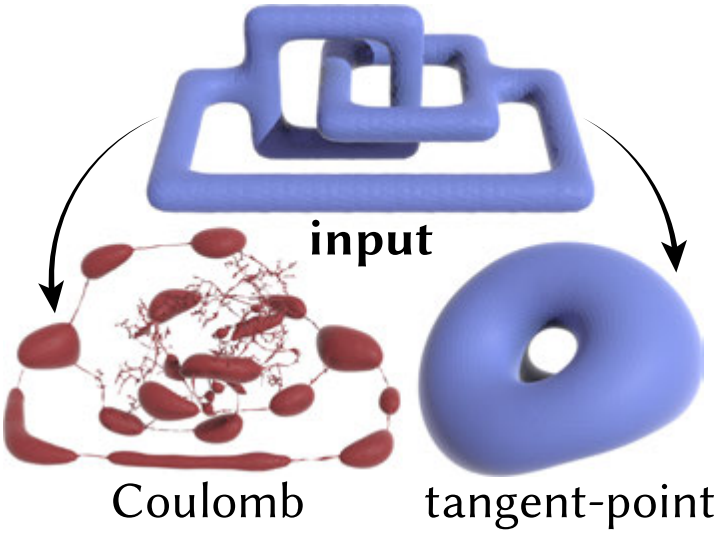


Figure 4.1: Ad-hoc schemes such as vertex-vertex Coulomb forces do not correspond to a meaningful smooth energy, and can be numerically unstable. Here we minimize Coulomb and tangent-point energies subject to a fixed area constraint.

"all-pairs" energies of the form

$$\mathcal{E}(f) = \int_{M \times M} k(x, y)\ dx_f\ dy_f,$$

where dx_f denotes the area element induced by f, and the kernel $k : M \times M \longrightarrow \mathbb{R}$ is designed to discourage self-contact.

We first review some natural possibilities for such energies. The first possibility one might consider is the Coulomb-like potential

$$k_{\text{Coulomb}}(x, y) = \frac{1}{|f(x) - f(y)|^\alpha}$$

for some falloff parameter $\alpha > 0$; on a triangle mesh, this amounts to just penalizing the distance between all pairs of vertices. As noted in Section 2.1.1, however, the resulting energy is too weak to prevent intersections for $\alpha < 2$, and yet ill-defined in the continuum limit for $\alpha \geq 1$. Numerically, ad-hoc vertex-vertex penalties are hence unstable and highly unpredictable (Figure 4.1).

The Möbius energy (Section 2.1.2) regularizes the Coulomb potential by subtracting the contribution of points that are nearby on the surface:

$$k_{\text{Möbius}}(x, y) = \frac{1}{|f(x) - f(y)|^2} - \frac{1}{d(x, y)^2}.$$

While this energy is well-defined and strong enough to prevent intersections (for suitable α), it

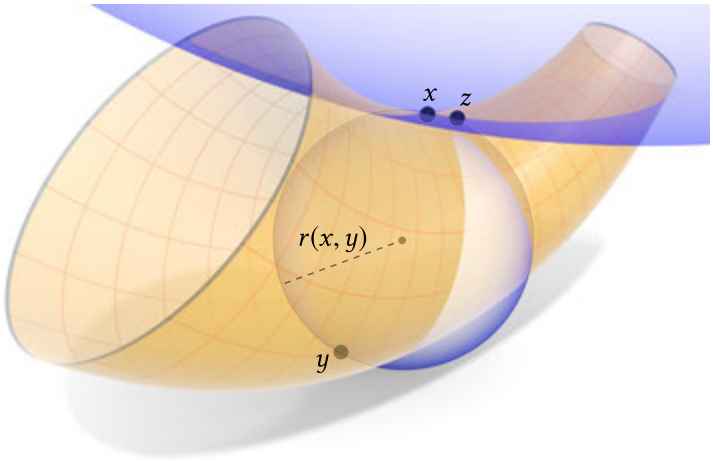


Figure 4.2: For each pair of points x, y on the surface, the tangent-point energy considers the radius $r(x, y)$ of the smallest sphere tangent to x and passing through y, penalizing $1/r(x, y)$. Hence, the contribution will be very large for points y close in space but distant along the surface—and small for points z nearby along the surface, where the radius is huge.

has two significant drawbacks for geometric modeling. First, like Willmore energy, Möbius energy is invariant to Möbius transformations—leading in this case not only to uneven curvature ([70, Figure 5]), but also "tight spots" where points distant in S^3 become arbitrarily close when projected into $\mathbb{R}^3$ (Figure 2.1). Second, the geodesic distance $d(x, y)$, though easy to compute for curves, is prohibitively expensive to compute for all pairs of points on a surface—much less to differentiate with respect to motions of the surface.

4.1.3 Tangent-Point Energy

For all these reasons (as also outlined in Section 2.1), we are prompted to instead consider the *tangent-point energy* introduced for curves by [25] and extended to higher dimensions by [106]. On surfaces as on curves, for each pair of points $x, y \in M$, this energy considers the radius $r(x, y)$ of the smallest sphere tangent to $f(x)$ and passing through $f(y)$ (Figure 4.2). This energy has several features that make it a prime candidate for repulsive surface optimization, namely:

- It provides an infinite barrier to self-intersection [106].
- Like Willmore energy, it penalizes bending (Section 4.2.1), preventing singularities and cusps.
- Unlike Willmore and Möbius energy, it is neither Möbius nor scale invariant, helping to evenly distribute curvature and avoid tight spots.
- Unlike Möbius energy it does not require geodesic distances, and instead depends only on quantities like surface normals N and extrinsic distances $|f(x) - f(y)|$ that are cheap

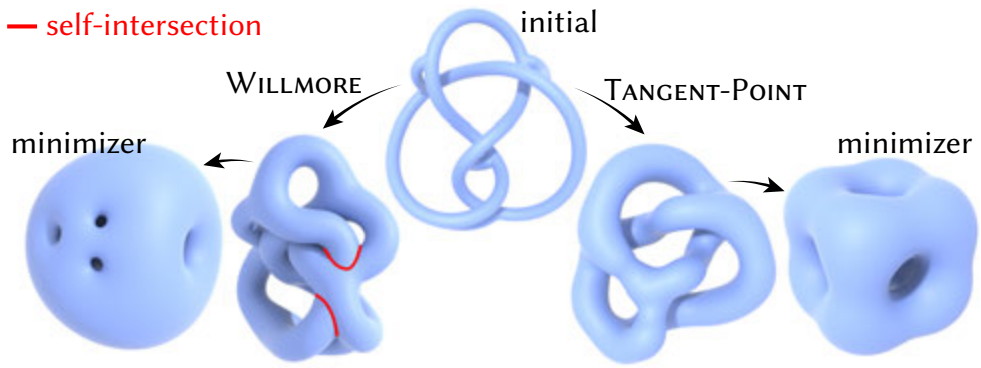


Figure 4.3: Willmore energy does nothing to prevent intersections (in red), and can have minimizers that asymmetrically distribute curvature over the surface. *Right:* tangent-point energy avoids intersections and tends to provide a more uniform curvature distribution.

to compute and easy to differentiate.

However, there are still two significant challenges in applying tangent-point energy to practical surface optimization, namely, (i) picking an inner product that accelerates optimization and (ii) efficiently inverting this inner product.

4.1.4 Accelerating Optimization

To integrate a parabolic gradient flow of order k with average node spacing h, one must typically take time steps of size around $O(1/h^k)$, which is prohibitively expensive for fine meshes. However, one can effectively transform gradient descent into a 0th-order equation by defining the gradient with respect to a different inner product—mitigating the time step restriction. This idea of *Sobolev gradients* has long been applied to surface flows [41, 78, 92, 95, 99, 102], and more recently to elastic energies in geometry processing [31, 68, 120]. However, all this work considers energies with integer-order differentials, whereas the tangent point energy has a differential of *fractional* order, as discussed in Chapter 2. The fractional inner product hence performs far better than even integer Sobolev schemes, especially for finely-tessellated or highly-knotted curves. We adopt the same basic strategy, adapting it to surfaces.

4.1.5 Efficient Evaluation

A second challenge is that there is a dramatic increase in problem size when going from curves to surfaces: rather than integrate an energy over all $O(n^2)$ pairs of elements on a curve, we now must consider $O(n^4)$ element pairs on a surface (where $n \approx 1/h$). Standard hierarchical Barnes-Hut approximation is still sufficient to approximate the energy and its differential (Section 4.4), but we must also invert the fractional Sobolev inner product, which is now a dense matrix with $O(n^4)$ entires. In Section 3.4.3, we used a multigrid solver based on a simple multiresolution curve hierarchy, but building a multiresolution *surface* mesh hierarchy on each optimization

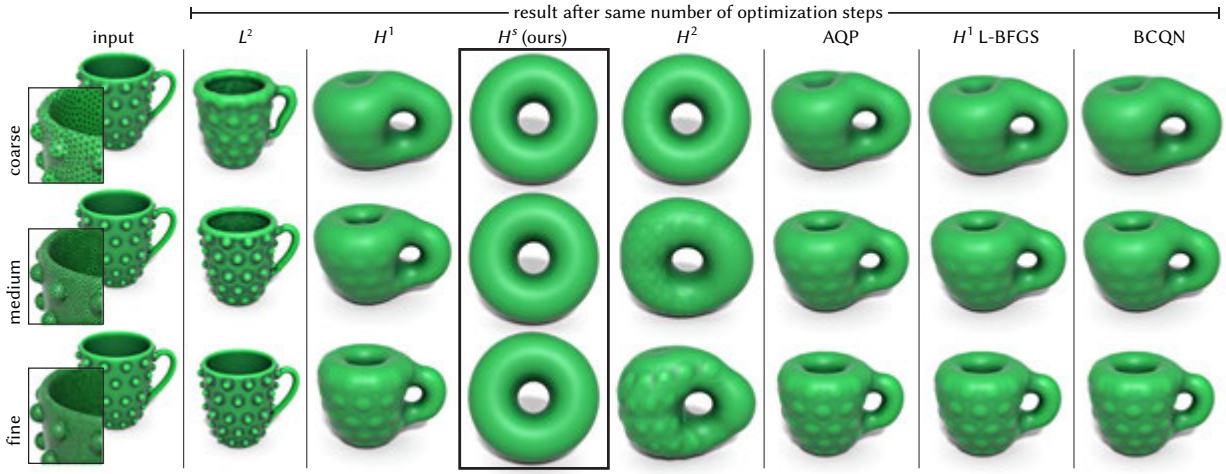


Figure 4.4: Unlike other schemes, our fractional preconditioner does not suffer from a mesh-dependent time step restriction. Here for example we take 300 optimization steps of maximum size (determined by line search) for each scheme. As resolution increases, all methods but H^s make slower and slower progress. Note also that schemes based on H^1 preconditioning (H^1, H^1 L-BFGS, AQP, BCQN) quickly eliminate high-frequency details but are slower to smooth the bulk shape; conversely, H^2 quickly smooths out the bulk shape but fine details remain. Using H^s for $1 < s < 2$ nicely handles both local and global features.

step is far more difficult and expensive. Our key insight is that the inverse of our fractional operator can be approximated by the inverse of two ordinary (integer-order) Laplace operators, together with *forward* application of a lower-order fractional derivative (Section 4.5.2). Since this decomposition is only approximate in the discrete setting, we use it to precondition an iterative linear solver (GMRES) that does not require a mesh hierarchy.

4.2 Smooth Formulation

In this section we define the smooth tangent-point energy $\mathcal{E}^p$, and give some remarks on the order of derivatives appearing in its differential $d\mathcal{E}^p$. Determining the order of the differential is essential to accelerating the gradient flow $\frac{d}{dt}f = -d\mathcal{E}^p(f)$, since it enables us to define a new inner product (in Section 4.2.4) with respect to which the gradient flow effectively becomes a 0th-order equation. Hence, the numerical integrator developed in Section 4.3 will be able to take dramatically larger time steps, of a size that does not depend strongly on mesh resolution (Figure 4.4).

4.2.1 Energy

As discussed in Section 4.1.3, we can define a repulsive energy by considering the *tangent-point radius* $r_f(x, y)$, defined as the radius of the smallest sphere tangent to $f(x)$ and passing through

$f(y)$ (Figure 4.2). Letting $N_f(x)$ be the unit normal at x, this radius can be computed as

$$r_f(x, y) = \frac{|f(x) - f(y)|^2}{2|P_f(x)\,(f(x) - f(y))|}, \tag{4.1}$$

where $P_f(x) = N_f(x)\,N_f(x)^\top$ denotes orthogonal projector onto the normal space at x. Note that expressing r_f via the projector avoids picking a sign for the normal, which will be useful in Section 4.5.1 (it is also valid for submanifolds of arbitrary dimension and codimension). Omitting the constant factor 2, the tangent-point kernel (due to Buck and Orloff [25]) is then given by

$$k_{f,p}(x, y) := \frac{2^p}{r_f(x, y)^p} = \frac{|P_f(x)\,(f(x) - f(y))|^p}{|f(x) - f(y)|^{2p}} \tag{4.2}$$

for some $p > 0$, and hence the energy itself is

$$\mathcal{E}^p(f) := \iint_{M^2} k_{f,p}(x, y)\; dx_f\; dy_f. \tag{4.3}$$

While in principle it is possible to allow the exponents in the numerator and denominator to vary independently [15], we use exponents $p, 2p$ (as above), which simplifies analysis. Note that, because $k_{f,p}$ has units m^{-p} (in meters) and $\mathcal{E}^p$ is a double integral over an n-dimensional manifold, $\mathcal{E}^p$ has units m^{2n-p}. Therefore, $p > 2n$ is required for the energy to be truly repulsive (i.e. to have units corresponding to inverse meters); otherwise, the energy could be reduced to 0 by simply shrinking the domain to a single point. As we deal with surfaces here ($n = 2$), $p > 4$ is sufficient. Unless otherwise noted, we use $p = 6$ for all examples in this paper.

4.2.2 Gradient Flow

Attempting to perform standard L^2 gradient descent on the tangent-point energy yields a flow

$$\frac{d}{dt} f = -d\mathcal{E}^p(f).$$

This flow exhibits poor convergence due to the presence of high-order spatial derivatives on the right-hand side, which even aggressive line search or general-purpose preconditioning (e.g. L-BFGS) cannot alleviate; see Figure 4.4. However, we can obtain a different descent strategy by defining the gradient with respect to a different inner product. In particular, if A is the linear operator defining the inner product, the descent equation becomes

$$\frac{d}{dt} f = -A^{-1} d\mathcal{E}^p(f). \tag{4.4}$$

An optimal choice of A will match the order of the differential, so that the right hand side no longer involves any spatial derivatives (hence avoiding a mesh-based time step restriction). We first establish the order of the differential $d\mathcal{E}^p$ in the surface case (Section 4.2.3), then define a

ALGORITHM 2: Assembly of the exact discrete fractional operator L^σ

initialize $L^\sigma \leftarrow 0$
forall *distinct pairs of faces* S, T **do**
  forall *vertices* i *adjacent to* S *or* T **do**
    forall *vertices* j *adjacent to* S *or* T **do**
      $L^\sigma_{ij} \leftarrow L^\sigma_{ij} + \frac{(\bar{\phi}_i(S)-\bar{\phi}_i(T))(\bar{\phi}_j(S)-\bar{\phi}_j(T))}{|X_f(S)-X_f(T)|^{2\sigma+2}}\, a_f(S)\, a_f(T)$
    end
  end
end
return L^σ

fractional Sobolev inner product that matches this order (Section 4.2.2).

4.2.3 Order of the Differential

Though originally defined for curves, the tangent-point energy $\mathcal{E}^p$ can be formulated for a quite broad class of n-dimensional sets $\Sigma \subset \mathbb{R}^m$ "with tangent planes," that need not even be manifolds [106]. In the case of 2-dimensional surfaces, one can argue (as discussed below) that $d\mathcal{E}^p$ is a nonlocal, nonlinear differential operator of *fractional* order $2(2-2/p) \in]3,4[$, rather than integer order. This distinguishes the tangent-point energy from standard geometric energies like Willmore, and it is why we have to develop special tools for it.

In more detail: Strzelecki and von der Mosel [106] show that if tangent-point energy is finite for some n-dimensional $\Sigma \subset \mathbb{R}^m$, then Σ must be an *embedded* submanifold of Hölder class $C^{1,\alpha}$, where $\alpha = 2 - 2n/p$. Intuitively: it must be free of self-intersections, and also fairly regular. This result is improved by [14], who establishes that $\mathcal{E}^p(\Sigma)$ is finite if and only if Σ is an embedded submanifold of *fractional Sobolev class* $W^{s,p}$, where $s = 2 - n/p$. In particular, this implies that Σ can be expressed as an embedding $f \in W^{s,p}(M;\mathbb{R}^m)$ for some smooth manifold M. For $n = 2$ we have $s \in]3/2, 2[$, so we inevitably have to deal with fractional Sobolev spaces. Knowing the natural habitat of $\mathcal{E}^p$ is key because it allows for the following observation: the differential $d\mathcal{E}^p$ is a mapping from $W^{s,p}$ to the dual space $(W^{s,p})^* = W^{-s,p}$. Hence it is plausible that $d\mathcal{E}^p$ reduces the differentiability of its argument by $2s = 2(2 - 2/p)$, as claimed above.

4.2.4 Inner Product

Standard (integer) Sobolev inner products are expressed via the Laplacian Δ. We likewise consider the *fractional Laplacian* of order $0 < 2\sigma < 2$ on $\mathbb{R}^n$, which can be expressed in integral form up to a constant factor as

$$\langle (-\Delta)^\sigma u, v\rangle_{L^2} = \iint_{\mathbb{R}^n} \frac{(u(x) - u(y))\,(v(x) - v(y))}{|x - y|^{2\sigma+n}}\, dx\, dy \tag{4.5}$$

for sufficiently smooth functions $u, v : \mathbb{R}^n \longrightarrow \mathbb{R}$ [71]. While this formula only relates to $\mathbb{R}^n$, we can obtain an analogous operator L^σ of fractional order 2σ on functions $u,\ v : M \longrightarrow \mathbb{R}$ by mimicking this expression on the n-dimensional manifold M:

$$\langle L^\sigma u, v\rangle_{L^2} = \iint_{M^2} \frac{(u(x) - u(y))\,(v(x) - v(y))}{|f(x) - f(y)|^{2\sigma+n}}\, dx_f\ dy_f. \tag{4.6}$$

Note that, for $p > 2n$, the order of $d\mathcal{E}^p$ is $2s = 2(2 - n/p) > 3$, which is outside the bounds $0 < 2\sigma < 2$. We can "boost" the order of this operator by introducing a first order derivative operator $\mathcal{D}_f$ in the numerator, yielding a "high-order" operator

$$\langle Bu, v\rangle_{L^2} = \iint_{M^2} \frac{\left\langle \mathcal{D}_f u(x) - \mathcal{D}_f u(y), \mathcal{D}_f v(x) - \mathcal{D}_f v(y)\right\rangle}{|f(x) - f(y)|^{2\sigma+n}} dx_f dy_f. \tag{4.7}$$

More precisely, we use $\mathcal{D}_f u(x) := du(x)\, df(x)^\dagger \in \mathrm{End}(\mathbb{R}^m)$, where $df(x)^\dagger \in \mathrm{Hom}(\mathbb{R}^m; T_x M)$ denotes the Moore-Penrose pseudoinverse of $df(x)$. If we now let $\sigma = s - 1$, then the operator B achieves the desired order $2s$.

Low order term. Similarly to the case of curves, we can get even better preconditioning in situations with close contact by adding an additional term of lower order, which in our case translates to

$$\langle B_0 u, v\rangle_{L^2} = \iint_{M^2} \frac{(u(x) - u(y))(v(x) - v(y))}{|f(x) - f(y)|^{2\sigma+n}} k_{f,2}(x, y) dx_f dy_f. \tag{4.8}$$

The inclusion of the tangent-point kernel $k_{f,2}(x, y)$ effectively distorts lengths in regions of high energy: as the local energy increases, so too does the apparent length induced by the inner product. As a result, self-intersecting configurations, having infinite energy, are so distant (if not infinitely so) that they are unlikely to be reached within a finite time. The kernel $k_{f,2}(x, y)$ is chosen here so that B and B_0 have the same units and thus behave similarly under scaling.

The overall operator $A = B + B_0$ will define the inner product we consider throughout this work. The order of this inner product matches that of the Sobolev space $W^{s,2} = H^s$, so we will occasionally use the term H^s to refer to our preconditioner.

4.3 Discretization

Here, we present discretizations of all components needed for our surface optimization scheme. The basic idea is to minimize tangent-point energy by following the gradient flow, preconditioned by our fractional inner product. In practice we will also want to incorporate a variety of constraints, which we do by both projecting the flow direction onto the tangent space of the constraint manifold, and by then projecting the surface itself onto this manifold. The overall algorithm for each descent step can be summarized as:

1. Assemble the derivative $d\hat{\mathcal{E}}^p(f)$ of the energy (Section 4.3.1).

2. Construct the fractional operator $A = B + B_0$ (Section 4.3.2).

3. Solve Equation 4.14 to obtain the descent direction $\mathbf{x}$.

4. Take a step in the direction of $\mathbf{x}$ using Armijo line search.

5. Project the resulting embedding onto the constraint manifold of Φ (Section 4.3.3).

The initial algorithm outlined in this section is of course quite inefficient; we will introduce accelerations in subsequent sections. For the final accelerated algorithm, see Section 4.5.4.

4.3.1 Discrete Energy

On a discrete triangle mesh $M = (V, E, F)$ with embedding $f : V \longrightarrow \mathbb{R}^3$, we evaluate the double integral of Equation 4.3 using simple mid-point quadrature on all faces. We define the *discrete tangent-point kernel* on a pair of faces S, T as

$$K_{f,p}(S, T) = \frac{|P_f(S)\,(X_f(S) - X_f(T))|^p}{|X_f(S) - X_f(T)|^{2p}}, \tag{4.9}$$

where $X_f(S)$ denotes the barycenter of face S under embedding f. The full energy is then defined as a double sum over faces

$$\hat{\mathcal{E}}^p(f) = \sum_{S \in F} \sum_{T \in F} K_{f,p}(S, T)\; a_f(S)\; a_f(T), \tag{4.10}$$

where $a_f(S)$ denotes the area of face S under embedding f. The differential $d\hat{\mathcal{E}}^p(f)$ of this energy with respect to $f \in \mathbb{R}^{3|V|}$ can be obtained via the chain rule.

4.3.2 Discrete Inner Product

The fractional operator L^σ can be discretized as a $|V| \times |V|$ matrix with entries obtained from the right-hand side of Equation 4.6. The rows and columns of L^σ are indexed by vertices, and each entry can naïvely be computed as

$$L^\sigma_{ij} = \sum_{\substack{S \in F \\ S \neq T}} \sum_{T \in F} \frac{(\bar{\phi}_i(S) - \bar{\phi}_i(T))\,(\bar{\phi}_j(S) - \bar{\phi}_j(T))}{|X_f(S) - X_f(T)|^{2\sigma+2}}\; a_f(S)\; a_f(T), \tag{4.11}$$

where ϕ_i denotes the piecewise linear hat-function centered at vertex i, and where $\bar{\phi}_i(S)$ denotes its evaluation on the barycenter of S, i.e. $\bar{\phi}_i(S)$ is 1/3 if vertex i is adjacent to face S and 0 otherwise. Assembling L^σ using Equation 4.11 would require quartic complexity; however, the integrand vanishes for most pairs S, T, so the assembly can be done in quadratic time by only considering nonzero contributions (Algorithm 2).

High- and Low-Order Terms

The high-order matrix B of our inner product (Equation 4.7) can be assembled using the same procedure as in Algorithm 2, simply using the summand

$$\frac{\left\langle \mathsf{D}_f \phi_i(S) - \mathsf{D}_f \phi_i(T), \mathsf{D}_f \phi_j(S) - \mathsf{D}_f \phi_j(T) \right\rangle}{|X_f(S) - X_f(T)|^{2\sigma+2}} \; a_f(S) \; a_f(T) \tag{4.12}$$

in place of the one in Equation 4.11. Here D_f is a discretization of $\boldsymbol{D}_f$. Intuitively, $\mathsf{D}_f u(S)$ is the derivative of the function $u = \sum_{i \in V} u_i \, \phi_i$ within the triangle S, and can be evaluated as

$$-2 \, a_f(S)^{-1} \left(N_f(S) \times (u_i e_{jk} + u_j e_{ki} + u_k e_{ij}) \right)^\top,$$

where i, j, k are the vertices of triangle S and where e_{jk}, e_{ki}, and e_{ij}) denote the unit edge vector of S. The low-order matrix B_0 can be assembled likewise with the summand

$$\frac{(\bar{\phi}_i(S) - \bar{\phi}_i(T)) \, (\bar{\phi}_j(S) - \bar{\phi}_j(T))}{|X_f(S) - X_f(T)|^{2\sigma+2}} \; K_{f,2}(S, T) \; a_f(S) \; a_f(T) \tag{4.13}$$

using the discrete tangent-point kernel $K_{f,2}(S, T)$ from Equation 4.9. We can then assemble $A = B + B_0$ by assembling both terms. The matrix A is $|V| \times |V|$ and thus applies to scalar functions, but we can construct a corresponding operator A_3 on vector-valued functions $u, v : V \longrightarrow \mathbb{R}^3$ by replacing each entry A_{ij} with the 3×3 block $A_{ij} \, \mathrm{I}_{3\times 3}$, thus obtaining a matrix of size $3\,|V| \times 3\,|V|$.

4.3.3 Constraints

Gradient Projection

Like the integer vector Laplacian, our operator A_3 possesses a nullspace consisting of uniform translations. A simple way to eliminate this nullspace is to define a constraint function $\Phi : \mathbb{R}^{3|V|} \longrightarrow \mathbb{R}^k$ and require (with some abuse of notation) that $\Phi(f) = 0$. The constrained descent direction $\mathbf{x}$ can then be obtained by solving the saddle point problem

$$\begin{bmatrix} A_3 & d\Phi(f)^\top \\ d\Phi(f) & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} -d\hat{\mathcal{E}}^p(f) \\ 0 \end{bmatrix}, \tag{4.14}$$

where $d\Phi(f)$ denotes the Jacobian of Φ. Assuming a suitable Φ is chosen, this saddle point matrix is invertible, and the result will be tangent to the constraint manifold $\{f \mid \Phi(f) = 0\}$. Beyond just eliminating nullspaces, such constraints can also be used to achieve design objectives such as control of areas or volumes.

Corrective Projection

The descent direction $\mathbf{x}$ obtained in Section 4.3.3 is tangent to the constraint manifold, but this does not prevent the embedding f itself from drifting away from the constraint manifold. To counteract this, after we have found a feasible step size $\tau > 0$ via line search, we project the current state $f + \tau\,\mathbf{x}$ back onto the constraint manifold of Φ. We reuse the left-hand side of Equation 4.14 and solve

$$\begin{bmatrix} A_3 & d\Phi(f)^\top \\ d\Phi(f) & 0 \end{bmatrix} \begin{bmatrix} \mathbf{h} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ -\Phi(f + \tau\,\mathbf{x}) \end{bmatrix} \tag{4.15}$$

to obtain a Newton step $\mathbf{h}$, which we add to the updated embedding $f + \tau\,\mathbf{x}$. With respect to the metric encoded by A_3, $\mathbf{h}$ is the least-norm solution of the linear equation $d\Phi(f)\,\mathbf{h} = -\Phi(f)$. This correction can be repeated several times if the constraint violation is not sufficiently close to 0. For the constraints we explored, however, a single step was always sufficient.

4.4 Fast Energy and Derivative Evaluation

The naïve algorithm of Section 4.3 is bottlenecked by several operations of at least quadratic complexity. The first such bottleneck is the evaluation of the energy and its derivative, which requires iteration over all pairs of elements. We thus use a Barnes-Hut hierarchical approximation [9] to evaluate the tangent-point energy $\hat{\mathcal{E}}^p$ and its derivative $d\hat{\mathcal{E}}^p$.

4.4.1 Approximate Energy

The kernel $K_{f,p}(S, T)$ (Equation 4.9) only requires three quantities to evaluate: the barycenters of S and T, and the normal projector of S. We can make this dependence clearer by rewriting it as $K_{f,p}(S, T) = K_p(X_f(S), P_f(S); X_f(T))$, with

$$K_p(X, P; Y) := \frac{|P\,(X - Y)|^p}{|X - Y|^{2p}}.$$

We can then hierarchically approximate the all-pairs interactions of K_p. We construct a bounding-volume hierarchy (BVH) on the face set F, where each node $\mathcal{I}$ computes the total area $a_{\mathcal{I}}$, the barycenter $X_{\mathcal{I}}$, and the average projector $P_{\mathcal{I}}$ (corresponding to the average normal) of its elements. To reduce the number of nodes, we stop splitting leaf nodes once they have l or fewer elements ($l = 8$ in our experiments). For a given $\theta \geq 0$, we say that $\mathcal{I}$ is *admissible* with respect to S if (1) it is a leaf node or if (2) it satisfies

$$\max(r(S), r(\mathcal{I})) < \operatorname{dist}(S, \operatorname{conv}(\mathcal{I})).$$

Here $r(S)$, $r(\mathcal{I})$ are the radii of the triangle S and the node $\mathcal{I}$, respectively, both measured from their barycenters; dist denotes the minimal Euclidean distance between two sets; and $\operatorname{conv}(\mathcal{I})$

denotes the convex hull of $\mathcal{I}$. In practice, we approximate these quantities by replacing the node $\mathcal{I}$ by its axis-aligned bounding boxes, leading to a slightly stricter admissability condition. Then, adm(S) is the set of all admissible nodes with respect to S with no admissible ancestors. The energy evaluation then becomes the sum

$$\widetilde{\mathcal{E}}^p(f) = \sum_{S \in F} \sum_{\mathcal{I} \in \mathrm{adm}(S)} K_p(X_f(S), P_f(S); X_{\mathcal{I}})\ a_f(S)\ a_{\mathcal{I}}. \tag{4.16}$$

The separation parameter θ controls the approximation quality; the higher θ is, the faster the computation, but the less accurate the result. For $\theta = 0$, the sum degenerates to an all-pairs exact computation. Unless otherwise noted, we use $\theta = 0.5$ for all experiments.

4.4.2 Approximate Derivative

Computing an approximate derivative with Barnes-Hut is not entirely analogous to computing the energy. For each vertex $v \in V$, we evaluate the sum

$$\widetilde{\partial_v \hat{\mathcal{E}}}^p(f) = \sum_{S \in F(v)} \sum_{\mathcal{I} \in \mathrm{adm}(S)} h_f(S, \mathcal{I})$$

where $F(v)$ denotes the set of faces containing v, and where

$$h_f(S, \mathcal{I}) = \partial_v \left[K_p \left(X_f(S), P_f(S); X_{\mathcal{I}} \right)\ a_f(S)\ a_{\mathcal{I}} \right] + \partial_v \left[K_p \left(X_{\mathcal{I}}, P_{\mathcal{I}}; X_f(S) \right)\ a_f(S)\ a_{\mathcal{I}} \right]$$

This approximates both the forward and reverse terms that would be differentiated by v in an exact computation. Note that the outer sum over all $S \in F(v)$ for both energy and derivative evaluations can be evaluated as a parallel reduction without modification.

4.5 Iterative Linear Solver

An even more significant bottleneck than the energy is the dense saddle point problem of Equation 4.14. Rather than solving this problem via dense matrix inversion, we will solve it instead using GMRES, an iterative method. In general, efficient iterative methods require two key ingredients: fast matrix-vector products, and effective preconditioners. Here, we will describe methods for both.

4.5.1 Hierarchical Matrices

We use hierarchical matrices [53] to perform fast multiplication with A without explicitly assembling the matrix. In this section, we present the special case of rank-1 compression of kernel matrices, while noting that the original method can also perform higher-rank approximations. In our setting, a *kernel matrix* H is a matrix of size $|F| \times |F|$ whose entries are defined by

$$H_{ST} = (1 - \delta_{ST})\, h(X_f(S), P_f(S); X_f(T), P_f(S)),$$

where $h : (\mathbb{R}^m \times \mathrm{End}(\mathbb{R}^m)) \times (\mathbb{R}^m \times \mathrm{End}(\mathbb{R}^m)) \longrightarrow \mathbb{R}$ is a suitable kernel function. To motivate this approach, we first reduce the actions of the operators L^σ, B, and B_0 to the multiplication with certain kernel matrices.

Applying the operator L^σ

An elementary computation shows (see Appendix A.1) that the action of the discrete linear operator L^σ on a vector $\mathbf{v} \in \mathbb{R}^{|V|}$ can be written as

$$L^\sigma \mathbf{v} = 2\,\mathsf{U}^\top \left[\operatorname{diag}\left(\operatorname{diag}(a_f)^{-1} H\, a_f \right) - H \right] \mathsf{U}\,\mathbf{v}.$$

Here a_f is the $|F|$-vector of face areas; U is the $|F| \times |V|$-matrix that averages values on vertices onto faces and multiplies with the face areas; and H is the kernel matrix of size $|F| \times |F|$ to the singular kernel $h(X, P; Y, Q) = |X - Y|^{-(2\sigma+2)}$. U is sparse, so we just need an efficient product with H to evaluate the full product with L^σ.

Applying the High-Order Term

To evaluate a matrix-vector product with $A = B + B_0$, it suffices to evaluate B and B_0 separately. This can be done in a similar fashion as for L^σ. For the higher order term B, we have the identity

$$B\mathbf{v} = 2\,\mathsf{V}^\top \left[\operatorname{diag}\left(\operatorname{diag}(a_f)^{-1} H\, a_f \right) - H \right] \mathsf{V}\,\mathbf{v},$$

where $\mathsf{V} = \operatorname{diag}(a_f)\,\mathsf{D}_f$ with the discrete derivative operator D_f described in Section 4.3.2 and where the kernel h of the kernel matrix H is given by $h(X, P; Y, Q) = |X - Y|^{-(2(s-1)+2)}$.

Applying the Low-Order Term

Likewise, we can write the action of B_0 as

$$B_0\mathbf{v} = 2\,\mathsf{V}^\top \left[\operatorname{diag}\left(\operatorname{diag}(a_f)^{-1} H\, a_f \right) - H \right] \mathsf{V}\,\mathbf{v},$$

where the kernel h of the kernel matrix H is given by

$$h(X, P; Y, Q) = \frac{k_2(X, P; Y) + k_2(Y, Q; X)}{2|X - Y|^{2(s-1)+2}}.$$

Block Cluster Tree

In order to compress these kernel matrices, we reuse the BVH from Section 4.4, but additionally compute the average projector $P_{\mathcal{I}} := a_{\mathcal{I}}^{-1} \sum_{S \in \mathcal{I}} a_f(S)\, P_f(S)$ for each node $\mathcal{I}$. From this, we construct a *block cluster tree*, whose nodes (termed *block clusters*) consist of pairs of BVH nodes (termed *clusters in the following*). For a given separation parameter $\chi \geq 0$, we say that two BVH

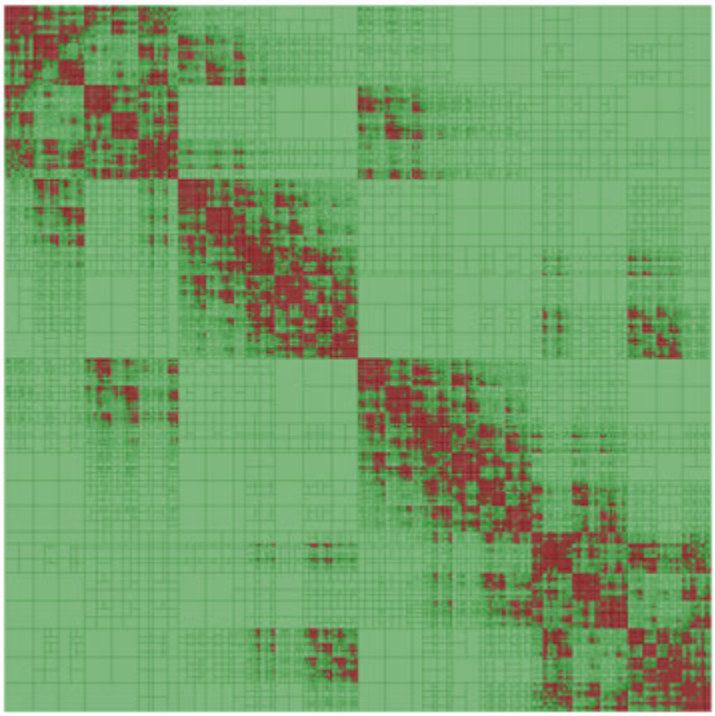


Figure 4.5: A block decomposition of a kernel matrix H induced by a block cluster tree. Admissible blocks are shown in green, while inadmissible blocks are in red.

clusters $\mathcal{I}$ and $\mathcal{J}$ are an *separated pair* if

$$\max\left(r(\mathcal{I}), r(\mathcal{J})\right) \leq \chi \operatorname{dist}(\operatorname{conv}(\mathcal{I}), \operatorname{conv}(\mathcal{J})).$$

Here again, $r(\mathcal{I})$, $r(\mathcal{J})$ are the radii of the nodes $\mathcal{I}$, $\mathcal{J}$ as measured from their barycenters. The parameter χ controls the accuracy of the approximation; it will be discussed further in the next section. Then, denoting the BVH root by $\mathcal{R}$, we construct the block cluster tree by starting with the single pair $(\mathcal{R}, \mathcal{R})$, and iteratively splitting nonseparated nodes $(\mathcal{I}, \mathcal{J})$ into the Cartesian products of their constituents' children until all leaf nodes are either separated or cannot be split any further. In practice, the tree structure is not important to maintain; only the lists of leaf nodes matter. We refer to the separated leaf nodes of the block cluster tree as *admissible blocks* and to the others as *inadmissible blocks*; Figure 4.5 illustrates the decomposition of the full matrix into these blocks.

Hierarchical Multiplication

The block cluster tree allows us to perform approximate multiplication with a *kernel matrix* H as follows. Every pair of BVH clusters $(\mathcal{I}, \mathcal{J})$ corresponds to a block of H with rows indexed by $\mathcal{I}$ and columns by $\mathcal{J}$. Let $H_{\mathcal{I}\mathcal{J}}$ denote this matrix block and let $\mathbf{x}_{\mathcal{I}}$ and $\mathbf{1}_{\mathcal{I}}$ denote the slices of $\mathbf{x} \in \mathbb{R}^{|F|}$ and of the all-ones vector indexed by $\mathcal{I}$, respectively. Then, for all leaf blocks $(\mathcal{I}, \mathcal{J})$,

we compute the product $\mathbf{y} = H\,\mathbf{x}$ in two steps:

1. If $(\mathcal{I}, \mathcal{J})$ is inadmissible, then we multiply exactly:

$$\mathbf{y}_{\mathcal{I}} \leftarrow \mathbf{y}_{\mathcal{I}} + H_{\mathcal{I}\mathcal{J}}\,\mathbf{x}_{\mathcal{J}}.$$

2. If $(\mathcal{I}, \mathcal{J})$ is admissible, we employ rank-one approximation:

$$\mathbf{y}_{\mathcal{I}} \leftarrow \mathbf{y}_{\mathcal{I}} + \mathbf{1}_{\mathcal{I}}\, h(X_{\mathcal{I}}, P_{\mathcal{I}}; X_{\mathcal{J}}, P_{\mathcal{J}})\, \mathbf{1}_{\mathcal{J}}^{\top}\,\mathbf{x}_{\mathcal{J}}.$$

Here, we can see more clearly the effect of χ. For $\chi = 0$, all blocks are considered inadmissible, and the action of H is evaluated exactly. For $\chi > 0$, the larger the value, the more blocks will be considered admissible and thus multiplied using the fast approximation in Step 2, leading to faster evaluation time – but also higher error, analogous to the θ parameter for Barnes-Hut. For our experiments, we found $\chi = 0.5$ to be a broadly acceptable value. Note that, while a straightforward implementation of these two steps is sufficient to evaluate the product, a much faster implementation can be obtained by employing multipole methods; see Appendix A.2 for details.

4.5.2 Preconditioner

While we can now evaluate matrix-vector products with A efficiently, this alone does not generally allow us to efficiently solve $Ax = b$. We further require a *preconditioner* whose action can be computed efficiently. As we never construct A, classical preconditioners such as incomplete Cholesky factorizations or even $\mathrm{diag}(A)$ are unusuable. Instead, we note that our operator A is closely related to the fractional Laplacian $(-\Delta_M)^s$, and has the same order $2s$. Assembling $(-\Delta_M)^s$ is infeasible, but we can obtain a cheap approximation of its inverse $(-\Delta_M)^{-s}$ by factoring it as

$$(-\Delta_M)^{-s} = (-\Delta_M)^{-1}(-\Delta_M)^{2-s}(-\Delta_M)^{-1},$$

where the two occurrences of the integer Laplace-Beltrami operator $(-\Delta_M)$ can then be replaced by the sparse cotan-weighted Laplace-Beltrami operator on meshes. What remains is a forward application of the fractional Laplacian $(-\Delta_M)^{2-s}$, to which we do not have direct access. Fortunately, since $0 < 2 - s < 2$ holds, we can replace $(-\Delta_M)^{2-s}$ with L^{2-s} (as per Section 4.2.4), whose action can efficiently approximated by Section 4.5.1. Thus, if we first pre-factorize $(-\Delta_M)$, we can apply our preconditioner

$$\widetilde{A}^{-1} := (-\Delta_M)^{-1} L^{2-s} (-\Delta_M)^{-1}$$

with just two back-substitutions and one hierarchical matrix-vector product per application, all of which can be evaluated reasonably efficiently. Note that, despite having the same order as our operator A (and therefore our energy), $\widetilde{A}^{-1}$ is not suitable for direct use as the inner product: as a direct approximation of the inverse operator (as opposed to the forward operator), it cannot

be added with other inner product terms such as those of Equation 4.8 or Section 4.7.3. As a preconditioner for GMRES, however, it is highly effective, allowing us to invert A (plus any auxiliary terms) efficiently.

4.5.3 Schur Complement

While we are now capable of solving the *unconstrained* problem $Ax = b$ iteratively, this does not immediately allow us to solve the saddle point problem (Section 4.3.3). While the method can be applied, we empirically found that it exhibited poor convergence when used on the constrained system. We instead use the *Schur complement* [119] to handle the additional rows. Let M be the saddle point matrix:

$$M := \begin{bmatrix} A_3 & d\Phi^\top \\ d\Phi & 0 \end{bmatrix}$$

Then, the Schur complement of M with respect to A_3 is given by

$$(M/A) = -d\Phi \left(A_3^{-1} d\Phi^\top\right) \tag{4.17}$$

Note that it is useful to cache $A_3^{-1} d\Phi^\top$ here for future reuse. Expressions for each block of M^{-1} are then given as

$$\begin{bmatrix} A_3^{-1} + \left(A_3^{-1} d\Phi^\top\right)(M/A_3)^{-1} d\Phi\, A_3^{-1} & -\left(A_3^{-1} d\Phi^\top\right)(M/A)^{-1} \\ -(M/A)^{-1}\left(A_3^{-1} d\Phi^\top\right)^\top & (M/A)^{-1} \end{bmatrix}$$

A_3^{-1} can be applied using the iterative method just outlined; a product with A_3 is equivalent to three separate products with A. The complement M/A is dense, but it has dimensions $k \times k$, corresponding to the number of scalar constraints. As long as k is a small constant, $(M/A)^{-1}$ can be computed quickly. Thus, all blocks of M^{-1} can be computed without having to invert a large matrix. Further, to obtain the constrained descent direction $\mathbf{x}$, we only require the top-left block. Let $\mathbf{g} := d\hat{\mathcal{E}}^p(f)$; then, we can compute the descent direction by directly applying the top-left block to $\mathbf{x}$, producing

$$\mathbf{x} = A_3^{-1}\mathbf{g} + \left(A_3^{-1} d\Phi^\top\right)(M/A)^{-1} d\Phi \left(A_3^{-1}\mathbf{g}\right). \tag{4.18}$$

Equation 4.17 requires one application A_3^{-1} per row of $d\Phi$. Equation 4.18 contains three occurrences of A_3^{-1}, but $A_3^{-1}\mathbf{g}$ can be reused in both places where it appears, and $A_3^{-1} d\Phi^\top$ can be reused from its earlier computation in Equation 4.17. Thus, the method requires $k + 1$ iterative solves, where k is the number of constraints. In our examples, we never have $k > 2$, so the cost remains acceptable.

Corrective Projection

We similarly use the Schur complement to solve Equation 4.15 for the corrective step $\mathbf{h}$. Only the top-right block of the Schur complement is needed, giving the expression

$$\mathbf{h} = -\left(A_3^{-1} d\Phi^\top\right)(M/A)^{-1}(-\Phi(f)). \tag{4.19}$$

(M/A) does not need to be recomputed, and $A_3^{-1} d\Phi^\top$ can again be reused. Thus, constraint projection incurs no significant costs.

4.5.4 Accelerated Algorithm Overview

The accelerated algorithm is as follows:

1. Assemble the (approximate) derivative $d\hat{\mathcal{E}}^p(f)$ of the energy using Barnes-Hut (Section 4.4).
2. Construct a BVH that partitions the faces of the mesh, and use it to create a block cluster tree (Section 4.5.1).
3. Use the Schur complement to solve the constrained saddle point problem (Equation 4.14).
   (a) Evaluate products with $(A_3)^{-1}$ by using a matrix-free iterative method (e.g. GMRES), with the preconditioner from Section 4.5.2, and an initial guess of 0.
   (b) Within the iterative method, evaluate products with A using the block cluster tree (Section 4.5.1, Section 4.5.1).
4. Take a step in the direction of $\mathbf{x}$ using standard line search.
5. Reuse the Schur complement to project the resulting embedding onto the constraint manifold of Φ (Section 4.5.3).

If no constraints are imposed, then the algorithm can be simplified: step 3 can be replaced by a single iterative solve $A_3 x = b$, and step 5 can be omitted entirely.

4.6 Dynamic Remeshing

Minimizing the tangent-point energy often induces large surface deformations that degrade triangle inequality. We therefore use a dynamic remeshing scheme similar to the approach of Chen and Holst [28]. The exact algorithm we use is as follows:

1. Edges with length greater than $3L_0/2$ are split and edges with length smaller than $L_0/2$ are collapsed, unless this operation would result in triangle foldover.
2. For N iterations:
   (a) All edges that violate the Delaunay condition are flipped until no such flippable edges can be found.

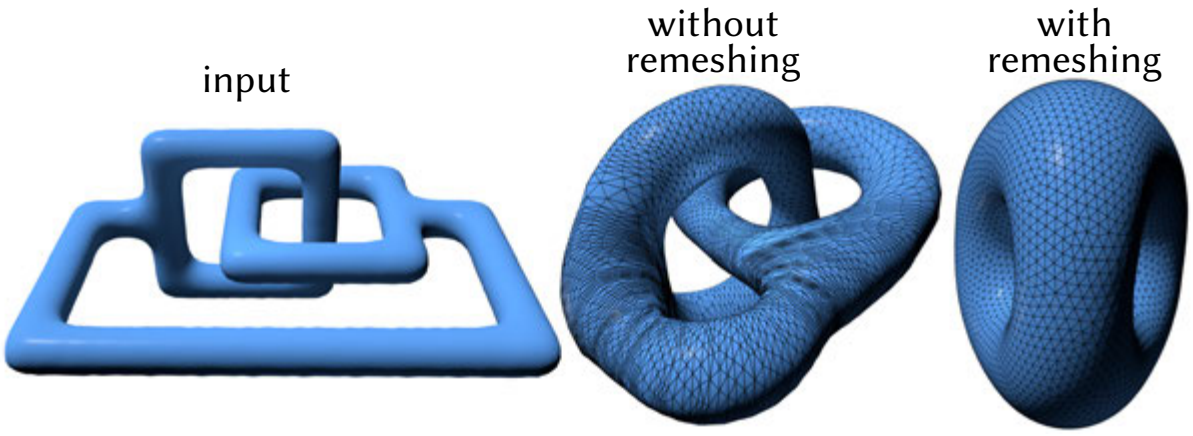


Figure 4.6: Adaptive remeshing not only improves element quality—it also helps to avoid local minima where the surface gets "stuck."

(b) Vertex positions are smoothed by computing a displacement vector from neighboring triangles

$$u_i = \rho \frac{\sum_{S \in F(i)} a_f(S)(c_f(S) - f(i))}{\sum_{S \in F(i)} a_f(S)}.$$

Here $F(i)$ denotes the set of faces containing vertex i, $c_f(S)$ is the circumcenter of the triangle S, and $\rho < 1$ is a constant. This displacement is projected onto the tangent space of the vertex and added to the original position.

Our implementation uses $\rho = 0.5$ and $N = 5$; L_0 is set to the average edge length of the initial mesh and remains constant throughout. We apply this remeshing procedure at the end of each iteration, after the final step of Section 4.5.4. Remeshing is crucial to reaching minimizers of the tangent-point energy; without it, degrading triangle quality can impede or even halt progress, as seen in Figure 4.6.

4.7 Constraints and Penalties

A variety of constraints and penalties can be imposed on the tangent-point energy, both for regularization of minimizers and for specific design purposes. In this section, we discuss the constraints and penalties that we have investigated; more are certainly possible, and in particular, combining the tangent-point energy with other classical surface energies could make for interesting future work.

4.7.1 Constraints

We consider four types of constraints: fixed barycenter, vertex pins, total area, and total volume.

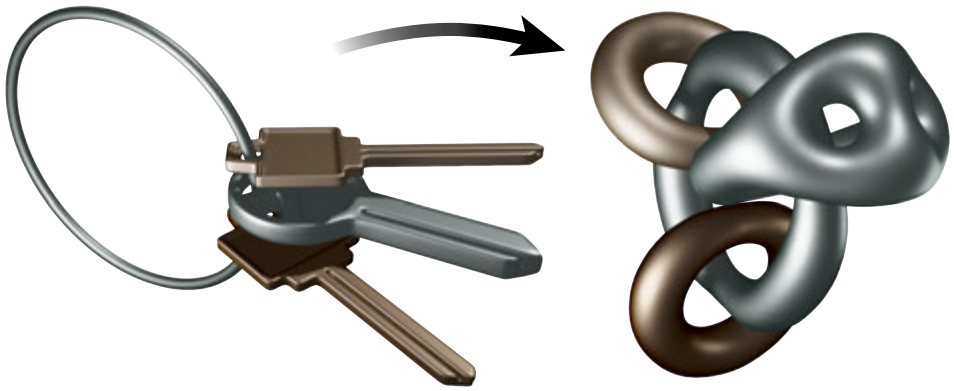


Figure 4.7: To handle multiple components (as shown here), we fix the barycenter of each one during preconditioning, then add back in the mean motion of each component from the original L^2-gradient after the solve.

Fixed Barycenter Constraint

A fixed barycenter constraint can be defined as

$$\Phi_C(f) = \frac{\sum_{i \in V} f(i)\, a_f(i)}{\sum_{i \in V} a_f(i)} - X_0,$$

where X_0 is the target barycenter location and $a_f(i)$ denotes the area associated to vertex i. Its Jacobian $d\Phi_C$ is a $3 \times 3|V|$ matrix consisting of $|V|$ copies of the 3×3 identity matrix appended horizontally. This constraint primarily serves to eliminate the nullspace of the fractional Laplacian (Section 4.3.3); either a barycenter constraint or at least one pin constraint must be added to every problem to be well-posed. For domains with multiple components, barycenters are constrained separately for each component.

Barycenter Motions. In some cases, it might be desirable to allow the barycenter to float freely, e.g. when a scene contains fixed obstacles for the surface to avoid. A simple modification enables this motion: compute the weighted average over all vertices of the L^2 gradient before projection, and then add the constant translation by that vector back to the descent direction after projection. For domains with multiple components, the average motion is computed separately for each component (Figure 4.7).

Vertex Pin Constraints

A vertex pin constraint simply fixes a vertex to a position. Every pinned vertex i produces a constraint function $\Phi_{P_i}(f) = f(i) - f_0(i)$, where $f_0(i)$ is the pinned position. The Jacobian $d\Phi_{P_i}$ is a $3 \times 3|V|$ matrix, but the only nonzero entries consist of a single copy of the identity matrix in the block indexed by i. A pin also eliminates the nullspace of the Laplacian, so if any pins are used, then a barycenter constraint is unneeded.

Total Area Constraint

A total area constraint preserves the total surface area of the mesh, and can be written as

$$\Phi_A(f) = \left(\sum_{T \in F} a_f(T)\right) - A_0,$$

where A_0 is the target area. The Jacobian $d\Phi_A$ is a $3|V|$ row vector with the area gradient at each vertex, which is equivalent to twice the mean curvature normal.

Total Volume Constraint

Likewise, a total (signed) volume constraint can be written as

$$\Phi_A(f) = \frac{1}{6}\Big(\sum_{(ijk) \in F} f(i) \cdot (f(j) \times f(k))\Big) - V_0,$$

where V_0 is the target volume. For each vertex, the Jacobian $d\Phi_A$ is proportional to the area-weighted vertex normal.

4.7.2 Fast Positional Constraints

As previously discussed, computing the Schur complement requires one iterative solve per row of the constraint block $d\Phi$. For linear positional constraints such as barycenters (3 rows per component) and vertex pins (3 rows per pinned vertex), this can be disproportionately expensive. Rather than handling these rows using the Schur complement, we include them directly in the matrix A, producing a smaller saddle point matrix with structure analogous to Equation 4.14. Forward matrix-vector products for the iterative solve require only sparse products with $d\Phi_C$ and $d\Phi_{P_i}$ in addition to the hierarchical products of Section 4.5.1. The same rows and columns are then appended to the integer Laplacians in the preconditioner (Section 4.5.2), and the system is solved iteratively as before.

Fast convergence in this scenario requires that orthogonality to these constraints be sufficiently similar under the two inner products defined by the integer Laplacian Δ and the fractional operator A. Empirically, this is the case for linear positional constraints, but is *not* the case for constraints such as total area and volume. Thus, we reserve the Schur complement for these more difficult constraints.

4.7.3 Penalties

In addition to hard constraints, a number of soft penalty potentials can be added to regularize the flow in some way. These potentials are added directly to the objective function with some weighting coefficient alongside the tangent-point energy, and their gradients are accumulated in the same step.

Total Area and Volume Potentials

Soft penalties for total area and volume can be used in place of hard constraints, encouraging these quantities to stay close to their initial values without enforcing this exactly. For total area, the potential is defined as

$$\mathcal{E}_{\text{area}}(f) = \left(\left(\sum_{T \in F} a_f(T)\right)/A_0 - 1\right)^2.$$

The raw deviation is normalized by the initial area A_0 to make the penalty scale invariant. The total volume potential is defined analogously.

Static Obstacles

For practical modeling purposes, it may be desirable not to design an object in isolation, but instead to design it within its intended environment. To that end, we provide the ability to place "obstacles", which are static meshes that exert a repulsive force on the optimization surface. These obstacles can be used to model surrounding environments such as rooms and the objects within them, which must be avoided by the object under design. From an obstacle O with embedding f_O, each point x in the domain experiences a repulsive potential equal to

$$\mathcal{E}_{\text{obs}}(x) = \sum_{S \in F_O} |f_O(S) - x|^{-p} \, a_{f_O}(S)$$

with p matching the exponent of the tangent-point energy. Naïvely, this requires iteration over all faces of O, but Barnes-Hut can be used as in Section 4.4 to approximate the obstacle potential.

Implicit Obstacles and Attractors

Similarly to static mesh obstacles, one can also use implicit surfaces defined by signed distance fields as obstacles or attractors. Given a signed distance field $d : \mathbb{R}^3 \longrightarrow \mathbb{R}$, the repulsive potential experienced at any point x due to the implicit obstacle defined by $d(x) = 0$ is simply

$$\mathcal{E}_i(x) = d(x)^{-p}.$$

An implicit attractor, rather than repelling other objects away from it, pulls objects towards it. The attractive potential experienced at any point x is simply the reciprocal of the above, or

$$\mathcal{E}_a(x) = d(x)^{p}.$$

Boundary Length and Curvature

For meshes with boundary (e.g. Figure 4.18), it may be beneficial to regularize the shape of the boundary curves. We support two potentials for this purpose. One is a regularizer on the length

of the boundary, defined as

$$\mathcal{E}_b = \left(L - \sum_{e \in \partial M} l(e)\right)^2,$$

where L is a target boundary length, and $l(e)$ is the length of boundary edge e. The other regularizes the curvature, and is defined as

$$\mathcal{E}_c = \sum_{v \in \partial M} \theta(v)^2 / \ell(v),$$

where $\theta(v)$ is the turning angle at vertex v, and $\ell(v)$ is the dual length (i.e. half the length of the two incident edges).

Willmore Energy

One can also add surface fairing energies such as the Willmore energy. For example, we use the following discrete variant of the squared mean curvature integral:

$$\mathcal{E}_{\text{Willmore}}(f) = f^\top \mathbf{A}\, \mathbf{M}^{-1} \mathbf{A}\, f.$$

Here $\mathbf{A}$ is the stiffness matrix of the cotan Laplacian and $\mathbf{M}$ is the lumped mass matrix. Up to mass lumping, this is the discrete Willmore energy from [40]. As suggested in [41, 99], we add an H^2 inner product term $\mathbf{A}\,\mathbf{M}^{-1}\,\mathbf{A}$ to the matrix that we invert in Section 4.5.

4.8 Evaluation and Comparisons

4.8.1 Consistency Testing

Evaluating convergence of our discretization and approximation scheme to minimizers is not straightforward, since to date there are only conjectures about what minimal solutions might look like (Section 4.9.1). Instead, we numerically investigate the *consistency* of our energy discretization: We generate several smooth surfaces, compute their true tangent-point energies, and compare to our discrete energy and its Barnes-Hut approximation.

The exact energy can be computed directly only for very simple shapes, like a round sphere or torus of revolution. To get a more generic picture, we took the parameterized torus of revolution $f_0(\phi, \theta) = \left((1 + \frac{1}{3}\cos(\phi))\,\cos(\phi), (1 + \frac{1}{3}\cos(\theta))\,\sin(\phi), \frac{1}{3}\,\sin(\theta)\right)$ and perturbed it by a random trigonometric polynomial $\Phi : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ of small magnitude (to ensure embeddedness) and small order (to obtain moderate curvature) to obtain the final smooth surface $f := f_0 + \Phi \circ f_0$. We computed $\mathcal{E}^p(f)$ up to 6 digits of precision by numerical integration with *Mathematica*'s `NIntegrate` command using the `"LocalAdaptive"` strategy. Afterwards, we computed an affinely squeezed Delaunay triangulation of $[0, 2\pi] \times [0, 2\pi]$ and used it to sample the surface f. The remaining nonuniformities in triangle size and aspect ratio were repaired by the remeshing routine from Section 4.6 followed by projecting each resulting vertex position back to the

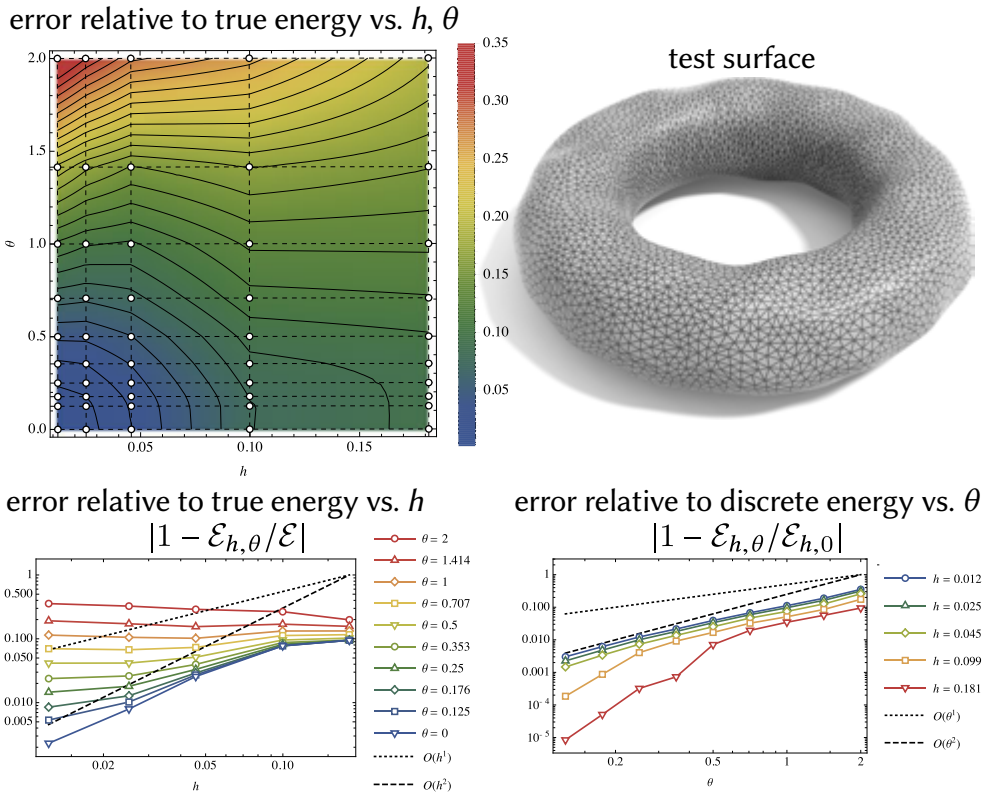


Figure 4.8: Empirically, our discrete tangent-point energy appears to converge to the true smooth energy at a rate somewhere between $O(h)$ and $O(h^2)$; as expected, our Barnes-Hut approximation also converges to the discrete energy as $\theta \to 0$. Reference values are obtained by applying highly accurate numerical integration to the tangent-point energy on a smooth parameterized surface (triangulated in *top right*).

surface f. For the resulting discrete surface f_h we computed its Barnes-Hut energy $\widetilde{\mathcal{E}}^p(f_h)$ (see Equation 4.16) for various values of the separation parameter θ; in the case $\theta = 0$, this is the all-pairs energy $\hat{\mathcal{E}}(f_h)$ from Equation 4.10. The resulting relative errors are shown in Figure 4.8.

The discrete energy $\hat{\mathcal{E}}^p(f_h)$ employs the face normals, which are known to be consistent of order 1 only. That means, their error is $O(h)$, where $h > 0$ denotes the longest edge length. So it is expected that the discretization error $e_h := |\hat{\mathcal{E}}^p(f_h) - \mathcal{E}^p(f)|$ is no better than $O(h)$. Surprisingly, the experiments show that the numerical rate is considerably better (see Figure 4.8, *bottom left* and for $\theta = 0$). Moreover, we use center of mass data on BVH nodes; so the deviation $e_{h,\theta} := |\widetilde{\mathcal{E}}^p(f_h) - \hat{\mathcal{E}}^p(f_h)|$ of the Barnes-Hut approximation from the discrete energy should be dominated by the midpoint rule's consistency error which is $O(\theta^2)$. Indeed our experiments seem to confirm this (see Figure 4.8, *bottom right*).

4.8.2 Comparison of Optimization Methods

We next compare to other accelerated descent strategies from geometry processing and geometric optimization. Our overall observations are consistent with those from Section 4.8: the fractional Sobolev scheme converges to local minimizers far quicker than general-purpose acceleration strategies (dramatically so, in the case of highly knotted configurations). This should not come as a surprise: the all-pairs energy we seek to minimize behaves very differently from those arising in e.g. curvature flows or elasticity, which are based on discrete differential operators with small local stencils.

To make a fair comparison, all methods use identical code for accelerated energy and differential evaluations (Section 4.4), and differ only in how they use these values. The same dynamic remeshing routine (Section 4.6) is also run at the end of each iteration for all methods. Note that edge splits and collapses invalidate the history of methods such as L-BFGS; here we use memory vectors for as long as they are valid, and reset them when edge splits or collapses occur. All experiments were run with barycenter and total area constraints. Since AQP and L-BFGS methods do not support nonlinear constraints such as total area—for these methods, we instead use stiff penalty functions (Section 4.7.3) to discourage excessive drift.

Comparison Methods. Our comparisons are guided by the extensive comparisons carried out in Section 4.8 (and published in Yu et al. [116]); here we compare with the best of those methods. As a baseline we consider ordinary L^2 gradient descent, which amounts to replacing A in Equation 4.14 with the mass matrix. Likewise, replacing A with the weak Laplacian Δ (encoded by the cotan matrix) yields standard H^1 Sobolev preconditioning; H^2 Sobolev preconditioning is achieved by solving Equation 4.14 with the weak formulation of the bi-Laplacian Δ^2 in place of A. (This latter preconditioner is essentially an ideal choice for Willmore flow [99].) Like H^1 preconditioning, the *accelerated quadratic proxy (AQP)* method uses the weak Laplacian Δ as the inner product, but also computes a Nesterov acceleration step from the previous two configurations; this strategy is compatible only with linear constraints [68, Section 2]. Another common strategy, which we refer to as H^1 L-BFGS, is to initialize L-BFGS with the weak

Laplacian rather than the identity matrix, and likewise use the Laplacian to evaluate inner products. Finally, *Blended cured quasi-Newton (BCQN)* essentially interpolates between ordinary H^1 Sobolev preconditioning and H^1 L-BFGS, together with barrier penalties to prevent triangle inversion. Since our gradient is almost H^s orthogonal with tangential motions of the surface (and do not experience element inversions), we omit these penalties.

4.8.3 Time Step Restriction

Figure 4.4 verifies that matching the order of the inner product to that of the energy differential essentially lifts the mesh-dependent time step restriction. Here, we sampled the same surface at three resolutions, and ran each method for the same number of iterations. Our H^s scheme makes more progress for an equal number of iterations—but more importantly, the per-iteration progress of H^s is largely unaffected by mesh resolution, whereas all other methods slow down as resolution increases. Hence, even if some of these methods could be further accelerated by a constant factor (e.g. via code-level optimization), asymptotic behavior would ultimately dominate.

4.8.4 Wall-Clock Performance

We also timed the real-world performance of each method on several challenge meshes, using an AMD Ryzen Threadripper 3990X with 32 GB of RAM. Though in practice our solver benefits from multiple threads (as per Section 4.4 and Section 4.5), we ran this benchmark single-threaded to ensure a fair comparison. Figure 4.9 plots energy as a function of time; we ran each method for 3600 seconds for the figure-8 and trefoil tunnels, and 2400 seconds for all others. Reference energy values were computed by evaluating the exact energy, without Barnes-Hut approximation. Our H^s projected gradient method gave the best performance in all cases, reliably reaching a minimum within the alloted time. In some cases the initial rate of decrease is faster for other methods, likely because there are initially many small local features to be smoothed out. Subsequently, however, these methods make much slower progress at evolving the global shape. Though AQP and BQN are also based on H^1 preconditioning, they do not do as well here as the "vanilla" H^1 preconditioner. One possible reason is that these methods do not support hard nonlinear constraints, and hence penalty forces may fight with the main objective. See Section 4.8 for much more extensive discussion and analysis of fractional methods versus a similar set of alternatives.

4.9 Examples and Applications

We here explore a variety of applications that help to further evaluate our method, show how it can be used in context, and also identify issues that might be improved in future work. These applications are also illustrated in the accompanying video—note that for many of these examples we take time steps far smaller than the optimal step determined by line search, in order to produce smooth animation.

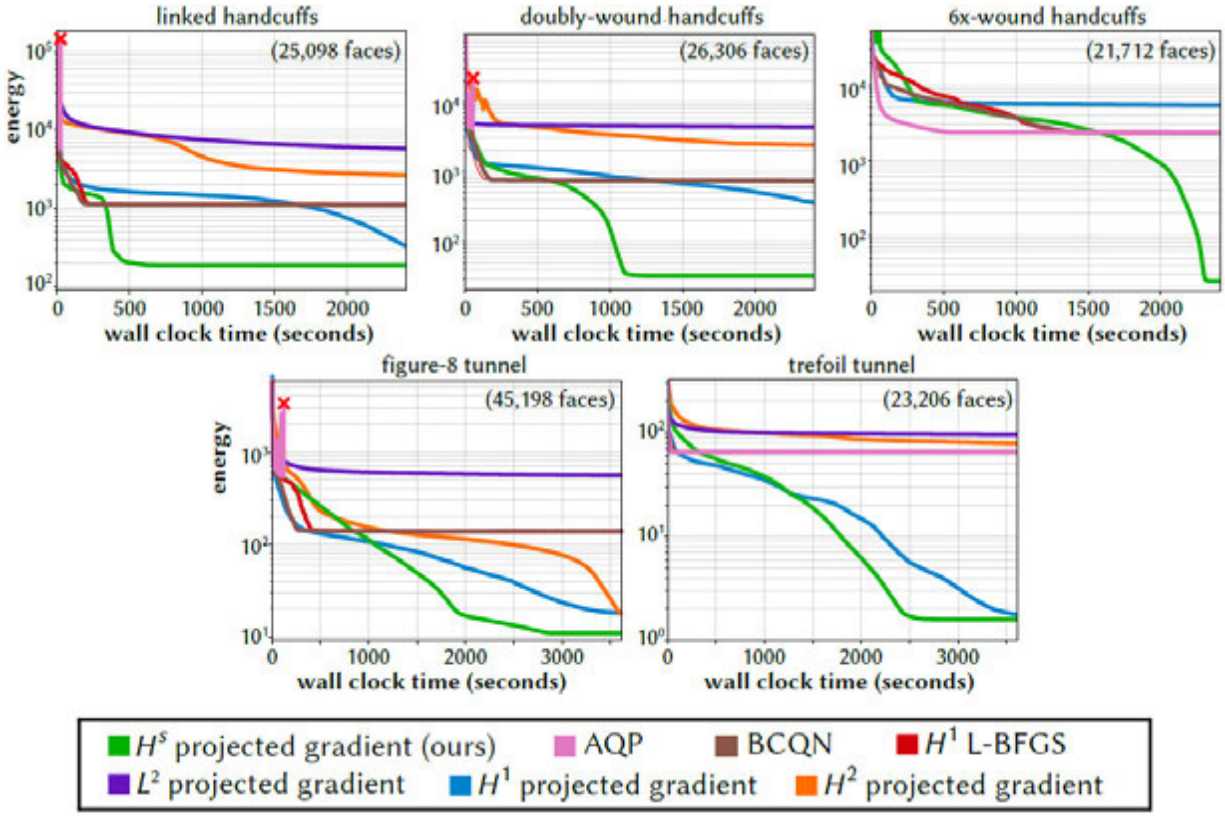


Figure 4.9: Energy plots showing the effectiveness of a suite of methods at minimizing the tangent-point energy. Our H^s method (in green) reaches minimizers more quickly and consistently than the alternatives. Points at which methods became unstable are marked with an X. Renderings of the meshes used and their minimizers can be seen in Figure 4.10.

4.9.1 Mathematical Visualization and Exploration

Mathematically, the motions computed by our method are ambient isotopies: given two embeddings $f_0, f_1 : M \longrightarrow \mathbb{R}^3$, an *ambient isotopy* is a continuous map $F : \mathbb{R}^3 \times [0,1] \longrightarrow \mathbb{R}^3$ such that for all $x \in M$, $F(x,0) = x$, $F(f_0(x),1) = f_1(x)$, and $F(x,t)$ is a homeomorphism from $\mathbb{R}^3$ to $\mathbb{R}^3$ for every time $0 \leq t \leq 1$. Intuitively, an ambient isotopy is a deformation of space that "drags along" f_0 with it, turning it into f_1 while avoiding any changes to the initial topology. A basic question in geometric topology is whether two embedded manifolds are ambiently isotopic, and in general this question can be quite hard to answer—for instance, even detecting whether an embedding of the circle in $\mathbb{R}^3$ is equivalent to the unit circle (or "unknot") has not yet admitted a polynomial time algorithm [73]. Hence, computational tools have been developed to explore such questions experimentally, with a notable example being the widely-used *KnotPlot* package for curve untangling [97]. The software developed for our project effectively provides the first "KnotPlot for surfaces." Especially the fact that our solver exhibits rapid convergence and excellent scaling enables us to investigate questions that would be impossible with naïve

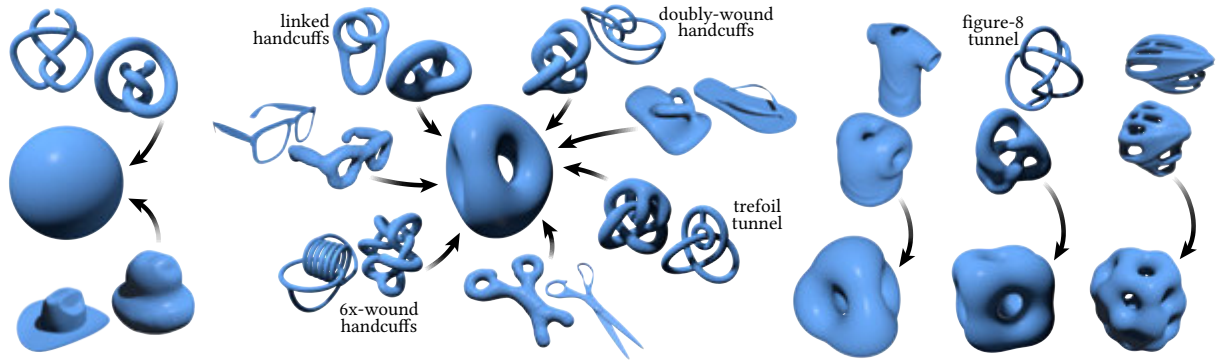


Figure 4.10: Gallery of isotopies obtained by minimizing tangent-point energy—notice that highly knotted surfaces, as well as surfaces with thin sheets and handles, successfully flow to their canonical embeddings. Surfaces are grouped by their isotopy equivalence classes, which are extremely difficult to determine via visual inspection (and also not simply determined by Euler characteristic—see Figure 4.13). Labeled meshes are used for performance comparisons in Figure 4.9.

numerical methods.

Canonical Embeddings

Global minimizers of geometric energies provide the "simplest" possible geometric representative of a given topological space. Such minimizers also play a critical role in geometric algorithms since they provide a canonical domain for e.g. surface correspondence and data transfer—see for instance recent algorithms in both the intrinsic [50, 98] and extrinsic [64, 114] settings. Formally proving that a given surface is a global minimizer is quite challenging. For instance, even the classic *Willmore conjecture* (which says that the *Clifford torus* minimizes Willmore energy for genus-1 surfaces) was resolved only very recently, after about 50 years of sustained effort [77]. Hence, numerical tools are essential for formulating hypotheses about the behavior of minimizers and other critical points. To date, there are no clear conjectures about tangent-point minimizers for surfaces of genus $g \geq 2$. For reasons discussed in Section 4.1.3, these minimizers likely exhibit symmetries in $\mathbb{R}^3$ rather than $\mathbb{S}^3$, making them potentially useful as a base domain for algorithms in extrinsic shape processing. To do so, one would simply need to track the parametric correspondence (e.g. via UV-coordinates), and perhaps minimize tangential distortion after flowing to a geometric minimizer (as in Schmidt et al. [98]).

Unknotted Minimizers Figure 4.12 shows a numerical study for untangled surfaces of increasing genus, initialized with a linear arrangement of handles. For genus 0, 1, and 2 we get a round sphere, a torus of revolution, and a surface with symmetries of a triangular prism. Other surfaces appear to exhibit symmetries of a highly regular polyhedron—for instance, for genus 3, 4, 5, 6, 8, 9, and 11 we get symmetries of the tetrahedron, triangular prism, cube, pentagonal prism, truncated bipyramid, rectangular prism, and dodecahedron, respectively. Symmetries (if any) for genus 7 and 10 are less clear—or we may have simply failed to reach a global minimum.

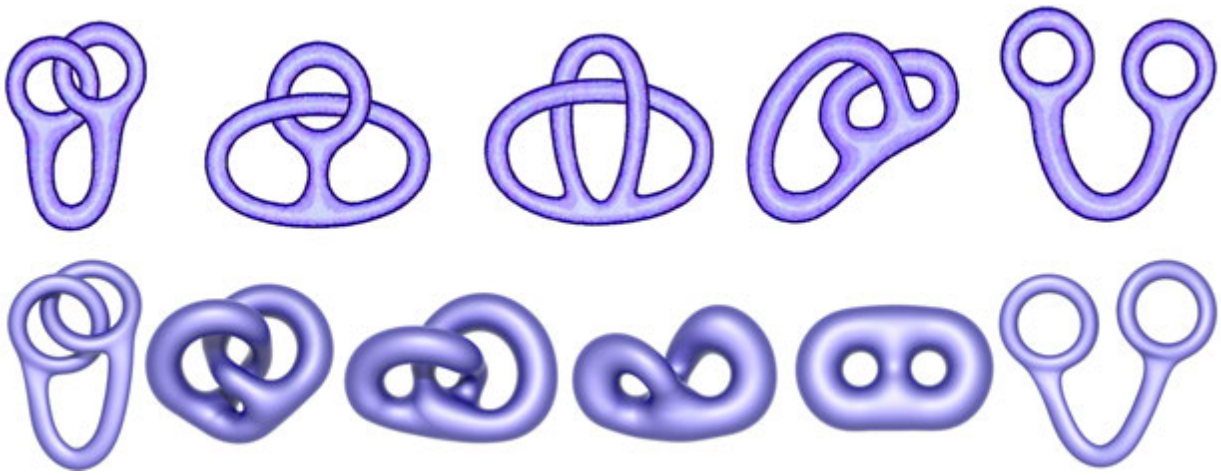
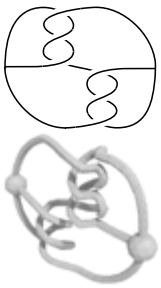
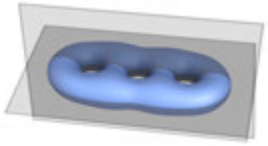


Figure 4.11: *Top:* even careful illustrations of topological phenomena (here drawn by mathematician Peter Lynch) can be difficult to understand without a good visual imagination. *Bottom:* our method automatically generates continuous motions that are easier to interpret (see video), enabling exploration by students and researchers who do not have significant artistic training.

Interestingly, an octahedral configuration does not appear to be a minimizer for genus 7, even if we start with a symmetric configuration (and similarly for the icosahedron, not shown). In general it seems that triangular "faces" are not preferred in higher-genus configurations due to the small angle between "edges"—much as electron repulsion maximizes bond angles in molecular geometries (e.g. stable compounds like graphite prefer bond angles near 120°, whereas only unstable compounds like white phosphorus exhibit tetrahedral symmetry).



Knotted Minimizers A key feature of tangent-point energy (versus, say, Willmore energy) is that it enables us to find minimizers within a given isotopy class. Hence, just as it is quite common to make tables of canonical knot embeddings, we can now make tables of canonical embeddings for knotted surfaces. For instance, Figure 4.13 shows the first-ever visualization of the different ways a genus-2 surface can be embedded in space. In the past, these isotopy classes have been depicted only as trivalent graphs—we take each such graph from [61, Table 1], and construct a topologically equivalent initial mesh that is optimized by our approach (see inset). As with knots most of these minimizers do not exhibit much extrinsic symmetry, except for e.g. 6_7 and 5_3 which exhibit bilateral and 3-fold symmetry, respectively.



Planar Representatives. Although minimizers exhibit a high degree of symmetry in $\mathbb{R}^3$, it can be hard to determine even the genus of a minimizer when viewed from just a single viewpoint. In contrast, topological figures depicted by expert illustrators tend to be somewhat "2.5-dimensional" so that they can be better understood when projected onto the image plane. We

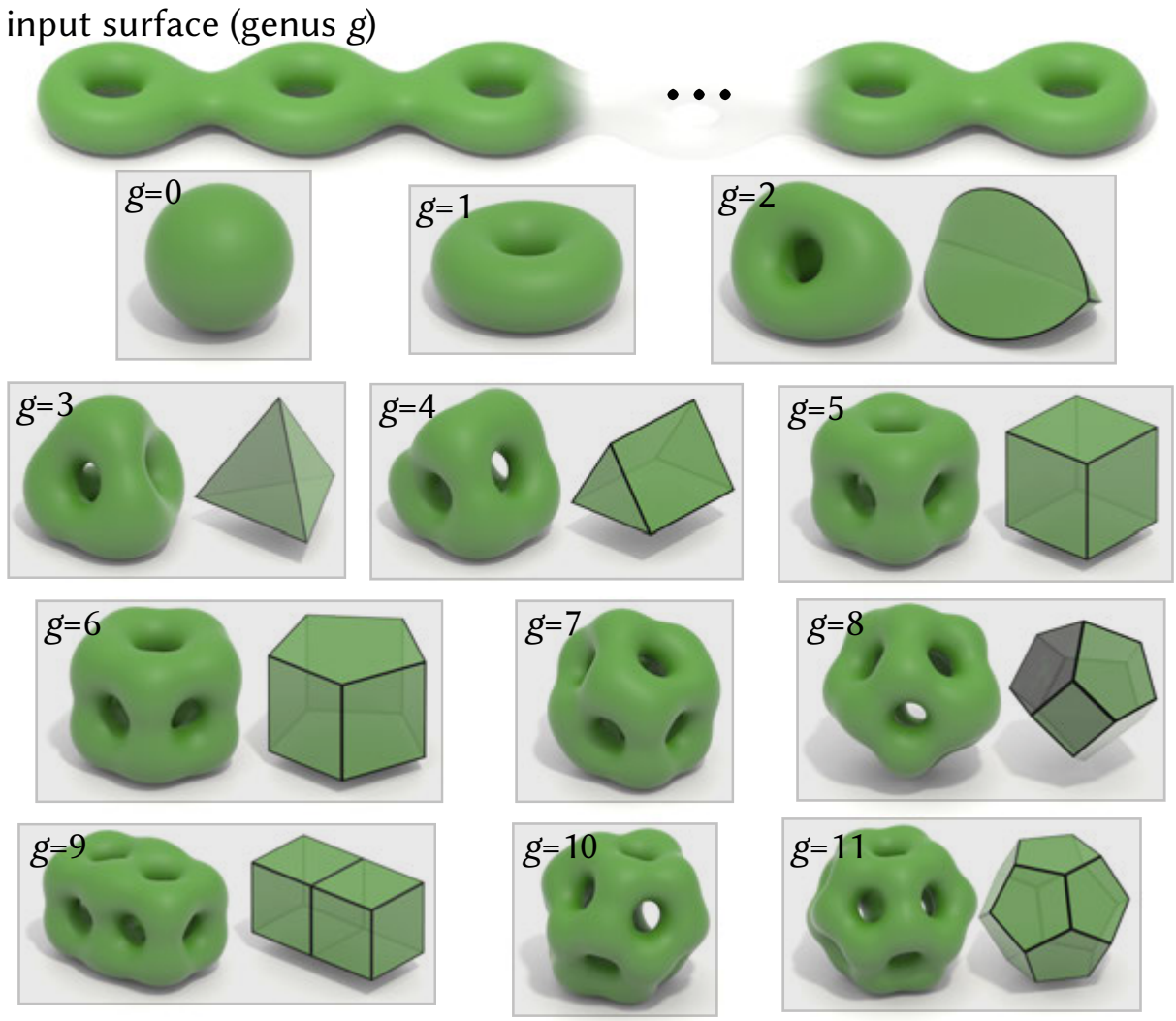


Figure 4.12: Global minimizers of geometric energies provide canonical domains that can be used to map between surfaces of the same topology, or simply help visualize a topological space. Here we show conjectured minimizers of tangent-point energy for unknotted surfaces of genus g; adjacent figures illustrate symmetries (when present).

Figure 4.13: Geometric functionals provide a bridge between topology and geometry by enabling one to construct canonical geometric representatives of a given topological space. Here, minimizers of tangent point energy are used to visualize nontrivial isotopy classes of a genus-2 surface. (Numbers indicate number of crossings; subscripts index trivalent graphs from [61, Table 1]).

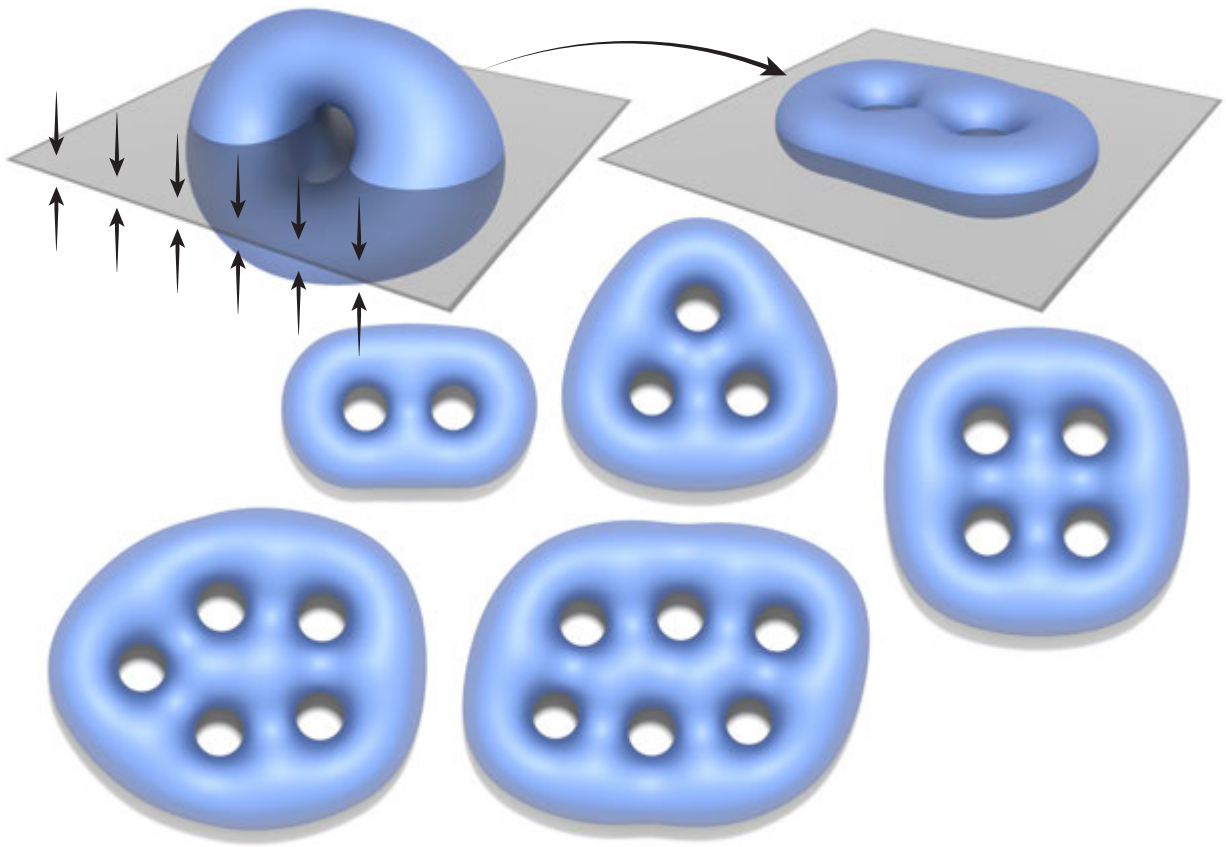
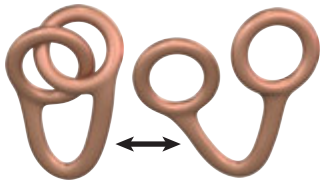


Figure 4.14: *Top:* minimizers of tangent-point energy often exhibit three-dimensional symmetries which can be difficult to understand from a single view—by adding an attracting plane, we get embeddings that can be nicely displayed in two-dimensional illustrations. *Bottom:* constrained minimizers for genus 2 through 6.

can replicate this behavior by adding a simple attractive plane potential, as depicted in Figure 4.14, yielding minimizers that are much easier to recognize (contrast with Figure 4.12). An additional plane constraint yields a linear arrangement of handles, as commonly drawn by hand (see inset).

Illustrating Isotopies

Our method also provides significant utility for mathematical visualization and illustration. Traditionally, interesting homotopies and isotopies are depicted by a sequence of drawings (or perhaps physical models) highlighting key moments of transition—a practice that has developed over time into a true art form [46]. However, even the best drawings can be difficult to understand without significant thought and visual imagination. To obtain continuous motions (that are more easily understood), a small number of carefully "hand-crafted" computer animations have been produced over the years by either artist keyframing, or explicit programming of meticulously derived parametric formulas [11, 75]. More recently, automatic optimization-based tools have been used to produce animations, such as the *minimax sphere eversion* [45], as well as recent work in computer graphics on metric embedding [30] and conformally-constrained Willmore surfaces [102]. Since these optimization-based tools are largely automatic, they help to democratize the creation of topological animations—our scheme extends such tools to the important and difficult case of ambient isotopies.



One classic example is "unlinking" a pair of handcuffs (as shown in the inset), though mathematically speaking these handcuffs are not actually linked: surprisingly, they belong to the same isotopy class. Figure 4.11 compares a hand drawing of this isotopy with a different isotopy automatically computed via our method—and which is much better depicted in the accompanying video. To create this animation we simply minimize tangent-point energy from both start and end configurations, together with a potential that encourages the surface to lay parallel to the view plane. Since we reach the same minimizer in both cases (seen in Figure 4.16, *far right*), we can compose these two sequences (one in reverse) to depict the complete motion. Other similar examples are shown in Figure 4.10, and in the video.

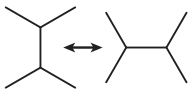


Figure 4.17: An IH-move.

Figure 4.15 shows another classic example: removing one handle of a pair of handcuffs from a rigid pole or ring. The hand-drawn illustration helps to indicate several stages of this isotopy, which are also captured in our animation. However, the remarkable fact about our version is that it is driven *purely* by energy minimization—we did not perform any keyframing, nor impose any boundary conditions, yet it still constructs an isotopy in several "stages": flatten out the two handles, perform a so-called *IH-move* (see Figure 4.17 and [60]), and then optimize the geometry of the untangled surface. Our specific setup here is to minimize tangent-point energy while fixing surface area, and incorporating an infinite repulsive cylinder (modeled by an implicit surface). As in the previous example we use an attractive plane orthogonal to

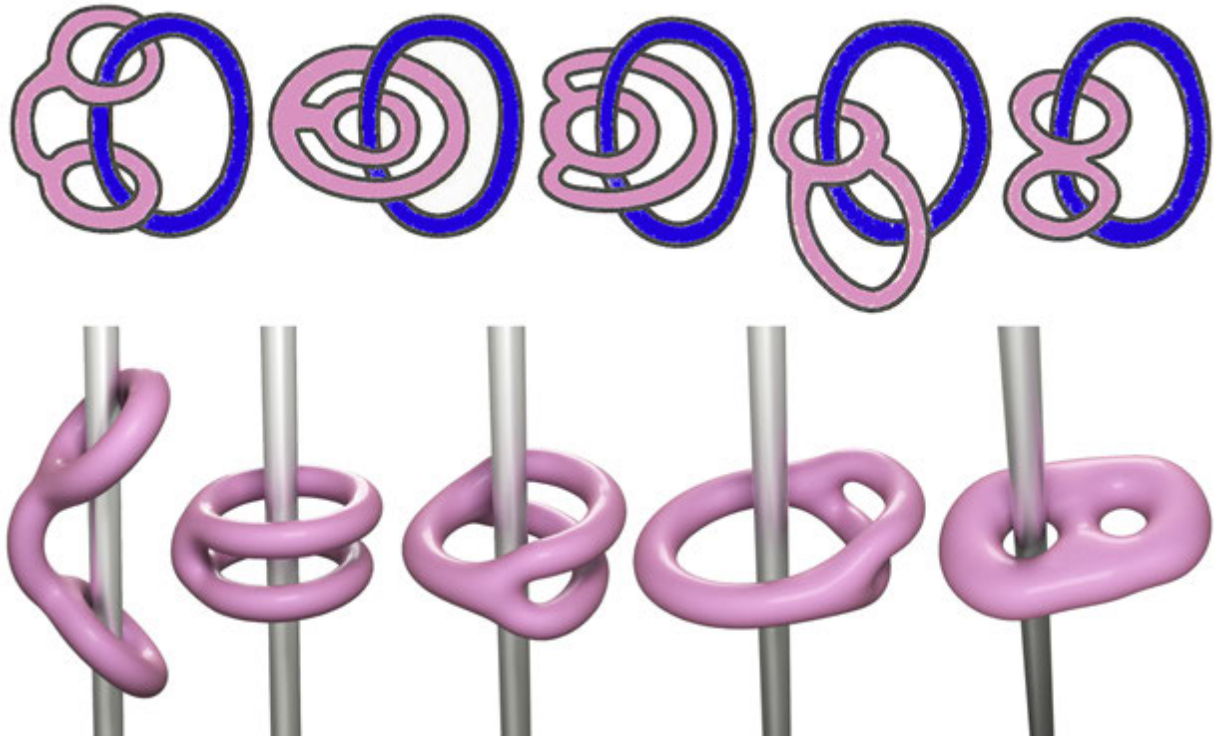


Figure 4.15: Surprisingly, one can remove a handle of a double torus from a loop or pole without cutting or pinching the surface. *Top:* hand-drawn illustration by Wells [111]. *Bottom:* isotopy computed automatically by our method (see video); no keyframing or boundary conditions were used.

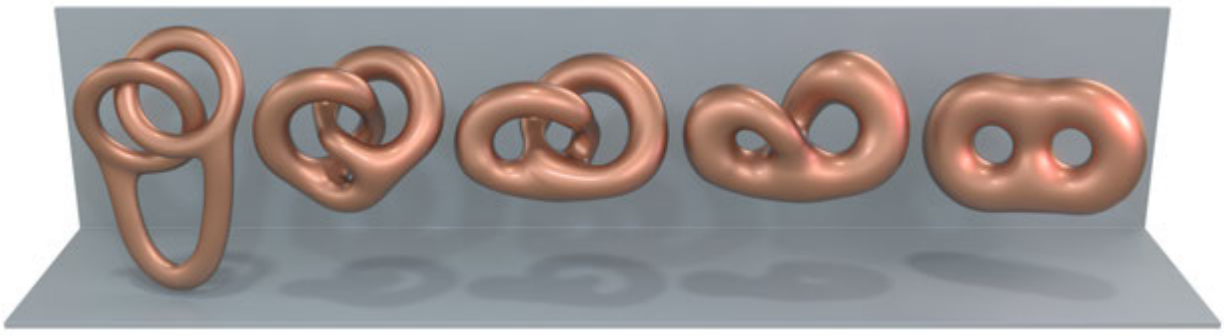


Figure 4.16: Here, we automatically find an unexpected transition between linked and unlinked states of a pair of "handcuffs" by simply minimizing a repulsive energy.

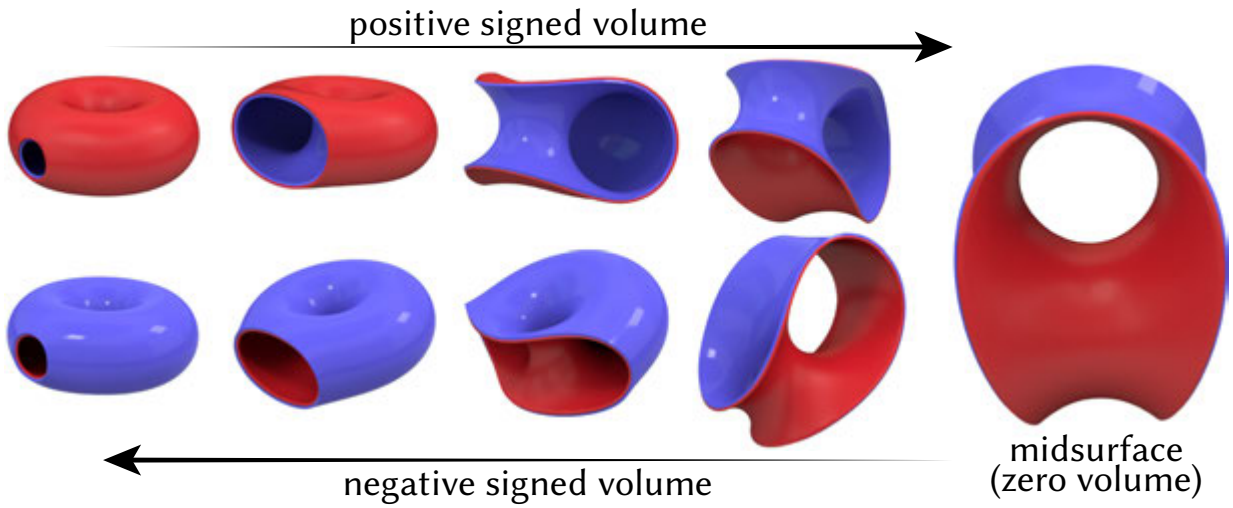


Figure 4.18: Minimizing the tangent-point energy of a punctured torus while pushing signed volume toward zero yields a surface with reflection symmetry. Applying a reflection and reversing the flow hence yields an eversion that turns the surface "inside-out" while avoiding self-intersections.

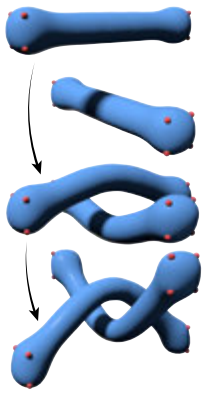
the pole to obtain a more canonical-looking minimizer. The only hand-tuning was reducing the repulsive strength of the cylinder near the end of the animation, to give the handles of the final surface a similar size. Importantly, allowing the barycenter to float freely (as per Section 4.7.1) is essential here, since the center of mass must ultimately move away from the pole.

Punctured Torus Eversion. Our discrete tangent-point energy can also be evaluated on surfaces with boundary, since we simply take a sum over pairs of triangles. Since we did not develop a careful treatment of boundary conditions, we simply penalize the total length and total squared curvature to ensure the boundary at least remains regular. In Figure 4.18 we use this setup to compute an isotopic *eversion* between the two orientations of a punctured torus. Unlike the classical sphere eversion, where one typically starts with a symmetric midsurface and flows toward the round sphere, we start with the punctured torus and use our flow to find the mid-surface. The key observation is that the *oriented* volume of the surface $\int_M \langle f(x), N_f(x) \rangle \, dx_f$ will be zero for a symmetric configuration; fixing the area ensures that our zero-volume penalty does not cause the surface to collapse to a point. Once we reach zero volume we transform the midsurface by a reflection and 90-degree rotation, and run the same flow in reverse (with opposite colors) to obtain the eversion.

4.9.2 Geometry Processing and Shape Modeling

The no-intersection condition is also natural in geometry processing and shape modeling, especially when a surface is meant to represent the boundary of a solid object (e.g. for computational fabrication). As noted in Section 4.1, there has been relatively little work on intersection-aware geometric modeling—see for instance Harmon et al. [56] and references therein. In contrast

to resolving local intersections, tangent-point energy adds the complementary functionality of global prevention of intersections to a broad range of existing tasks. Here we present several aspirational examples—importantly, our goal is not to outperform more specialized, mature solutions, but rather to explore how a tangent-point regularizer might serve as a unified approach to intersection-free modeling across many disparate applications.



Proximity-Aware Variational Modeling As a basic example, the inset figure above shows a simple example of interactive surface editing, where surface geometry is guided by point constraints, and nearby geometry is moved out of the way by the tangent-point energy. To better preserve the details of an initial mesh one might also combine tangent-point energy with a discrete shell energy [52], which would entail transferring the material configuration across meshing operations (a question which is beyond the scope of this work). Figure 4.19 shows another example where pinned points and edges are interpolated while optimizing the rest of the geometry. (Here we disable remeshing, but could easily modify remeshing to ignore pinned vertices). Unlike harmonic interpolation or area minimization, for which point constraints are ill-posed, we get nice curvature behavior even near the pins; unlike Willmore flow (which provides good curvature behavior), we avoid self-intersection. Tangent-point energy could also in principle be used as a regularizer to discourage intersection in other common modeling paradigms, such as *as rigid as possible (ARAP)* modeling [103].

Shrink Wrapping

One class of methods for reconstructing a surface from a collection of points is to "shrink-wrap" them with a triangle mesh [54, 66]; such methods are especially suitable in problems where one wishes to fit a high-quality template mesh to a known class of shapes (e.g. head or body scans). A basic problem, however, is that the mesh can get "tangled" during wrapping, inhibiting progress or requiring intricate remeshing to resolve self-intersections. Tangent-point energy may prove useful as a regularizer for such methods—Figure 4.20 shows a basic shrink wrapping example on a point cloud, and on polygon soup with severe holes. Here we minimize tangent-point energy with a gradually decreasing volume constraint.

Nested Envelopes

In a similar vein, nested sequences of solids $U_1 \subset \cdots \subset U_k \subset \mathbb{R}^n$ represented by progressively coarser meshes have applications in multiresolution solvers, cage-based editing, and physical simulation [96]. In Figure 4.21 we construct each surface ∂U_k by minimizing tangent point energy plus a volume constraint, and gradually adjusting the constrained volume to achieve a fixed constant factor (here, 1.15x) of the volume of ∂U_{k-1}. This variational approach may offer interesting generalizations of ordinary nested cages, since it can easily incorporate constraints and objectives beyond just the absence of intersections.

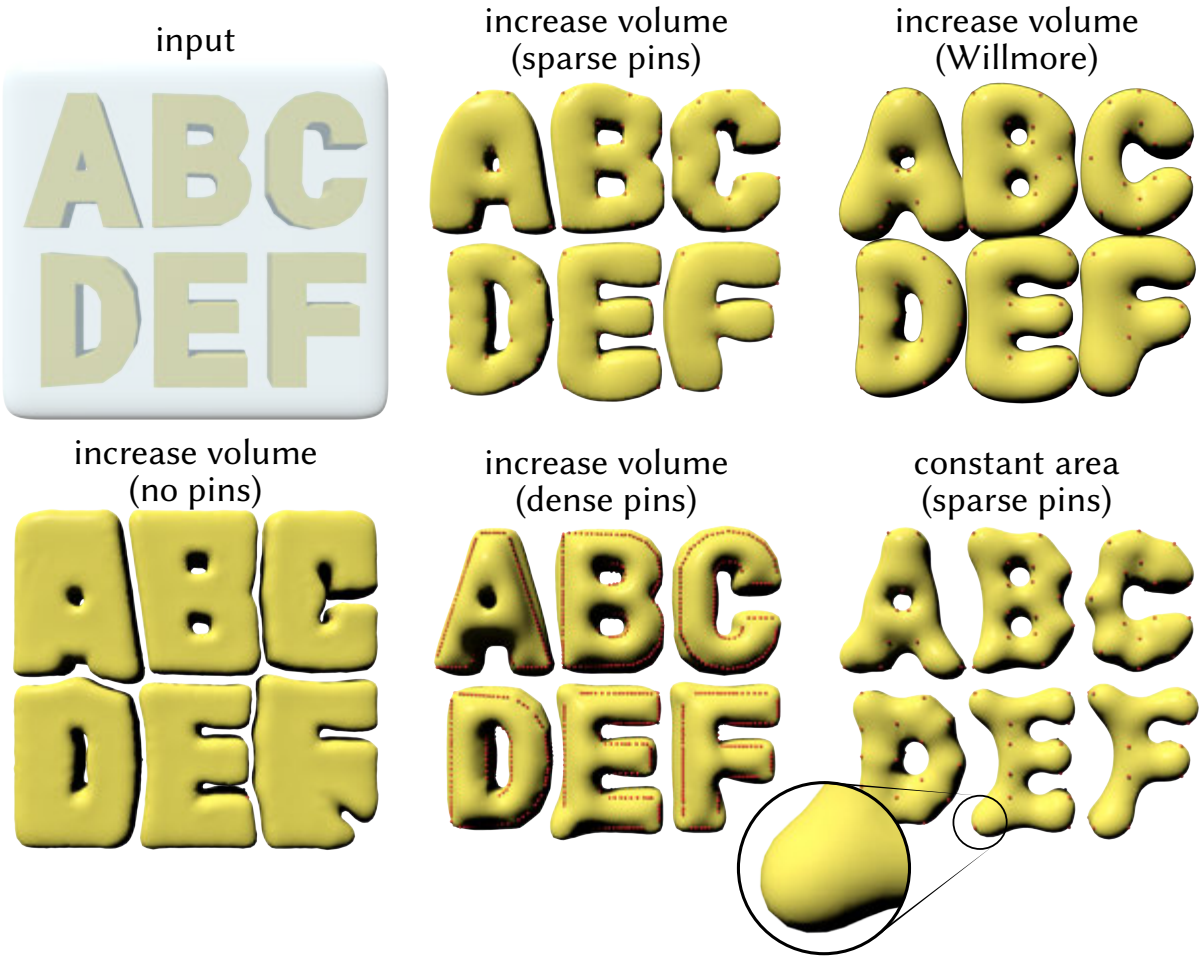


Figure 4.19: The tangent-point energy can be used to make variational surface modeling responsive to proximity, rather than just intersections. Here for instance we pin a sparse or dense set of points and modify volume and surface area to adjust the appearance of some text (in some cases enclosed in a box). Like Willmore energy *(top right)*, we get smooth behavior near point constraints (see magnified portion), but avoid overlap.

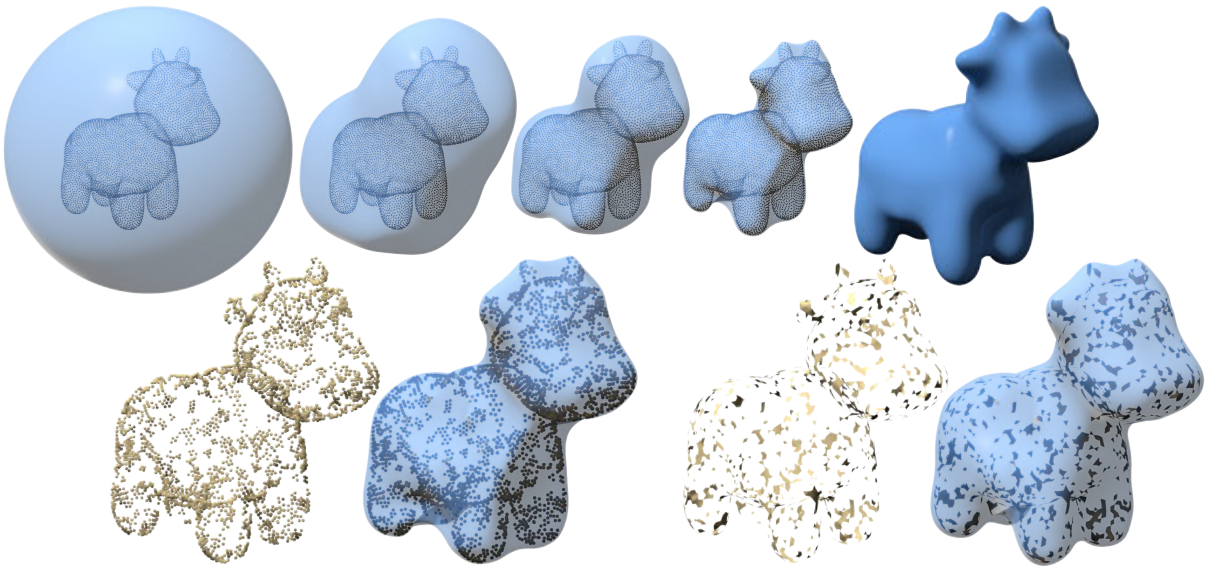


Figure 4.20: Here we perform a simple "shrink wrapping" to obtain a manifold, intersection-free reconstruction *(top)*, which works well even for points or polygon soup with severe holes and missing data.

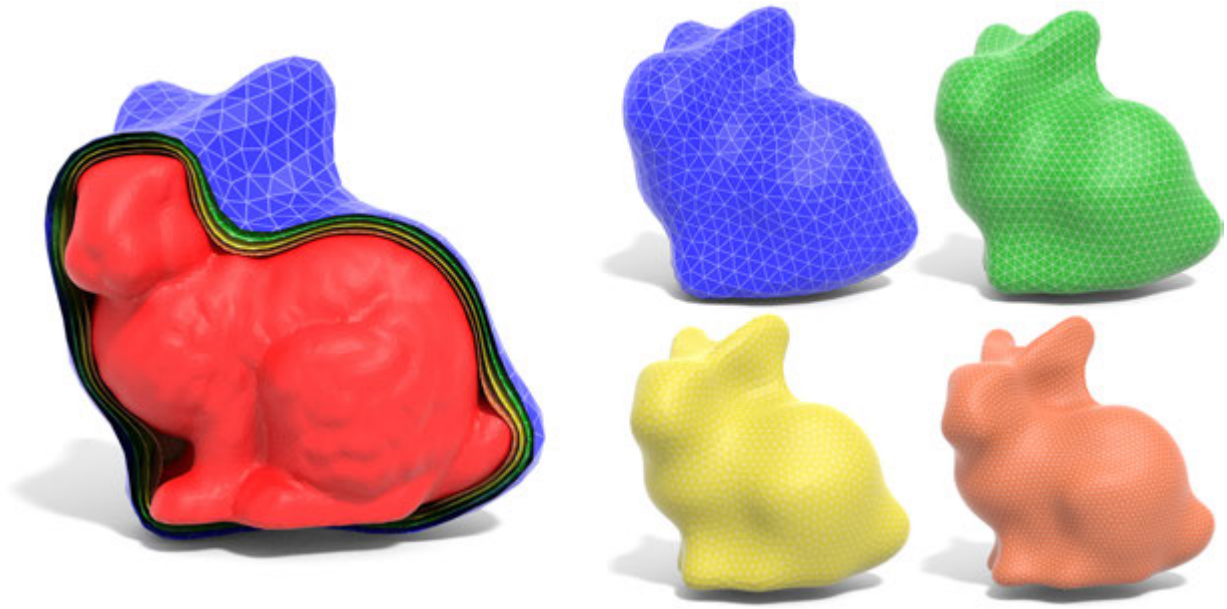


Figure 4.21: We can "shrink wrap" a model to get a sequence of progressively coarser approximating envelopes that exhibit a strict containment property, and are free of self-intersection. Here we aim for a 1.15x increase in volume at each level.

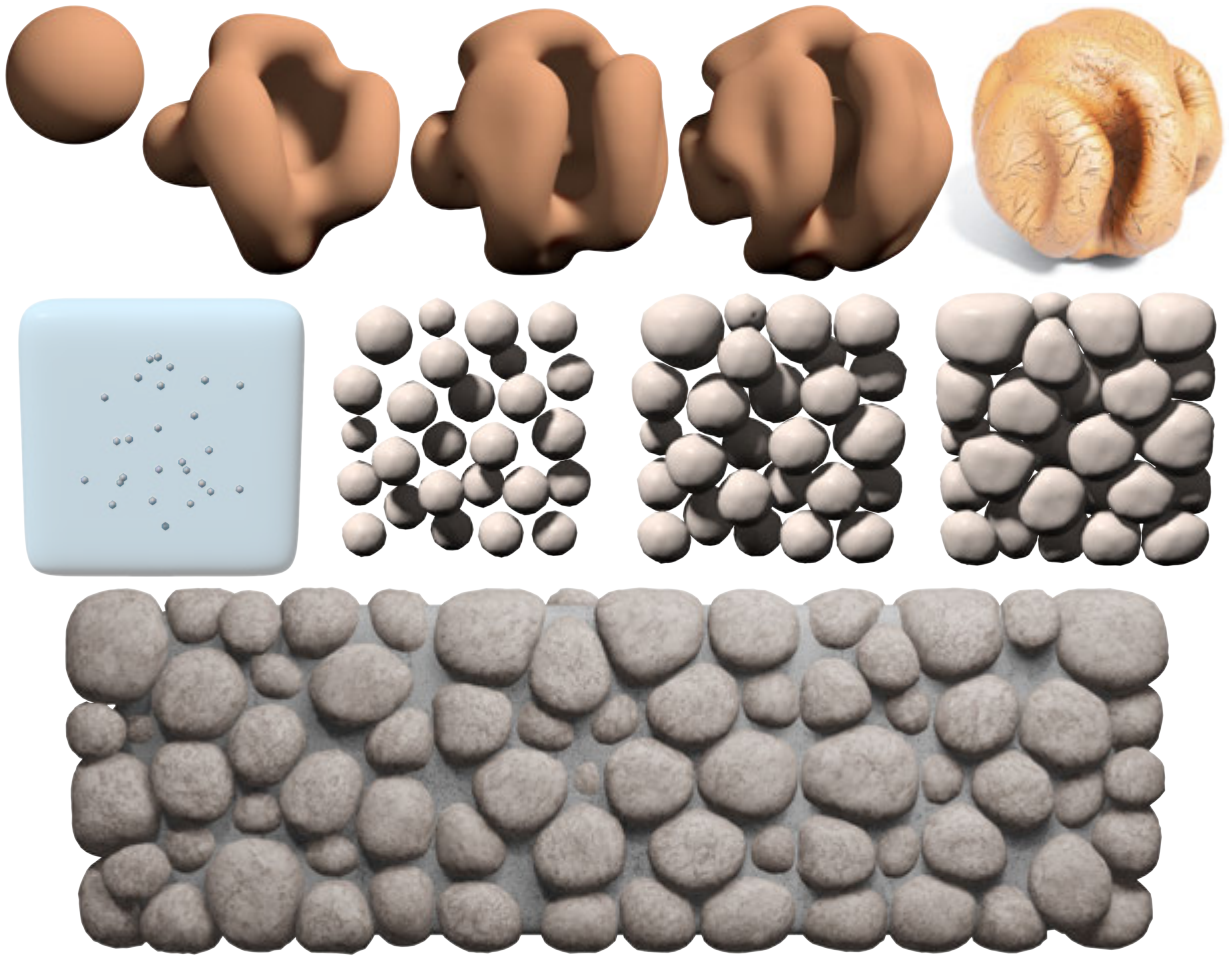


Figure 4.22: We can also use tangent-point energy for generative modeling by "growing" a surface subject to constraints. *Top:* confining to a sphere while increasing area leads to a wrinkled shape reminiscent of a walnut. *Bottom:* growing many small spheres inside a slab yields a tileable cobblestone pattern.

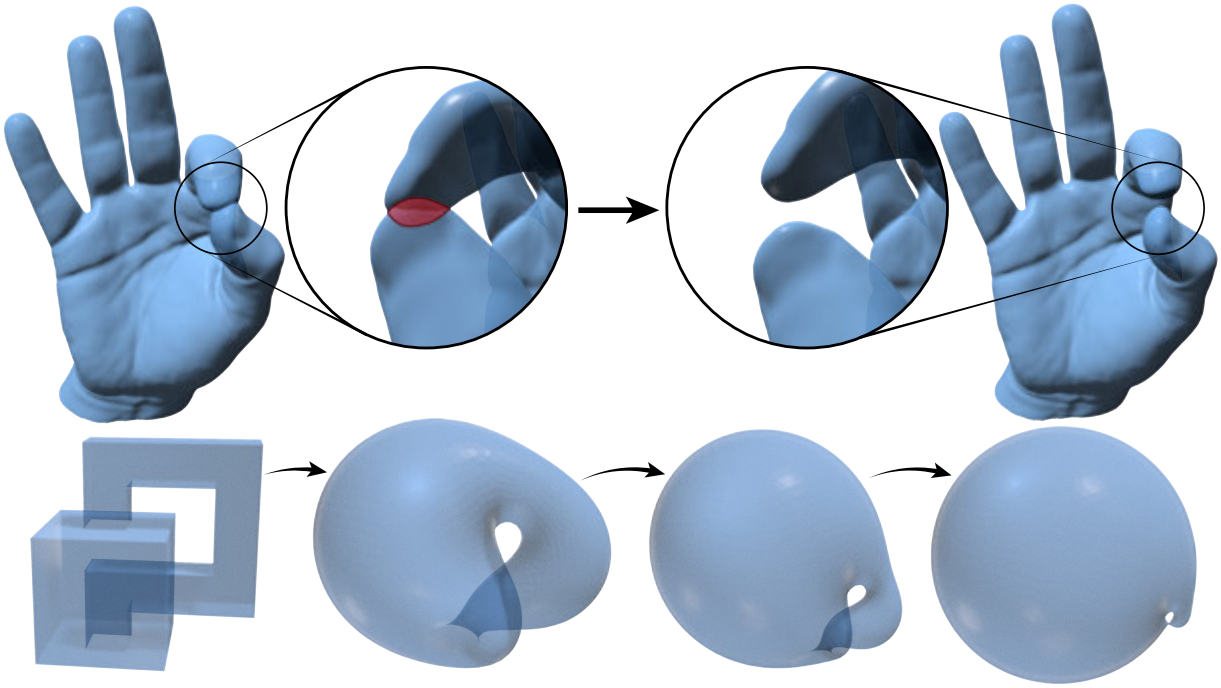


Figure 4.23: For exponents $p < 4$, the tangent-point energy $\mathcal{E}^p$ is no longer infinite for self-intersecting surfaces, but still discourages overlap. Here we try using this "subcritical" energy to resolve intersections, which works for small intersections *(top)*, but fails for an unembeddable surface like the Klein bottle *(bottom)*.

Generative Modeling

Rather than using the tangent-point energy to edit or process existing data, we can also use it to generate new geometry. In nature, the growth of organic shapes is often governed by simple combinations of objectives, e.g. a balance between area and volume while avoiding self-intersection. We can likewise use such forces to drive the growth of organic-looking objects, such as the "walnut" depicted in Figure 4.22 (*top*). The same technique is used in Figure 4.22 (*bottom*) where multiple objects are packed into a volume to create a repeating organic pattern.

Self-Intersection Resolution

In many geometry processing tasks, input data is not free of self-intersections. For exponents $p > 4$, the tangent-point energy $\mathcal{E}^p$ of a non-embedded surface is infinite; to *resolve* intersections in the input, we can try reducing the exponent to a value $p < 4$, at which point $\mathcal{E}^p$ becomes finite but still discourages intersection. Here we find that it also helps to disable the low-order term from Equation 4.8. Empirically, the same system framework now appears capable of eliminating small self-intersections (Figure 4.23, *top*), through struggles in more difficult scenarios like the Klein bottle depicted in Figure 4.23 (*bottom*), which cannot be globally embedded without self-intersection. Further analysis of the energy for these "subcritical" values may help to provide

more robust tools for global intersection resolution.

Chapter 5

Conclusion

In this thesis, we have proposed the tangent-point energy as a tool for finding intersection-free solutions to optimization problems on curves and surfaces, and built an optimization system demonstrating its practicality and applicability across numerous tasks therein. The success of our system at producing compelling results in such a broad range of tasks demonstrates the effectiveness of the tangent-point energy at geometric modeling, particularly when it comes to preventing self-intersections. Further, our ability to achieve reasonable runtimes despite the "all-pairs" nature of the tangent-point energy demonstrates the effectiveness of our numerical techniques at optimizing such challenging energies. Both of these points extend beyond the scope of this thesis; there are certainly many more tasks for which intersection-free optimization by way of the tangent-point energy could be a valuable tool, and many more challenging energies capturing other interesting objectives that could be optimized using the techniques we have developed.

5.1 Limitations and Future Work

Since we approximate the tangent-point energy via numerical quadrature, it is possible for a very coarse curve or surface to reach a self-intersecting state. These events are more easily observable on coarse surfaces, which effectively have fewer quadrature points per unit surface area; in particular, in situations such as "shrink wrapping" where we force very tight contact (Section 4.9.2), the occurrence of these intersections effectively becomes the primary obstacle to achieving a tighter fit. Adding additional quadrature points, either overall or to elements in near-contact through some form of adaptive refinement, may help alleviate this problem. Intersections can also be prevented via continuous time collision detection (Section 3.3.5); to maintain accuracy, one could try adding more quadrature points at the previous time step if any intersections occur.

Even in the absence of intersections, elements that are extremely close together can temporarily get stuck in a configuration very close to self-contact (Figure 5.1). In this situation, the term k^2

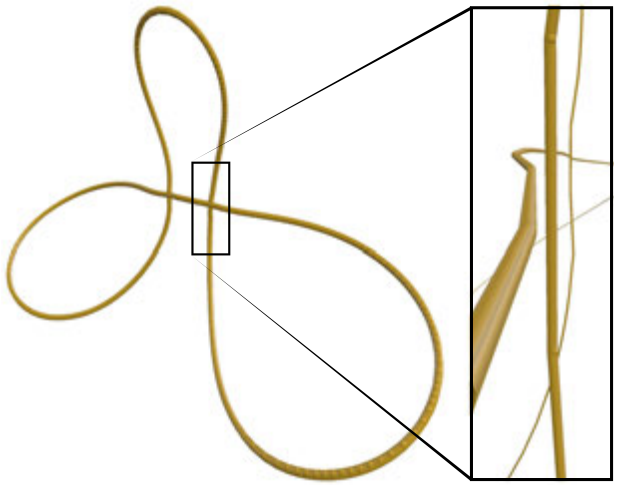


Figure 5.1: A nearly crossing initial configuration that can be difficult for the flow to escape, due to the behavior of the low-order term.

from the low-order term (Equation 4.8) is very large, causing the inverse of A—and hence the Sobolev gradient—to be very small. One idea is to use adaptive quadrature for pairs that are close in space, which would better resolve the near-infinite high-order term and hence push the curve apart. Given the scalability of our approach, however, a more pragmatic solution may simply be to increase the overall resolution.

There are also many ways to further accelerate our solver; for instance, we did not make use of the GPU, and only made limited use of vectorized arithmetic. For small time steps one might refit rather than re-build the BVH; likewise, it may be beneficial to incrementally update the BCT. Significant performance gains could also be achieved purely through better software engineering, e.g. by improving our parallel implementation of hierarchical matrix multiplication (which is currently bottlenecked around 4–8 threads). On an algorithmic level, better line search or descent direction heuristics may also reduce the overall number of steps.

The relatively complex remeshing algorithm on surfaces compared to curves (where we simply split each edge in half) also presents some challenges specific to the surface case. For instance, it would also be quite useful to track mesh attributes across remeshing operations, to enable (for instance) mapping of data from one shape to another through the canonical minimizer. Implementing curvature-adaptive remeshing (as in [39]) might also provide some benefits over our current approach of a uniform target edge length, as it would allow finely tessellating only those areas where resolution is required to resolve details, rather than the entire domain.

Several other issues specific to meshes require deeper investigation. For one thing, unlike in the case of curves, our preconditioning strategy for surfaces cannot easily accommodate dense constraints (e.g. preservation of each triangle area), which would require a prohibitive number of iterative solves. Revisiting the multigrid approach via hierarchical coarsening [17, 100] may prove fruitful in this regard. Currently, we present two options for inverting the fractional

Laplacian: a multigrid method with simple CG or GMRES as the smoother (Section 3.4.3), and a preconditioned iterative GMRES solver with no multiresolution hierarchy (Section 4.5). In reality, these options are not mutually exclusive; a preconditioned GMRES solver could be used as the smoother on each level of the multigrid hierarchy. Such an approach could potentially achieve superior convergence to either option alone, while also potentially handling dense constraints more efficiently if it allows for omitting the Schur complement (as in Section 3.4.3).

For shape interpolation and mathematical visualization, it would be quite useful to find the trajectory that minimizes overall tangent-point energy, rather than just flowing to a common minimizer—here, ideas about shell-space geodesics may prove valuable [58]. Likewise, integrating repulsive regularization into a thin shell model might provide better proximity-aware shape editing by retaining a "memory" of the initial shape. Finally, we do not directly treat boundary conditions or more general arrangements of repulsive curves *and* surfaces, both of which could have interesting modeling applications.

5.2 Final Remarks

As we have seen, the tangent-point energy goes a long way towards making intersection-free modeling practical in a variety of problem settings involving non-rigid curves and meshes. Further, the numerical techniques that we have developed for the tangent-point energy could also make the optimization of other global all-pairs energies a more tractable task. In general, we hope that the insights provided in this thesis will pave the way for further integration of geometry and optimization techniques into pipelines for modeling and computational design. Our hope is that, by providing a rich lexicon of intuitive objectives that can be specified and combined – in the form of constraints, obstacles, potentials, and so on – we can make optimization algorithms more and more accessible to end users. As an energy-based solution to intersection-free optimization that can easily be deployed on curves and meshes, the tangent-point energy is one major step in that direction.

Appendix A

Derivations

A.1 Action of the Fractional Operators

In Section 4.5.1, we claimed that the actions of the fractional operators L^σ, B, and B_0 can be expressed by suitable kernel matrices that we then compress by hierarchical methods. We include a brief derivation here to substantiate this claim. Consider the kernel matrix

$$H_{ST} := (1 - \delta_{ST})\,|X_f(S) - X_f(T)|^{-(2\sigma+2)}.$$

Rewriting Equation 4.11 for general $\mathbf{u}$ and $\mathbf{v} \in \mathbb{R}^{|V|}$ in terms of this kernel yields

$$\mathbf{u}^\top L^\sigma \mathbf{v} = \sum_{S\in F} \sum_{T\in F} (\bar{u}(S) - \bar{u}(T))\,(\bar{v}(S) - \bar{v}(T))\, a_f(S)\, H_{ST}\, a_f(T).$$

Multiplying the product inside the sum gives

$$\big(\bar{u}(S)\,\bar{v}(S) + \bar{u}(T)\,\bar{v}(T) - \bar{u}(T)\,\bar{v}(S) - \bar{u}(S)\,\bar{v}(T)\big)\, a_f(S)\, H_{ST}\, a_f(T)$$

for the pair (S, T). Because $H_{ST} = H_{TS}$, we can move some terms between the summands for (S, T) and (T, S), and thus reorganize the sum into

$$\begin{aligned}
\mathbf{u}^\top L^\sigma \mathbf{v} &= 2 \sum_{S\in F} \sum_{T\in F} \big(\bar{u}(S)\,\bar{v}(S) - \bar{u}(S)\,\bar{v}(T)\big)\, a_f(S)\, H_{ST}\, a_f(T) \\
&= 2 \sum_{S\in F} \bar{u}(S) a_f(S) \Big(a_f(S)^{-1} \sum_{T\in F} H_{ST} a_f(T)\Big)\, a_f(S)\bar{v}(S) \\
&\quad - 2 \sum_{S\in F} \sum_{T\in F} \bar{u}(S)\, a_f(S)\, H_{ST}\, a_f(T)\, \bar{v}(T).
\end{aligned}$$

Recall that $\mathsf{U} \in \mathrm{Hom}(\mathbb{R}^{|V|}; \mathbb{R}^{|F|})$ is defined by $(\mathsf{U}\mathbf{u})(S) = a_f(S)\bar{u}(S)$. Thus the above collapses to

$$\begin{aligned}
\mathbf{u}^\top L^\sigma \mathbf{v} &= 2\,\mathbf{u}^\top \mathsf{U}^\top \operatorname{diag}(a_f)^{-1} \operatorname{diag}(H\, a_f)\, \mathsf{U}\,\mathbf{v} - 2\,\mathbf{u}^\top \mathsf{U}^\top H\, \mathsf{U}\,\mathbf{v} \\
&= 2\,\mathbf{u}^\top \mathsf{U}^\top \Big[\operatorname{diag}\big(\operatorname{diag}(a_f)^{-1} H\, a_f\big) - H\Big]\, \mathsf{U}\,\mathbf{v}.
\end{aligned}$$

The derivation follows analogously for the high- and low-order matrices B and B_0, with the substitution of the operator $\mathsf{V} = \operatorname{diag}(a_f)\mathsf{D}_f$ for U in the case of B.

A.2 Fast Matrix-Vector Multiplication

Step 1 of Section 4.5.1 corresponds to thinning out the matrix shown in Figure 4.5 by removing all the green parts. The remainder is a sparse block matrix with variable block size. We store this sparse matrix in CSR format and perform matrix-vector multiplication via sparse BLAS routines.

In Step 2 the kernel matrix $H_{\mathcal{I}\mathcal{J}}$ is compressed into the rank-one-matrix $\mathbf{1}_{\mathcal{I}}\, h(X_{\mathcal{I}}, P_{\mathcal{I}}; X_{\mathcal{J}}, P_{\mathcal{J}})\, \mathbf{1}_{\mathcal{J}}^{\top}$. In this step, we are cautious not to move the input data $\mathbf{x}_{\mathcal{J}}$ and output data $\mathbf{y}_{\mathcal{I}}$ directly to and from the clusters $\mathcal{I}$ and $\mathcal{J}$. Instead, we employ a common technique for fast multipole and hierarchical matrix methods and use the BVH for that. For each cluster $\mathcal{I}, \mathcal{J}$, we allocate scalars $\tilde{x}_{\mathcal{J}}$ and $\tilde{y}_{\mathcal{I}}$. We start only with the leaf clusters and set

$$\tilde{x}_{\mathcal{J}} \leftarrow \textstyle\sum_{T \in \mathcal{J}} \mathbf{x}(T) \quad \text{for each leaf cluster } \mathcal{J}.$$

Then, during a parallel traversal of the BVH in post-order, for each cluster $\mathcal{J}$, we add the $\tilde{x}$-values of its children into $\tilde{x}_{\mathcal{J}}$. After this *upward pass* is finished, we loop over all clusters $\mathcal{I}$ and set

$$\tilde{y}_{\mathcal{I}} \leftarrow \textstyle\sum_{\mathcal{J}} h(X_{\mathcal{I}}, P_{\mathcal{I}}; X_{\mathcal{J}}, P_{\mathcal{J}})\, \tilde{x}_{\mathcal{J}}, \tag{A.1}$$

where the sum runs over the $\mathcal{J}$ such that $(\mathcal{I}, \mathcal{J})$ is admissible. This operation is also best performed by a sparse matrix multiplication. To this end, we fix an ordering of the BVH clusters, e.g. depth-first ordering. Then we assemble a sparse matrix $\tilde{H}$ with the nonzero value $h(X_{\mathcal{I}}, P_{\mathcal{I}}; X_{\mathcal{J}}, P_{\mathcal{J}})$ at the position that correspond to the admissible block cluster $(\mathcal{I}, \mathcal{J})$. Storing $\tilde{x}$ and $\tilde{y}$ as vectors, Equation A.1 amounts to

$$\tilde{y} \leftarrow \tilde{H}\, \tilde{x}.$$

Afterwards, we use a *downward pass* through the BVH to distribute the $\tilde{y}$-values back into the vector $\mathbf{y}$: We traverse the BVH in pre-order and let each cluster $\mathcal{I}$ add its $\tilde{y}$-value into each of its children's $\tilde{y}$-values. Finally each leaf cluster adds its value into each of its member's $\mathbf{y}$-entry:

$$\mathbf{y}(S) \leftarrow \mathbf{y}(S) + \tilde{y}_{\mathcal{I}} \quad \text{for each leaf } \mathcal{I} \text{ and each } S \in \mathcal{I}.$$

The structure of the kernel matrices of L^{σ}, B, and B_0 is very similar. This allows us to use a single block cluster tree to compress all of them. Moreover, the sparsity patterns for the two sparse matrices used to perform Steps 1 and 2 can be shared and the corresponding nonzero values can be computed in a single parallelized loop over the admissible and inadmissible blocks, respectively.

For the application of A_3 to a vector $\mathbf{v}$ of size $3|V|$, we could apply $A = B + B_0$ separately on three vectors $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ of size $|V|$ that each store only one spatial component of the vertex positions. However, it turns out to be more efficient to store $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ as columns of a matrix of size $|V| \times 3$ and to replace the sparse matrix-vector products by sparse matrix-dense matrix products.

Bibliography

[1] M. Ainsworth and C. Glusa. Aspects of an adaptive finite element method for the fractional Laplacian. *Comput. Methods Appl. Mech. Eng.*, 327, 2017. ISSN 0045-7825. 3.1.3, 3.4.3

[2] B. Andrews, B. Chow, C. Guenther, and M. Langford. *Extrinsic Geometric Flows*, volume 206 of *Graduate Studies in Mathematics*. 2020. ISBN 147045596X. 2.2.1

[3] A. Angelidis and F. Neyret. Simulation of smoke based on vortex filament primitives. In *Symp. Comp. Anim.*, pages 87–96, 2005. 3.1.1

[4] T. Ashton and J. Cantarella. A fast octree-based algorithm for computing ropelength. In *Physical And Numerical Models In Knot Theory*, pages 323–341. 2005. 3.1.2

[5] T. Ashton, J. Cantarella, M. Piatek, and E. Rawdon. Knot tightening by constrained gradient descent. *Experimental Mathematics*, 20(1):57–90, 2011. 3.1.2

[6] Trygve Bærland. An auxiliary space preconditioner for fractional laplacian of negative order. *arXiv preprint arXiv:1908.04498*, 2019. 3.1.3

[7] Trygve Bærland, Miroslav Kuchta, and Kent-Andre Mardal. Multigrid methods for discrete fractional Sobolev spaces. *SIAM J. Sci. Comput.*, 41(2):A948–A972, 2019. ISSN 1064-8275. doi: 10.1137/18M1191488. URL https://doi.org/10.1137/18M1191488. 3.1.3

[8] Gavin Barill, Neil Dickson, Ryan Schmidt, David I.W. Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics*, 2018. 1.1.3

[9] J. Barnes and P. Hut. A hierarchical o(n log n) force-calculation algorithm. *Nature*, 324 (6096):446–449, 1986. 1, 3.4.1, 4.4

[10] S. Bartels, P. Reiter, and J. Riege. A simple scheme for the approximation of self-avoiding inextensible curves. *IMA J. Num. Anal.*, 38(2):543–565, 2018. 3.1.2, 3.3

[11] Adam Bednorz and Witold Bednorz. Analytic sphere eversion using ruled surfaces. *Differential Geometry and its Applications*, 64:59–79, 2019. 4.9.1

[12] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete elastic rods. In *ACM Trans. Graph.*, volume 27, page 63. ACM, 2008. 3.1.1

[13] S. Blatt. Boundedness and regularizing effects of o'hara's knot energies. *Journal of Knot Theory and Its Ramifications*, 21(01), 2012. 3.1.2

[14] S. Blatt. The energy spaces of the tangent point energies. *Journal of Topology and Analysis*, 5(3):261–270, 2013. ISSN 1793-5253. 1.1, 2.1.3, 2.3.2, 2.3.2, 3.1.2, 4.2.3

[15] S. Blatt and P. Reiter. Regularity theory for tangent-point energies: the non-degenerate sub-critical case. *Adv. Calc. Var.*, 8(2):93–116, 2015. ISSN 1864-8258. 2.1.3, 2.3.2, 3.1.2, 4.2.1

[16] Alexander I. Bobenko and Peter Schröder. Discrete willmore flow. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, pages 101–es, Goslar, DEU, 2005. Eurographics Association. ISBN 390567324X. 4.1.1

[17] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 185–192, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 3905673134. 5.1

[18] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0521833787. 3.3.5

[19] D. Braess and R. Sarazin. An efficient smoother for the stokes problem. *Applied Numerical Mathematics*, 23(1):3–19, 1997. 3.4.3

[20] K. Brakke. The surface evolver. *Experimental mathematics*, 1(2), 1992. 3.1.2

[21] K. Brakke. Surface evolver manual, 1994. 3.1.3

[22] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, page 594âĂŞ603, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581135211. 1

[23] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 594–603, 2002. 4.1.2

[24] J. Brown, J. Latombe, and K. Montgomery. Real-time knot-tying simulation. *The Visual Computer*, 20(2):165–179, May 2004. ISSN 1432-2315. 3.1.2

[25] G. Buck and J. Orloff. A simple energy function for knots. *Top. Appl.*, 61(3), 1995. ISSN 0166-8641. 1, 1.1, 2.1.3, 4.1.3, 4.2.1

[26] Dorin Bucur and Giuseppe Butazzo. Variational methods in shape optimization problems, 2006. 1, 4.1

[27] J. Calvo, K. Millett, and E. Rawdon. *Physical Knots: Knotting, Linking, and Folding Geometric Objects in* $\mathbb{R}^3$, volume 304. American Mathematical Society, 2002. 3.1.2

[28] Long Chen and Michael Holst. Efficient mesh optimization schemes based on optimal delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering*, 200 (9):967 – 984, 2011. ISSN 0045-7825. 1.1.2, 4.6

[29] A. Chern, F. Knöppel, U. Pinkall, P. Schröder, and S. Weißmann. Schrödinger's smoke. *ACM Trans. Graph.*, 35(4):77, 2016. 3.1.1

[30] Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. Shape from metric. *ACM Transactions on Graphics (TOG)*, 37(4):1–17, 2018. 1, 4.1, 4.9.1

[31] S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon. Isometry-aware preconditioning for mesh parameterization. In *Comp. Graph. Forum*, volume 36, 2017. 3.1.3, 3.5.2, 4.1.4

[32] Ulrich Clarenz, Udo Diewald, Gerhard Dziuk, Martin Rumpf, and R Rusu. A finite element method for surface restoration with smooth boundary conditions. *Computer Aided Geometric Design*, 21(5):427–445, 2004. 1, 4.1

[33] K. Crane, U. Pinkall, and P. Schröder. Robust fairing via conformal curvature flow. *ACM Trans. Graph.*, 32(4), 2013. 1.1.2, 3.1.3, 4.1.1

[34] B. de Wilde, A. ter Mors, and C. Witteveen. Push and rotate: cooperative multi-agent path planning. In *Proc. Conf. Auton. Agents and Multi-agent Sys.*, 2013. 3.6.5

[35] C. DeForest and C. Kankelborg. Fluxon modeling of low-beta plasmas. *J. Atm. Sol.-Terr. Phys.*, 69(1-2):116–128, 2007. 3.1.1

[36] M. Desbrun, M. Meyer, P. Schröder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. ACM SIGGRAPH*, 1999. ISBN 0-201-48560-5. 3.1.3, 4.1.1

[37] E. Di Nezza, G. Palatucci, and E. Valdinoci. Hitchhiker's guide to the fractional Sobolev spaces. *Bull. Sci. Math.*, 136(5):521–573, 2012. ISSN 0007-4497. 2.3.1, 2.3.1

[38] Marc Droske and Martin Rumpf. A level set formulation for willmore flow. *Interfaces and free boundaries*, 6(3):361–378, 2004. 4.1.1

[39] Marion Dunyach, David Vanderhaeghe, Loïc Barthe, and Mario Botsch. Adaptive remeshing for real-time mesh deformation. In *Eurographics 2013*. The Eurographics Association, 2013. 5.1

[40] Gerhard Dziuk. Computational parametric Willmore flow. *Numer. Math.*, 111(1):55–80, 2008. ISSN 0029-599X. 4.7.3

[41] I. Eckstein, J. Pons, Y. Tong, C. Kuo, and M. Desbrun. Generalized Surface Flows for Mesh Processing. In Alexander Belyaev and Michael Garland, editors, *Geometry Processing*. The Eurographics Association, 2007. ISBN 978-3-905673-46-3. 3.1.3, 4.1.4, 4.7.3

[42] J. Ellson, E. Gansner, L. Koutsofios, S. North, and G. Woodhull. Graphviz: Open source graph drawing tools. In *Int. Symp. on Graph Drawing*, pages 483–484, 2001. 3.22

[43] Matthew Elsey and Selim Esedoḡlu. Analogue of the total variation denoising model in the context of geometry processing. *Multiscale Modeling & Simulation*, 7(4):1549–1573, 2009. 1, 4.1

[44] R. Fletcher and C. Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964. 3.5.2

[45] George Francis, John M Sullivan, Rob B Kusner, Ken A Brakke, Chris Hartman, and Glenn Chappell. The minimax sphere eversion. In *Visualization and mathematics*, pages 3–20. Springer, 1997. 4.9.1

[46] George K Francis and GK Francis. *A topological picturebook*, volume 2. Springer, 1987. 4.9.1

[47] M. Freedman, Z. He, and Z. Wang. Mobius energy of knots and unknots. *Annals of Mathematics*, 139(1):1–50, 1994. ISSN 0003486X. 2.1.2

[48] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991. 3.6.3

[49] S. S. Ge and Y. J. Cui. New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation*, 16(5):615–620, 2000. 1

[50] Mark Gillespie, Boris Springborn, and Keenan Crane. Discrete conformal equivalence of polyhedral surfaces. *ACM Trans. Graph.*, 40(4), 2021. 4.9.1

[51] B. Grigoryan, S. Paulsen, et al. Multivascular networks and functional intravascular topologies within biocompatible hydrogels. *Science*, 364(6439):458–464, 2019. 3.6.3

[52] Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67. Citeseer, 2003. 4.9.2

[53] W. Hackbusch. *Hierarchical matrices: algorithms and analysis*, volume 49. Springer, 2015. 1.1.3, 3.4.2, 4.5.1

[54] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4), July 2020. ISSN 0730-0301. 4.9.2

[55] D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun. Asynchronous contact mechanics. In *ACM Trans. Graph.*, volume 28, page 87. ACM, 2009. 3.1.2

[56] David Harmon, Daniele Panozzo, Olga Sorkine, and Denis Zorin. Interference-aware geometric modeling. *ACM Transactions on Graphics (TOG)*, 30(6):1–10, 2011. 4.9.2

[57] Y. Hassan, S. Easa, and A. Abd El Halim. State-of-the-art of three-dimensional highway geometric design. *Can. J. Civ. Eng.*, 25(3):500–511, 1998. 3.1

[58] Behrend Heeren, Martin Rumpf, Max Wardetzky, and Benedikt Wirth. Time-discrete geodesics in the space of shells. In *Computer Graphics Forum*, volume 31, pages 1755–1764. Wiley Online Library, 2012. 5.1

[59] Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32, 1992. 1

[60] Atsushi Ishii. Moves and invariants for knotted handlebodies. *Algebraic & Geometric Topology*, 8(3):1403–1418, 2008. 4.9.1

[61] Atsushi Ishii, Kengo Kishimoto, Hiromasa Moriuchi, and Masaaki Suzuki. A table of genus two handlebody-knots up to six crossings. *Journal of Knot Theory and Its Ramifications*, 21(04):1250035, 2012. 4.9.1, 4.13

[62] Alec Jacobson, Ladislav Kavan, and Olga Sorkine. Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph.*, 32(4), 2013. 1.1.3

[63] Pushkar Joshi and Carlo Séquin. Energy minimizers for curvature-based surface functionals. *Computer-Aided Design and Applications*, 4(5):607–617, 2007. 4.1.1

[64] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. Can mean-curvature flow be modified to be non-singular? In *Computer Graphics Forum*, volume 31, pages 1745–1754. Wiley Online Library, 2012. 4.1.1, 4.9.1

[65] D. Kleckner, L. Kauffman, and W. Irvine. How superfluid vortex knots untie. *Nature Physics*, 12(7):650, 2016. 3.1.1

[66] Leif P. Kobbelt, Jens Vorsatz, and Ulf Labsik. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18(3):119–130, 1999. 4.9.2

[67] Sławomir Kolasiński, Paweł Strzelecki, and Heiko von der Mosel. Compactness and isotopy finiteness for submanifolds with uniformly bounded geometric curvature energies, 2015. 4

[68] S. Kovalsky, M. Galun, and Y. Lipman. Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.*, 35(4):1–11, 2016. 1.1.2, 3.1.3, 3.5.2, 4.1.4, 4.8.2

[69] B. Kubiak, N. Pietroni, F. Ganovelli, and M. Fratarcangeli. A robust method for real-time thread simulation. In *Proc. ACM Symp. Virt. Real. Soft. Tech.*, pages 85–88. ACM, 2007. 3.1.2

[70] R. Kusner and J. Sullivan. Möbius-invariant knot energies. *Ideal knots*, 19, 1998. 2.1, 3.1.2, 4.1.2

[71] M. Kwaśnicki. Ten equivalent definitions of the fractional Laplace operator. *Fract. Calc. Appl. Anal.*, 20(1):7–51, 2017. ISSN 1311-0454. 2.3.1, 2.3.4, 4.2.4

[72] M. Lackenby. Elementary knot theory. *Clay Mathematics Institute*, 2014. 3.1.2

[73] Marc Lackenby. Elementary knot theory. *arXiv preprint arXiv:1604.03778*, 2016. 4.9.1

[74] A. Ladd and L. Kavraki. *Motion Planning for Knot Untangling*, pages 7–23. 2004. ISBN 978-3-540-45058-0. 3.1.2

[75] Silvio Levy and William P Thurston. *Making waves: A guide to the ideas behind Outside In.* Geometry Center, 1995. 4.9.1

[76] C. Liang and K. Mislow. On amphicheiral knots. *J. Math. Chem.*, 15(1), 1994. 3.5.3

[77] Fernando C Marques and André Neves. Min-max theory and the willmore conjecture. *Annals of mathematics*, pages 683–782, 2014. 4.9.1

[78] T. Martin, P. Joshi, M. Bergou, and N. Carr. Efficient non-linear optimization via multi-scale gradient filtering. In *Comp. Grap. Forum*, volume 32, pages 89–100, 2013. 3.1.3, 4.1.4

[79] F. Maucher and P. Sutcliffe. Untangling knots via reaction-diffusion dynamics of vortex strings. *Physical review letters*, 116(17):178101, 2016. 3.1.1

[80] J. McCrae and K. Singh. Sketching piecewise clothoid curves. *Computers & Graphics*, 33 (4):452–461, 2009. 3.1

[81] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *IEEE Visualization*, pages 479–486, Oct 2005. 3.6.6

[82] H. Moreton. *Minimum curvature variation curves, networks, and surfaces for fair free-form shape design.* PhD thesis, University of California, Berkeley, 1992. 3.1

[83] Henry P Moreton and Carlo H Séquin. Functional optimization for fair surface design. *ACM SIGGRAPH Computer Graphics*, 26(2):167–176, 1992. 4.1.1

[84] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Comm. and Im. Repr.*, 18(2):109–118, 2007. 3.5.2

[85] J. O'Hara. Energy of a knot. *Topology*, 30(2):241–247, 1991. 2.1.2

[86] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006. 4.1.2

[87] M. Padilla, A. Chern, F. Knöppel, U. Pinkall, and P. Schröder. On bubble rings and ink chandeliers. *ACM Trans. Graph.*, 38(4):129, 2019. 3.1.1

[88] S. Pellegrino. *Deployable Structures.* CISM International Centre for Mechanical Sciences. Springer Vienna, 2002. ISBN 9783211836859. 1

[89] J. Pérez, B. Thomaszewski, et al. Design and fabrication of flexible rod meshes. *ACM Trans. Graph.*, 34(4), 2015. 3.1, 3.1.1

[90] J. Pérez, M. Otaduy, and B. Thomaszewski. Computational design and automated fabrication of kirchhoff-plateau surfaces. *ACM Trans. Graph.*, 36(4):62, 2017. 3.1.1

[91] P. Pierański. In search of ideal knots. In A. Stasiak, V. Katritch, and L. Kauffman, editors, *Ideal Knots*, volume 19. World Scientific, 1998. 3.1.2, 3.5.2

[92] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993. 1.1.2, 3.1.3, 4.1.4

[93] E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Math. Model. Num. Anal.*, 3(R1):35–43, 1969. 3.5.2

[94] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. In *Comp. Graph. Forum*, volume 21, pages 279–287, 2002. 3.3.5

[95] R. Renka and J. Neuberger. Minimal surfaces and sobolev gradients. *SIAM Journal on Scientific Computing*, 16(6):1412–1427, 1995. 3.1.3, 4.1.4

[96] Leonardo Sacht, Etienne Vouga, and Alec Jacobson. Nested cages. *ACM Transactions on Graphics (TOG)*, 34(6):1–14, 2015. 4.9.2

[97] R. Scharein. *Interactive Topological Drawing*. PhD thesis, University of British Columbia, 1998. 3.1.2, 3.3.5, 3.5.1, 3.5.2, 4.9.1

[98] Patrick Schmidt, Marcel Campen, Janis Born, and Leif Kobbelt. Inter-surface maps via constant-curvature metrics. *ACM Transactions on Graphics (TOG)*, 39(4):119–1, 2020. 4.9.1

[99] H. Schumacher. On H^2-gradient Flows for the Willmore Energy. *arXiv e-prints*, Mar 2017. 3.1.3, 4.1.4, 4.7.3, 4.8.2

[100] Lin Shi, Yizhou Yu, Nathan Bell, and Wei-Wen Feng. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph.*, 25(3):1108–1117, 2006. ISSN 0730-0301. 5.1

[101] J. Smith and S. Schaefer. Bijective parameterization with free boundaries. *ACM Trans. Graph.*, 34(4):1–9, 2015. 3.3.5

[102] Yousuf Soliman, Albert Chern, Olga Diamanti, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. Constrained willmore surfaces. *ACM Trans. Graph.*, 40(4), 2021. 4.1.1, 4.1.4, 4.9.1

[103] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007. 4.9.2

[104] P. Strzelecki and H. von der Mosel. Tangent-point self-avoidance energies for curves. *J. Knot Theory Ramifications*, 21(5), 2012. ISSN 0218-2165. 1.1

[105] P. Strzelecki and H. von der Mosel. Geometric curvature energies: facts, trends, and open problems. In *New Directions in Geometric and Applied Knot Theory*. 2017. 3.3.1

[106] PawełStrzelecki and Heiko von der Mosel. Tangent-point repulsive potentials for a class of non-smooth m-dimensional sets in $\mathbb{R}^n$. Part I: Smoothing and self-avoidance effects. *J. Geom. Anal.*, 23(3):1085–1139, 2013. ISSN 1050-6926. doi: 10.1007/s12220-011-9275-z. 4.1.3, 4.2.3

[107] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Symp. Comp. Anim.*, pages 181–190, 2005. 3.5.2

[108] H. Triebel. *Theory of function spaces*, volume 78 of *Monographs in Mathematics*. 1983. ISBN 3-7643-1381-1. 2.3.1

[109] S. Walker. Shape optimization of self-avoiding curves. *J. Comp. Phys.*, 311, 2016. 3.1.2

[110] S. Weißmann and U. Pinkall. Filament-based smoke with vortex shedding and variational reconnection. In *ACM Trans. Graph.*, volume 29, 2010. 3.1.1

[111] David Wells. *The Penguin dictionary of curious and interesting numbers*. Penguin, 1997. 4.15

[112] Peter Wriggers and Giorgio Zavarise. Computational contact mechanics. *Encyclopedia of computational mechanics*, 2004. 4.1.2

[113] Z. Yan, S. Schiller, G. Wilensky, N. Carr, and S. Schaefer. k-curves: interpolation at local maximum curvature. *ACM Trans. Graph.*, 36(4), 2017. 3.6.4

[114] Zi Ye, Olga Diamanti, Chengcheng Tang, Leonidas Guibas, and Tim Hoffmann. A unified discrete framework for intrinsic and extrinsic dirac operators for geometry processing. In *Computer Graphics Forum*, volume 37, pages 93–106. Wiley Online Library, 2018. 4.9.1

[115] C. Yu, K. Crane, and S. Coros. Computational design of telescoping structures. *ACM Trans. Graph.*, 36(4), 2017. 3.1

[116] Chris Yu, Henrik Schumacher, and Keenan Crane. Repulsive curves. *ACM Trans. Graph.*, 40(2), May 2021. ISSN 0730-0301. doi: 10.1145/3439429. 1.1.2, 2, 3.5, 4.8.2

[117] J. Yu and S. LaValle. Multi-agent path planning and network flow. In *Algorithmic Foundations of Robotics X*, pages 157–173. Springer, 2013. 3.6.5

[118] J. Zehnder, S. Coros, and B. Thomaszewski. Designing structurally-sound ornamental curve networks. *ACM Trans. Graph.*, 35(4):99, 2016. 3.1, 3.1.1

[119] Fuzhen Zhang. *The Schur Complement and its Applications*, volume 4 of *Numerical Methods and Algorithms*. Springer, New York, 2005. 4.5.3

[120] Y. Zhu, R. Bridson, and D. Kaufman. Blended cured quasi-newton for distortion optimization. *ACM Trans. Graph.*, 37(4):1–14, 2018. 1.1.2, 3.1.3, 3.5.2, 4.1.4