Discrete Differential Geometry: An Applied Introduction

SIGGRAPH 2006
COURSE NOTES

ORGANIZER
Eitan Grinspun

LECTURERS
Mathieu Desbrun
Konrad Polthier
Peter Schröder
Ari Stern

Preface

The behavior of physical systems is typically described by a set of continuous equations using tools such as geometric mechanics and differential geometry to analyze and capture their properties. For purposes of computation one must derive discrete (in space and time) representations of the underlying equations. Researchers in a variety of areas have discovered that theories, which are discrete from the start, and have key geometric properties built into their discrete description can often more readily yield robust numerical simulations which are true to the underlying continuous systems: they exactly preserve invariants of the continuous systems in the discrete computational realm.

This volume documents the full day course Discrete Differential Geometry: An Applied Introduction *at SIGGRAPH '06 on 30 July 2006. These notes supplement the lectures given by Mathieu Desbrun, Eitan Grinspun, Konrad Polthier, Peter Schröder, and Ari Stern. These notes include contributions by Miklos Bergou, Alexander I. Bobenko, Sharif Elcott, Matthew Fisher, David Harmon, Eva Kanso, Markus Schmies, Adrian Secord, Boris Springborn, Ari Stern, John M. Sullivan, Yiying Tong, Max Wardetzky, and Denis Zorin, and build on the ideas of many others.*

A chapter-by-chapter synopsis

The course notes are organized similarly to the lectures. We introduce discrete differential geometry in the context of discrete curves and curvature (Chapter 1). The overarching themes introduced here, *convergence* and *structure preservation*, make repeated appearances throughout the entire volume. We ask the question of which quantities one should measure on a discrete object such as a triangle mesh, and how one should define such measurements (Chapter 2). This exploration yields a host of measurements such as length, area, mean curvature, etc., and these in turn form the basis for various applications described later on. We conclude the introduction with a summary of curvature measures for discrete surfaces (Chapter 3).

The discussion of immersed surfaces paves the way for a discrete treatment of thin-shell mechanics (Chapter 4), and more generally of isometric curvature energies, with applications to fast cloth simulation and Willmore flow (Chapter 5). Continuing with the theme of discrete surfaces, we define straightest geodesics on polyhedral surfaces with applications to integration of vector fields (Chapter 6).

At this point we shift down to explore the low-level approach of discrete exterior calculus: after an overview of the field (Chapter 7), we lay out the simple tools for implementing DEC (Chapter 8). With this in place, numerically robust and efficient simulations of the Navier-Stokes equations of fluids become possible (Chapter 9).

Simulations of thin-shells, cloth, and fluids, and geometric modeling problems such as fairing and parametrization, require robust numerical treatment in both space and time: we describe the intrinsic Laplace-Beltrami operator, which satisfies a local maximum principle and leads to better conditioned linear systems (Chapter 10), and we overview the variational time integrators offered by the geometric mechanics paradigm (Chapter 11).

Eitan Grinspun
with Mathieu Desbrun, Konrad Polthier, and Peter Schröder
11 May 2006

Contents

Chapter 1: Introduction to Discrete Differential Geometry: The Geometry of Plane Curves 1
Eitan Grinspun and Adrian Secord

Chapter 2: What can we measure?..... 5
Peter Schröder

Chapter 3: Curvature measures for discrete surfaces..... 10
John M . Sullivan

Chapter 4: A discrete model for thin shells..... 14
Eitan Grinspun

Chapter 5: Discrete Quadratic Curvature Energies..... 20
Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun

Chapter 6: Straightest Geodesics on Polyhedral Surface..... 30
Konrad Polthier and Markus Schmies

Chapter 7: Discrete Differential Forms for Computational Modeling..... 39
Mathieu Desbrun, Eva Kanso, and Yiying Tong

Chapter 8: Building Your Own DEC at Home..... 55
Sharif Elcott and Peter Schröder

Chapter 9: Stable, Circulation-Preserving, Simplicial Fluids 60
Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun

Chapter 10: An Algorithm for the Construction of Intrinsic Delaunay Triangulations
with Applications to Digital Geometry Processing 69
Matthew Fisher, Boris Springborn, Alexander I. Bobenko, Peter Schröder

Chapter 11: Discrete Geometric Mechanics for Variational Time Integrators 75
Ari Stern and Mathieu Desbrun

Chapter 1: Introduction to Discrete Differential Geometry: The Geometry of Plane Curves

Eitan Grinspun
Columbia University

Adrian Secord
New York University

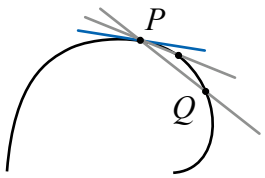
1 Introduction

The nascent field of discrete differential geometry deals with discrete geometric objects (such as polygons) which act as analogues to continuous geometric objects (such as curves). The discrete objects can be measured (length, area) and can interact with other discrete objects (collision/response). From a computational standpoint, the discrete objects are attractive, because they have been designed from the ground up with data-structures and algorithms in mind. From a mathematical standpoint, they present a great challenge: the discrete objects should have properties which are analogues of the properties of continuous objects. One important property of curves and surfaces is their curvature, which plays a significant role in many application areas (see, *e.g.*, Chapters 4 and 5). In the continuous domain there are remarkable theorems dealing with curvature; a key requirement for a discrete curve with discrete curvature is that it satisfies analogous theorems. In this chapter we examine the curvature of continuous and discrete curves on the plane.

The notes in this chapter draw from a lecture given by John Sullivan in May 2004 at Oberwolfach, and from the writings of David Hilbert in his book *Geometry and the Imagination*.

2 Geometry of the Plane Curve

Consider a plane curve, in particular a small piece of curve which does not cross itself (a *simple* curve). Choose two points, P and Q, on this curve and connect them with a straight line: a *secant*. Fixing P as the "hinge," rotate the secant about P so that Q slides along the curve toward Q. If the curve is sufficiently smooth ("*tangent-continuous* at P") then the secant approaches a definite line: the *tangent*. Of all the straight lines passing through P, the tangent is the best approximation to the curve. Consequently we define the *direction* of the curve at P to be the direction of the tangent, so that if two curves intersect at a point P their angle of intersection is given by the angle formed by their tangents at P. If both curves have identical tangents at P then we say "the curves are tangent at P." Returning to our single curve, the line perpendicular to the tangent and passing through P is called the *normal* to the curve at P. Together the tangent and normal form the axes of a local rectangular coordinate system. In addition, the tangent can be thought of as a local approximation to the curve at P.



A better approximation than the tangent is the *circle of curvature*: consider a circle through P and two neighboring points on the curve, and slide the neighboring points towards P. If the curve is sufficiently smooth ("*curvature-continuous* at P") then the circle thus approaches a definite position known as the circle of curvature or *osculating circle*; the center and radius of the osculating circle are the *center of curvature* and *radius of curvature* associated to point P on the curve. The inverse of the radius is κ, the *curvature* of the curve at P.

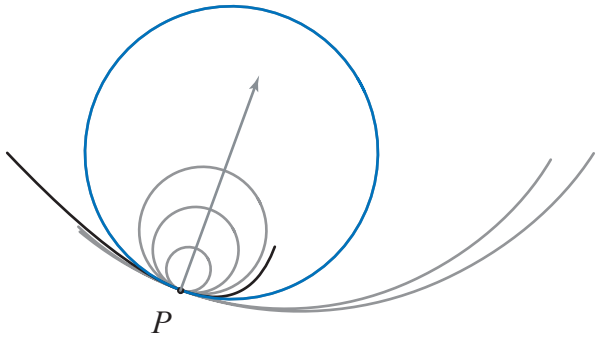


Figure 1: The family of tangent circles to the curve at point P. The circle of curvature is the only one crossing the curve at P.

If we also consider a sense of traversal along the curve segment (think of adding an arrowhead at one end of the segment) then we may measure the *signed* curvature, identical in magnitude to the curvature, but negative in sign whenever the curve is turning clockwise (think of riding a bicycle along the curve: when we turn to the right, it is because the center of curvature lies to the right, and the curvature is negative).

Another way to define the circle of curvature is by considering the infinite family of circles which are tangent to the curve at P (see Figure 1). Every point on the normal to the curve at P serves as the center for one circle in this family. In a small neighborhood around P the curve divides the plane into two sides. Every circle (but one!) in our family lies entirely in one side or the other. Only the circle of curvature however spans both sides, crossing the curve at P. It divides the family of tangent circles into two sets: those with radius smaller than the radius of curvature lying on one side, and those with greater radius lying on the other side. There may exist special points on the curve at which the circle of curvature does not locally cross the curve, and in general these are finite and isolated points where the curve has a (local) axis of symmetry (there are four such points on an ellipse). However on a circle, or a circular arc, the special points are infinitely many and not isolated.

That the circle of curvature crosses the curve may be reasoned by various arguments. As we traverse the curve past

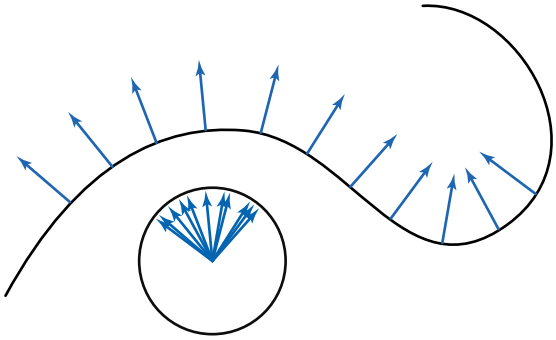


Figure 2: The Gauss map assigns to every point on the curve a corresponding point on the unit circle.

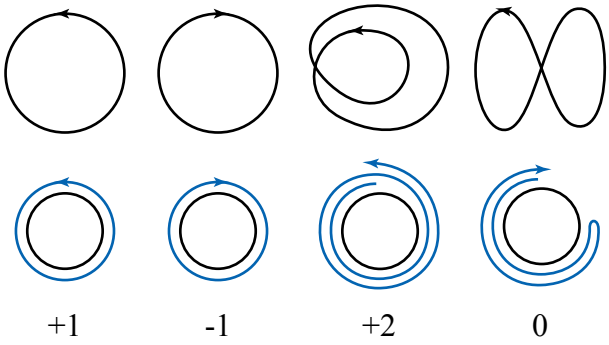


Figure 3: Turning numbers of various closed curves. *Top row:* Two simple curves with opposite sense of traversal, and two self-intersecting curves, one of which "undoes" the turn. *Bottom row:* Gaussian image of the curves, and the associated turning numbers.

point P, the curvature is typically either increasing or decreasing, so that in the local neighborhood of P, so that the osculating circle in comparison to the curve will have a higher curvature on one side and lower on the other. An alternative argument considers our three point construction. Trace along a circle passing through three consecutive points on the curve to observe that the circle must pass from side A to side B on the first point, B to A on the second, and A to B on the third. Similar reasoning of our two-point construction shows that in general the tangent does not cross the curve—the isolated exceptions are the *points of inflection*, where the radius of curvature is infinite and the circle of curvature is identical to the tangent.

Informally we say that P, the tangent at P, and the osculating circle at P have one, two, and three coincident points in common with the curve, respectively. Each construction in sequence considers an additional approaching point in the neighborhood of P and the so-called *order of approximation* (0, 1, and 2 respectively) is identical to the number of additional points.

In 1825 Karl F. Gauss introduced a new tool for thinking about the shape of curves and surfaces. Begin by fixing a sense of traversal for the curve, naturally inducing for every point on the curve a direction for the tangent. By convention, the normal points a quarter turn counterclockwise from tangent direction. Gauss's idea is to draw a unit circle on the plane of the curve, and for any point on the curve, to represent the normal by the radius of the circle parallel to the normal and having the same sense as the normal. To any point P on the curve, the *Gauss map* assigns a point Q on the unit circle, namely the point where the *radius* meets the circle (here, radius means the line segment from the center of the circle to a point on the circumference). Observe that the normal at P is parallel to the radius of the circle, and the tangent to the curve at P is parallel to the tangent to the circle at Q. That the tangent at P and Q are parallel is used to simplify important definitions in differential geometry (see, *e.g.*, the definition of the shape operator in the chapter on discrete shells). While the Gauss map assigns exactly one point on the unit circle to any point on the curve, there may be multiple points on the curve that map to the same point on the circle, *i.e.*the map is not one-to-one.

Consider the image of the curve under the Gauss map: the *Gaussian image* of a curve is the union of all points on the unit circle corresponding to all points on the given curve. For an open curve, the Gaussian image may be an arc or may be the unit circle. Consider a closed simple plane curve: the image is always the unit circle. If we allow the closed curve to intersect itself, we can count how many times the image completely "wraps around" the unit circle (and in which sense): this is the *turning number* or the *index of rotation*, denoted k. It is unity for a simple closed curve traversed counterclockwise. It is zero or ± 2 for curve that self-intersects once, depending on the sense of traversal and on whether or not the winding is "undone."

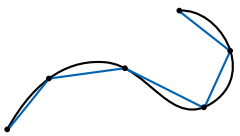
Turning Number Theorem. An old and well-known fact about curves is that the integral of signed curvature over a closed curve, Ω, is dependent *only* on the turning number:

$$\int_\Omega \kappa \, ds = 2\pi k \ .$$

No matter how much we wiggle and bend the curve, if we do not change its turning number we do not change its *total signed curvature*[1]. To change the total signed curvature of Ω we are forced to alter its turning number by adjusting the curve to introduce (or rearrange) self-intersecting loops. This theorem about the significance of the turning number is a piece of mathematical *structure*: together all the structure we discover embodies our understanding of differential geometry. Consequently, our computational algorithms will take advantage of this structure. In computing with discrete approximations of continuous geometry, we will strive to keep key pieces of structure intact.

3 Geometry of the Discrete Plane Curve

Given a curve, r, approximate it drawing an *inscribed polygon* p: a finite sequence of (point) *vertices*, $V_1, V_2, \ldots V_n$, ordered by a traversal of the curve, and line segments connecting successive vertices[2].



[1] Beware that in the context of space curves, the phrase "total curvature" is occasionally used to denote the Pythagorean sum of torsion and curvature—a pointwise quantity like curvature. In contrast, here we mean the integral of curvature over the curve.

[2] While we concern ourselves here only with plane curves, this treatment may be extended to curves in a higher-dimensional am-

The length of the inscribed polygon is given by

$$len(p) = \sum_{i=0}^{n} d(V_i, V_{i+1}) \ ,$$

where $d(\cdot,\cdot)$ measures the euclidean distance[3] between two points. We find the length of the continuous curve by taking the supremum over all possible inscriptions:

$$len(r) = \sup_{p \text{ inscribed in } r} len(p) \ .$$

Next, choose a sense of traversal along the curve, naturally inducing a sense for the inscribed polygon. The (discrete) *total signed curvature* of the inscribed polygon is given by

$$tsc(p) = \sum_{i=0}^{n} \alpha_i \ ,$$

where α_i is the signed *turning angle* at vertex V_i, measured in the sense that a clockwise turn has negative sign; if p is open then $\alpha_0 = \alpha_n = 0$. (N.B.: the turning *angle* is a local quantity at each vertex, whereas the turning *number* is a global quantity of a curve—these are two distinct concepts). Again, we may express the total signed curvature of the continuous curve by taking the supremum over all possible inscribed polygons:

$$tsc(r) = \sup_{p \text{ inscribed in } r} tsc(p) \ .$$

A definition based on suprema serves as an elegant foundation for defining the (integral quantities) length and total curvature of a smooth curve using only very simple polygonal geometry; however suprema are typically is not well suited for computation. For an equivalent, computationally meaningful definition, we construct an infinite sequence of inscribed polygons, p_1, p_2, p_3, ..., that approaches the position of r; analogous definitions of $len(r)$ and $tsc(r)$ are formulated as limits of measurements over elements of the sequence.

To clarify what we mean by "the inscribed polygon p approaches the position of r," define the *geometric mesh size* of p by the length of its longest line segment:

$$h(p) = \max_{0 \le i < n} d(V_i, V_{i+1}) \ .$$

Suppose that r is a smooth simple curve. By smooth we mean that every point on the curve has a unique well-defined tangent[4]. Then one can show that given a sequence p_1, p_2, p_3, ... such that $h(p_i)$ vanishes in the limit of the sequence, then $len(p_i)$ approaches $len(r)$. An analogous statement holds for total curvature, as summarized by the following statement:[5]

bient space, $M^m \subseteq \mathbb{R}^d$, by replacing line segments with shortest geodesics in this definition, and straight-line distance by length of geodesic in subsequent definitions.

[3]It measures distance using the metric of the ambient space, in our case $\mathbb{R}^2$.

[4]Observe that smoothness here is in a purely geometric sense—the notion of parametric smoothness in the context of parameterized curves is a different matter altogether.

[5]Note that there are sequences of pathological polygons whose mesh size vanishes yet the limit of the sequence does not approach the curve. For example, if the curve is a circle, consider a polygon whose vertices all cluster about a single point of the circle.

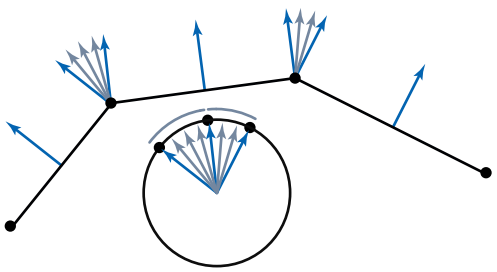


Figure 4: The discrete Gauss map assigns to every edge of the polygon a corresponding point on the unit circle, and to every vertex of the polygon a corresponding arc on the unit circle.

Convergence. A key recurring theme in discrete differential geometry is the *convergence* of a measurement taken over a sequence of discrete objects each better approximating a particular smooth object. In the case of a plane curve, a sequence of inscribed polygons, each closer in position to the curve, generates a sequence of measurements that approach that of the curve:

$$len(r) = \lim_{h(p_i) \to 0} len(p_i) \ ,$$

$$tsc(r) = \lim_{h(p_i) \to 0} tsc(p_i) \ .$$

Establishing convergence is a key step towards numerical computations which use discrete objects as approximations to continuous counterparts. Indeed, one might argue that the notion of continuous *counterpart* is only meaningful in the context of established convergence. Put simply, if we choose an inscribed polygon as our discrete analogue of a curve, then as the position of the approximating polygon approaches the curve, the measurements taken on the approximant should approach those of the underlying curve.

Next, consider the tangents, normals, and Gaussian image of a closed polygon p. Repeating the two-point limiting process we used to define the tangent for a point on the curve, we observe that every vertex of the polygon has two limiting tangents (thus two normals), depending on the direction from which the limit is taken (see Figure 4). Define the Gaussian image of p by assigning to every vertex V_i the arc on the unit circle whose endpoints are the two limiting normals and whose signed angle equals the signed turning angle α_i, *i.e.*, as if one "smoothly interpolated" the two normals in the Gaussian image. Every point on the polygon away from the vertices has a unique normal which corresponds in the Gaussian image to the meeting point of consecutive arcs. The sense of traversal along the polygon induces a natural sense of traversal along the arcs of the Gaussian image. With this construction in place, our definition of turning number for a smooth plane curve carries over naturally to the setting of closed polygons. Not that for for open polygons, the Gaussian image of vertices at the endpoints is a point on the unit circle (a degenerate arc).

As long as the length of the longest line segment shrinks, *i.e.* the polygon clusters more tightly around the point, then this sequence of polygons will satisfy our definition but will clearly not converge to a circle. One may introduce stronger requirements on the polygon sequence to exclude such pathological sequences.

Structure preservation. Does the Turning Number Theorem hold for discrete curves? Yes. Recall that the sum of exterior angles of a simple closed polygon is 2π. This observation may be generalized to show that $tsc(p) = 2\pi k$ where k is the turning number of the polygon. We stress a key point: the Turning Number Theorem is *not* a claim that the total signed curvature converges to a multiple of 2π in the limit of a finely refined inscribed polygon. The Turning Number Theorem is preserved *exactly* and it holds for *any* (arbitrarily coarse) closed polygon. Note, however, that the turning number of an inscribed polygon may not match that of the smooth curve, at least until sufficiently many vertices are added (in the right places) to capture the topology of the curve.

4 Parameterization of the Plane Curve

So far in our exploration of curves our arguments have never explicitly made reference to a system of coordinates. This was to stress the point that the geometry (or *shape*) of the curve can be described without reference to coordinates. Nevertheless, the idea of *parameterizing* a curve occurs throughout applied mathematics. Unfortunately, parameterization can sometimes obscure geometric insight. At the same time, it is an exceedingly useful computational tool, and as such we complete our exploration of curves with this topic.

In working with curves it is useful to be able to indicate particular points and their neighborhoods on the curve. To that end we *parameterize* a curve over a real interval mapping each parameter point, $t \in [0, a]$, to a point $R(t)$ on the plane:

$$R : [0, a] \to \mathbb{R}^2 \ .$$

Thus the endpoints of finite open curve are $R(0)$ and $R(a)$; for closed curves we require $R(0) = R(a)$.

The parameterization of a curve is not unique. Besides the geometric information encoded in the image of R, the parameterization also encodes a parameterization-dependent *velocity*. To visualize this, observe that moving the parameter at unit velocity slides a point $R(t)$ along the curve: the rate of change of $R(t)$, or velocity, is the vector $\vec{v}(t) = \frac{d}{dt}R(t)$. Indeed, given any strictly increasing function $t(s) : [0, b] \to [0, a]$ we *reparameterize* the curve as $R(t(s))$ so that moving along $s \in [0, b]$ generates the same points along the curve; the geometry remains the same, but by chain rule of the calculus the velocity is now $\vec{v}(s) = \frac{d}{ds}R(t(s)) = \frac{d}{dt}R(t(s))\frac{d}{ds}t(s)$: at every point the $R(t(s))$ reparameterization scales the velocity by $\frac{d}{ds}t(s)$.

Given a parameterized curve there is a unique reparameterization, $\hat{R}(s) = R(t(s))$, with the property that $\|\vec{v}(s)\| = 1,\ s \in [0, b]$. In *arc-length* parameterization of a curve, unit motion along the parameter s corresponds to unit motion along the length of the curve. Consequently, s is the length traveled along the curve walking from $\hat{R}(0)$ to $\hat{R}(s)$, therefore b is the length of the entire curve.

In the special setting of an arc-length parameterization the curvature at a point $R(s)$ is identical to the second derivative $\frac{d^2}{ds^2}R(s)$. It is a grave error to identify curvatures with second derivatives in general. The former is a geometric quantity only, and we defined it without reference to a parameterization; the latter encodes both geometry and velocity, and is parameterization-dependent. Here a spaceship analogy is helpful. If a spaceship travels at unit speed along a curved path, the curvature give the acceleration of the spaceship. Now if the spaceship travels at a nonuniform velocity along the path, then part of the acceleration is due to curvature, and part is due to speeding up and slowing down. A parameterization encodes velocity—this can be extremely useful for some applications.

Parameterization enables us to reformulate our statement of convergence. Given a sequence of parameter values, $0 = t_1 \leq t_2 \ldots \leq t_{n-1} \leq t_n = b$, for a "sufficiently well-behaved" parameterization of a "sufficiently well-behaved" curve[6], we may form an inscribed polygon taking $V_i = R(t_i)$. Then the *parametric mesh size* of the inscribed polygon is the greatest of all parameter intervals $[t_i, t_i + 1]$:

$$h_R(p) = \max_i (t_{i+1} - t_i) \ .$$

Unlike *geometric* mesh size, *parametric* mesh size is dependent on the chosen parameterization.

As before, consider a sequence of inscribed polygons, each sampling the curve at more parameter points, and in the limit sampling the curve at all parameter points: the associated sequences of discrete measurements approach their continuous analogs:

$$len(r) = \lim_{h_R(p_i)\to 0} len(p_i) \ ,$$

$$tsc(r) = \lim_{h_R(p_i)\to 0} tsc(p_i) \ .$$

5 Conclusion and Overview

So far we have looked at the geometry of a plane curve and demonstrated that it is possible to define its discrete analogue. The formulas for length and curvature of a discrete curve (a polygon) are immediately amenable to computation. *Convergence* guarantees that in the presence of abundant computational resources we may refine our discrete curve until the measurements we take match to arbitrary precision their counterparts on a smooth curve. We discussed an example of *structure preservation*, namely that the Turning Number Theorem holds exactly for discrete curves, even for coarse mesh sizes. If we wrote an algorithm whose correctness relied on the Turning Number Theorem, then the algorithm could be applied to our discrete curve.

The following chapters will extend our exploration of discrete analogues of the objects of differential geometry to the settings of surfaces and volumes and to application areas spanning physical simulation (thin shells and fluids) and geometric modeling (remeshing and parameterization). In each application area algorithms make use of mathematical structures that are carried over from the continuous to the discrete realm. We are *not* interested in preserving structure just for mathematical elegance—each application demonstrates that by carrying over the right structures from the continuous to the discrete setting, the resulting algorithms exhibit impressive computational and numerical performance.

[6] Indeed, the following theorems depend on the parameterization being *Lipschitz*, meaning that small changes in parameter value lead to small motions along the curve:

$$d(R(a), R(b)) \leq C|a - b| \ ,$$

for some constant C. The existence of a Lipschitz parameterization is equivalent to the curve being *rectifiable*, or having finite arclength. Further care must be taken in allowing non-continuous curves with finitely many isolated jump points.

What Can We Measure?

Peter Schröder
Caltech

1 Introduction

When characterizing a shape or changes in shape we must first ask, what can we measure about a shape? For example, for a region in $\mathbb{R}^3$ we may ask for its volume or its surface area. If the object at hand undergoes deformation due to forces acting on it we may need to formulate the laws governing the change in shape in terms of measurable quantities and their change over time. Usually such measurable quantities for a shape are defined with the help of integral calculus and often require some amount of smoothness on the object to be well defined. In this chapter we will take a more abstract approach to the question of measurable quantities which will allow us to define notions such as mean curvature integrals and the curvature tensor for piecewise linear meshes without having to worry about the meaning of second derivatives in settings in which they do not exist. In fact in this chapter we will give an account of a classical result due to Hadwiger, which shows that for a convex, compact set in $\mathbb{R}^n$ there are only $n+1$ unique measurements if we require that the measurements be invariant under Euclidian motions (and satisfy certain "sanity" conditions). We will see how these measurements are constructed in a very straightforward and elementary manner and that they can be read off from a characteristic polynomial due to Steiner. This polynomial describes the volume of a family of shapes which arise when we "grow" a given shape. As a practical tool arising from these consideration we will see that there is a well defined notion of the curvature tensor for piecewise linear meshes and we will see very simple formulas for quantities needed in physical simulation with piecewise linear meshes. Much of the treatment here will be limited to convex bodies to keep things simple.

The treatment in this chapter draws heavily upon work by Gian-Carlo Rota and Daniel Klein, Hadwigers pioneering work, and some recent work by David Cohen-Steiner and colleagues.

2 Geometric Measures

To begin with let us define what we mean by a measure. A measure is a function μ defined on a family of subsets of some set S, and it takes on real values: $\mu : L \to \mathbb{R}$. Here L denotes this family of subsets and we require of L that it is closed under finite set union and intersection as well as that it contains the empty set, $\emptyset \in L$. The measure μ must satisfy two axioms: (1) $\mu(\emptyset) = 0$; and (2) $\mu(A \cup B) = \mu(A) + \mu(B) - \mu(A \cap B)$ whenever A and B are measureable. The first axiom is required to get anything that has a hope of being well defined. If $\mu(\emptyset)$ was not equal to zero the measure of some set $\mu(A) = \mu(A \cup \emptyset) = \mu(A) + \mu(\emptyset)$ could not be defined. The second axiom captures the idea that the measure of the union of two sets should be the sum of the measures minus the measure of their overlap. For example, consider the volume of the union of two sets which clearly has this property. It will also turn out that the additivity property is the key to reducing measurements for complicated sets to measurements on simple sets. We will furthermore require that all measures we consider be invariant under Euclidian motions, *i.e.*, translations and rotations. This is so that our measurements do not depend on where we place the coordinate origin and how we orient the coordinate axes. A measure which depended on these wouldn't be very useful.

Let's see some examples. A well known example of such a measure is the volume of bodies in $\mathbb{R}^3$. Clearly the volume of the empty body is zero and the volume satisfies the additivity axiom. The volume also does not depend on where the coordinate origin is placed and how the coordinate frame is rotated. To uniquely tie down the volume there is one final ambiguity due to the units of measurement being used, which we must remove. To this end we enforce a normalization which states that the volume of the unit, coordinate axis aligned parallelipiped in $\mathbb{R}^n$ be one. With this we get

$$\mu_n^n(x_1, \ldots, x_n) = x_1 \cdot \ldots \cdot x_n$$

for x_1 to x_n the side lengths of a given axis aligned parallelipiped. The superscript n denotes this as a measure on $\mathbb{R}^n$, while the subscript denotes the type of measurement being taken. Clearly the definition of μ_n^n is translation invariant. It also does not depend on rotations of the global coordinate system since these do not change the side lengths of our parallelipiped. Notice that we have only defined the meaning of μ_n^n for axis aligned parallelipipeds as well as finite unions and intersections of such parallelipipeds. The definition can be extended to more general bodies through a limiting process akin to how Riemann integration fills the domain with ever smaller boxes to approach the entire domain in the limit. There is nothing here that prevents us from performing the same limit process. In fact we will see later that once we add this final requirement, that the measure is continuous in the limit, the class of such measures is completely tied down. This is Hadwiger's famous theorem. But, more on that later.

Of course the next question is, are there other such invariant measures? Here is a proposal:

$$\mu_{n-1}^n(x_1, \ldots, x_n) = \\ x_1x_2 + x_1x_3 + \ldots + x_1x_n + x_2x_3 + \ldots + x_2x_n \ldots$$

For an axis aligned parallelipiped in $\mathbb{R}^3$ we'd get

$$\mu_2^3(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3x_1$$

which is just half the surface area of the paralellipiped with sides x_1, x_2, and x_3. Since we have the additivity property we can certainly extend this definition to more general bodies through a limiting process and find that we get, up to normalization, the surface area.

Continuing in this fashion we are next led to consider

$$\mu_1^3(x_1, x_2, x_3) = x_1 + x_2 + x_3$$

(and similarly for μ_1^n). For a parallelipiped this function measures one quarter the sum of lengths of its bounding edges. Once again this new measure is clearly rigid motion invariant. What we need to check is whether it satisfies the additivity theorem. Indeed it does, which is easily checked for the union of two paralellipipeds. What is less clear is what this measure represents if we extend it to more general shapes where the notion of "sum of edge lengths" is not clear. The resulting continuous measure is sometimes referred to as the *mean width*.

From these simple examples we can see a pattern. For Euclidian n-space we can use the elementary symmetric polynomials in edge lengths to define n invariant measures

$$\mu_k^n(x_1, \ldots, x_n) = \sum_{1 \le i_1 < i_2 < \ldots < i_k \le n} x_{i_1} x_{i_2} \ldots x_{i_k}$$

for $k = 1, \ldots, n$ for parallelipipeds. To extend this definition to more general bodies we'll follow ideas from geometric probability. In particular we will extend these measures to the ring of compact convex bodies, *i.e.*, finite unions and intersections of compact convex sets in $\mathbb{R}^n$.

3 How Many Points, Lines, Planes,... Hit a Body?

Consider a compact convex set, a *convex body*, in $\mathbb{R}^n$ and surround it by a box. One way to measure its volume is to count the number of points which, when randomly thrown into the box, hit the body versus those that hit empty space inside the box. To generalize this idea we consider *affine subspaces* of dimension $k < n$ in $\mathbb{R}^n$. Recall that an affine subspace of dimension k is spanned by $k+1$ points $p_i \in \mathbb{R}^n$ (in general position), *i.e.*, the space consists of all points q which can be written as affine combinations $q = \sum_i \alpha_i p_i$, $\sum_i \alpha_i = 1$. Such an affine subspace is simply a linear subspace translated, *i.e.*, it does not necessarily go through the origin. For example, for $k = 1$, $n = 3$ we will consider all lines—a line being the set of points one can generate as affine combinations of two points on the line—in three space. Let $\lambda_1^3(R)$ be the measure of all lines going through a rectangle in $\mathbb{R}^3$. Then

$$\lambda_1^3(R) = c\mu_2^3(R),$$

i.e., the measure of all lines which meet the rectangle is proportional to the area of the rectangle. To see this, note that a given line (in general position) either meets the rectangle once or not at all. Conversely for a given point in the rectangle there is a whole set of lines—a sphere's worth—which "pierce" the rectangle in the given point. The measure of those lines is proportional to the area of the unit sphere. Since this is true for all points in the rectangle we see that the total measure of all such lines must be proportional to the area of the rectangle with a constant of proportionality depending on the measure of the sphere. For now such constants are irrelevant for our considerations so we will just set them to unity. Given a more complicated shape C in a plane nothing prevents from performaing a limiting process and we see that the measure of lines meeting C is

$$\lambda_1^3(C) = \mu_2^3(C),$$

i.e., it is proportional to the area of the region C. Given a union of rectangles $D = \cup_i R_i$, each living in a different plane, we get

$$\int X_D(\omega)\, d\lambda_1^3(\omega) = \sum_i \mu_2^3(R_i).$$

Here $X_D(\omega)$ counts the number of times a line ω meets the set D and the integration is performed over all lines. Going to the limit we find for some convex body E a measure proportional to its surface area

$$\int X_E(\omega)\, d\lambda_1^3(\omega) = \mu_2^3(E).$$

Using planes ($k = 2$) we can now generalize the mean width. For a straight line $c \in \mathbb{R}^3$ we find $\lambda_2^3(c) = \mu_1^3(c) = l(c)$, *i.e.*, the measure of all planes that meet the straight line is proportional—as before we set the constant of proportionality to unity—to the length of the line. The argument mimics what we said above: a plane either meets the line once or not at all. For a given point on the line there is once again a whole set of planes going through that point. Considering the normals to such planes we see that this set of planes is proportional in measure to the unit sphere without being more precise about the actual constant of proportionality. Once again this can be generalized with a limiting process giving us the measure of all planes hitting an arbitrary curve in space as proportional to its length

$$\int X_F(\omega)\, d\lambda_2^3(\omega) = \mu_1^3(F).$$

Here the integration is performed over all planes $\omega \in \mathbb{R}^3$, and X_F counts the number of times a given plane touches the curve F.

It is easy to see that this way of measuring recovers the perimeter of a parallelipiped as we had defined it before

$$\lambda_2^3(P) = \mu_1^3(P).$$

To see this consider the integration over all planes but taken in groups. With the parallelipiped having one corner at the origin—and being axis aligned—first consider all planes whose normal (n_x, n_y, n_z) has either all non-negative or non-positive entries (*i.e.*, the normal, or its negative, points into the first octant). Any such plane, if it meets the parallelipiped, meets it in a point along either x_1, x_2 or x_3 giving us the desired $\mu_1^3(P) = x_1 + x_2 + x_3$ as the measure of all such planes. The same argument holds for the remaining seven octants giving us the desired result up to a constant. We can now see that $\mu_1^3(E)$ for some convex body E can be written as

$$\int X_E(\omega)\, d\lambda_2^3(\omega) = \mu_1^3(E),$$

i.e., the measure of all planes which meet E. With this we have generalized the notion of perimeter to more general sets.

All this can be summarized as follows. Let μ be a measure which is Euclidean motion invariant. Then it can be written, up to normalization, as a linear combination of the measures $\mu_k^n(C)$ of all affine subspaces of dimension $n-k$ meeting $C \subset \mathbb{R}^n$ for $k = 1, \ldots, n$. These measures are called the *intrinsic volumes*.

Are these all such measures? It turns out there is one measure missing, which corresponds to the elementary symmetric function of order zero

$$\mu_0(x_1, \ldots, x_n) = \begin{cases} 1 & n > 0 \\ 0 & n = 0 \end{cases}$$

This very special measure is the Euler characteristic of a convex body. It takes the value 1 on all non-empty convex bodies. The main trick is to prove that μ_0 is indeed well defined. This can be done by inducation. In dimension $n = 1$ we consider closed intervals $[a, b]$, $a < b$. Instead of working with the set directly we consider a functional on the characteristic function $f_{[a,b]}$ of the set which does the trick

$$\chi_1(f) = \int_{\mathbb{R}} f(\omega) - f(\omega+)\, d\omega.$$

Here $f(\omega+)$ denotes the right limiting value of f at ω: $\lim_{\epsilon\to 0} f(\omega + \epsilon)$, $\epsilon > 0$. For the set $[a, b]$, $f(\omega) - f(\omega+)$ is zero for all $\omega \in \mathbb{R}$ except b since $f(b) = 1$ and $f(b+) = 0$. For higher dimensions we proceed by induction. In $\mathbb{R}^n$ take a straight line L and consider the affine subspaces A_ω of dimension $n - 1$ which are orthogonal to L and parameterized by ω along L. Letting f be the characteristic function of a convex body in $\mathbb{R}^n$ we get

$$\chi_n(f) = \int_{\mathbb{R}} \chi_{n-1}(f_\omega) - \chi_{n-1}(f_{\omega+})\, d\omega.$$

Here f_ω is the restriction of f to the affine space A_ω or alternatively the characteristic function of the intersection of A_ω and the convex body of interest. With this we define $\mu_0^n(G) = \chi_n(f)$ for any finite union of convex bodies G and f the characteristic function of the set $G \in \mathbb{R}^n$.

That this definition of μ_0^n amounts to the Euler characteristic is not immediately clear, but it is easy to show, if we convince ourselves that for any non empty convex body $C \in \mathbb{R}^n$

$$\mu_0^n(\text{Int}(C)) = (-1)^n.$$

For $n = 1$, *i.e.*, the case of open intervals on the real line, this statement is obviously correct. We can now apply the recursive definition to the characteristic function of the interior of C and get

$$\mu_0^n(\text{Int}(C)) = \int_\omega \chi_{n-1}(f_\omega) - \chi_{n-1}(f_{\omega+})\, d\omega.$$

By induction the right hand side is zero except for the first ω at which $A_\omega \cap C$ is non-empty. There $\chi_{n-1}(f_{\omega+}) = (-1)^{n-1}$, thus proving our assertion for all n.

The Euler-Poincaré formula for a polyhedron

$$|F| - |E| - |V| = 2(1 - g)$$

which relates the number of faces, edges, and vertices to the genus now follows easily. Given a polyhedron simply write it as the non-overlapping union of the interiors of all its cells from dimension n down to dimension 0, where the interior of a vertex (0-cell) is the vertex itself. Then

$$\mu_0^n(P) = \sum_{c\in P} \mu_0^n(\text{Int}(c)) = c_0 - c_1 + c_2 - \dots$$

where c_i equals the number of cells of dimension i. For the case of a polyhedron in $\mathbb{R}^3$ this is exactly the Euler-Poincaré formula.

4 The Intrinsic Volumes and Hadwiger's Theorem

The above machinery can now be used to define the intrinsic volumes as functions of the Euler characteristic alone for all finite unions of convex bodies G

$$\mu_k^n(G) = \int \mu_0^n(G \cap \omega)\, d\lambda_{n-k}^n(\omega).$$

Here $\mu_0^n(G \cap \omega)$ plays the role of $X_G(\omega)$ we used earlier to count the number of times ω hits G.

There is one final ingredient missing, continuity in the limit. Suppose C_n is a sequence of convex bodies which converges to C in the limit as $n \to \infty$. (In fact we already used this when we appealed to the Riemann sum argument of recovering the usual notion of area of a region.) Hadwiger's theorem says that if a Euclidean motion invariant measure μ of convex bodies in $\mathbb{R}^n$ is continuous in the sense that

$$\lim_{C_n\to C} \mu(C_n) = \mu(C)$$

then μ must be a linear combination of the intrinsic volumes μ_k^n, $k = 0, \dots, n$. In other words, the intrinsic volumes, under the additional assumption of continuity, are the only linearly independent, Euclidean motion invariant, additive measures on finite unions and intersections of convex bodies in $\mathbb{R}^n$.

What does all of this have to do with the applications we have in mind? A consequence of Hadwiger's theorem assures us that if we want to take measurements of piecewise linear geometry (surface or volume meshes, for example) such measurements should be functions of the intrinsic volumes. This assumes of course that we are looking for additive measurements which are Euclidean motion invariant and continuous in the limit. For a triangle for example this would be area, edge length, and Euler characteristic. Similarly for a tetrahedron with its volume, surface area, mean width, and Euler characteristic. As the name suggests all of these measurements are intrinsic. For a 2-manifold mesh which is the boundary of a solid one of these measurements is an *extrinsic* quantity corresponding to the dihedral angle between triangles meeting at an edge (see below).

5 Steiner's Formula

We return now to questions of discrete differential geometry by showing that the intrinsic volumes are intricately linked to curvature integrals and represent their generalization to the non-smooth setting. This connection is established by Steiner's formula.

Consider a non-empty convex body $K \in \mathbb{R}^n$ together with its parallel bodies

$$K_\epsilon = \{x \in \mathbb{R}^n : d(x, K) \le \epsilon\}$$

where $d(x, K)$ denotes the Euclidean distance from x to the set K. In effect K_ϵ is the body K thickened by ϵ. Steiner's formula gives the volume of K_ϵ as a polynomial in ϵ

$$V(K_\epsilon) = \sum_{j=0}^{n} V(\mathbb{B}_{n-j}) V_j(K) \epsilon^{n-j}.$$

Here the $V_j(K)$ are the measures μ_k^n we have seen earlier. For this formula to be correct the $V_j(K)$ are normalized so that they compute the j-dimensional volume when restricted to a j-dimensional subspace of $\mathbb{R}^n$. $V(\mathbb{B}_{n-j}) = \pi^{n/2}/\Gamma(1 + 1/2n)$ denotes the $(n - j)$-volume of the $(n - j)$-unit ball. In

particular we have $V(\mathbb{B}_0) = 1$, $V(\mathbb{B}_1) = 2$, $V(\mathbb{B}_2) = \pi$, and $V(\mathbb{B}_3) = 4\pi/3$.

In the case of a polyhedron we can verify Steiner's formula "by hand." Consider a tetrahedron in $T \in \mathbb{R}^3$ and the volume of its parallel bodies T_ϵ. For $\epsilon = 0$ we have the volume of T itself ($V_3(T)$). The first order term in ϵ, $2V_2(T)$, is controlled by area measures: above each triangle a displacement along the normal creates additional volume proportional to ϵ and the area of the triangle. The second order term in ϵ, $\pi V_1(T)$, corresponds to edge lengths. Above each edge the parallel bodies form a wedge with opening angle θ which is the exterior angle of the faces meeting at that edge and radius ϵ (this is the extrinsic measurement alluded to above). The volume of each such wedge is proportional to edge length, exterior angle, and ϵ^2. Finally the third order term in ϵ, $4\pi/3V_0(T)$, corresponds to the volume of the parallel bodies formed over vertices. Each vertex gives rise to additional volume spanned by the vertex and a spherical cap above it. The spherical cap corresponds to a spherical triangle formed by the three incident triangle normals. The volume of such a spherical wedge is proportional to its solid angle and ϵ^3.

If we have a convex body with a boundary which is C^2 we can give a different representation of Steiner's formula. Consider such a convex $M \in \mathbb{R}^n$ and define the offset function

$$g(p) = p + t\vec{n}(p)$$

for $0 \leq t \leq \epsilon$, $p \in \partial M$ and $\vec{n}(p)$ the outward normal to M in p. We can now directly compute the volume of M_ϵ as the sum of $V_n(M)$ and the volume between the surfaces ∂M and ∂M_ϵ. The latter can be written as an integral of the determinant of the Jacobian of g

$$\int_{\partial M} \left(\int_0^\epsilon \left| \frac{\partial g(p)}{\partial p} \right| dt \right) dp.$$

Since we have a choice of coordinate frame in which to do this integration we may assume wlog that we use principal curvature coordinates on ∂M, *i.e.*, a set of orthogonal directions in which the curvature tensor diagonalises. In that case

$$\begin{aligned} \left| \frac{\partial g(p)}{\partial p} \right| &= |\mathbb{I} + t\mathbb{K}(p)| \\ &= \prod_{i=1}^{n-1} (1 + \kappa_i(p)t) \\ &= \sum_{i=0}^{n-1} \mu_i^{n-1}(\kappa_1(p), \ldots, \kappa_{n-1}(p)) t^i. \end{aligned}$$

In other words, the determinant of the Jacobian is a polynomial in t whose coefficients are the elementary symmetric functions in the principal curvatures. With this subsitution we can trivially integrate over the variable t and get

$$V(M_\epsilon) = V_n(M) + \sum_{i=0}^{n-1} \frac{\epsilon^{i+1}}{i+1} \int_{\partial M} \mu_i^{n-1}(\kappa_1(p), \ldots, \kappa_{n-1}(p))\, dp.$$

Comparing the two versions of Steiner's formula we see that the intrinsic volumes generalize curvature integrals. For example, for $n = 3$ and some arbitrary convex body K we get

$$V(K_\epsilon) = 1V_3(K) + 2V_2(K)\epsilon + \pi V_1(K)\epsilon^2 + \frac{4\pi}{3} V_0(K)\epsilon^3$$

while for a convex body M with C^2 smooth boundary the formula reads as

$$\begin{aligned} V(M_\epsilon) = V_3(M) + & \underbrace{\left(\int_{\partial M} \underbrace{\mu_0^2(\kappa_1(p), \kappa_2(p))}_{=1}\, dp \right)}_{=A} \epsilon + \\ & \left(\int_{\partial M} \underbrace{\mu_1^2(\kappa_1(p), \kappa_2(p))}_{=2H}\, dp \right) \frac{\epsilon^2}{2} + \\ & \underbrace{\left(\int_{\partial M} \underbrace{\mu_2^2(\kappa_1(p), \kappa_2(p))}_{=K}\, dp \right)}_{=4\pi} \frac{\epsilon^2}{3}. \end{aligned}$$

6 What All This Machinery Tells Us

We began this section by considering the question of what additive, continuous, rigid motion invariant measurements there are for convex bodies in $\mathbb{R}^n$ and learned that the $n + 1$ intrinsic volumes are the only ones and any such measure must be a linear combination of these. We have also seen that the intrinsic volumes in a natural way extend the idea of curvature integrals over the boundary of a smooth body to general convex bodies without regard to a differentiable structure. These considerations become one possible basis on which to claim that integrals of Gaussian curvature on a triangle mesh become sums over excess angle at vertices and that integrals of mean curvature can be identified with sums over edges of dihedral angle weighted by edge length. These quantities are always *integrals*. Consequently *they do not make sense as pointwise quantities*. In the case of smooth geometry we can define quantities such as mean and Gaussian curvature as pointwise quantities. On a simplicial mesh they are only defined as integral quantities.

All this machinery was developed for convex bodies. If a given mesh is not convex the additivity property allows us to compute the quantities no less by writing the mesh as a finite union and intersection of convex bodies and then tracking the corresponding sums and differences of measures. For example, $V(K_\epsilon)$ is well defined for an individual triangle K and we know how to identify the coefficients involving intrinsic volumes with the integrals of elementary polynomials in the principal curvatures. Glueing two triangles together we can perform a similar identification carefully teasing apart the intrinsic volumes of the union of the two triangles. In this way the convexity requirement is relaxed so long as the shape of interest can be decomposed into a finite union of convex bodies.

This machinery was used by Cohen-Steiner and Morvan to give formulas for integrals of a discrete curvature tensor. We give these here together with some fairly straightforward intuition regarding the underlying geometry.

Let P be a polyhedron with vertex set V and edge set E and B a ball in $\mathbb{R}^3$ then we can define integrated Gaussian and mean curvature measures as

$$\phi_P^G(B) = \sum_{v \in V \cap B} K_v \quad \text{and} \quad \phi_P^H(B) = \sum_{e \in E} l(e \cap B)\theta_e,$$

where $K_v = 2\pi - \sum_j \alpha_j$ is the excess angle sum at vertex v defined through all the incident triangle angles at v, while $l(.)$ denotes the length and θ_e is the signed dihedral angle at e made between the incident triangle normals. Its sign is positive for convex edges and negative for concave edges (note that this requires an orientation on the polyhedron). In essence this is simply a restatement of the Steiner polynomial coefficients restricted to the intersection of the ball B and the polyhedron P. To talk about the second fundamental form II_p at some point p in the surface, it is convenient to first extend it to all of $\mathbb{R}^3$. This is done by setting it to zero if one of its arguments is parallel to the normal p. With this one may define

$$\bar{II}_P(B) = \sum_{e \in E} l(e \cap B)\theta_e e_n \otimes e_n, \quad e_n = e/\|e\|.$$

The dyad $e_n \otimes e_n(u, v) = \langle u, e_n \rangle \langle v, e_n \rangle$ projects given vectors u and v along the normalized edge. What is the geometric interpretation of the summands? Consider a single edge and the associated dyad. The curvature along this edge is zero while it is θ orthogonal to the edge. A vector aligned with the edge is mapped to θ_e while one orthogonal to the edge is mapped to zero. These are the principal curvatures *except they are reversed*. Hence $\bar{II}_P(B)$ is an integral measure of the curvature tensor *with the principal curvature values exchanged*. For example we can assign each vertex a three by three tensor by summing the edge terms for each incident edge. As a tangent plane at the vertex, which we need to project the three by three tensor to the expected two by two tensor in the tangent plane, we may take a vector parallel to the area gradient at the vertex. Alternatively we could defined $\bar{II}_P(B)$ for balls containing a single triangle and its three edges each. In that case the natural choice for the tangent plane is the support plane of the triangle. In practice one often finds that noise in the mesh vertex positions makes these discrete computations noisy. It is then a simple matter of enlarging B to stabilize the computations.

Cohen-Steiner and Morvan show that this definition can be rigorously derived from considering the coefficients of the Steiner polynomial in particular in the presence of non-convexities (which requires some fancy footwork...). They also show that if the polyhedron is a sufficiently fine sample of a smooth surface the discrete curvature tensor integrals have linear precision with regards to continuous curvature tensor integrals. They also provide a formula for a discrete curvature tensor which does not have the principal curvatures swapped.

7 Further Reading

The material in this section only gives the rough outlines of what is a very fundamental theory in probability and geometric measure theory. In particular there are many other consequences which follow from relationships between intrinsic volumes which we have not touched upon. A rigorous derivation of the results of Hadwiger, but much shorter than the original can be found in [Klain 1995]. A complete and rigorous account of the derivation of intrinsic volumes from first principles in geometric probability can be found in the short book by Klain and Rota [Klain and Rota 1997], while the details of the discrete curvature tensor integrals can be found in [Cohen-Steiner and Morvan 2003]. Approximation results which discuss the accuracy of these measure vis-a-vis an underlying smooth surface are treated by Cohen-Steiner and Morvan in a series of tech reports available at http://www-sop.inria.fr/geometrica/publications/.

Acknowledgments This work was supported in part by NSF (DMS-0220905, DMS-0138458, ACI-0219979), DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Alias, and Pixar.

References

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted Delaunay Triangulations and Normal Cycle. In *Proceedings of the 19th Annual Symposium on Computational Geometry*, 312–321.

KLAIN, D. A., AND ROTA, G.-C. 1997. *Introduction to Geometric Probability*. Cambridge University Press.

KLAIN, D. A. 1995. A Short Proof of Hadwiger's Characterization Theorem. *Mathematika 42*, 84, 329–339.

Chapter 3: Curvature Measures for Discrete Surfaces

John M. Sullivan
sullivan@math.tu-berlin.de
Institut für Mathematik, TU Berlin MA 3–2
Str. des 17. Juni 136, 10623 Berlin

The curvatures of a smooth curve or surface are local measures of its shape. Here we consider analogous measures for discrete curves and surfaces, meaning polygonal curves and triangulated polyhedral surfaces. We find that the most useful analogs are those which preserve integral relations for curvature, like the Gauß–Bonnet theorem. For simplicity, we usually restrict our attention to curves and surfaces in euclidean space $\mathbb{R}^3$, although many of the results would easily generalize to other ambient manifolds of arbitrary dimension.

These notes are based on work by many people, but unfortunately do not include proper citations to the literature.

1 Smooth Curves and Surfaces

Before discussing discrete analogs, we briefly review the usual theory of curvatures for smooth curves and surfaces in space.

1.1 Smooth curves

The curvatures of a smooth curve γ are the local properties of its shape invariant under Euclidean motions. The only first-order information is the tangent line; since all lines in space are equivalent, there are no first-order invariants. Second-order information is given by the osculating circle, and the invariant is its curvature $\kappa = 1/r$.

For a plane curve given as a graph $y = f(x)$ let us contrast the notions of curvature and second derivative. At a point p on the curve, we can find either one by translating p to the origin, transforming so the curve is horizontal there, and then comparing to a standard set of reference curves. The difference is that for curvature, the transformation is a Euclidean rotation, while for second derivative, it is a shear $(x, y) \mapsto (x, y - ax)$. A parabola has constant second derivative f'' because it looks the same at any two points after a shear. A circle, on the other hand, has constant curvature because it looks the same at any two points after a rotation.

A plane curve is completely determined (up to rigid motion) by its (signed) curvature $\kappa(s)$ as a function of arclength s. For a space curve, however, we need to look at the third-order invariants, which are the torsion τ and the derivative κ' (which of course gives no new information). These are now a complete set of invariants: a space curve is determined by $\kappa(s)$ and $\tau(s)$. (Generally, for higher codimension, higher-order invariants are needed. For curves in $\mathbb{R}^n$, we need $n - 1$ curvatures, of order up to n, to characterize the shape.)

A smooth space curve γ is often described by its orthonormal *Frenet frame* (T, N, B). With respect to an arclength parameter s, the defining equations are $T := \gamma'$ and

$$\begin{pmatrix} T \\ N \\ B \end{pmatrix}' = \begin{pmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{pmatrix} \begin{pmatrix} T \\ N \\ B \end{pmatrix}.$$

For a curve γ lying on a surface M, it is often more useful to consider the *Darboux frame* (T, η, ν), adapted to this situation. This orthonormal frame includes the tangent vector T to γ and the normal vector ν to M. Its third element is thus $\eta := \nu \times T$, called the *cornormal*. The curvature vector of γ decomposes into parts tangent and normal to M as $T' = \kappa N = \kappa_g \eta + \kappa_n \nu$. Here in fact, κ_n measures the normal curvature of M in the direction T, and is independent of γ.

1.2 Smooth surfaces

Given a (two-dimensional, oriented) surface M (immersed) in $\mathbb{R}^3$, we understand its local shape by looking at the Gauß map $\nu : M \to \mathbb{S}^2$ given by the unit normal vector $\nu = \nu_p$ at each point $p \in M$. We can view its derivative at p as a linear endomorphism $-S_p : T_pM \to T_pM$, since T_pM and $T_{\nu_p}\mathbb{S}^2$ are naturally identified, being parallel planes in $\mathbb{R}^3$. The map S_p is called the shape operator (or Weingarten map).

The shape operator is the second-order invariant (or curvature) which completely determines the original surface M. However, it is usually more convenient to work with scalar quantities. The eigenvalues κ_1 and κ_2 of S_p are called principal curvatures, and (since they cannot be globally distinguished) it is their symmetric functions which have geometric meaning.

We define the *Gauß curvature* $K := \kappa_1\kappa_2$ as the determinant of S_p and the *mean curvature* $H := \kappa_1 + \kappa_2$ as its trace. Note that the sign of H depends on the choice of unit normal ν, and so often it is more natural to work with the *vector mean curvature* (or mean curvature vector) $\mathbf{H} := H\nu$. Note furthermore that some authors use the opposite sign on S_p and thus H, and many use $H = (\kappa_1 + \kappa_2)/2$, justifying the name *mean* curvature. Our conventions mean that the mean curvature vector for a convex surface points inwards (like the curvature vector for a circle). For a unit sphere oriented with inward normal, the Gauß map ν is the antipodal map, $S_p = I$, and $H = 2$.

The Gauß curvature is an intrinsic notion, depending only on the pullback metric on the surface M, and not on the immersion into space. That is, K is unchanged by bending the surface without stretching it. For instance, a developable surface like a cylinder or cone has $K = 0$ because it is obtained by bending a flat plane. One intrinsic definition of $K(p)$ is obtained by considering the circumferences C_ε of (intrinsic) ε-balls around p. We get

$$\frac{C_\varepsilon}{2\pi\varepsilon} = 1 - \frac{\varepsilon^2}{6}K + \mathcal{O}(\varepsilon^3).$$

Mean curvature is certainly not intrinsic, but it has a nice variational interpretation. Consider a variation vectorfield V on M, compactly supported away from any boundary. Then $H = -\delta \text{ Area} / \delta \text{ Vol}$ in the sense that

$$\delta_V \text{ Vol} = \int V \cdot \nu \, dA, \quad \delta_V \text{ Area} = -\int V \cdot H\nu \, dA.$$

With respect to the L^2 inner product $\langle U, V \rangle := \int U_p \cdot V_p \, dA$ on vectorfields, the vector mean curvature is the negative gradient of the area functional, often called the first variation of area: $\mathbf{H} = -\nabla \text{ Area}$. (Similarly, the negative gradient of length for a curve is κN.)

Just as κ is the geometric version of second derivative for curves, mean curvature is the geometric version of the Laplacian Δ. Indeed, if a surface M is written locally as the graph of a height function f over its tangent plane T_pM then $H(p) = \Delta f$. Alternatively, we can write $\mathbf{H} = \nabla_M \cdot \nu = \Delta_M \mathbf{x}$, where $\mathbf{x}$ is the position vector in $\mathbb{R}^3$ and Δ_M is Beltrami's surface Laplacian.

If we flow a curve or surface to reduce its length or area, by following these gradients κN and $H\nu$, the resulting parabolic heat flow is slightly nonlinear in a natural geometric way. This so-called *mean-curvature flow* has been extensively studied as a geometric smoothing flow.

1.3 Integral curvature relations for curves

The *total curvature* of a curve is $\int \kappa \, ds$. For closed curves, the total curvature is at least 2π (Fenchel) and for knotted space curves the total curvature is at least 4π (Fáry/Milnor). For plane curves, we can consider instead the signed curvature, and find that $\int \kappa \, ds$ is always an integral multiple of 2π.

Suppose we define (following Milnor) the total curvature of a polygonal curve simply to be the sum of the turning angles at the vertices. Then all the theorems for smooth curves mentioned in the previous paragraph remain true for polygonal curves. Our goal, when defining curvatures for polyhedral surfaces, will be to again ensure that integral relations for these curvatures remain exactly true.

1.4 Integral curvature relations for surfaces

For surfaces, the integral curvature relations we want to consider relate area integrals over a region $D \subset M$ to arclength integrals over the boundary $\gamma = \partial D$. The Gauß–Bonnet theorem says, when D is a disk,

$$2\pi - \iint_D K \, dA = \oint_\gamma \kappa_g \, ds = \oint_\gamma T' \cdot \eta \, ds = -\oint \eta' \cdot \, d\mathbf{x},$$

where $d\mathbf{x} = T \, ds$ is the vector line element along γ. This implies that the total Gauß curvature of D depends only on a collar neighborhood of γ: if we make any modification to D supported away from the boundary, the total curvature is unchanged (as long as D remains topologically a disk). We will extend the notion of Gauß curvature from smooth surfaces to more general surfaces (in particular polyhedral surfaces) by requiring this property to remain true.

The other relations are all proved by Stokes' Theorem, and thus only depend on γ being the boundary of D in a homological sense; for these D is not required to be a disk. First consider the vector area

$$\mathbf{A}_\gamma := \tfrac{1}{2} \oint_\gamma \mathbf{x} \times d\mathbf{x} = \iint_D \nu \, dA.$$

The right-hand side represents the total vector area of any surface spanning γ, and the relation shows this to depend only on γ (and this time not even on a collar neighborhood). The integrand on the left-hand side depends on a choice of origin for the coordinates, but because we integrate over a closed loop, the integral is independent of this choice. Both sides of this vector area formula can be interpreted directly for a polyhedral surface, and the equation remains true in that case.

A simple integral for curve γ from p to q says that

$$T(q) - T(p) = \int_p^q T'(s) \, ds = \int \kappa N \, ds.$$

This can be viewed as a balance between tension forces trying to shrink the curve, and sideways forces holding it in place. It is the relation used in proving that κ is the first variation of length.

The analog for a surface patch D is the mean curvature force balance equation

$$\oint_\gamma \eta \, ds = -\oint_\gamma \nu \times d\mathbf{x} = \iint_D H\nu \, dA = \iint_D \mathbf{H} \, dA.$$

Again this represents a balance between surface tension forces acting in the conormal direction along the boundary of D and what can be considered as pressure forces (espcially in the case of constant H) acting normally across D. We will use this equation to develop the analog of mean curvature for discrete surfaces.

Two other similar relations that we will not need later are the torque balance

$$\oint_\gamma \mathbf{x} \times \eta \, ds = \oint_\gamma \mathbf{x} \times (\nu \times d\mathbf{x}) = \iint_D H(\mathbf{x} \times \nu) \, dA$$

and the area relation

$$\oint_\gamma \mathbf{x} \cdot \eta \, ds = \oint_\gamma \mathbf{x} \cdot (\nu \times d\mathbf{x}) = \iint_D (\mathbf{H} \cdot \mathbf{x} - 2) \, dA.$$

2 Discrete Surfaces

For us, a discrete or polyhedral surface $M \subset \mathbb{R}^3$ will mean a triangulated surface with a PL map into space. In more detail, we start with an abstract combinatorial triangulation—a simplicial complex—representing a 2-manifold with boundary. We then pick positions $p \in \mathbb{R}^3$ for every vertex, which uniquely determine a linear map on each triangle; these fit together to form the PL map.

2.1 Gauß curvature

It is well known how the notion of Gauß curvature extends to such discrete surfaces M. Any two adjacent triangles (or, more generally, any simply connected region in M not including any vertices) can be flattened—developed isometrically into the plane. Thus the Gauß curvature is supported on the vertices $p \in M$. In fact, to keep the Gauß–Bonnet theorem true, we must take

$$\iint_D K \, dA := \sum_{p \in D} K_p; \qquad K_p := 2\pi - \sum_i \theta_i.$$

Here, the angles θ_i are the interior angles at p of the triangles meeting there, and K_p is often known as the angle defect

at p. If D is any neighborhood of p contained in $\mathrm{Star}(p)$, then $\oint_{\partial D} \eta\, ds = \sum \theta_i$; when the triangles are acute, this is most easily seen by letting ∂D be the path connecting their circumcenters and crossing each edge perpendicularly.

(Similar arguments lead to a notion of Gauß curvature—defined as a measure—for any rectifiable surface. For our polyhedral surface, this measure consists of point masses at vertices. Surfaces can also be built from intrinsically flat pieces joined along curved edges. Their Gauß curvature is spread out with a linear density along these edges. This technique is often used in designing clothes, where corners would be undesirable.)

Note that K_p is clearly an intrinsic notion, as it should be, depending only on the angles of each triangle and not on the precise embedding into $\mathbb{R}^3$. Sometimes it is useful to have a notion of combinatorial curvature, independent of all geometric information. Given just a combinatorial triangulation, we can pretend that each triangle is equilateral with angles $\theta = 60^\circ$, whether or not that geometry could be embedded in space. The resulting *combinatorial curvature* is $K_p = \frac{\pi}{3}(6 - \deg p)$. In this context, the global form $\sum K_p = 2\pi\chi(M)$ of Gauß–Bonnet amounts to nothing more than the definition of the Euler characteristic χ.

2.2 Vector area

The vector area formula

$$\mathbf{A}_\gamma := \tfrac{1}{2} \oint_\gamma \mathbf{x} \times d\mathbf{x} = \iint_D \nu\, dA$$

needs no special interpretation for discrete surfaces: both sides of the equation make sense directly, since the surface normal ν is well-defined almost everywhere. However, it is worth interpreting this formula for the case when D is the star of a vertex p. More generally, suppose γ is any closed curve (smooth or polygonal), and D is the cone from p to γ (the union of all line segments pq for $q \in \gamma$). Fixing γ and letting p vary, we find that the volume enclosed by this cone is a linear function of p, and $\mathbf{A}_p := \nabla_p \operatorname{Vol} D = \mathbf{A}/3 = \frac{1}{6} \oint_\gamma \mathbf{x} \times d\mathbf{x}$. We also note that any such cone D is intrinsically flat except at the cone point p, and that $2\pi - K_p$ is the cone angle at p.

2.3 Mean curvature

The mean curvature of a discrete surface M is supported along the edges. If e is an edge, and $e \subset D \subset \mathrm{Star}(e) = T_1 \cup T_2$, then

$$\mathbf{H}_e := \iint_D \mathbf{H}\, dA = \oint_{\partial D} \eta\, ds = e \times \nu_1 - e \times \nu_2 = J_1 e - J_2 e.$$

Here ν_i is the normal vector to the triangle T_i, and J_i is rotation by 90° in the plane of that triangle. Note that $|\mathbf{H}_e| = 2|e| \sin \frac{\theta_e}{2}$ where θ_e is the exterior dihedral angle along the edge, defined by $\cos\theta_e = \nu_1 \cdot \nu_2$.

No nonplanar discrete surface has $\mathbf{H}_e = 0$ along every edge. But this discrete mean curvature can cancel out around the vertices. We set

$$2\mathbf{H}_p := \sum_{e \ni p} \mathbf{H}_e = \iint_{\mathrm{Star}(p)} \mathbf{H}\, dA = \oint_{\mathrm{Link}(p)} \eta\, ds.$$

The area of the discrete surface is a function of the vertex positions; if we vary only one vertex p, we find that $\nabla_p \operatorname{Area}(M) = -\mathbf{H}_p$.

Suppose that vertices adjacent to p are $p_1, \ldots, p_n$. Then we have

$$\begin{aligned} 3\mathbf{A}_p = 3\nabla_p \operatorname{Vol} &= \iint_{\mathrm{Star}\, p} \nu\, dA \\ &= \tfrac{1}{2} \oint_{\mathrm{Link}\, p} \mathbf{x} \times d\mathbf{x} = \tfrac{1}{2} \sum_i p_i \times p_{i+1} \end{aligned}$$

and similarly

$$\begin{aligned} 2\mathbf{H}_p = \sum \mathbf{H}_{pp_i} = -2\nabla_p \operatorname{Area} &= \sum J_i(p_{i+1} - p_i) \\ &= \sum_i (\cot\alpha_i + \cot\beta_i)(p - p_i), \end{aligned}$$

where α_i and β_i are the angles opposite edge pp_i in the two incident triangles.

Note that if we change the combinatorics of a discrete surface M by introducing a new vertex p along an existing edge e, and subdividing the two incident triangles, then $\mathbf{H}_p$ in the new surface equals the original $\mathbf{H}_e$, independent of where along e we place p. This allows a variational interpretation of $\mathbf{H}_e$.

2.4 Minkowski mixed volumes

A somewhat different interpretation of mean curvature for convex polyhedra is suggested by Minkowski's theory of mixed volumes (which actually dates in this form well earlier). If X is a smooth convex body in $\mathbb{R}^3$ and $B_t(X)$ denotes its t-neighborhood, then

$$\operatorname{Vol}(B_t(X)) = \operatorname{Vol} X + t \operatorname{Area} X + \frac{t^2}{2} \int_X H\, dA + \frac{t^3}{3} \int_X K\, dA.$$

Here, the last integral is always 4π.

When X is instead a convex polyhedron, the only term that needs a new interpretation is $\int_X H\, dA$. The correct replacement for this term is then $\sum_e |e|\, \theta_e$. This suggests $H_e := |e|\, \theta_e$ as a notion of total mean curvature for the edge e.

We note the difference between this formula and our earlier $|\mathbf{H}_e| = 2|e| \sin\theta_e/2$. Either one can be derived by replacing the edge e with a sector of a cylinder of length $|e|$ and arbitrary (small) radius r. We find then

$$\iint \mathbf{H}\, dA = \mathbf{H}_e, \qquad \iint H\, dA = H_e.$$

The difference is explained by the fact that one formula integrates the scalar mean curvature while the other integrates the vector mean curvature.

2.5 CMC surfaces and Willmore surfaces

A smooth surface which minimizes area under a volume constraint has constant mean curvature; the constant H can be understood as the Lagrange multiplier for the constrained minimization problem. A discrete surface which minimizes area among surfaces of fixed combinatorial type and fixed volume will have constant discrete mean curvature H in the sense that at every vertex, $\mathbf{H}_p = H\mathbf{A}_p$, or equivalently $\nabla_p \operatorname{Area} = -H\nabla_p \operatorname{Vol}$. In general, of course, the vectors $\mathbf{H}_p$ and $\mathbf{A}_p$ are not even parallel: they give two competing notions of a normal vector at p.

Still,

$$h_p := \frac{|\nabla_p \operatorname{Area}|}{|\nabla_p \operatorname{Vol}|} = \frac{|\mathbf{H}_p|}{|\mathbf{A}_p|} = \frac{|\iint_{\operatorname{Star} p} \mathbf{H}\, dA|}{|\iint_{\operatorname{Star} p} \nu\, dA|}$$

gives a better notion of mean curvature near p than, say, the smaller quantity $3|\mathbf{H}_p|/\operatorname{Area}(\operatorname{Star}(p)) = |\iint \mathbf{H}\, dA|/\iint 1\, dA$.

For this reason, a good discretization of the Willmore elastic energy $\iint H^2\, dA$ is given by $\sum_p h_p^2 \frac{1}{3} \operatorname{Area}(\operatorname{Star}(p))$.

2.6 Relation to discrete harmonic maps

Discrete minimal surfaces minimize area, but also have other properties similar to those of smooth minimal surfaces. For instance, in a conformal parameterization, their coordinate functions are harmonic. We don't know when in general a discrete map should be considered conformal, but the identity map is certainly conformal. We have that M is discrete minimal if and only if $\mathrm{Id} : M \to \mathbb{R}^3$ is discrete harmonic. Here a PL map $f : M \to N$ is called discrete harmonic if it is a critical point for the Dirichlet energy $E(f) := \sum_T |\nabla f_T|^2 \operatorname{Area}_M(T)$. We find that $E(f) - \operatorname{Area} N$ is a measure of nonconformality. For the identity map, $E(\mathrm{Id}_M) = \operatorname{Area}(M)$ and $\nabla_p E(\mathrm{Id}_M) = \nabla_p \operatorname{Area}(M)$ confirming that M is minimal if and only if Id_M is harmonic.

A Discrete Model of Thin Shells

Eitan Grinspun
Columbia University

Figure 1: A measure of discrete strain is used to fracture a thin shell in this simulation of a shattering lightbulb.

Abstract

We describe a discrete model for the dynamics of thin flexible structures, such as hats, leaves, and aluminum cans, which are characterized by a curved undeformed configuration. Previously such *thin shell* models required complex continuum mechanics formulations and correspondingly complex algorithms. We show that a simple shell model can be derived geometrically for triangle meshes and implemented quickly by modifying a standard cloth simulator. Our technique convincingly simulates a variety of curved objects with materials ranging from paper to metal, as we demonstrate with several examples including a comparison of a real and simulated falling hat.

This chapter is based on the publication by Eitan Grinspun, Anil Hirani, Mathieu Desbrun, and Peter Schröder which appeared in the Proceedings of the Symposium for Computer Animation 2003*, and on subsequent collaborations with the group of Denis Zorin at New York University and Zoë Wood at Cal Poly San Luis Obispo.*

1 Introduction

Thin shells are thin flexible structures with a high ratio of width to thickness (> 100) [Ciarlet 2000]. While their well-known counterparts, thin *plates*, relax to a *flat* shape when unstressed, thin *shells* are characterized by a *curved undeformed configuration*. Cloth, recently studied in the computer animation literature, may be modeled as a thin plate, since garments are typically constructed from flat textiles. In stark contrast, thin-walled objects which are naturally curved (*e.g.*, leaves, fingernails), or put into that shape through plastic deformation (*e.g.*, hats, cans, carton boxes, pans, car bodies, detergent bottles) are thin shells and cannot be modeled using plate formulations.

Thin shells are remarkably difficult to simulate. Because of their degeneracy in one dimension, shells do not admit to straightforward tessellation and treatment as three-dimensional solids; indeed, the numerics of such approaches become catastrophically ill-conditioned, foiling numerical convergence and/or accuracy. Robust finite element methods for thin shell equations continue to be an active and challenging research area.

In this chapter we develop a simple model for thin shells with applications to computer animation. Our *discrete model* of shells captures the same characteristic behaviors as more complex models, with a surprisingly simple implementation. We demonstrate the realism of our approach through various examples including comparisons with real world footage (see Figure 4).

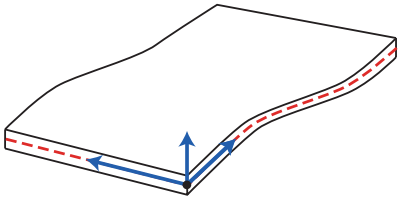


Figure 2: The local coordinate frame in a small neighborhood of a thin shell: two axes span the *middle surface*, and the normal *shell director* spans the thickness.

2 Kinematics

Since it is thin, the geometry of the shell is well described by its *middle-surface* (see Figure 2). At any point on the middle surface the local tangent plane and surface normal induce a coordinate frame in which to describe "motion along the surface" and "motion along thickness."

In the discrete setting, the topology of the middle surface is represented by the combinatorics of an oriented 2-manifold simplicial complex, $M = \{v, e, f\}$, where $v = \{v_1, v_2, \ldots\}$, $e = \{e_1, e_2, \ldots\}$, $f = \{f_1, f_2, \ldots\}$ are sets of vertices, edges and faces respectively. The geometry of the middle surface is given by the discrete *configuration map*, $C : v \mapsto \mathbb{R}^3$, which assigns to every vertex, v_i, a position, $C(v_i)$, in the ambient space. Together M and C correspond to the usual notion of a *triangle mesh* in $\mathbb{R}^3$; in our exposition we assume fixed combinatorics M, and discuss a temporally evolving configuration, C_t where the subscript refers to a specific instant in time.

Restricting our attention to elastic ("memory-less") materials, the physics can be understood in terms of the *undeformed* configuration (the stress-free shape) and the *deformed* configuration (the stressed shape at the current instant in time), C_0 and C_1, respectively. The elastic response of a material depends on the *change in*

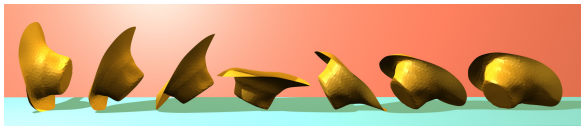


Figure 3: Frames from the simulation of tumbling thin shell.

shape of the elastic body, and on the *constitutive laws* that describe the restoring forces associated to this change in shape. The former is a purely geometric quantity.

What is the change in shape between C_0 and C_1? Since rigid motions (translations and rotations) do not affect shape, the answer must be invariant under composition of C_0 (likewise C_1) with any rigid body transformation. A simple theorem is that any reasonable measure of change in shape, or *generalized strain*, may be written as a function of only the edge lengths and dihedral angles of C_0 and C_1. The proof lies in showing that the configuration can be completely recovered from the edge lengths and dihedral angles, up to an unknown (but here inconsequential) rigid body transformation. We will also expect our measure of strain to be zero when shape has not changed, and non-zero whenever shape has changed. In particular, strain should "see" *any local change in shape*.

Perhaps the simplest forms of generalized strain which satisfy these desiderata are two expressions that are evaluated at a specific edge e_i. Comparing C_0 to C_1, let $s^e(e_i)$ be the difference in length of edge e_i, and let $s^\theta(e_i)$ be the difference in dihedral angle at e_i.

While these are perhaps the simplest possible measures of generalized strain, other more complex formulas can offer attendant advantages. Recent research in discrete shell models has focused on functions evaluated over mesh faces which aggregate in one term the configuration of all the incident edge lengths and dihedral angles [Gingold et al. 2004]. Nevertheless, our goal here is to develop the simplest discrete model of thin shells capturing their qualitative elastic behavior.

3 Constitutive Model

Having defined the *geometry* of thin shells, we turn our attention to the governing *physical* equations. The *stored elastic energy* of a thin shell is at the heart of the equations which govern its response to elastic deformations. The stored energy, $W[C_0, C_1]$, should be a function of the local strain, integrated over the middle surface.

We choose the simplest expression for energy that is consistent with Hookean mechanics. In 1676 Robert Hooke stated

> The power [*sic.*] of any springy body is in the same proportion with the extension.

This statement was the birth of modern elasticity, which states that a first order approximation for the response of a material is a force proportional to strain, and consequently (by the definition of work as force over distance) that the first approximation of stored energy is quadratic in strain. We propose an energy with two kinds of terms, measuring stretching and bending modes respectively:

$$W_M[C_0, C_1] = \sum_{e_i \in M} k_i^e \cdot \left(s^e(e_i)\right)^2 + \sum_{e_k \in M} k_k^\theta \cdot \left(s^\theta(e_k)\right)^2 ,$$

This expression has several desirable properties. First, it is positive whenever the shapes of C_0 and C_1 differ, and zero otherwise. Second, evaluations over subsets of M satisfy the usual inclusion/exclusion principle: for $A, B \subset M$, $W_M = W_A + W_B - W_{A \cap B}$, which is consistent with continuum formulations in which energy is defined as an integral of energy density over the middle surface. Third, because strain is invariant under rigid body transformations of the undeformed and deformed configurations, Nöther's theorem guarantees that the resulting dynamics will conserve linear and angular momentum. Consider the following interpretations of the membrane and bending terms:

Membrane Elastic surfaces resist stretching (local change in length). While some materials such as rubber sheets may undergo significant deformations in the stretching or shearing (*membrane*) modes, we focus on inextensible shells which are characterized by nearly *isometric* deformations, *i.e.*, possibly significant deformations in bending but unnoticeable deformation in the membrane modes. Most membrane models for triangle meshes satisfy this small-membrane-strain assumption with choice of suitably large membrane stiffness coefficient, k_i^e.

Rewriting the membrane term in the following form permits an alternative interpretation:

$$W^e(e_i) = k^e \left(|e_i| - |\bar{e}_i|\right)^2 = k^e |\bar{e}_i|^2 \left(\frac{|e_i|}{|\bar{e}_i|} - 1\right)^2$$

where $|e_i|$ is the length of edge i, quantities with a bar (such as $\bar{e}_i$) refer to the undeformed configuration C_0 and remaining quantities are with respect to C_1; note that we have dropped the subscript on k_i^e indicating a uniform material stiffness over the domain of interest. This is a unitless strain measurement, squared, and then integrated over the area of the local neighborhood, and multiplied by the material-dependent parameters. Observe that under regular refinement of a triangle mesh, the local area indeed scales as $|\bar{e}_j|^2$, which has units of area. The units of the material parameters are energy per unit area, *i.e.*, surface energy density. In engineering models of shells, the material parameter is given as a volume energy density, and the energy is integrated over shell thickness yielding a surface energy density. Efficient implementations of this formula precompute the quantities $k^e|\bar{e}_i|^2$, which depend only on the undeformed configuration.

Bending Consider the proposed discrete bending energy in relation to its continuous analogues. Models in mechanics are typically based on tensors, and in particular shell models use the difference of the second fundamental forms [Gray 1998] in the deformed and undeformed configurations (pulling back the deformed tensor onto the undeformed configuration). These treatments derive tensorial expressions over smooth manifolds, and as a final step discretize to carry out the numerics.

The shape operator [Gray 1998] is the derivative of the Gauss map[1]: geometrically, it measures the local curvature at a point on a smooth surface. Our bending energy is an extrinsic measure of the difference between the shape operator evaluated on the deformed and undeformed surfaces. We express this difference as the *squared difference of mean curvature*:

$$\left[\mathrm{Tr}(\varphi^* \mathrm{S}) - \mathrm{Tr}(\bar{\mathrm{S}})\right]^2 = 4(H \circ \varphi - \bar{H})^2 , \qquad (1)$$

where $\bar{\mathrm{S}}$ and S are the shape operators evaluated over the undeformed and deformed configurations respectively; likewise $\bar{H}$ and H are the mean curvatures; $\varphi^*\mathrm{S}$ is the pull-back of S onto $\bar{\Omega}$, and we use $\mathrm{Tr}(\varphi^* S) = \varphi^* \mathrm{Tr}(S) = \mathrm{Tr}(S) \circ \varphi = H \circ \varphi$ for a diffeomorphism φ. This measure is *extrinsic*: it sees only changes in the *embedding* of the surface in $\mathbb{R}^3$. Integrating (1) over the reference domain we find the continuous flexural energy $\int_{\bar{\Omega}} 4(H \circ \varphi - \bar{H})^2 d\bar{A}$. Next, we discretize this integral over the *piecewise linear mesh* that represents the shell.

We derive the discrete, integral *mean-curvature squared* operator as follows. We first partition the undeformed surface

[1] This is the map from the surface to the unit sphere, mapping each surface point to its unit surface normal.

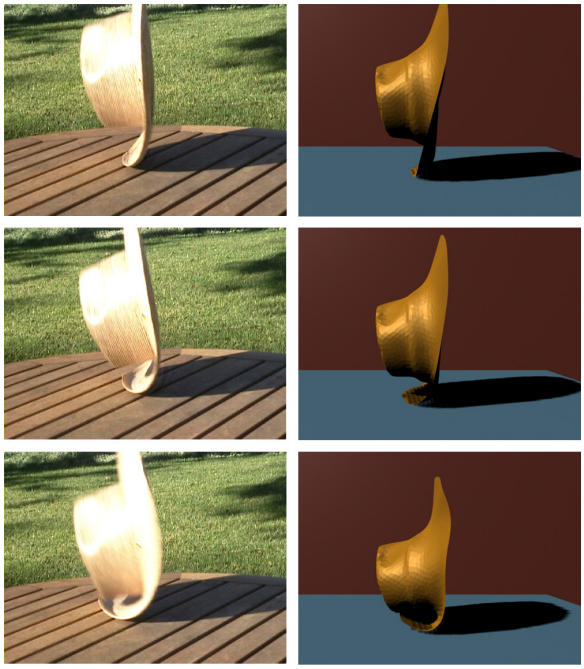
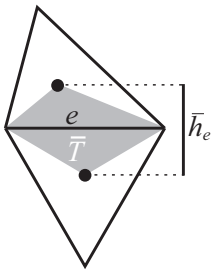


Figure 4: Real footage vs. Simulation: left, a real hat is dropped on a table; right, our shell simulation captures the bending of the brim. Notice that volumetric-elasticity, plate, or cloth simulations could not capture this behavior, while earlier work on shell simulation required significant implementation and expertise.

into a disjoint union of diamond-shaped tiles, $\bar{T}$, associated to each mesh edge, e, as indicated on the side figure. Following [Meyer et al. 2003], one can use the barycenter of each triangle to define these regions—or alternatively, the circumcenters. Over such a diamond, the mean curvature integral is $\int_{\bar{T}} \bar{H} d\bar{A} = \bar{\theta}|\bar{e}|$ (for a proof see [Cohen-Steiner and Morvan 2003]). A similar argument leads to: $\int_{\bar{T}} (H \circ \varphi - \bar{H}) d\bar{A} = (\theta - \bar{\theta})|\bar{e}|$. Using the notion of area-averaged value from [Meyer et al. 2003], we deduce that $(H \circ \varphi - \bar{H})|_{\bar{T}} = (\theta - \bar{\theta})/\bar{h}_e$, where $\bar{h}_e$ is the span of the undeformed tile, which is one sixth of the sum of the heights of the two triangles sharing $\bar{e}$. For a sufficiently fine, non-degenerate tessellation approximating a smooth surface, the average over a tile (converging pointwise to its continuous counterpart) *squared* is equal to the squared average, leading to: $\int_{\bar{T}} (H \circ \varphi - \bar{H})^2 d\bar{A} = (\theta - \bar{\theta})^2 |\bar{e}|/\bar{h}_e$.

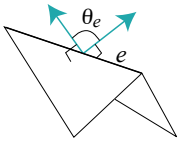


We might instead consider a formula of the form $(\theta - \bar{\theta})^2 |\bar{e}|$. Here the energy functional becomes dependent only on its piecewise planar geometry *not* on the underlying triangulation. An attractive claim, this is appealing in that a material's physical energy should depend on its shape, *not* on the discretization of the shape. Unfortunately, there is no discretization of (1) that simultaneously is (a) dependent only on the geometry *not* its triangulation, and (b) converges to its continuous equivalent under refinement. Indeed, the area integral of (1) is in general unbounded for a piecewise planar geometry! A discrete energy satisfying both (a) and (b) may exist for smoother surfaces, but our focus is piecewise planar (triangle mesh) geometry.

Following the argument found in [Meyer et al. 2003], there may be numerical advantages in using circumcenters instead of barycenters for the definition of the diamond tiles (except in triangles with obtuse angles). This affects the definition of $\bar{h}_e$ and of the lumped mass. Since we only need to compute these values for the undeformed shape, the implementation and performance *only of initialization code* would be affected. Bobenko notes that when circumcenters are used, this formulation of discrete shells coincides (for flat undeformed configurations) with the derivation of the discrete Willmore energy based on circle packing [Bobenko 2004],.

As we have just seen, we can express our *discrete flexural energy* as a summation over mesh edges,

$$W^{\theta}(e_k) = K^{\theta} \left(\theta_k - \bar{\theta}_k\right)^2 \frac{|\bar{e}_k|}{\bar{h}_k}\,, \tag{2}$$



where the term for edge e_k is where θ_k and $\bar{\theta}_k$ are corresponding complements of the dihedral angle of edge e_k measured in the deformed and undeformed configuration respectively, K^{θ} is the material bending stiffness, and $\bar{h}_k$ is a third of the average of the heights of the two triangles incident to the edge e_k (see the appendix for another possible definition of $\bar{h}_k$). Note that the unit of K^{θ} is energy (not surface energy density). This formulation is consistent with the physical scaling laws of thin shells: if the (deformed and undeformed) geometry of a thin shell is uniformly scaled by λ along each axis, then surface area scales as λ^2 as does the total membrane energy, *however* the total bending (curvature squared) energy is *invariant* under uniform scaling.

Following the reasoning for (1), we could have formed a second energy term taking the determinant instead of the trace of S. This would lead to a difference of Gaussian curvatures, but this is always zero under isometric deformations (pure bending). This is not surprising, as Gaussian curvature is an *intrinsic* quantity, *i.e.*, it is independent of the embedding of the two-dimensional surface into its ambient three-dimensional space. In contrast, flexural energy measures *extrinsic* deformations.

4 Dynamics

The treatment of the temporal evolution of a thin shell is beyond the scope of this chapter; we briefly summarize the basic components required to simulate the motion of thin shells.

Our dynamic system is governed by the ordinary differential equation of motion $\ddot{\mathbf{x}} = -\mathbf{M}^{-1} \nabla W(\mathbf{x})$ where $\mathbf{x}$ is the vector of unknown DOFs (*i.e.*, the vertices of the deformed geometry) and $\mathbf{M}$ is the mass matrix. We use the conventional simplifying hypothesis that the mass distribution is lumped at vertices: the matrix M is then diagonal, and the mass assigned to a vertex is a third of the total area of the incident triangles, scaled by the area mass density.

Newmark Time Stepping We adopt the Newmark scheme [Newmark 1959] for ODE integration,

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{x}_i + \Delta t_i \dot{\mathbf{x}}_i + \Delta t_i^2 \left((1/2 - \beta)\ddot{\mathbf{x}}_i + \beta \ddot{\mathbf{x}}_{i+1}\right), \\ \dot{\mathbf{x}}_{i+1} &= \dot{\mathbf{x}}_i + \Delta t_i \left((1 - \gamma)\ddot{\mathbf{x}}_i + \gamma \ddot{\mathbf{x}}_{i+1}\right), \end{aligned}$$

where Δt_i is the duration of the i^{th} timestep, $\dot{\mathbf{x}}_i$ and $\ddot{\mathbf{x}}_i$ are configuration velocity and acceleration at the beginning of the i^{th} timestep, respectively, and β and γ are adjustable parameters linked to the accuracy and stability of the time scheme. Newmark is either an explicit ($\beta = 0$) or implicit ($\beta > 0$) integrator: we used $\beta = 1/4$ for final production, and $\beta = 0$ to aid in debugging. Newmark gives control over numerical damping via its second parameter γ. We obtained the best results by minimizing numerical damping

($\gamma = 1/2$); this matches Baraff and Witkin's observation that numerical damping causes undesirable effects to rigid body motions. See also [West et al. 2000] for a discussion of the numerical advantages of the Newmark scheme.

Dissipation Shells dissipate energy via flexural oscillations. To that end we complete our model with an *optional* damping force proportional to $(\dot{\theta} - \dot{\bar{\theta}})\nabla\theta$ where $\dot{\bar{\theta}} = 0$ for elastic deformations but is in general non-zero for plastoelastic deformations. This is consistent with standard derivations of Rayleigh-type damping forces using the strain rate tensor [Baraff and Witkin 1998].

Discussion This discrete flexural energy (2) generalizes established formulations for (*flat*) plates both continuous and discrete: (a) Ge and coworkers presented a geometric argument that the stored energy of a continuous inextensible plate has the form $\int_{\bar{\Omega}} c_H H^2 + c_K K dA$ for material-specific coefficients c_H and c_K [Ge et al. 1996]; (b) Haumann used a discrete hinge energy [Haumann 1987], similarly Baraff and Witkin used a discrete constraint-based energy [Baraff and Witkin 1998], of the form $W_B(\mathbf{x}) = \sum_{\bar{e}} \theta_{\bar{e}}^2$. Our approach generalizes both (a) and (b), and produces convincing simulations beyond the regime of thin plate and cloth models (see Section 5).

The proposed discrete model has three salient features: (a) the energy is invariant under rigid body transformation of both the undeformed and the deformed shape: our system conserves linear and angular momenta; (b) the piecewise nature of our geometry description is fully captured by the purely intrinsic membrane terms, and the purely extrinsic bending term; most importantly, (c) it is *simple* to implement.

5 Results

We exercised our implementation on various problems, including fixed beams, falling hats, and pinned paper. Computation time, on a 2GHz Pentium 4 CPU, ranged from 0.25s–3.0s per frame; timings are based on a research implementation that relies on automatic differentiation techniques.

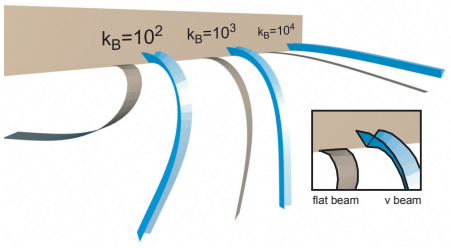


Figure 5: Three pairs of flat and v-beams with increasing flexural stiffness K^θ (left to right) of 100, 1000, and 10000; the non-flat cross section of the v-beam contributes to structural rigidity.

Beams We pinned to a wall one end of a v-beam, and released it under gravity. Figure 5 demonstrates the effect of varying flexural stiffness on oscillation amplitude and frequency. The flexural energy coefficient has a high dynamic range; extreme values (from pure-membrane to near-rigid) remain numerically and physically well-behaved. Observe that increasing flexural stiffness augments structural rigidity. Compare the behavior of beams: the non-flat cross section of the v-beam contributes to structural rigidity. This difference is most pronounced in the operating regime of low flexural stiffness (but high membrane stiffness). Here the material does not inherently resist bending, but a V-shaped cross-section effectively converts a bending deformation into a stretching deformation.

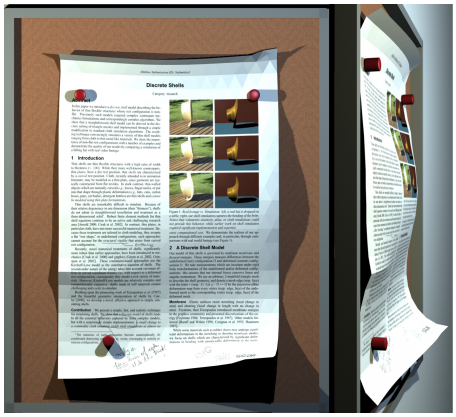


Figure 6: Modeling a curled, creased, and pinned sheet of paper: by altering dihedral angles of the reference configuration, we effect plastic deformation. While the rendering is texture-mapped we kept flat-shaded triangles to show the underlying mesh structure.

One can mimic this experiment by holding a simple paper strip by its end; repeat after folding a v-shaped cross-section.

Elastic hats We dropped both real and virtual hats and compared (see Figure 4): the deformation is qualitatively the same, during impact, compression, and rebound. Adjusting the damping parameter, we capture or damp away the brim's vibrations. Adjusting the flexural stiffness, we can make a hat made of hard rubber or textile of a nearly-rigid hat and a floppy hat).

Plastoelasticity As discussed in the early work of Kergosien and coworkers, a compelling simulation of paper would require a mechanical shell model [Kergosien et al. 1994]. Using our simple shell model, we can easily simulate a sheet of paper that is rolled, then creased, then pinned (see Figure 6). Here the physics require *plastic* as well as elastic deformations. We begin with a flat surface, and gradually increase the undeformed angles, $\bar{\theta}_e$. Notice: modifying the *undeformed* configuration effects a *plastic* deformation. The kinematics of changing $\bar{\theta}_e$ span only physically-realizable bending, *i.e.*, *inextensible* plastic deformations. In contrast, directly modifying $\bar{\mathbf{x}}$ could introduce plastic deformations with unwanted membrane modes. We introduced elastic effects by applying three pin constraints to the *deformed* configuration. Observe the half-crease on the left side. The energy of the undeformed state is no longer zero! The (plastically-deformed) left and (untouched) right halves have incompatible undeformed shapes, consequently the undeformed configuration is *not* stress-free.

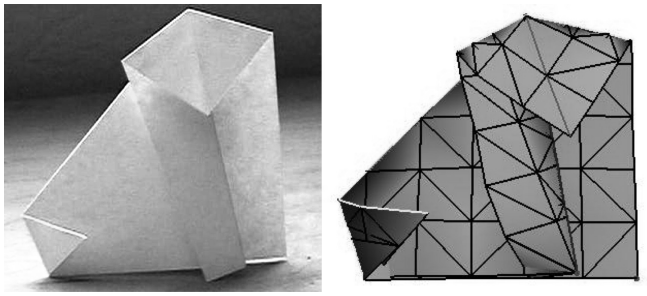


Figure 7: Virtual origami: user-guided simulated folding of a paper sheet produces a classical origami dog.

Recent extensions More recently, we demonstrated that simple, discrete models of thin shells can also produce striking examples of shattering glass (see Figure 1) [Gingold et al. 2004], and paper origami (see Figure 7) [Burgoon et al. 2006].

Implementation An attractive practical aspect of the proposed model is that it may be easily incorporated into working code of a standard cloth or thin-plate simulator such as those commonly used by the computer graphics community [Baraff and Witkin 1998]. One must replace the bending energy with (2). From an implementation point of view, this involves minimal work. For example, consider that [Baraff and Witkin 1998] already required all the computations relating to θ_e. These and other implementation details were outlined in [Grinspun et al. 2003].

6 Further Reading

A comprehensive survey of this expansive body of literature is far beyond the scope of this chapter; as a starting point see [Arnold 2000; Cirak et al. 2002] and references therein. Here we highlight only a few results from the graphics and engineering literature.

Recently, novel numerical treatments of shells, significantly more robust than earlier approaches, have been introduced in mechanics by Cirak *et al.* [Cirak et al. 2000] and in graphics by Green *et al.*and Grinspun *et al.* [Green et al. 2002; Grinspun et al. 2002] among others. These continuum-based approaches use the Kirchoff-Love constitutive equations, whose energy captures curvature effects in curved coordinate frames; consequently they model a rich variety of materials. In contrast, thin *plate* equations assume a flat undeformed configuration. Thin plate models are commonly used for cloth and garment simulations and have seen successful numerical treatment in the computer graphics literature (see [House and Breen 2000] and references therein). Thin plates have also been useful for variational geometric modeling [Celniker and Gossard 1991; Greiner 1994; Welch and Witkin 1992] and intuitive direct manipulation of surfaces [Qin and Terzopoulos 1996; Terzopoulos and Qin 1994]. In graphics, researchers have used two kinds of approaches to modeling plates: finite-elements and mass-spring networks. In the latter resistance to bending is effected by springs connected to opposite corners of adjacent mesh faces. Unfortunately, this simple approach does not carry over to curved undeformed configurations: the diagonal springs are insensitive to the sign of the dihedral angles between faces.

In this chapter we have developed a very simple discrete model of thin shells. One price that must be paid for this simplicity is that, while we have taken care to ensure the correct scaling factors for each energy term, for an arbitrary triangle mesh we cannot guarantee the convergence of this model to its continuum equivalent. In [Grinspun et al.] we present experimental results comparing the convergence of the discrete shell and other discrete curvature operators.

7 Acknowledgments

The work described here is the fruit of a collaboration with Mathieu Desbrun, Anil Hirani, and Peter Schröder [Grinspun et al. 2003]. Recent work on modeling thin shells is part of an active collaboration with Miklos Bergou, Rob Burgoon, Akash Garg, David Harmon, Adrian Secord, Yotam Gingold, Jason Reisman, Max Wardetzky, Zoë Wood, and Denis Zorin. The author is indebted to Alexander Bobenko, Jerry Marsden, and Anastasios Vayonakis for insightful discussions. Pierre Alliez, Ilja Friedel, Harper Langston, Anthony Santoro and Steven Schkolne were pivotal in the production of the images shown here. The author is grateful for the generous support provided by the National Science Foundation (MSPA-MCS 0528402), Elsevier, and nVidia.

References

ARNOLD, D., 2000. Questions on shell theory. Workshop on Elastic Shells: Modeling, Analysis, and Computation. Mathematical Sciences Research Institute, Bekeley.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH*, 43–54.

BOBENKO, A. I. 2004. A Conformal Energy for Simplicial Surfaces. Published online at http://arxiv.org/abs/math.DG/0406128, August.

BURGOON, R., GRINSPUN, E., AND WOOD, Z. 2006. Discrete shells origami. In *Proceedings of CATA*.

CELNIKER, G., AND GOSSARD, D. 1991. Deformable Curve and Surface Finite Elements for Free-Form Shape Design. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4, 257–266.

CIARLET, P. 2000. *Mathematical Elasticity. Vol. III*, vol. 29 of *Studies in Mathematics and its Applications*. Amsterdam. Theory of shells.

CIRAK, F., ORTIZ, M., AND SCHRÖDER, P. 2000. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg. 47*, 12, 2039–2072.

CIRAK, F., SCOTT, M., ANTONSSON, E., ORTIZ, M., AND SCHRÖDER, P. 2002. Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. *CAD 34*, 2, 137–148.

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted delaunay triangulations and normal cycle. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, 237–246.

GE, Z., KRUSE, H. P., AND MARSDEN, J. E. 1996. The limits of hamiltonian structures in three-dimensional elasticity, shells, and rods. *Journal of Nonlinear Science 6*, 19–57.

GINGOLD, Y., SECORD, A., HAN, J. Y., GRINSPUN, E., AND ZORIN, D. 2004. Poster: A discrete model for inelastic deformation of thin shells. In *ACM/Eurographics Symposium on Computer Animation '04*.

GRAY, A. 1998. *Modern Differential Geometry of Curves and Surfaces*. Second edition. CRC Press.

GREEN, S., TURKIYYAH, G., AND STORTI, D. 2002. Subdivision-based multilevel methods for large scale engineering simulation of thin shells. In *Proceedings of ACM Solid Modeling*, 265–272.

GREINER, G. 1994. Variational design and fairing of spline surfaces. *Computer Graphics Forum 13*, 3, 143–154.

GRINSPUN, E., GINGOLD, Y., REISMAN, J., AND ZORIN, D. Computing discrete shape operators on general meshes. submitted for publication.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: a simple framework for adaptive simulation. *ACM Transactions on Graphics 21*, 3 (July), 281–290.

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *ACM SIGGRAPH Symposium on Computer Animation*. to appear.

HAUMANN, R. 1987. Modeling the physical behavior of flexible objects. In *Topics in Physically-based Modeling,* Eds. Barr, Barrel, Haumann, Kass, Platt, Terzopoulos, and Witkin*, SIGGRAPH Course Notes*.

HOUSE, D. H., AND BREEN, D. E., Eds. 2000. *Coth Modeling and Animation*. A.K. Peters.

KERGOSIEN, Y. L., GOTODA, H., AND KUNII, T. L. 1994. Bending and creasing virtual paper. *IEEE Computer Graphics and Applications*, 40–48.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, H.-C. Hege and K. Polthier, Eds. Springer-Verlag, Heidelberg, 35–57.

NEWMARK, N. M. 1959. A method of computation for structural dynamics. *ASCE J. of the Engineering Mechanics Division 85*, EM 3, 67–94.

QIN, H., AND TERZOPOULOS, D. 1996. D-NURBS: A physics-based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics 2*, 1, 85–96.

TERZOPOULOS, D., AND QIN, H. 1994. Dynamic NURBS with Geometric Constraints for Interactive Sculpting. *ACM Transactions on Graphics 13*, 2, 103–136.

WELCH, W., AND WITKIN, A. 1992. Variational Surface Modeling. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 157–166.

WEST, M., KANE, C., MARSDEN, J. E., AND ORTIZ, M. 2000. Variational integrators, the newmark scheme, and dissipative systems. In *International Conference on Differential Equations 1999*, World Scientific, Berlin, 1009 – 1011.

Discrete Quadratic Curvature Energies

Miklos Bergou, Columbia University
Max Wardetzky, Freie Universität Berlin
David Harmon, Columbia University
Denis Zorin, New York University
Eitan Grinspun, Columbia University

Abstract

Efficient computation of curvature-based energies is important for practical implementations of geometric modeling and physical simulation applications. Building on a simple geometric observation, we provide a version of a curvature-based energy expressed in terms of the Laplace operator acting on the embedding of the surface. The corresponding energy—being quadratic in positions—gives rise to a constant Hessian in the context of isometric deformations. The resulting isometric bending model is shown to significantly speed up common cloth solvers, and when applied to geometric modeling situations built on Willmore flow to provide runtimes which are close to interactive rates.

1 Introduction

Efficient computation of curvature-based energies is important for practical implementations of physical simulation and geometric modeling applications. Any coordinate-invariant curvature energy may be expressed in terms of principal curvatures (eigenvalues of the shape operator), and desired symmetries often lead to expressions in terms of the elementary symmetric functions: mean ($H = \kappa_1 + \kappa_2$), Gaussian ($K = \kappa_1\kappa_2$), and total ($\kappa_1^2 + \kappa_2^2$) curvature. Energies assembled from these elementary expressions are widely used in geometric modeling, e.g. [Schneider and Kobbelt 2001; Clarenz et al. 2004; Hildebrandt and Polthier 2004; Bobenko et al. 2005; Deckelnick et al. 2005] as well as physical simulation of 2D deformable objects, such as cloth and thin shells (see the recent survey of Thomaszewsi and Wacker [2006] and the theory of Ciarlet [2000]). As a model problem we consider the bending energy functional

$$E(S) = \int_S H^2 \mathrm{d}vol\,. \tag{1}$$

In the continuous case, this energy is invariant under rigid motions and uniform scaling of the surface S (in fact, it is invariant under all Möbius transformations of ambient 3-space [White 2000]).

In this chapter we explore the interrelation between isometry, differential operators, and curvature energy. The importance of isometry for simplification of energy was previously acknowledged in the context of surface fairing and modeling, *e.g.*, by Desbrun *et al.* [1999b] and later Schneider *et al.* [2001]. We focus primarily on the simulation of inextensible plates and shells where physics dictates quasi-isometry: membrane stiffness typically is greater than bending stiffness by four or more orders of magnitude (cf. Koiter's model [Ciarlet 2000]) and the advantages of isometry can be exploited in full.

Although ultimately we are interested in discretizing the functional (1), we begin with a central observation and guiding principle in the continuous domain: $E(S)$ is "quadratic in positions" in the following sense. If $\mathbf{I} : S \to \mathbb{E}^3$ denotes the embedding of the surface, the *mean curvature normal* $\mathbf{H}$ of S can be written as the intrinsic Laplacian Δ (induced by the Riemannian metric of S) applied to the embedding of the surface, $\mathbf{H} = \Delta\mathbf{I}$ (note that this is a vector-valued quantity; here and henceforth vector quantities are typeset in boldface). Then we write our bending model as

$$E(S) = \int_S \langle \Delta\mathbf{I}, \Delta\mathbf{I}\rangle \mathrm{d}vol. \tag{2}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product of $\mathbb{E}^3$.

We pay special attention to isometric deformations of a surface—such deformations are a reasonable assumption, *e.g.*, for various textiles and more generally inextensible thin-plates and -shells. Since the intrinsic Laplacian Δ remains unchanged under isometric deformations of the surface, we infer that $E(S)$ is quadratic in positions under isometric deformations; therefore, in the discrete case the corresponding Hessian is constant. We will refer to $E(S)$ written in the form of (2) as the *isometric bending model* (IBM).

From a differential geometry point of view, these observations offer no surprises. However, they have a number of very practical consequences when carried over to discrete differential geometry (DDG). In this setting one begins with a discrete mean curvature or discrete Laplacian if one seeks a discrete version of (1) or (2), respectively. While DDG operators are extensively studied [Pinkall and Polthier 1993; Mercat 2001; Meyer et al. 2003; Bobenko 2005], our aim here is to seek operators which satisfy a discrete IBM: a bending energy quadratic in positions under the class of discrete isometric deformations. Therefore we must begin with a reasonable definition of isometry, which in the case of triangulated surfaces we take to be that (a) bending occurs only along edges, and (b) edges may not stretch. In reality, the second clause is relaxed *i.e.*, we consider discrete quasi-isometry.

In analogy to the smooth case, a discrete IBM will be based on Laplacians which are—by construction—invariant under discrete isometric deformations. Fortunately, the necessary discrete operators are readily available, either by building on a mixture of conforming (vertex-based) and nonconforming (edge-based) linear finite elements, but also, as we shall explain, by linearizing well-established bending models such as [Bridson et al. 2003; Grinspun et al. 2003] about the planar rest state.

In short, the main purpose of the current chapter is twofold: showing the transition of the continuous IBM to a discrete one, and exploiting this discrete model for speeding up computations involving inextensible thin materials.

Main observations Our central observation is that the isometric bending energy has a constant Hessian corresponding to Δ^2 (or more generally a squared second-order differential operator). Based on this observation,

- We describe a family of discretizations of the IBM which are (i) *invariant under rigid transformations and uniform scaling* and (ii) *quadratic in mesh positions*.
- We show that virtually any cloth or thin plate simulator which requires the computation of bending forces can easily incorporate this discrete IBM, and in doing so, we observed a 2-3$\times$ speedup for representative solvers.
- We remark that using the IBM as a surrogate energy in the semi-implicit solution of Willmore flow enables hole-filling and non-shrinking smoothing applications at nearly interactive rates.

Figure 1: Snapshots from our simulation of a billowing flag. The accompanying movie demonstrates that despite its economy of cost, the proposed bending model achieves qualitatively the same dynamics as popular nonlinear models.

1.1 Physical models of cloth and thin plates

Efficient simulation of clothing and thin plates is important for feature film production, fashion design, and manufacturing, among other applications. The pioneering work of Terzopoulos *et al.* [1987], who introduced tensorial treatment of elastica in graphics, and of Baraff and Witkin [1998], who demonstrated the power of a semi-implicit integration framework for cloth, spurred numerous significant developments in the treatment of numerics [Ascher and Boxerman 2003; Hauth et al. 2003], contact response [Bridson et al. 2002; Baraff et al. 2003], and constitutive models [Choi and Ko 2002]. The body of literature on simulation of cloth is too broad to survey here, and we refer the reader to the recent surveys [Choi and Ko 2005; Magnenat-Thalmann and Volino 2005; Zhu et al. 2004; Ng and Grimsdale 1996] and the text of House and Breen [2000].

In general, computer graphics methods have formulated and discretized separately the membrane and bending modes of deformation (an approach mirrored by treatments of thin plates in the finite element community [Zienkiewicz and Taylor 2000; Hughes 1987]).

Although bending forces are much weaker then stretching forces, their interaction with membrane forces determines the shape of folds and wrinkles that we associate with garments and other thin flexible bodies [Thomaszewski and Wacker 2006; Ciarlet 2000].

Membrane modes Feynman [Feynman 1986] introduced discretizations of membrane energy for computer animation, and other models followed [Terzopoulos et al. 1987; Haumann 1987; Carignan et al. 1992; Baraff and Witkin 1998]. Any good treatment of membrane response satisfies the assumption of small in-plane strains. To achieve this, some practicioners use a stiff membrane response, whereas others prefer a reduced stiffness combined with a strain limiting procedure [Provot 1995; Bridson et al. 2002; Desbrun et al. 1999a]. In summary, we may safely view the planar strain (membrane) forces as a mechanism (in the spirit of Lagrange multipliers or penalty forces) that enforces an isometry constraint on the manifold of permissible deformations. In this light, our concern here is impartial to the choice of membrane model.

Bending modes For a thorough overview of bending models in graphics we refer to [Thomaszewski and Wacker 2006]. Discretizations of bending may be loosely categorized as involving (a) particles interconnected by springs and dampers [Breen et al. 1994; Volino et al. 1995; Choi and Ko 2002], (b) linear or higher-order finite-elements [Etzmuss et al. 2003; Cirak et al. 2000], or (c) based on dihedral angles [Baraff and Witkin 1998; Bridson et al. 2003; Grinspun et al. 2003].

For comparison, we implemented the widely-adopted bending model described in [Baraff and Witkin 1998], henceforth the "nonlinear hinge." To our knowledge all the commonly-used models are either inherently nonlinear in positions—they usually involve expressions in terms of lengths or angles —or they assume small displacements and factor out a rigid motion each time step. The bulk of our performance advantage results from the observation that IBM's forces are naturally linear in position, without assuming small displacements or disposing of rotational invariance.

Before going into details of these applications, we start by outlining our discretization scheme. This scheme will yield an IBM based on discrete differential operators.

2 Geometric discretization of curvature energy

Our guiding principles for discretization are: (a) to maintain the geometric description of the continuous operators defined above, (b) to maintain the simple quadratic structure of bending energy when expressed in these terms, and (c) to preserve key symmetries of the continuous energy: invariance under rigid transformations and uniform scaling.

2.1 Discrete mean curvature

Several discrete versions of mean curvature have been brought forward recently. We review some of the most important models for triangulated meshes here. Observe that in all these models discrete curvature is an *integrated quantity*. In this setting curvatures do not correspond to functions (quantities which can be evaluated pointwise). Instead, they correspond to evaluations of *functionals*—functions integrated over some domain Ω. The need for a clear distinction between functions and functionals becomes immediate when applying operations, in our case when squaring H: squaring the pointwise mean curvature and then integrating, $\int_\Omega H^2$, is *not* the same as integrating the mean curvature and then squaring, $\left(\int_\Omega H\right)^2$.

In more general terms, we think of a functional F to associate to a function f its integrated value over some domain Ω. The opposite direction, going from a functional F to a function f, involves *averaging* (dividing) by the area of Ω. This underscores the relevance of Voronoi regions and barycentric area in the geometric modeling literature. Extending this notation, we may evaluate the functional over an array of domains, Ω_i. In matrix-vector form, this reads $f = M^{-1}F$, where M is a matrix, with diagonal entries $|\Omega_i|$ corresponding to areas. The key observation is that M^{-1} projects evaluations of a functional to their corresponding ("averaged") function values. This relation may be derived in a finite-element framework where M (now not necessarily diagonal) retains the meaning shown here (cf. Appendix B). In working with triangle meshes, the domains Ω_i are usually associated to neighborhoods of either edges (e_i) or vertices (v_i).

Edge-based Edges are the atomic entities across which a triangulated surface may be considered flexible. In graphics, a common model for integrated discrete mean curvature associated with an edge e is given by the product of dihedral angle and edge length, $H(e) = (\pi - \theta_e) \cdot \|e\|$. This version was derived in the context of geometric measure theory [Cohen-Steiner and Morvan 2003] and found high-quality applications in modeling thin shells [Grinspun et al. 2003]. In [Bridson et al. 2003] a similar model was used, by replacing $(\pi - \theta_e)$ by $\cos(\theta_e/2)$. Obviously, these two versions agree (up to a factor of two) in the limit of small angles between normals ($\theta_e \to \pi$). Coming from a different direction, in [Bobenko 2005] a version of integrated mean curvature *squared* was introduced, based on the intersection angles of circumcircles. This model is a priori a functional version of H^2, as opposed to starting out with a functional representation of H—a nontrivial insight. Note that this is done in a functional perspective, so it does not a priori give a model of discrete mean curvature itself. However, this concept is still found to be intimately linked to the previous two in the limit of the planar case [Bobenko 2005].

Linear edge-based model Although various nonlinear discrete models for mean curvature agree for small deformations, we are interested in a *linear* model which is also valid for large deformations. Consequently, one might be tempted to *impose* the linear model for small displacements onto the case of large deformations. This ad hoc view can be formalized by considering so-called "Crouzeix-Raviart linear finite elements" and obtaining a linear model for the mean curvature *normal* (cf. Appendix B). Such a linear model was proposed in [Hildebrandt and Polthier 2004],

$$\mathbf{H}(e) = -2\cos\frac{\theta_e}{2} \cdot \|e\| \cdot \mathbf{N}_e,$$

where $\mathbf{N}_e$ is the (angle-bisecting) normal vector to e of unit length. Note that unlike the scalar-valued mean curvature, $\mathbf{H}$ is a vector, and the norm of $\mathbf{H}$ corresponds to the (scalar) mean curvature. We make two practical observations: The first observation about this version is that it is *linear in vertex positions of the mesh* (cf. Appendix B). The second observation is that the norm of this operator, $\|\mathbf{H}(e)\|$, agrees with the above nonlinear models, up to first order, in the limit of the planar case. This linear version of the mean curvature vector will hence serve as the basis for our desired discrete IBM. As a corollary, we find that in the limit of small dihedral angles, the discrete IBM will match the bending stiffness of the original nonlinear model. Consequently, the material stiffness coefficients of the original model will carry over to the discrete IBM, eliminating a need to search for new material parameters.

Vertex-based Vertex-centered discrete mean curvatures are obtained by choosing one of the above edge-based models and summing the contributions from all edge-based curvatures of incident edges,

$$\mathbf{H}(p) = \frac{1}{2}\sum_{e \ni p} \mathbf{H}(e).$$

The factor $1/2$ is due to the fact that we treat integrated quantities here, and each triangle in the vertex star of a vertex p is seen by exactly two edges.

Discrete Laplacian Recall that in the continuous setting, the mean curvature normal is inimately related to the Laplacian: $\mathbf{H} = \Delta\mathbf{I}$. Given a *discrete model* of a Laplacian L (such as obtained by finite elements, cf. Appendix B), one can view $\mathrm{L}\vec{\mathbf{I}}$ as a discrete mean curvature functional, where the vector $\vec{\mathbf{I}}$ corresponds to the coefficient vector of positional degrees of freedom. By the above, the corresponding discrete mean curvature function reads

$$\boxed{\vec{\mathbf{H}}_d = \mathrm{M}^{-1}(\mathrm{L}\vec{\mathbf{I}})\,,} \tag{3}$$

where the vector $\vec{\mathbf{H}}_d$ corresponds to pointwise (average, not integrated) mean curvature normals, and the mass matrix M is the area-scaling matrix we discussed above. In Appendix B we give a concise finite element view of these relations. While the path from L to $\vec{\mathbf{H}}_d$ is well-understood, the opposite direction is relatively unexplored. Given a discrete mean curvature normal, such as from [Bridson et al. 2003; Grinspun et al. 2003; Cohen-Steiner and Morvan 2003], we may consult (3) to define L—here again (3) will hold *by construction*. In Appendix C we outline a procedure for uniquely recovering L given a (possibly nonlinear in positions) discrete mean curvature. To summarize, we have imposed that the mathematical relation in the continous setting ($\mathbf{H} = \Delta\mathbf{I}$) by construction carries over to the discrete setting ($\vec{\mathbf{H}}_d = \mathrm{M}^{-1}(\mathrm{L}\vec{\mathbf{I}})$).

2.2 Discrete isometric bending model

Given a discrete mean curvature normal, $\vec{\mathbf{H}}_d$, we can—in perfect analogy to the smooth case—define the discrete IBM

$$E_d = \int_S \langle \mathbf{H}_d, \mathbf{H}_d \rangle \mathrm{d}vol\,.$$

The concrete value of this energy will depend on which version of L (consequently $\mathbf{H}_d$) is used. In general, this discrete energy takes the form

$$\boxed{E_d(\mathbf{I}) = \vec{\mathbf{I}}^T(\mathrm{L}^T\,\mathrm{M}^{-1}\,\mathrm{L})\vec{\mathbf{I}}\,.}$$

The form of $E_d(\mathbf{I})$ reveals two crucial properties: (i) it is *invariant under rigid transformations and uniform scaling*, and (ii) it is *quadratic in positions* in the class of isometric deformations, which include arbitrarily large bending away from the initial shape. A third property is that the energy Hessian $\mathbf{K} = \mathrm{L}^T\,\mathrm{M}^{-1}\,\mathrm{L}$ allows for a geometric interpretation: L is invariant under *conformal* deformations (it only depends on *angles*), whereas the entries of M correspond to *area*.

We now show how to apply the discrete IBM to obtain significant speedups in situations of a cloth solver and geometric modeling application.

3 Application to the bending of cloth and plates

Isometric bending model For cloth simulation, we chose to implement the simplest "hinge" IBM, whose local stencil matches that of the nonlinear hinge model. The simple tools required to implement this IBM are given in Appendix A.

The hinge IBM uses the nonconforming (edge-based) finite element theory summarized in Appendix B. Despite deferring to an edge-based FEM, we emphasize that all DOFs are associated to vertices. In other applications, a vertex-based (conforming FEM) formulation was used by [Clarenz et al. 2004] among others. Here we are primarily interested in the nonconforming FEM formulation since it (a) reflects the flexibility of triangle meshes at *edge hinges* and by the above discussion it (b) leads to an energy which in the limit of planar rest state coincides with previously considered bending energies built from nonlinear models up to second order.

Numerical treatment Whether in the high-fidelity or interactive setting, rapid simulation requires an efficient numerical integrator for second order initial value problems (IVPs) of the form

$$M\ddot{\mathbf{I}}(t) = f(t, \mathbf{I}(t), \dot{\mathbf{I}}(t))\,, \tag{4}$$

where M is the (physical) mass matrix, $f(t, \mathbf{I}(t), \dot{\mathbf{I}}(t))$ are time-, position, and velocity-dependent forces, and the initial position and

velocity of the cloth are prescribed [Hauth 2004]. For analysis one often introduces velocity as a separate variable, rewriting the above as a coupled first-order system,

$$\begin{pmatrix} Id & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} \dot{\mathbf{I}}(t) \\ \dot{\mathbf{V}}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{V}(t) \\ f(t, \mathbf{I}(t), \mathbf{V}(t)) \end{pmatrix},$$

with appropriate initial conditions. The temporal discretization of this system is a well-studied and active area of applied mathematics and computational mechanics, with a host of attendent methods. For treatments tailored to cloth simulation we refer the reader to [Baraff and Witkin 1998; Etzmuss et al. 2000; Hauth and Etzmuss 2001; Choi and Ko 2002; Ascher and Boxerman 2003; Bridson et al. 2003; Hauth et al. 2003; Boxerman and Ascher 2004; Hauth 2004]).

As a general observation, each force may be treated *explicitly* or *implicitly*. An explicit treatment requires the (possibly repeated) evaluation of the force per discrete time step; an implicit method requires additionally the (possibly repeated) evaluation of the force Jacobian per discrete time step. In the case of a conservative force, such as elastic bending, the force Jacobian is minus the energy Hessian. In the case of a dissipative Rayleigh force, such as damping of the bending modes, the Jacobian is expressed as a linear combination of the physical mass and energy Hessian. Etzmuss, Hauth *et al.* [2000; 2001; 2003; 2004] as well as Ascher and Boxerman [2003; 2004] analyzed the behavior of time integration schemes and presented arguments for the adoption of implicit-explicit (IMEX) integrators; Bridson *et al.* [2003] presented a semi-implicit method that treats damping implicitly and elastic forces explicitly. Our results indicate that IBM accelerates both explicit or implicit treatments of bending.

Results We compare the computational cost and visual quality of the isometric bending energy to the widely-used bending ("hinge") energy described in [Baraff and Witkin 1998; Bridson et al. 2003; Grinspun et al. 2003], for regular- and irregular-meshes ranging from 400 to 25600 vertices, on a draping problem as well as a dynamic billowing flag. Figures 1-2 and the accompanying movie provide a qualitative point of comparison between IBM and the nonlinear hinge model.

Implementing the isometric bending model is straightforward, requiring the (precomputed) assembly of the Hessian matrix (coefficients of $\mathrm{L}^T\,\mathrm{M}^{-1}\,\mathrm{L}$ are given in the Appendix), and a matrix-vector multiplication to compute the forces $(-\mathrm{L}^T\,\mathrm{M}^{-1}\,\mathrm{L})\vec{\mathbf{I}}$. In contrast, an efficient implementation of a nonlinear model such as a hinge spring requires both significant (human and computer) gradient calculations, or leads to adoption of costly automatic-differentiation techniques [Grinspun et al. 2003]. For comparison of computational efficiency, our implementation of the hinge forces and Jacobians was hand tuned and not based on automated methods.

Replacing the nonlinear hinge model with quadratic IBM reduces bending force computation cost by seven- to eleven-fold. Where a force Jacobian is required, IBM's constant Hessian is precomputed once and stored in a sparse matrix datastructure, eliminating the computational cost of bending Jacobian assembly. Of course, the actual net performance improvement in any given application depends on the fraction of total computation associated to bending.

Test setup To estimate this, we implemented both the implicit solver framework of [Baraff and Witkin 1998] as well as a simple explicit Euler solver. Our choice of solvers is motivated by a desire to estimate profitability of incorporating IBM whether or not a framework requires bending force Jacobians. The test framework incorporates: (a) the usual constant strain linear finite element for membrane response [Zienkiewicz and Taylor 2000; Hughes 1987], which computationally is at least as costly as the membrane model proposed in [Baraff and Witkin 1998]; (b) the robust collision detection and response methods of [Bridson et al. 2002], augmented with k-DOP trees [Klosowski et al. 1998]; (c) the PETSc solver library [Balay et al. 2001]. Further optimization of our collision detection using [Govindaraju et al. 2005] is likely to reduce the overhead of collision computations for large meshes. For a more imformed comparison, in Table 1 we report costs both including as well as excluding collision handling. We observe a consistent reduction in total computational cost compared to the nonlinear hinge model, across the four problem setups, two mesh types and resolutions ranging from 400 to 25600 vertices.

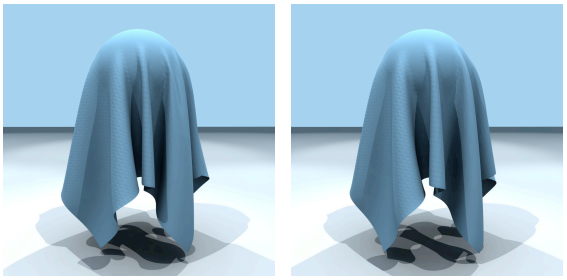


Figure 2: Final rest state of a cloth draped over a sphere, for *(left)* the proposed isometric bending model and *(right)* the widely-adopted nonlinear hinge model.

Draping cloth. We simulated the draping of a square sheet over a sphere. As shown in Figure 2, the draped cloths are qualitatively similar in their configuration and distribution of wrinkles and folds. Since only the final draped state was important, we introduced a dissipative Rayleigh force allowing for larger time steps [Hughes 1987]. For this example we might also consider a quasistatic method, as recently considered in [Teran et al. 2005]. Since quasistatics requires repeated evaluation of forces and their Jacobian, the proposed bending model will also be profitable in the quasitatic setting. Together Figures 2-3 capture the behaviour of IBM under a range of bending stiffnesses.

Billowing flag. We simulated the dynamics of a flag under wind. As shown in the accompanying movie (see also Fig. 1) the motion of the flag is qualitatively unaffected when we substitute the more economical quadratic bending energy. Furthermore, we found that there is no need to readjust material parameters when switching from the nonlinear hinge to the IBM model; this is not unexpected in light of the link between the two energies, as discussed in §2. We modeled wind by a constant homogeneous velocity field, with force proportional to the projection of the wind velocity onto the surface-normal. For more sophisticated effects of wind fields on cloth see [Keckeisen et al. 2004]. Purely for contrast with the draping example, we did not introduce any specific damping forces; although the implicit Euler method is known to exhibit numerical damping.

IMEX methods. We stress that the proposed model is not specialized to our test framework. Consider for example the integration of the IBM into the framework described by Bridson *et al.* [2003], which treats elastic (position-dependent, velocity-independent) forces explicitly, and treats damping forces implicitly. With IBM both force and Jacobian computation is greatly reduced, therefore both explicit, implicit, and semi-implicit treatments are prime candidates. Bridson *et al.* describe a very efficient matrix-free solver strategy, involving as few as one to two conjugate gradient iterations: this heightens the relevance of fast (elastic

Draping problem		regular mesh (resolution, in no. vertices)				irregular mesh (resolution, in no. vertices)			
		400	1600	6400	25600	450	2100	6500	22500
Gradient	nonlinear hinge	0.937	3.45	16.4	66.6	1.10	5.43	17.6	67.8
cost (ms)	quadratic IBM	0.081	0.338	2.19	9.15	0.098	0.494	2.32	9.68
Hessian	nonlinear hinge	12.8	54.2	218	890.	15.2	77.2	246	888
cost (ms)	quadratic IBM	0.237	0.963	3.87	15.7	0.266	1.28	3.99	13.6
Explicit step	nonlinear hinge	3.81	6.64	27.5	112.	2.16	9.53	31.4	140.
cost (ms)	quadratic IBM	2.63	2.90	11.9	48.8	0.964	4.35	15.2	76.5
Implicit step	nonlinear hinge	28.6	138	470.	1730	33.9	219	557	1880
cost (ms)	quadratic IBM	11.0	62.7	168	505	13.6	103	219	612

Flag problem		regular mesh (resolution, in no. vertices)				irregular mesh (resolution, in no. vertices)			
		400	1600	6400	25600	450	2100	6500	22500
Gradient	nonlinear hinge	0.975	3.99	16.0	64.0	1.10	5.43	17.8	68.7
cost (ms)	quadratic IBM	0.085	0.341	2.14	8.75	0.099	0.490	2.31	9.28
Hessian	nonlinear hinge	13.4	54.8	212	849	15.2	77.4	247	887
cost (ms)	quadratic IBM	0.251	0.974	3.79	14.99	0.267	1.30	3.96	13.7
Explicit step	nonlinear hinge	1.73	7.05	27.7	112.	1.97	9.80	32.7	134
cost (ms)	quadratic IBM	0.780	3.26	13.3	53.4	0.900	4.54	16.1	70.0
Implicit step	nonlinear hinge	27.6	106	420.	1680	33.5	155	513	1880
cost (ms)	quadratic IBM	9.53	32.9	127	490	12.5	50.4	166	608

Table 1: Computational cost per time step for a variety of regular- and irregular-mesh resolutions, comparing our quadratic energy to the a popular nonlinear hinge energy. Quantities in parentheses correspond to timings that include an implementation of collision detection and response. These tests were conducted on a single process, Pentium D 3.4GHz, 2GB RAM.

and damping) bending force computations, since with fewer conjugate gradient iterations, the balance of computation shifts further to the computation of the cached nonlinear position-dependent terms as well as the algebraic system's right hand side. In incorporating IBM in this context, the requisite matrix-vector multiplication may be distributed, $(\mathbf{K}_m + \mathbf{K}_b)\vec{\mathbf{I}} = \mathbf{K}_m\vec{\mathbf{I}} + \mathbf{K}_b\vec{\mathbf{I}}$, where the precomputed K_b is a multiple of the constant bending Jacobian, and K_m is the membrane Jacobian.

In any practical application, the final decision on incorporating a proposed technique hinges on the implementation cost and perfomance benefit of the method. We have presented a model for bending that involves simply computing the entries of a sparse matrix. This computation is straightforward, and all details are outlined in the Appendix. Finally, incorporation of the proposed model is a minimally-invasive task. Given these considerations, we suggest that it will be straightforward to evaluate the efficacy of the proposed model within the reader's preferred cloth simulation framework.

3.1 Geometric Modeling

The *Willmore energy* of a surface is given as

$$E_W(\mathbf{I}) = \int_S (H^2 - K)\mathrm{d}vol \;=\; \frac{1}{4}\int_S (\kappa_1 - \kappa_2)^2 \mathrm{d}vol.$$

As noted in [Bobenko et al. 2005], immersions which minimize Willmore energy are of interest in a range of areas, including the study of conformal geometry [Blascke 1929; Willmore 2000], physical modeling of fluid membranes [Canham 1970; Helfrich 1973], and our focus in this example, geometric modeling. For surfaces without boundary and surfaces whose boundary is fixed up to first order, the Willmore functional is variationally equivalent to the functional (1). The corresponding geometric flow

$$\dot{S} = -\nabla E(S),$$

has spurred many applications for surface fairing and restoration [Yoshizawa and Belyaev 2002; Bobenko et al. 2005; Clarenz et al. 2004; Schneider and Kobbelt 2001]. One of the earliest examples of discrete Willmore flow was implemented in Ken Brakke's *Surface Evolver* [Hsu et al. 1992]. This work uses the cotangent formula to discretize mean curvature, and explicit time stepping. Yoshizawa *et al.* [2002] discretize explicit expressions for forces obtained from the variation of the Willmore energy, and use the cotangent formula to discretize both the mean curvature and Laplace-Beltrami operator in the force expression. Explicit timestepping is used and an additional tangential component is added to the forces to improve the quality of the evolving mesh. Clarenz *et al.* [2004] discretize the variation of the Willmore energy in terms of finite elements and treat the corresponding L^2-flow by a coupled system of second order equations. Finally, Bobenko *et al.* [2005] introduced a circle-based conformally invariant Willmore energy discretization in their flow formulation.

In geometric hole-filling applications (see Fig. 5) the flow is integrated to its stationary limit. In smoothing applications (see Figure 4) the flow is integrated over a prescribed duration, and longer integration times smooth progressively coarser spatial frequencies. Thus the flow duration effectively controls a trade-off between keeping and discarding the initial geometry in exchange for the "smoothest" immersion of given genus satisfying prescribed G^1 boundary conditions.

Discrete IBM in Willmore solver Our discrete isometric bending model can be readily used for an efficient Willmore solver. For the forces we can consider the complete gradient of the energy, not assuming any specific type of deformation. For the Jacobian of forces we use our discrete IBM, together with a Cholesky direct solver. We need only compute the local stiffness matrices *once* at the beginning of time, allowing us to pre-factor (symbolically and numerically) the Jacobian matrix before the flow begins. Our focus here is on a formulation that facilitates rapid computation at nearly interactive rates while visually maintaining good surface quality (see Figs. 4 and 5).

Implementation Our implementation uses semi-implict backwards Euler time integration using as a linear solver the PARDISO [Schenk and Gärtner 2004] LL^T direct solver. We report computation times for a single process running on a 2GHz notebook with 2GB RAM.

One avenue for extending our application would be to incorporate a reparameterization flow, such as the special tangent speed component that Yoshizawa and Belyaev used to improve the quality of the evolving mesh [Yoshizawa and Belyaev 2002].

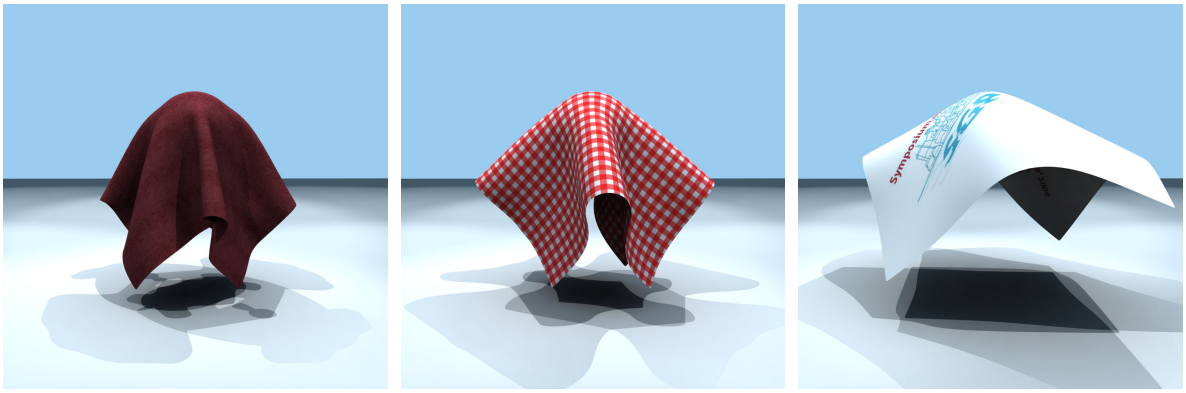


Figure 3: Deformations are primarily characterized by the ratio of bending to membrane stiffness. Despite having forces linear in positions, the quadratic bending model is valid over a broad range of stiffness values. Shown here *(left to right)*: $10^{-5} : 1$, $10^{-3} : 1$, and $10^{-2} : 1$.

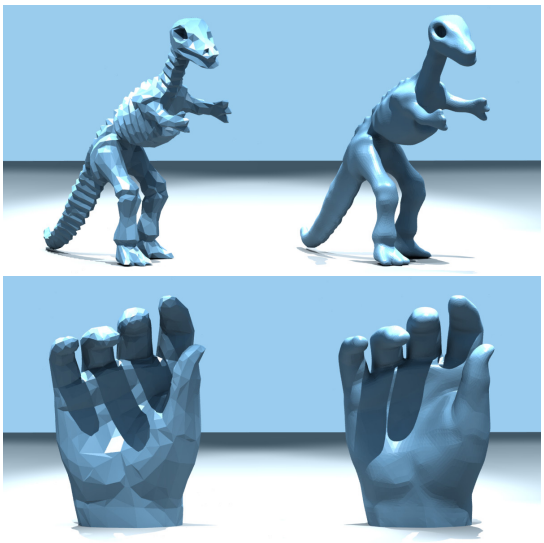


Figure 4: Initial and final frames of Willmore flow applied to smooth *(top)* a 44928 triangle dinosaur and *(bottom)* a 24192 triangle hand at interactive rates. 16 smoothing steps require a total of 7.47s and 4.42s, with one-time factorization costing 8.77s and 5.31s, for the dinosaur and the hand, respectively. Images rendered with flat shading.

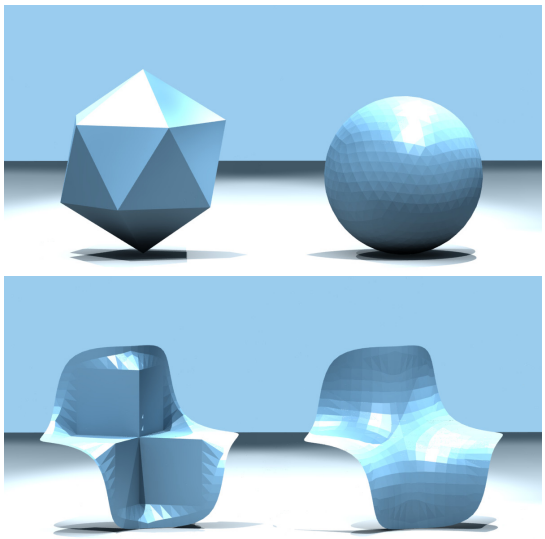


Figure 5: Initial and final frames of Willmore flow applied to solve two tasks posed by Bobenko and Schröder. *(top)* Smoothing a 4-times subdivided icosahedron into a sphere and *(bottom)* a hole filling problem. The sphere converged in 120ms (12ms $\times$ 10 smoothing steps), with 200ms for Hessian prefactorization. The hole filling problem required 640ms after 120ms for prefactorization. Rendered with flat shading.

4 Conclusion and Future Work

The idea presented above is to formulate the bending energy in terms of the Laplace-Beltrami operator endowed with the metric of the deformed surface. We leave as open directions two generalizations of this idea. The first is to replace the Laplace-Beltrami with an arbitrary second-order differential operator. The second is to consider bending energies that are minimized by a nonflat reference surface.

Anisotropic energy A generalization of $\mathbf{H} = \Delta\mathbf{I}$ holds for any directional curvature, κ_i, along a direction v_i: $\kappa_i n = \frac{\mathrm{d}^2}{\mathrm{d}s_i^2} I$, where n is the surface normal, and derivatives are taken with respect to an arc length, s_i, of a curve with the direction v_i at the point of interest. One may interpret this as the one-dimensional case of $\mathbf{H} = \Delta\mathbf{I}$ applied to the intersection curve of the surface and a plane passing through n, v_i. Indeed, mean curvature may be recovered by summing the applications of the above equation evaluated for the two principal curvature directions. In analogy to (1), we may use $\kappa_i\kappa_j = g(\mathrm{d}^2/\mathrm{d}s_i^2 I, \mathrm{d}^2/\mathrm{d}s_j^2 I)$ to define anisotropic energies of the form:

$$E_{aniso}(S) \quad = \quad \int_S c_1\kappa_1^2 + c_2\kappa_1\kappa_2 + c_3\kappa_2^2 \, \mathrm{d}vol \; .$$

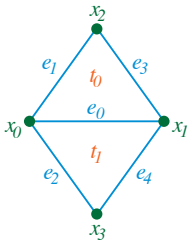
Spontaneous curvature or nonflat reference surface We have observed that our isometric simplification is applicable to any energy obtained as a linear combination of squared directional curvatures, or, in other words, squared entries of the shape operator matrix in a local parametrization.

For nonflat configurations, a typical energy expression remains the same but with the shape operator S replaced with the difference of the shape operators $S - S_0$ where S_0 is the shape operator of the undeformed surface. In this case, the energy is no longer quadratic

in embedding coordinate functions of the deformed surface; one can easily show however that it is cubic. This fact can potentially be used to obtain simpler computations for gradients and Hessians of the discretized bending: while the energy matrix is no longer constant, it is linear in deformed coordinates so can be computed more efficiently than in the general case.

Appendix A

Isometric quadratic hinge model We briefly lay out the complete set of tools required to assemble the hinge-stenciled isometric bending model. The Hessian, $\mathbf{K} = \mathrm{L}^T \mathrm{M}^{-1} \mathrm{L}$, is assembled in the usual manner of finite-element stiffness matrices [Zienkiewicz and Taylor 2000], *i.e.*, by considering contributions from each *local* stiffness matrix, $\mathbf{K}_i$, centered about edge e_i with stencil consisting of the triangle(s), $t_0,\ t_1$, incident to e_i and their incident vertices, $v_0,\ v_1,\ v_2,\ v_3$. With reference to the illustration,



$$\mathbf{A} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix},$$

we build $\mathbf{K}_i = \mathbf{A}^T(\mathrm{L}_i^T \mathrm{M}_i^{-1} \mathrm{L}_i)\mathbf{A}$, where $\mathrm{M}_i = \frac{4}{3}(\mathrm{area}(t_0) + \mathrm{area}(t_1))$ is the *scalar* mass of e_i, $\mathbf{A}$ is the change of basis matrix taking vertex DOFs to edge DOFs, and L_i corresponds to the i^{th} row of the edge-based Laplacian,

$$\mathrm{L}_i = (c_{01} + c_{02} + c_{03} + c_{04}, -c_{01}, -c_{02}, -c_{03}, -c_{04}),$$

where $c_{jk} = 2\cot \angle e_j, e_k$. The *local* energy is obtained by applying the quadratic form $\mathbf{K}_i$ to the vertex vector $(v_0, v_1, v_2, v_3, v_4)$; the *global* (total) energy of the system is obtained by summing over all local contributions corresponding to interior edges e_i. These formulas are derived in the context of nonconforming (edge-based) finite elements. The necessary background is provided in Appendix B.

Appendix B

Discrete Laplacians We provide the necessary background on finite elements. In particular, we give a discussion of conforming and nonconforming elements. The difference is in the degree of freedoms (DOFs): conforming elements have their DOFs at vertices, and nonconforming ones have their DOFs at edges. Moreover, we relate the functional perspective of the linear model for the mean curvature vector to a discrete Laplacian. Finally, we derive a representation of the mean curvature normal as a (3-valued) function.

Conforming setting Our surfaces are discerized by *piecewise linear* elements, consisting of vertices, edges, and triangular faces. Along this line, it is natural to discretize functions over such surfaces in a piecewise linear fashion, with degrees of freedom assigned to vertices. In particular the embedding of a mesh $\mathbf{I} : S \to \mathbb{E}^3$ becomes a (3-valued) PL function. Using PL functions, it naturally follows to discretize the Laplace-Beltrami according to a finite element (FEM) representation. Disregarding boundary vertices, this reads

$$\Delta u(p) = -\int_S \langle \nabla u, \nabla \Phi_p \rangle \mathrm{d}vol,$$

where the Φ_p are Lagrange nodal basis functions (taking value 1 at p and 0 at all other vertices). This is well-known to exactly correspond to the *cotan representation* [Pinkall and Polthier 1993] at each inner vertex p of S:

$$\Delta u(p) = -\frac{1}{2}\sum_q (\cot \alpha_{pq} + \cot \beta_{pq})(u(p) - u(q)),$$

where the sum is taken over all vertices q sharing an edge with p, and α_{pq} and β_{pq} are the opposite angles to the edge $\bar{pq}$ in the two triangles sharing that edge. It is important to note that Δu is no longer a function since differentiating a PL function twice takes values in the space of distributions. Instead, according to (1), $\Delta u(p) = (\Delta u)(\Phi_p)$ represents the value of the *functional* Δu applied to the Lagrange basis function Φ_p. The conforming stiffness matrix is given by

$$\mathrm{L}_{pq} = \int_S \langle \nabla \Phi_p, \nabla \Phi_q \rangle \mathrm{d}vol.$$

Dealing with zero-boundary conditions throughout, the number of rows of (L_{pq}) correspond to inner vertices, its columns to all vertices of the mesh.

Nonconforming setting In the nonconforming setting, the DOFs are associated with edges. In particular, a basis is given by PL mid-edge functions (Φ_e) (so-called Crouzeix-Raviart elements [1973]): To be precise, Φ_e is given by putting the value 1 at the edge midpoint of edge e and the value zero at all other edge midpoints, followed by PL interpolation over triangles. Note carefully that unlike the conforming functions Φ_p, the mid-edge functions Φ_e are *not continuous*. Instead they are merely *mid-edge continuous*. The associated discrete Laplacian is edge-based,

$$\Delta u(e) = -\int_S \langle \nabla u, \nabla \Phi_e \rangle \mathrm{d}vol,$$

where u is some linear combination of nonconforming basis functions associated with *internal edges*. Again, the object Δu is a functional rather than a function. The corresponding stiffness matrix is given as

$$\mathrm{L}_{e_i e_j} = \int_S \langle \nabla \Phi_{e_i}, \nabla \Phi_{e_j} \rangle \mathrm{d}vol,$$

the number of rows of $(\mathrm{L}_{e_i e_j})$ correspond to inner edges, its columns to all edges of the mesh.

Note that the space of conforming elements is a *subspace* of the space of nonconfroming elements. In particular, the (continuous) embedding $\mathbf{I}$ of the mesh can be written in a nonconforming basis; and at inner edges one obtains

$$\mathbf{H}(e) = \Delta \mathbf{I}(e).$$

This proofs the claim that the linear mean curvature vector is *linear in mesh positions*.

Mass matrix and mean curvature function So far we have dealt with the functional viewpoint of the mean curvature normal. This is indispensable for a mathematical analysis, such as treating convergence [Hildebrandt et al. 2005]. However, for bending energy, such as defined in (1), we are forced to find a representation of $\mathbf{H}_d$ as a (3-valued) PL function over S. Again, there is a conforming and a nonconforming version. By definition, this amounts to finding the unique conforming (resp. nonconfroming) function $\mathbf{H}_d$ having zero boundary conditions and for each interior vertex p (resp. each inner edge e) satisfying

$$\mathbf{H}(p) = \int_S \mathbf{H}_d^c \cdot \Phi_p \qquad \left(\text{resp. } \mathbf{H}(e) = \int_S \mathbf{H}_d^{nc} \cdot \Phi_e\right).$$

By linearity this is to say that the L^2 inner product of $\mathbf{H}_d$ with a PL function u that vanishes at the boundary gives the same value as the functional $\mathbf{H}$ applied to u. In basis representation, this L^2 inner product is given by the *mass matrix* M, that is

$$\mathrm{M}_{pq} = \int_S \Phi_p \cdot \Phi_q, \qquad \left(\text{resp. } \mathrm{M}_{e_i e_j} = \int_S \Phi_{e_i} \cdot \Phi_{e_j}\right).$$

In both cases, M is square, symmetric and invertible, its dimension being the number of inner vertices (resp. inner edges). The continuous PL embedding $\mathbf{I}$ can be expressed as

$$\mathbf{I} = \sum_p \vec{\mathbf{I}}_p \cdot \Phi_p \qquad \left(\text{resp. } \mathbf{I} = \sum_e \vec{\mathbf{I}}_e \cdot \Phi_e\right),$$

the sum being taken over all vertices (resp. edges). In the conforming case, the vector $\vec{\mathbf{I}}$ just corresponds to the vertex positions of the mesh. The implementation of the nonconforming version is equally simple: a basis change from conforming to nonconforming elements is simply obtained by a matrix whose rows correspond to the number of edges and whose columns correspond to the number of vertices - the only nonzero entries of this matrix take the value $1/2$, where each edge sees exactly its two boundary vertices, cf. Appendix A. Similarly, the representation of $\mathbf{H}_d$ in terms of basis functions reads

$$\mathbf{H}_d^c = \sum_p (\vec{\mathbf{H}}_d^c)_p \cdot \Phi_p \qquad \left(\text{resp. } \mathbf{H}_d^{nc} = \sum_e (\vec{\mathbf{H}}_d^{nc})_e \cdot \Phi_e\right),$$

summing over all interior vertices (resp. interior edges).

In both cases the result may be written as $\vec{\mathbf{H}}_d = (\mathrm{M}^{-1}\,\mathrm{L})\vec{\mathbf{I}}$, where the L and M are the conforming (resp. nonconforming) stiffness and mass matrix. Since

$$E_d = \int_S \langle \mathbf{H}_d, \mathbf{H}_d \rangle \mathrm{d}vol,$$

this leads to

$$E_d(\mathbf{I}) = (\mathrm{M}^{-1}\,\mathrm{L}\vec{\mathbf{I}})^T\,\mathrm{M}(\mathrm{M}^{-1}\,\mathrm{L}\vec{\mathbf{I}}) = \vec{\mathbf{I}}^T(\mathrm{L}^T\,\mathrm{M}^{-1}\,\mathrm{L})\vec{\mathbf{I}}.$$

The concrete value of this energy will depend on which version of $\mathbf{H}_d$ is used - the conforming or the nonconforming one. In Appendix A we used the *nonconforming* version. The idea of employing nonconforming elements for treating discrete operators has previously been employed for a theory of discrete minimal surfaces [Polthier 2002]. The *conforming* case of bending energy was treated in the exact same manner as outlined above in [Clarenz et al. 2004] (and used therein for Willmore flow).

Appendix C

Laplacians vs. mean curvature We briefly sketch how to move from a notion of a (discrete) Laplacian to a notion of (discrete) mean curvature and vice-versa. As a first observation, *any* model of a (discrete) Laplacian provides a model for a (discrete) mean curvature vector by applying the Laplacian to the postion vector of the surface,

$$\mathbf{H} = \Delta\mathbf{I}.$$

Examples of this observation include a Laplacian which is discretized as the umbrella operator, and $\mathbf{H} = \Delta\mathbf{I}$ is used to translate between the position vector $\mathbf{I}$ and the so called *delta coordinates* $\mathbf{H}$ (cf. Sorkine [2005] and references therein). Vice-versa, since we are primarily interested in applications such as cloth simulation that deal with flat (planar) metrics, we briefly sketch how to recover the Laplacian from the knowledge of the mean curvature operator in the case of small normal displacements u about a *planar domain*. Consider the mean curvature operator H applied to a height field $u \in C^2(\mathbb{R}^2)$. We can regard this as $H : C^2(\mathbb{R}^2) \to C^0(\mathbb{R}^2)$. It is well-known that the *linearization* of this operator is the Laplacian Δu. This can be transferred to the discrete case: given a triangulation of the plane and considering a vector of normal displacements, (u_i), and a discrete operator H acting on (u_i) that computes discrete curvatures at vertices, the linearization of this operator (uniquely) determines a discrete model for a Laplacian.

References

ASCHER, U. M., AND BOXERMAN, E. 2003. On the modified conjugate gradient method in cloth simulation. *Visual Computer 19*, 7-8, 526–531.

BALAY, S., BUSCHELMAN, K., GROPP, W. D., KAUSHIK, D., KNEPLEY, M., MCINNES, L. C., SMITH, B. F., AND ZHANG, H. 2001. PETSc homepage. http://www.mcs.anl.gov/petsc.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH*, 43–54.

BARAFF, D., WITKIN, A., AND KASS, M. 2003. Untangling cloth. *ACM Transactions on Graphics 22*, 3, 862–870.

BLASCKE, W. 1929. *Vorlesungen über Differentialgeometrie III*. Springer.

BOBENKO, A. I., , AND SCHRÖDER, P. 2005. Discrete Willmore flow. *SGP*, 101–110.

BOBENKO, A. I. 2005. A conformal energy for simplicial surfaces. *Combinatorial and Computational Geometry*, 133–143.

BOXERMAN, E., AND ASCHER, U. 2004. Decomposing cloth. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 153–161.

BREEN, D. E., HOUSE, D. H., AND WOZNY, M. J. 1994. Predicting the drape of woven cloth using interacting particles. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, 365–372.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM TOG 21*, 3, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. *SCA '03*, 28–36.

CANHAM, P. 1970. The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell. *Journal of Theoretical Biology 26*, 61–81.

CARIGNAN, M., YANG, Y., THALMANN, N. M., AND THALMANN, D. 1992. Dressing animated synthetic actors with complex deformable clothes. In *Proceedings of SIGGRAPH*, 99–104.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. *ACM Transactions on Graphics 21*, 3 (July), 604–611.

CHOI, K.-J., AND KO, H.-S. 2005. Research problems in clothing simulation. *CAD 37*, 6, 585–592.

CIARLET, P. 2000. *Mathematical Elasticity, Vol III*. North-Holland.

CIRAK, F., ORTIZ, M., AND SCHRÖDER, P. 2000. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg. 47*, 12, 2039–2072.

CLARENZ, U., DIEWALD, U., DZIUK, G., RUMPF, M., AND RUSU, R. 2004. A finite element method for surface restoration with smooth boundary conditions. *CAGD*, 427–445.

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted Delaunay triangulations and normal cycle. *SoCG 2003*, 312–321.

CROUZEIX, M., AND RAVIART, P. A. 1973. Conforming and nonconforming finite elements for solving stationary Stokes equations. *RAIRO Anal. Numer. 7*, 33–76.

DECKELNICK, K., DZIUK, G., AND ELLIOTT, C. M. 2005. Fully discrete semi-implicit second order splitting for anisotropic surface diffusion of graphs. *SINUM 43*, 1112–1138.

DESBRUN, M., SCHRODER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. *Proceedings. Graphics Interface '99*, 1–8.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. *SIGGRAPH'99 Conference Proceedings*, 317–324.

ETZMUSS, O., EBERHARDT, B., AND HAUTH, M. 2000. Implicit-explicit schemes for fast animation with particle systems. In *Computer Animation and Simulation 2000*, 138–151.

ETZMUSS, O., KECKEISEN, M., AND STRASSER, W. 2003. A fast finite element solution for cloth modelling. *Proceedings 11th Pacific Conference on Computer Graphics and Applications*, 244–251.

FEYNMAN, C. 1986. *Modeling the Appearance of Cloth*. MSc thesis, MIT.

GOVINDARAJU, N. K., KNOTT, D., JAIN, N., KABUL, I., TAMSTORF, R., GAYLE, R., LIN, M. C., AND MANOCHA, D. 2005. Interactive collision detection between deformable models using chromatic decomposition. *ACM Transactions on Graphics 24*, 3 (Aug.), 991–999.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. *SCA '03*, 62–67.

HAUMANN, R. 1987. Modeling the physical behavior of flexible objects. In *Topics in Physically-based Modeling,* Eds. Barr, Barrel, Haumann, Kass, Platt, Terzopoulos, and Witkin*, SIGGRAPH Course Notes*.

HAUTH, M., AND ETZMUSS, O. 2001. A high performance solver for the animation of deformable objects using advanced numerical methods. *Computer Graphics Forum 20*, 3, 319–328.

HAUTH, M., ETZMUSS, O., AND STRASSER, W. 2003. Analysis of numerical methods for the simulation of deformable models. *Visual Computer 19*, 7-8, 581–600.

HAUTH, M. 2004. *Visual Simulation of Deformable Models*. PhD thesis, University of Tübingen.

HELFRICH, W. 1973. Elastic properties of lipid bilayers: Theory and possible experiments. *Zeitschrift fr Naturforschung Teil C 28*, 693–703.

HILDEBRANDT, K., AND POLTHIER, K. 2004. Anisotropic filtering of non-linear surface features. *CGF 23*, 3, 391–400.

HILDEBRANDT, K., POLTHIER, K., AND WARDETZKY, M. 2005. On the convergence of metric and geometric properties of polyhedral surfaces. ZIB-Report ZR-05-24.

HOUSE, D. H., AND BREEN, D. E., Eds. 2000. *Cloth modeling and animation*.

HSU, L., KUSNER, R., AND SULLIVAN, J. 1992. Minimizing the squared mean curvature integral for surfaces in space forms. *Experiment. Math. 1*, 3, 191–207.

HUGHES, T. J. R. 1987. *Finite Element Method - Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, Englewood Cliffs.

KECKEISEN, M., KIMMERLE, S., THOMASZEWSKI, B., AND WACKER, M. 2004. Modelling Effects of Wind Fields in Cloth Animations. In *Journal of WSCG*, vol. Vol. 12, 205–212.

KLOSOWSKI, J. T., HELD, M., MITCHELL, J. S. B., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE TVCG 4*, 1 (January-March), 21–36.

MAGNENAT-THALMANN, N., AND VOLINO, P. 2005. From early draping to haute couture models: 20 years of research. *The Visual Computer 21*, 8-10, 506–519.

MERCAT, C. 2001. Discrete Riemann surfaces and the Ising model. *Communications in Mathematical Physics 218*, 1, 177–216.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, H.-C. Hege and K. Polthier, Eds. Springer-Verlag, Heidelberg, 113–134.

NG, H. N., AND GRIMSDALE, R. L. 1996. Computer graphics techniques for modeling cloth. *IEEE CG&A 16*, 5 (Sept.), 28–41.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experim. Math. 2*, 15–36.

POLTHIER, K., 2002. Polyhedral surfaces of constant mean curvature. Habilitationsschrift, TU Berlin.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, 147–154.

SCHENK, O., AND GÄRTNER, K. 2004. Solving unsymmetric sparse systems of linear equations with PARDISO. *Journal of Future Generation Computer Systems 20*, 3, 475–487.

SCHNEIDER, R., AND KOBBELT, L. 2001. Geometric fairing of irregular meshes for free-from surface design. *CAGD 18*, 359–379.

SORKINE, O. 2005. Laplacian mesh processing. *Eurographics STAR - State of The Art Report*, 53–70.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 181–190.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proceedings of SIGGRAPH*, 205–214.

THOMASZEWSKI, B., AND WACKER, M. 2006. Bending Models for Thin Flexible Objects. In *WSCG Short Communication proceedings*.

VOLINO, P., COURCHESNE, M., AND THALMANN, N. M. 1995. Versatile and efficient techniques for simulating cloth and other deformable objects. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 137–144.

WHITE, J. H. 2000. A global invariant of conformal mappings in space. *Proceedings of the American Mathematical Society 38*, 162–164.

WILLMORE, T. J. 2000. Surfaces in conformal geometry. *Annals of Global Analysis and Geometry 18*, 255–264.

YOSHIZAWA, S., AND BELYAEV, A. 2002. Fair triangle mesh generation via discrete elastica. In *GMP2002*, IEEE, 119–123.

ZHU, H., JIN, X., FENG, J., AND PENG, Q. 2004. Survey on cloth animation. *Journal of Computer Aided Design & Computer Graphics 16*, 5, 613–618.

ZIENKIEWICZ, O. C., AND TAYLOR, R. L. 2000. *The finite element method: The basis*, 5th ed., vol. 1. Butterworth and Heinemann.

Straightest Geodesics on Polyhedral Surfaces*

Konrad Polthier†
Freie Universität Berlin

Markus Schmies‡
Technische Universität Berlin

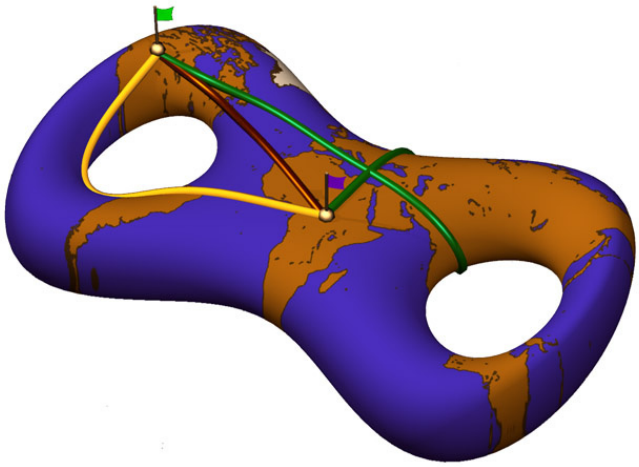


Figure 1: Geodesics on a surface and multiple locally shortest connections. Straightest geodesics are unique solutions to the initial value problem for geodesics on polyhedral surface.

Abstract

Geodesic curves are the fundamental concept in geometry to generalize the idea of straight lines to curved surfaces and arbitrary manifolds. On polyhedral surfaces we introduce the notion of discrete geodesic curvature of curves and define straightest geodesics. This allows a unique solution of the initial value problem for geodesics, and therefore a unique movement in a given tangential direction, a property not available in the well-known concept of locally shortest geodesics.

An immediate application is the definition of parallel translation of vectors and a discrete Runge-Kutta method for the integration of vector fields on polyhedral surfaces. Our definitions only use intrinsic geometric properties of the polyhedral surface without reference to the underlying discrete triangulation of the surface or to an ambient space.

Keywords: discrete geodesics, straightest geodesics, shortest geodesics, polyhedral surfaces, intrinsic curves, parallel translation, curvature.

1 Introduction

Geodesics on smooth surfaces are the straightest and locally shortest curves. They generalize the concept of Euclidean straight lines and play a fundamental role in the study of smoothly curved manifolds. Two basic properties are responsible for their importance: first, geodesics solve the initial value problem which states, that from any point of a manifold there starts a unique geodesic in any direction. Second, the length minimization property provides a solution of the boundary value problem of connecting two given points on a manifold with a locally shortest curve. On smooth surfaces geodesics possess both properties, in contrast to the situation on polyhedral surfaces.

The aim of this paper is to define *straightest curves* on two-dimensional polyhedral surfaces, as opposed to the concepts of locally shortest and quasi-geodesics. Such straightest geodesics will uniquely solve the initial value problem on polyhedral surfaces, and therefore allow to move uniquely on a polyhedral surface in a given direction along a straightest geodesic until the boundary is reached, a property not available for locally shortest geodesics. An application of straightest geodesics is the definition of parallel translation of vectors and higher order numerical integration methods for tangential vector fields. This allows the extension of Runge Kutta methods to polyhedral surfaces.

We consider polyhedral surfaces as two-dimensional simplicial complexes consisting of triangles. Each triangle has a flat metric and the common edge of two neighbouring triangles has the same length in both triangles. The definition of a metric on the polyhedral surface only requires the specification of edge lengths and does not refer to an immersion of the surface in an ambient space. This intrinsic approach allows the definition of straightest geodesics, discrete geodesic curvature, vector fields, and parallel translation of vectors in terms of the geometric data of the surface, such as edge lengths, triangle angles, and discrete curvature properties.

Geodesics on polyhedral surfaces were intensively studied using different definitions. The Russian school of A.D. Alexandrov [Aleksandrov and Zalgaller 1967] defines geodesics on polyhedral

*Originally published in: Mathematical Visualization, H.C. Hege and K. Polthier (Eds.), Springer Verlag 1997. Reprinted in modified form with kind permission of the publisher.

†e-mail: polthier@mi.fu-berlin.de

‡e-mail:schmies@math.tu-berlin.de

surfaces as locally shortest curves which leads to important implications in the study of non-regular and regular differential geometry. But shortest geodesics cannot be extended as shortest curves across a spherical vertex with positive Gauß curvature as, for example, the vertex of a cube. Beyond a hyperbolic vertex with negative Gauß curvature there even exists a continuum of extensions. Therefore, shortest geodesics fail to solve the initial value problem for geodesics at vertices of a polyhedral surface.

A.D. Alexandrov also introduced the concept of quasi-geodesics which are limit curves of geodesics on a family of converging smooth surfaces. They form a wider class than shortest geodesics and were amongst others studied by Pogorelov [Pogorelov 1952] on convex polyhedral surfaces. A quasi-geodesic through a spherical vertex is a curve with right and left angles both less than π, and therefore an inbound direction has multiple extensions.

Shortest geodesics appear in many practical applications. For example, the optimal movement of a robot should have minimal length in its parameter space. Such discrete minimization problems are studied in computational geometry, see for example Dijkstra [Dijkstra 1959], Sharir and Schorr [Sharir and Schorr 1986], and Mitchell et.al. [Mitchell et al. 1987] for efficient algorithms on the computation of the shortest path in graphs and in polyhedral spaces.

Our paper starts in section 2 with a review of geodesics on smooth surfaces, especially since some of their properties differ from those of geodesics on polyhedral surfaces. In section 3 we will introduce polyhedral surfaces as metric spaces and recall basic facts. Straightest geodesics are defined in section 4 and discussed as solutions of the initial value problem. In section 5 we imbed the notion of straightest lines into the concept of discrete geodesic curvature of arbitrary curves on polyhedral surfaces. This general setting is more appropriate for our later discussions, and straightest geodesics turn out to be those class of curves with vanishing discrete geodesic curvature. As a validation of the definition we prove the Gauß-Bonnet theorem using our notion of discrete geodesic curvature. In section 6 we apply the concept to the definition of parallel translation of tangential vector fields and in section 7 we generalize Runge Kutta methods to the numerical integration of ordinary differential equations on polyhedral surfaces.

Applications of this paper are given in the video *Geodesics and Waves* [Polthier et al. 1997]. The numerics were developed within the visualization environment OORANGE [Gunn et al. 1997].

2 Review of Geodesics on Smooth Surfaces

Geodesics on smooth surfaces can be characterized by different equivalent properties. The generalized properties on polyhedral surfaces will no longer be equivalent and lead to different classes of discrete geodesics. The following material can be found in any introductory text book on differential geometry, see for example [Carmo 1976].

Let M be a smooth surface and $\gamma : I = [a,b] \to M$ a curve parametrized over an interval I. To avoid accelerations tangential to the curve we assume arc length parametrization, i.e. the tangent vector has constant length $|\gamma'| = 1$. A curve γ is called *locally shortest* if it is a critical point of the length functional $L(\gamma_{|[a,b]}) := length(\gamma_{|[a,b]})$ with respect to variations tangential to M which leave the endpoints fixed. Formally, if $\phi : I \to T_\gamma M$ is a tangential vector field along γ with $\phi(a) = 0$ and $\phi(b) = 0$, then we have $\frac{\partial}{\partial \varepsilon} L(\gamma + \varepsilon\phi)_{|\varepsilon=0} = 0$. A critical point of the length functional is usually not a global minimizer compared to curves with the same endpoints.

On smooth manifolds the length minimizing property of geodesics can be reformulated as an ordinary differential equation for γ, namely $\gamma''(s)^{\tan M} = 0$, the Euler-Lagrange equations of the variational problem.

The curvature $\kappa(s) = |\gamma''(s)|$ of a curve measures the infinitesimal turning of the tangent vector at every point $\gamma(s)$. For curves γ on surfaces $M \subset \mathbb{R}^3$, the curvature can be decomposed into the curve's bending in the normal direction n of the surface and its bending in the tangent space in direction of the binormal b. This decomposition leads to the definition of the geodesic curvature κ_g and the normal curvature κ_n of a curve:

$$\begin{aligned} \kappa^2(s) &= \left|\gamma''(s)\right|^2 \\ &= \left|\gamma''(s)^{\tan M}\right|^2 + \left|\gamma''(s)^{\text{nor M}}\right|^2 \\ &= \kappa_g^2(s) + \kappa_n^2(s). \end{aligned} \tag{1}$$

The geodesic curvature κ_g of a curve γ measures the tangential acceleration. If $\kappa_g = 0$ then the curve varies up to second order only in direction of the surface normal, therefore it is a *straightest curve* on the surface. The normal curvature κ_n is related with the bending of the surface itself and can be neglected from an intrinsic point of view.

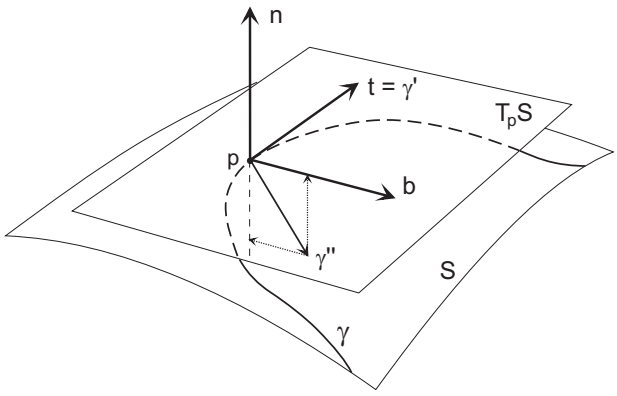


Figure 2: Geodesic and normal curvature of a curve on a smooth surface.

Summarizing, one characterizes smooth geodesics as follows:

Definition 1 *Let M be a smooth two-dimensional surface. A smooth curve $\gamma : I \to M$ with $|\gamma'| = 1$ is a* geodesic *if one of the equivalent properties holds:*

1. *γ is a locally shortest curve.*
2. *γ'' is parallel to the surface normal, i.e.*

$$\gamma''(s)^{tan\ M} = 0. \tag{2}$$

3. *γ has vanishing geodesic curvature $\kappa_g = 0$.*

In section 4 we will consider geodesics on polyhedral surfaces and notice that the polygonal equivalents of the above properties lead to different notions of discrete geodesics.

The boundary value problem for geodesics has a solution in every homotopy class and is usually not unique. On the other hand, we have a unique solution for the initial value problem derived from equation (2):

Lemma 1 *Let M be a smooth manifold. Then for any point $p \in \mathring{M}$ in the interior of M and any tangent direction $v \in T_pM$ the initial value problem*

$$\begin{aligned} \gamma''(s)^{tan\,M} &= 0 \\ \gamma(0) &= p \\ \gamma'(0) &= v \end{aligned} \tag{3}$$

has a unique solution $\gamma : [0,\ell) \to M$, where ℓ is the length of the maximal interval of existence.

3 Curvature of Polyhedral Surfaces

In this section we review some facts on the geometry of polyhedral surfaces. Basic references are for example A.D. Alexandrov and Zalgaller [Aleksandrov and Zalgaller 1967] and Reshetnyak [Reshetnyak 1993]. For simplification we restrict ourselves to two-dimensional surfaces consisting of planar triangles. A topological triangle f in a two-dimensional manifold S is a simple domain $f \subset S$ whose boundary is split by three vertices into three edges with no common interior points.

Definition 2 *A* polyhedral surface *S is a two-dimensional manifold (with boundary) consisting of a finite or denumerable set F of topological triangles and an intrinsic metric $\rho(X,Y)$ such that*

1. *Any point $p \in S$ lies in at least one triangle $f \in F$.*
2. *Each point $p \in S$ has a neighbourhood that intersects only finitely many triangles $f \in F$.*
3. *The intersection of any two non-identical triangles $g, h \in F$ is either empty, or consists of a common vertex, or of a simple arc that is an edge of each of the two triangles.*
4. *The intrinsic metric ρ is flat on each triangle, i.e. each triangle is isometric to a triangle in $\mathbb{R}^2$.*

Remark 1 *Most of our considerations apply to a more general class of length spaces. Each face may have an arbitrary metric as long as the metrics of two adjacent faces are compatible, i.e. if the common edge has the same length in both faces, and the triangle inequality holds.*

Let $\gamma \subset S$ be a curve whose segments on each face are rectifiable. Then the *length of* γ is well-defined and given by

$$Length(\gamma) = \sum_{f \in F} Length(\gamma_{|f}). \tag{4}$$

The neighbourhood of a vertex is isometric to a cone and is characterized by the total vertex angle:

Definition 3 *Let S be a polyhedral surface and $v \in S$ a vertex. Let $F = \{f_1,..., f_m\}$ be the set of faces containing p as a vertex, and θ_i be the interior angle of the face f_i at the vertex p, compare figure 3. Then the* total vertex angle *$\theta(p)$ is given by*

$$\theta(p) = \sum_{i=1}^{m} \theta_i(p). \tag{5}$$

Interior points p of a face or of an open edge have a neighbourhood which is isometric to a planar Euclidean domain and we define $\theta(p) = 2\pi$.

All points of a polyhedral surface can be classified according to the sign of the *vertex angle excess* $2\pi - \theta(p)$:

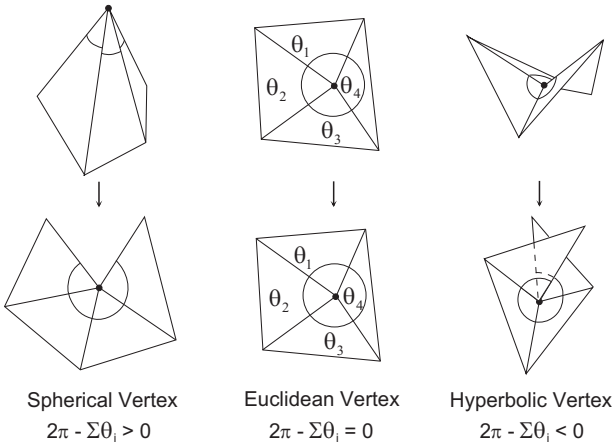


Figure 3: Classification of vertices on a polyhedral surface according to the excess of the vertex angle, and their unfolding in a planar domain.

Definition 4 *A vertex p of a polyhedral surface S with total vertex angle $\theta(p)$ is called* Euclidean, spherical, *or* hyperbolic *if its angle excess $2\pi - \theta(p)$ is $= 0$, > 0, or < 0. Respectively, interior points of a face or of an open edge are Euclidean.*

The neighbourhood of a vertex can be isometrically unfolded to a (partial or multiple) covering of a part of the Euclidean plane. There exist three situations as shown in figure 3 which metrically characterize the vertex. For example, the tip of a convex cone is a spherical vertex and a saddle point is hyperbolic. On the other hand, a spherical vertex need not be the tip of a convex cone. The isometric unfolding of sets of a polyhedral surface is a common procedure to study the geometry.

The Gauß curvature of a general manifold is a central intrinsic property of the geometry and can be computed in terms of the metric. It influences, for example, the parallel translation of vectors along curves. The Gauß curvature of a piecewise linear surface is concentrated at the isolated vertices since all other points on the surface have a neighbourhood isometric to a planar Euclidean domain with zero curvature. It is therefore more appropriate to work with the concept of total Gauß curvature.

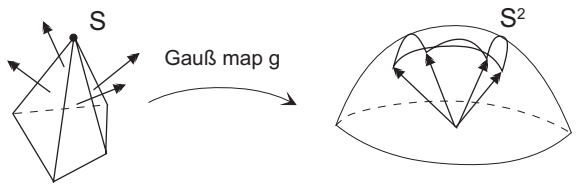


Figure 4: The Gauß map assigns to each point $p \in S$ of a surface its normal vector $n(p) \in \mathbb{S}^2$. At edges and vertices of a polyhedral surface the image of the Gauß map is the spherical convex hull of the normal vectors of adjacent faces.

Using following definition the curvature can be measured directly in metrical terms of the surface S.

Definition 5 *The* (total) Gauß curvature *$K(p)$ of a vertex p on a*

polyhedral surface S is defined as the vertex angle excess

$$\begin{aligned} K(p) &= 2\pi - \theta(p) \\ &= 2\pi - \sum_{i=1}^{m} \theta_i(p). \end{aligned} \tag{6}$$

An immediate consequence is that Euclidean vertices have curvature $K = 0$, spherical vertices have $K > 0$, and hyperbolic vertices have $K < 0$. For example, the vertices of a cube each have Gauß curvature $\frac{\pi}{2}$.

For a smooth surface S embedded into $\mathbb{R}^3$ the curvature measures the infinitesimal turn of the normal vector of the surface and can be defined via the *Gauß map* $g : S \to \mathbb{S}^2$ which assigns to each point p on a surface S its normal vector $n(p)$, see figure 4. The total Gauß curvature $K(\Omega)$ of a domain $\Omega \subset S$ is given by the area of its spherical image: $K(\Omega) = area\ g(\Omega)$. It is an easy calculation to show that this relation also holds for the Gauß curvature of a vertex on a polyhedral surface.

4 Discrete Straightest Geodesics

Our approach to discrete geodesics on polyhedral surfaces concentrates on the property of a curve to be straightest rather than locally shortest. Both properties are equivalent for geodesics on smooth surfaces, as mentioned in section 2, but locally shortest curves on polygonal surfaces do not allow a unique extension, for example, beyond spherical vertices of the surface. The original motivation for our study was to define a unique way to move straight ahead in a given direction on a polyhedral surface. Applications are, for example, the tracing of moving particles restricted to flow along a polyhedral surface, the solution of initial value problems on polyhedral surfaces related with given tangential vector fields, and the intrinsic generalization of numerical algorithms for ordinary differential equations to polygonal surfaces.

The concept of shortest geodesics in graphs, polyhedral manifolds, and more general length spaces has been studied by a number of authors in different fields, see for example [Dijkstra 1959][Mitchell et al. 1987][Aleksandrov and Zalgaller 1967][Alexander and Bishop 1996]. For our applications this concept has a central missing property, namely, the initial value problem for geodesics is not uniquely solvable and in some cases has no solution: first, no shortest geodesics can be extended through a spherical vertex since it could be shortened by moving off the corner, and second, there exists a family of possible extensions of a geodesic as a shortest curve through a hyperbolic vertex: every extension with curve angles $\theta_l, \theta_r \in [\pi, \theta - \pi]$ is locally shortest where θ is the total vertex angle. See lemma 2 and figure 5.

Quasi-geodesics are a different approach which was introduced by A.D. Alexandrov (see the references to the original Russian literature in [Aleksandrov and Zalgaller 1967]) and investigated on convex surfaces by Pogorelov [Pogorelov 1952] and others. They appear as limit sets of smooth geodesics when smooth surfaces approximate, for example, a polyhedral surface. On polyhedral surfaces quasi-geodesics are characterized by their fulfillment of the inequation $|\pi - \theta_l| + |\pi - \theta_r| - |2\pi - \theta_l - \theta_r| \geq 0$ at each point, where θ_l and θ_r are the two angles of the curve, and $\theta_l + \theta_r = \theta$ is the total vertex angle of the point. Compare figure 5 for the notation. At hyperbolic vertices with $\theta > 2\pi$ the definition is identical to that for shortest geodesics, while at spherical vertices with $\theta < 2\pi$ curves with $\pi - \theta_l \geq 0$ and $\pi - \theta_r \geq 0$ are quasi-geodesics.

In the following definition we introduce straightest geodesics which are a new class of discrete geodesics on polyhedral surfaces. This class has a non-empty intersection with the set of shortest geodesics and is a subset of quasi-geodesics.

Definition 6 *Let S be a polyhedral surface and $\gamma \subset S$ a curve. Then γ is a* straightest geodesic *on S if for each point $p \in \gamma$ the left and right curve angles θ_l and θ_r at p are equal, see figure 5.*

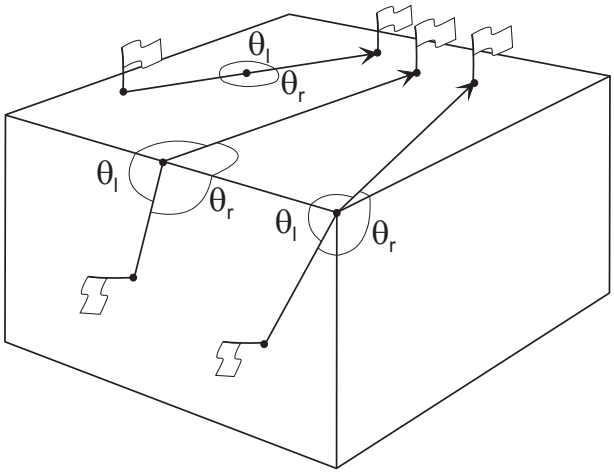


Figure 5: Notion of left and right curve angles θ_l and θ_r with $\theta_l + \theta_r = \theta$.

A straightest geodesic in the interior of a face is locally a straight line, and across an edge it has equal angles on opposite sides. The definition of straightest geodesics on faces and through edges is identical to the concept of shortest geodesics but at vertices the concepts differ. Our definition fits into the more general discussion of discrete geodesic curvature of curves on a polyhedral surface, this will be discussed in detail in section 5.

The following theorem proves the unique solvability of the initial value problem for straightest geodesics. To state the problem we start with the notion of a tangent vector on a polyhedral surface:

Definition 7 *Let S be a polyhedral surface and $p \in S$ a point. A* polyhedral tangent vector *v with base point p lies in the plane of an adjacent face and locally points into the face. The polyhedral tangent space T_pS consists of all polyhedral tangent vectors at p.*

We remark, that the polyhedral tangent bundle TS can be equipped with the structure of a topological vector bundle by introducing normalized angles as in definition 10, but do not pursue this property. Instead, we use the fact that polyhedral tangent vectors are characterized solely by intrinsic properties of the geometry rather than by reference to an ambient space.

Theorem 1 (Discrete Initial Value Problem) *Let S be a polyhedral surface and $p \in S$ a point with polyhedral tangent vector $v \in T_pS$. Then there exists a unique straightest geodesic γ with*

$$\begin{aligned} \gamma(0) &= p \\ \gamma'(0) &= v, \end{aligned} \tag{7}$$

and the geodesic extends to the boundary of S.

Proof. There exists a face f of S which contains the initial point p and, for a small number $\varepsilon > 0$, the straight line $\gamma(t) := p + tv$ with $t \in [0, \varepsilon)$. γ is a straightest geodesic and a solution of equation 7. If we extend γ beyond the small interval and γ reaches an edge or a vertex of S for larger values of t then definition 6 of straightest

geodesics uniquely defines how to extend γ beyond the edge or vertex. That is to proceed in that direction for which the left and right curve angles of γ at the vertex are equal.■

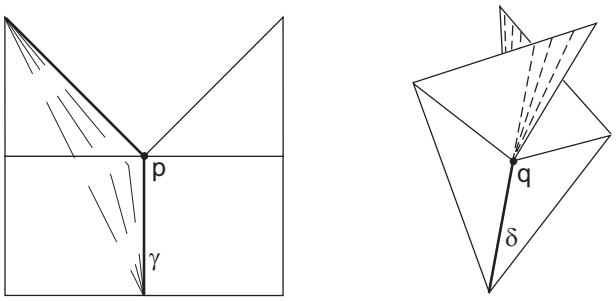


Figure 6: Locally shortest geodesics cannot be extended through a spherical vertex p and there exist multiple continuations at a hyperbolic vertex q.

The concepts of straightest and shortest geodesics differ on polyhedral surfaces. For example, as shown in the following lemma, the theorem above does not hold for locally shortest geodesics approaching a spherical or hyperbolic vertex. As long as a geodesic γ does not meet a vertex of a polyhedral surface both concepts are equal and γ is both, straightest and locally shortest. The following lemma comprehends the differences:

Lemma 2 *On a polyhedral surface S the concepts of straightest and locally shortest geodesics differ in the following way (see figure 6):*

1. *A geodesic γ containing no surface vertex is both straightest and locally shortest.*
2. *A straightest geodesic γ through a spherical vertex is not locally shortest.*
3. *There exists a family of shortest geodesics γ_θ through a hyperbolic vertex with the same inbound direction. Only one of the shortest geodesics extends the inbound direction as straightest geodesic.*
4. *Straightest geodesics do not solve the boundary value problem for geodesics since there exist shadow regions in the neighbourhood of a hyperbolic vertex where two points cannot be joined by a straightest geodesic.*

Proof. Ad 1.) We unfold the faces met by the geodesic to an isometric strip of faces in the Euclidean plane. The geodesic γ is unfolded to a Euclidean straight line in the interior of the strip which is locally shortest and fulfills the angle condition of definition 6.

Ad 2.) Let γ be a straightest geodesic through a spherical vertex with curvature $K > 0$. We unfold the adjacent faces to a planar domain by cutting along the outbound direction of γ. The image of γ in the plane has a corner at the vertex with curve angle $\frac{\theta}{2} = \pi - \frac{K}{2} < \pi$ at both sides. Therefore, γ is not locally shortest since it can be shortened by smoothing the corner in either direction as shown on the left in figure 6.

Ad 3.) A hyperbolic vertex has curvature $K < 0$. Let γ_0 be the unique straightest geodesic though the vertex which extends the inbound direction. We unfold the adjacent faces to a planar domain by cutting along the outbound direction of γ_0, then γ_0 has a curve angle $\frac{\theta}{2} = \pi - \frac{K}{2} > \pi$ at both sides of the corner. Assume a curve with the same inbound but a different outbound direction. Whenever both angles between the inbound and outbound direction are bigger than or equal to π, we cannot locally shorten the curve. Therefore all such curves are locally shortest.■

5 Discrete Geodesic Curvature

We define the notion of geodesic curvature of curves on piecewise linear surfaces with the later aim of defining parallel translation of vectors along arbitrary curves. Additionally, vanishing geodesic curvature should characterize straightest geodesics. The definition should comply with the known (total) curvature of polygons in the Euclidean plane, and the Gauß-Bonnet equation should hold. In the following, we assume curves to be smooth on faces and to have well-defined polyhedral tangent directions at the edges and vertices of the surface. Similar to the discrete Gauß curvature for surfaces, the discrete geodesic curvature is the equivalent of the total geodesic curvature of smooth surfaces.

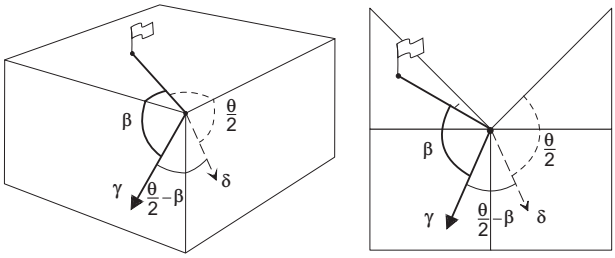


Figure 7: The discrete geodesic curvature of a curve γ is the normalized angle between γ and a discrete straightest geodesic δ.

Definition 8 *Let γ be a curve on a polyhedral surface S. Let θ be the total vertex angle and β one of the two curve angles of γ at p. Then the* discrete geodesic curvature *κ_g of γ at p is given by*

$$\kappa_g = \frac{2\pi}{\theta}\left(\frac{\theta}{2} - \beta\right). \tag{8}$$

Choosing the other curve angle $\beta' = \theta - \beta$ changes the sign of κ_g.

Using the notion of discrete geodesic curvature we obtain a new characterization of straightest geodesics since they bisect the total vertex angle θ, i.e. $\beta = \frac{\theta}{2}$:

Lemma 3 *Let S be a polyhedral surface and $\gamma \subset S$ a curve. Then γ is a straightest geodesic if and only if γ has vanishing discrete geodesic curvature.*

Remark 2 *1.) Let γ be a polygon in the Euclidean plane S and $p \in \gamma$ be a vertex with curve angle β. Then the discrete geodesic curvature equals the total curvature of γ at p defined by the spherical image of its normal vectors.*
2.) Let S be a polyhedral surface and let γ touch a vertex $p \in S$, i.e. $\beta = 0$. Then the geodesic curvature of γ at p is $\kappa_g = \pi$, i.e. it can be measured in the Euclidean face and without influence of the vertex angle θ at p.
3.) Shortest geodesics through a hyperbolic vertex with vertex angle $\theta > 2\pi$ have geodesic curvatures κ_g in the interval $\left[-\pi(1-\frac{2\pi}{\theta}), \pi(1-\frac{2\pi}{\theta})\right]$.

Straightest geodesics are natural generalizations of straight lines in Euclidean space. For example, geodesic triangles on surfaces can be defined as simply connected regions bounded by three straightest segments, and geodesic polygons as piecewise straightest curves.

The Gauß-Bonnet theorem relates the topology and geometry of surfaces. It is a remarkable consequence of the definition of discrete geodesic curvature that this fundamental theorem still holds. In fact, one can even reverse the arguments and derive our formula for geodesic curvature from the requirement that the equation of Gauß-Bonnet should hold.

There have been different formulations of the Gauß-Bonnet theorem on polyhedral surfaces, each expressing the Euler characteristic $\chi(\Omega)$ of a domain Ω using different curvature terms. For example, Reshetnyak [Reshetnyak 1993] only uses the Gauß curvature of interior vertices and defines the curvature of the boundary curve by $\kappa = \pi - \beta$, where β is the inner curve angle of the boundary. We refine this approach and split his definition of boundary curvature in two components, a geodesic curvature of the boundary curve and a partial Gauß curvature, where the vertices $p \in \partial\Omega$ contribute to the total Gauß curvature of Ω. The following natural definition determines the contribution of boundary vertices to the total Gauß curvature of Ω. The contribution is proportional to the curve angle β:

Definition 9 *Let $\Omega \subset S$ be a domain on a polyhedral surface with boundary $\Gamma = \partial\Omega$. If $\theta(p)$ is the total vertex angle and $\beta(p)$ the inner curve angle at a vertex $p \in \Gamma$, then the* partial Gauß curvature *$K_{|\Omega}$ of Ω at p is proportional to β:*

$$K_{|\Omega}(p) = \frac{\beta}{\theta} K(p). \tag{9}$$

If $\beta = 0$ then the vertex has no partial Gauß curvature, and $\beta = \theta$ leads to a full contribution of the total Gauß curvature $K = 2\pi - \theta$ to Ω. In the following we simplify the notation by omitting the subindex $_{|\Omega}$.

Theorem 2 (Discrete Gauss-Bonnet) *Let S be a polyhedral surface and $\Omega \subset S$ a domain with boundary curve Γ and Euler characteristic $\chi(\Omega)$. Then the equation*

$$\sum_{p \in \Omega} K(p) + \kappa_g(\Gamma) = 2\pi\chi(\Omega) \tag{10}$$

holds where the total Gauß curvature of Ω includes the partial Gauß curvature at boundary points. If Γ is piecewise straightest then the total geodesic curvature is the sum of the geodesic curvature at the vertices of Γ.

Proof. For the proof we use the version

$$\sum_{p \in \mathring{\Omega}} K(p) + \sum_{p \in \Gamma} (\pi - \beta(p)) = 2\pi\chi(\Omega)$$

proved by Reshetnyak [Reshetnyak 1993] where only interior vertices of Ω contribute to the total Gauß curvature. Let $p \in \Gamma$ be a boundary vertex, then we have the splitting

$$K_{|\Omega}(p) - \kappa_g(p) = \pi - \beta(p)$$

which proves the assumption.■

6 Parallel Translation of Vectors

Numerical methods for the integration of ordinary differential equations rely on the possibility for parallel translation of vectors in the Euclidean plane. For example, higher order Runge-Kutta methods do several trial shots in a single integration step to compute the final shooting direction and translate direction vectors to their current positions. When transferring such integration methods to surfaces, which are not described by local charts, it is necessary to compare vectors with different base points on the curved surface.

We use the notion of polyhedral tangent vectors formulated in definition 7 and define an intrinsic version of parallel translation of vectors which uses no ambient space as reference. We start with two definitions of angles:

Definition 10 *Let S be a polyhedral surface and $p \in S$ a point with total vertex angle θ. The* Euclidean angle $\measuredangle(v,w)$ *between tangent vectors $v, w \in T_pS$ is the angle between corresponding vectors in the unfolded neighbourhood of p measured in $\mathbb{R}^2$, i.e. $\measuredangle(v,w) \in \left[-\frac{\theta}{2}, \frac{\theta}{2}\right]$. The* normalized angle $\alpha(v,w)$ *is obtained by scaling:*

$$\alpha(v,w) := \frac{2\pi}{\theta} \measuredangle(v,w). \tag{11}$$

The normalized and Euclidean angles are identical at points which are not vertices of the surface. In practical applications one measures the Euclidean angle at first, and then uses the normalized angle to avoid case distinctions at vertices of the surface as seen, for example, in the following lemma:

Lemma 4 *Let Δ be a geodesic triangle on a polyhedral surface S whose edges are straightest segments. If α_1, α_2, and α_3 are the normalized angles of Δ then we have*

$$\alpha_1 + \alpha_2 + \alpha_3 - \pi = \int_\Delta K. \tag{12}$$

Proof. Denote the Euclidean angles of Δ with β_i and the vertex angles with θ_i . Then the geodesic curvature of the boundary of Δ at one of its vertices is given by

$$\kappa_g = \frac{2\pi}{\theta}\left(\frac{\theta}{2} - \beta\right) = \pi - \alpha \tag{13}$$

and the assumption follows directly from the discrete Gauß-Bonnet equation (10).■

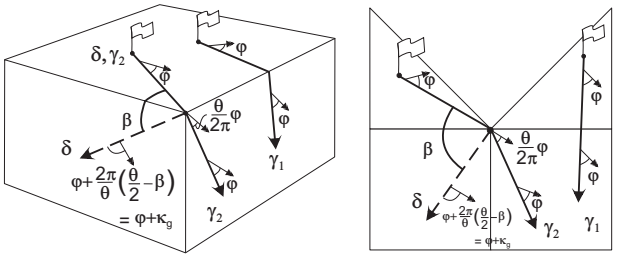


Figure 8: Parallel translation of vectors along straightest geodesics γ_1, γ_2 and an arbitrary curve δ.

On polyhedral surfaces we can use the concept of straightest geodesics and normalized angles to define parallel translation along geodesics and arbitrary curves similar to the smooth case:

Definition 11 *Let $\gamma : I \to S$ be a parametrized straightest geodesic on a polyhedral surface S. A tangential vector field $v : I \to TS$ with $v(s) \in T_{\gamma(s)}S$ is a* parallel vector field *along γ if the normalized angle $\alpha(v(s), \gamma'(s))$ is constant.*

Definition 12 *Let κ_g be the geodesic curvature of a curve $\gamma : I \to S$ with $\gamma(0) = p$ and let $v_0 \in T_pS$ be a tangent vector with normalized*

angle $\alpha(0) := \frac{2\pi}{\theta(p)} \measuredangle(v_0, \gamma'(0))$. *Then* v_0 *uniquely extends to a parallel vector field* v *with* $v(s) \in T_{\gamma(s)}S$ *along* γ *with* $v(0) = v_0$. $v(s)$ *is defined by the normalized angle* $\alpha(s)$ *it encloses with* $\gamma'(s)$*:*

$$\alpha(s) = \alpha(0) + \int_0^s \kappa_g(t)\, dt. \tag{14}$$

The formula is well-known for curves on smooth surfaces. In the discrete situation we have made direct use of the definition of discrete geodesic curvature and the notion of normalized angles at vertices.

7 Runge Kutta on Discrete Surfaces

The tracing of particles on a surface by integrating a given vector field with Euler or Runge Kutta methods requires an additional effort to keep the trace on the surface. For example, one may use local coordinate charts of a surface to transform the integration to the planar Euclidean domain. Here the metrical distortion between surface and Euclidean domain must be respected and a preprocessing step to generate the charts and transitions between neighbouring charts is required.

If the vector field is given on a curved surface in an ambient space, say $\mathbb{R}^3$, then a usual tangent vector "points into the ambient space", leading the numerical particle trace off the surface without additional projection methods.

The concepts of straightest geodesics and polyhedral tangent vectors offer an intrinsic tool to solve these problems. In Euclidean methods, the vector $v|_{\gamma(s)}$ is interpreted as tangent vector to the particle trace $\gamma(s)$, and the straight line through $\gamma(s)$ with direction $v|_{\gamma(s)}$ is the first order approximation of γ. The idea on surfaces is to use polyhedral tangent vectors defined in definition 7 and to replace the straight line with a straightest geodesic through $\gamma(s)$ with initial direction $v|_{\gamma(s)}$:

Definition 13 (Geodesic Euler Method) *Let* S *be a polyhedral surface with a polyhedral tangential vector field* v *on* S*, let* $y_0 \in S$ *be an initial point, and let* $h > 0$ *a (possibly varying) stepsize. For each point* $p \in S$ *let* $\delta(t, p, v(p))$ *denote the unique straightest geodesic through* p *with initial direction* $v(p)$ *and evaluated at the parameter value* t*. A single iteration step of the* geodesic Euler method *is given by*

$$y_{i+1} := \delta(h, y_i, v(y_i)). \tag{15}$$

This produces a sequence of points $\{y_0, y_1, \ldots\}$ *on* S *which are connected by straightest geodesic segments of length* h*. For each* $i \in \{0, 1, \ldots\}$ *we define*

$$\gamma(ih + t) := \delta(t, y_i, v(y_i)),\ t \in [0, h] \tag{16}$$

and obtain a piecewise straightest, continuous curve $\gamma : [0, \ell) \to S$ *of some length* ℓ *such that each segment* $\gamma_{|[ih,\,(i+1)h]}$ *is a straightest geodesics.*

The definition of the geodesic Euler method is intrinsic and no projection of the tangent vectors or tangent directions onto the surface are required during integration. If the original vector field is not a polyhedral tangential field then an initial generation of a polyhedral tangential vector field is required in a preprocessing step, however, this step is part of the formulation of the numerical problem and not of the integration method.

Using the concept of parallel translation it is straight forward to define higher order integration methods in a similar intrinsic way.

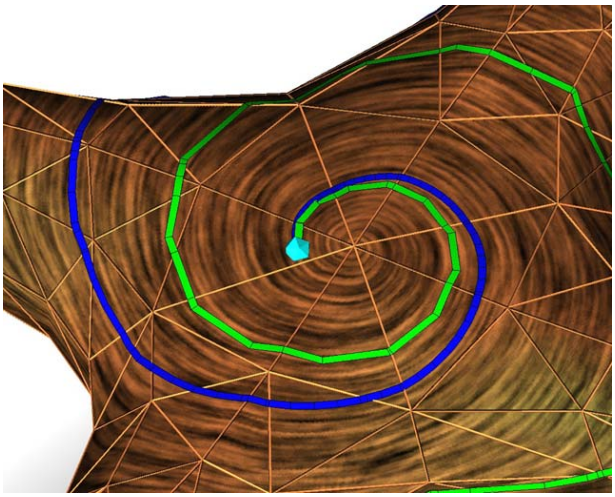


Figure 9: The two piecewise straightest geodesics are solutions computed with the geodesic Euler method (outer curve, stepsize h) and 4th order Runge Kutta method (inner curve, stepsize $4h$). Note, that the geodesic segments extend across triangle edges and vertices. Also, a comparison with the underlying flow shows the expected better approximation quality of the geodesic Runge-Kutta method.

For simplicity, we restrict to a 4-th order geodesic Runge Kutta method:

Definition 14 (Geodesic Runge-Kutta Method) *Let* S *be a polyhedral surface with a polyhedral tangential vector field* v *on* S*, let* $y_0 \in S$ *be an initial point, and let* $h > 0$ *a (possibly varying) stepsize. For each point* $p \in S$ *let* $\delta(t, p, v(p))$ *denote the unique straightest geodesic through* p *with initial direction* $v(p)$ *and evaluated at the parameter value* t*. A single iteration step of the* geodesic Runge Kutta method *is given by*

$$y_{i+1} := \delta(h, y_i, v_i) \tag{17}$$

where the direction v_i *is a polyhedral tangent vector at* y_i *obtained as follows: we denote the parallel translation of vectors along a geodesic* δ *to* $\delta(0)$ *by* $\pi_{|\delta}$ *and iteratively define*

$$\begin{aligned} v_i^1 &: = v(y_i) \\ v_i^2 &: = \pi_{|\delta_1} \circ v(\delta_1(\frac{h}{2}, y_i, v_i^1)) \\ v_i^3 &: = \pi_{|\delta_2} \circ v(\delta_2(\frac{h}{2}, y_i, v_i^2)) \\ v_i^4 &: = \pi_{|\delta_3} \circ v(\delta_3(h, y_i, v_i^3)) \end{aligned} \tag{18}$$

and

$$v_i := \frac{1}{6}(v_i^1 + 2v_i^2 + 2v_i^3 + v_i^4) \tag{19}$$

where the curves δ_i *are straightest geodesics through* y_i *with initial direction* v_i^j *for* $j \in \{1, 2, 3\}$.

8 Conclusion

On polyhedral surfaces we introduced the concept of straightest geodesics and discrete geodesic curvature of curves. We applied the concept to define the parallel translation of tangential vectors and

generalized Runge Kutta methods to polyhedral surfaces. These concepts allow a uniform and intrinsic description of geometric and numerical properties on polyhedral surfaces.

References

ALEKSANDROV, A. D., AND ZALGALLER, V. A. 1967. *Intrinsic Geometry of Surfaces*, vol. 15 of *Translation of Mathematical Monographs*. AMS.

ALEXANDER, S. B., AND BISHOP, R. L. 1996. Comparison theorems for curves of bounded geodesic curvature in metric spaces of curvature bounded above. *Diff. Geom. Appl. 6*, 1, 67–86.

CARMO, M. P. D. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.

DIJKSTRA, E. 1959. A note on two problems in connection with graphs. *Numer. Math. 1*, 269–271.

GUNN, C., ORTMANN, A., PINKALL, U., POLTHIER, K., AND SCHWARZ, U. 1997. Oorange - a visualization environment for mathematical experiments. In *Visualization and Mathematics*, H.-C. Hege and K. Polthier, Eds. Springer Verlag, Heidelberg, 249–265.

MAX, N. L. 1977. *Turning a Sphere Inside Out*. International Film Bureau, Chicago. narrated videotape (21 min).

MITCHELL, J. S. B., MOUNT, D. M., AND PAPADIMITRIOU, C. H. 1987. The discrete geodesic problem. *SIAM J. Comput. 16*, 4, 647–668.

POGORELOV, A. V. 1952. Quasigeodesic lines on a convex surface. *Amer. Math. Soc. Transl. I. Ser. 6*, 72, 430–473.

POLTHIER, K., SCHMIES, M., STEFFENS, M., AND TEITZEL, C., 1997. Geodesics and waves. Siggraph'97 Video Review 120 and VideoMath Festival at ICM'98. Video, 4:40 min.

RESHETNYAK, Y. G. 1993. *Geometry IV*, vol. 70 of *Encyclopaedia of Mathematical Sciences*. Springer Verlag, ch. 1. Two-Dimensional Manifolds of Bounded Curvature, 3–164.

SHARIR, M., AND SCHORR, A. 1986. On shortest paths in polyhedral space. *SIAM J. Comput. 15*, 1, 193–215.

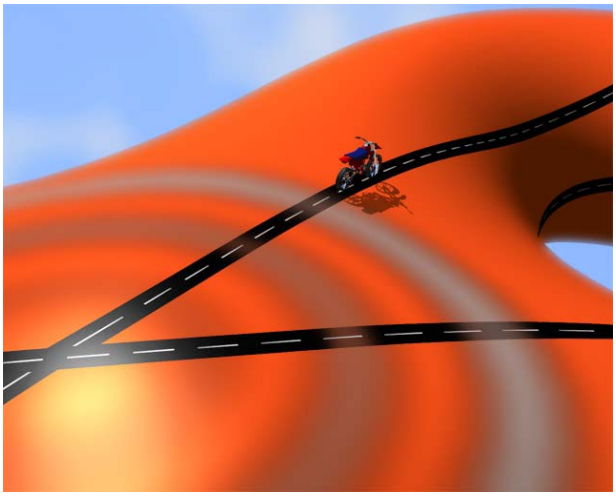


Figure 10: Point waves on surfaces evolve through distance circles. Each particle of the front moves along a straightest geodesic.

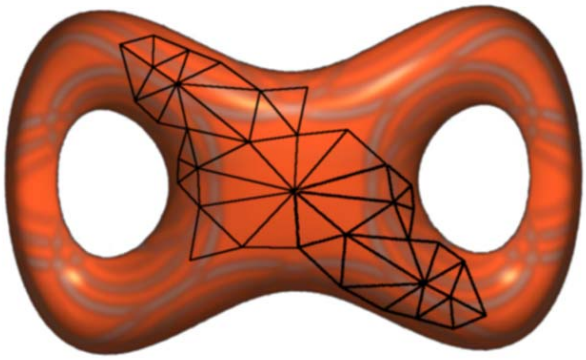


Figure 11: The front of a point wave on a polyhedral surface may branch and hit itself depending on the curvature and topology of the surface.

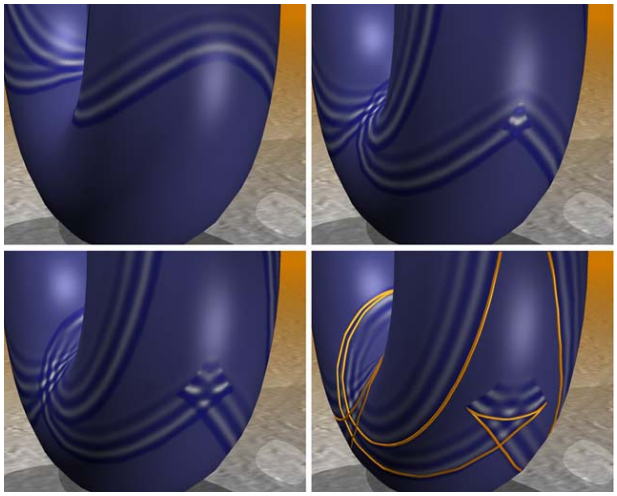


Figure 12: Branching of point waves at the conjugate point on a torus.

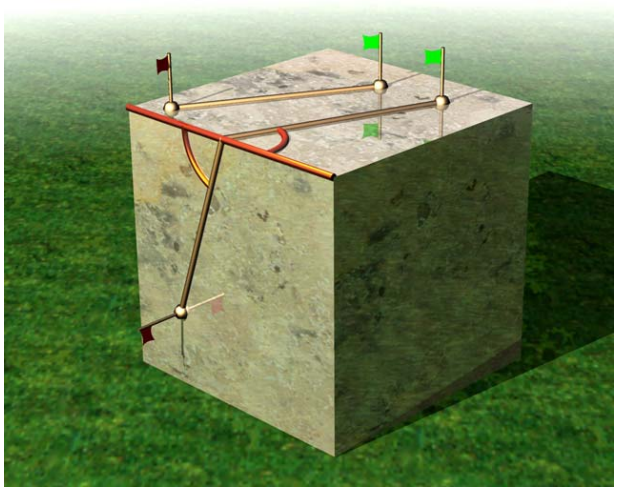


Figure 13: Discrete geodesics on faces and at edges are natural generalizations of straight lines.

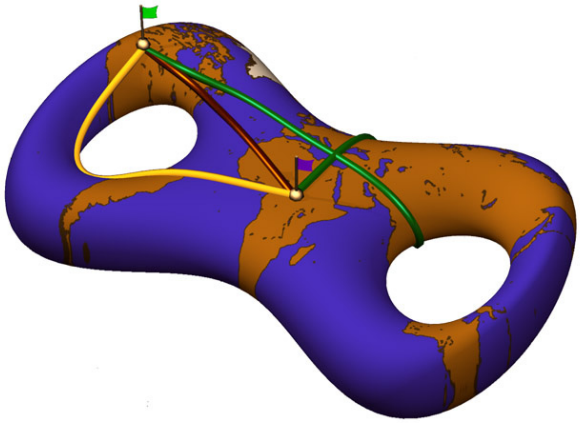


Figure 14: Minimizing length of curves may lead to different local minimizers.

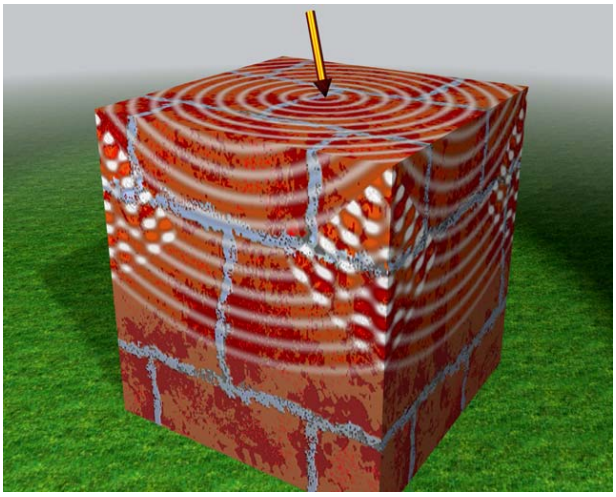


Figure 15: Each positively curved vertex on a polyhedral surface is a conjugate point where all point waves branch.

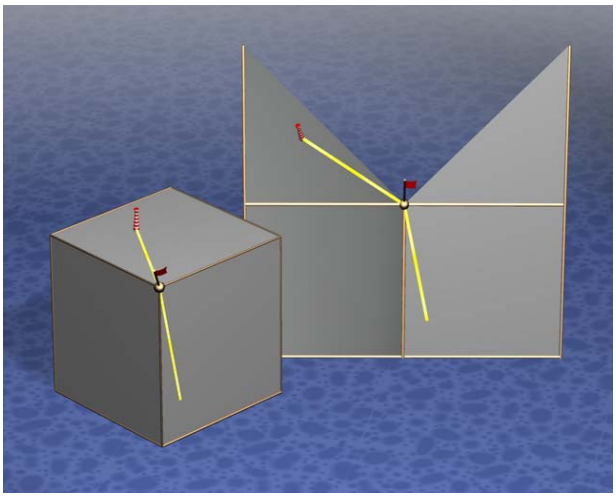


Figure 16: Straightest geodesics are able to pass through vertices in contrast to locally shortest geodesics.

Figure 17: Straightest geodesics have unique extensions like light rays.

Discrete Differential Forms for Computational Modeling

Mathieu Desbrun Eva Kanso* Yiying Tong
Applied Geometry Lab
Caltech†

1 Motivation

The emergence of computers as an essential tool in scientific research has shaken the very foundations of differential modeling. Indeed, the deeply-rooted abstraction of smoothness, or *differentiability*, seems to inherently clash with a computer's ability of storing only finite sets of numbers. While there has been a series of computational techniques that proposed discretizations of differential equations, the geometric structures they are simulating are often lost in the process.

1.1 The Role of Geometry in Science

Geometry is the study of space and of the properties of shapes in space. Dating back to Euclid, models of our surroundings have been formulated using simple, geometric descriptions, formalizing apparent *symmetries* and experimental *invariants*. Consequently, geometry is at the foundation of many current physical theories: general relativity, electromagnetism (E&M), gauge theory as well as solid and fluid mechanics all have strong underlying geometrical structures. Einstein's theory for instance states that gravitational field strength is directly proportional to the *curvature of space-time*. In other words, the physics of relativity is *directly modelled* by the shape of our 4-dimensional world, just as the behavior of soap bubbles is modeled by their shapes. Differential geometry is thus, de facto, the mother tongue of numerous physical and mathematical theories.

Unfortunately, the inherent geometric nature of such theories is often obstructed by their formulation in vectorial or tensorial notations: the traditional use of a coordinate system, in which the defining equations are expressed, often obscures the underlying structures by an overwhelming usage of indices. Moreover, such complex expressions entangle the topological and geometrical content of the model.

1.2 Geometry-based Exterior Calculus

The geometric nature of these models is best expressed and elucidated through the use of the *Exterior Calculus of Differential Forms*, first introduced by Cartan [Cartan 1945]. This geometry-based calculus was further developed and refined over the twentieth century to become the foundation of modern differential geometry. The calculus of exterior forms allows one to express differential and integral equations on smooth and curved spaces in a consistent manner, while revealing the geometrical invariants at play. For example, the classical operations of gradient, divergence, and curl as well as the theorems of Green, Gauss and Stokes can all be expressed concisely in terms of differential forms and an operator on these forms called the exterior derivative—hinting at the generality of this approach.

Compared to classical tensorial calculus, this exterior calculus has several advantages. First, it is often difficult to recognize the coordinate-independent nature of quantities written in tensorial notation: local and global invariants are hard to notice by just staring at the indices. On the other hand, invariants are easily discovered when expressed as differential forms by invoking either Stokes' theorem, Poincaré lemma, or by applying exterior differentiation. Note also that the exterior derivative of differential forms—the antisymmetric part of derivatives—is one of the most important parts of differentiation, since it is invariant under coordinate system change. In fact, Sharpe states in [Sharpe 1997] that every differential equation may be expressed in term of the exterior derivative of differential forms. As a consequence, several recent initiatives have been aimed at formulating the physical laws in terms of differential forms. For recent work along these lines, the reader is invited to refer to [Burke 1985; Abraham et al. 1988; Lovelock and Rund 1993; Flanders 1990; Morita 2001; Carroll 2003; Frankel 2004] for books offering a theoretical treatment of various physical theories using differential forms.

1.3 Differential vs. Discrete Modeling

We have seen that a large amount of our scientific knowledge relies on a deeply-rooted differential (*i.e.*, smooth) comprehension of the world. This abstraction of differentiability allows researchers to model complex physical systems via concise equations. With the sudden advent of the digital age, it was therefore only natural to resort to computations based on such differential equations.

However, since digital computers can only manipulate finite sets of numbers, their capabilities seem to clash with the basic foundations of differential modeling. In order to overcome this hurdle, a first set of computational techniques (*e.g.*, finite difference or particle methods) focused on satisfying the continuous equations at a discrete set of spatial and temporal samples. Unfortunately, focusing on accurately discretizing the local laws often fails to respect important global structures and invariants. Later methods such as Finite Elements (FEM), drawing from developments in the calculus of variations, remedied this inadequacy to some extent by satisfying local conservation laws on average and preserving some important invariants. Coupled with a finer ability to deal with arbitrary boundaries, FEM became the de facto computational tool for engineers. Even with significant advances in error control, convergence, and stability of these finite approximations, the underlying structures of the simulated continuous systems are often destroyed: a moving rigid body may gain or loose momentum; or a cavity may exhibit fictitious eigenmodes in an electromagnetism (E&M) simulation. Such examples illustrate some of the loss of fidelity that can follow from a standard discretization process, failing to preserve some fundamental geometric and topological structures of the underlying continuous models.

The cultural gap between theoretical and applied science communities may be partially responsible for the current lack of proper discrete, computational modeling that could mirror and leverage the rich developments of its differential counterpart. In particular, it is striking that the calculus of differential forms has not yet had an impact on the mainstream computational fields, despite excellent initial results in E&M [Bossavit 1998] or Lagrangian mechanics [Marsden and West 2001]. It should also be noticed that some basic tools necessary for the definition of a discrete calculus already exist, probably initiated by Poincaré when he defined his

*Now at the University of Southern California.
†E-mail: {mathieu|eva|yiying}@caltech.edu

cell decomposition of smooth manifolds. The study of the structure of ordered sets or simplices now belongs to the well-studied branch of mathematics known as *Combinatorial Differential Topology and Geometry*, which is still an active area of research (see, e.g., [Forman] and [Björner and Welker 1995] and references therein).

1.4 Calculus ex Geometrica

Given the overwhelming geometric nature of the most fundamental and successful calculus of these last few centuries, it seems relevant to *approach computations from a geometric standpoint*.

One of the key insights that percolated down from the theory of differential forms is rather simple and intuitive: one needs to recognize that different physical quantities have different properties, and must be treated accordingly. Fluid mechanics or electromagnetism, for instance, make heavy use of line integrals, as well as surface and volume integrals; even physical measurements are performed as specific local integrations or averages (think flux for magnetic field, or current for electricity, or pressure for atoms' collisions). Pointwise evaluations or approximations for such quantities are not the appropriate discrete analogs, since the defining geometric properties of their physical meaning cannot be enforced naturally. Instead, *one should store and manipulate those quantities at their geometrically-meaningful location*: in other words, we should consider values on vertices, edges, faces, and tetrahedra as proper discrete versions of respectively pointwise functions, line integrals, surface integrals, and volume integrals: only then will we be able to manipulate those values without violating the symmetries that the differential modeling tried to exploit for predictive purposes.

1.5 Similar Endeavors

The need for improved numerics have recently sprung a (still limited) number of interesting related developments in various fields. Although we will not try to be exhaustive, we wish to point the reader to a few of the most successful investigations with the same "flavor" as our discrete geometry-based calculus, albeit their approaches are rarely similar to ours. First, the field of *Mimetic Discretizations of Continuum Mechanics*, led by Shashkov, Steinberg, and Hyman [Hyman and Shashkov 1997], started on the premise that spurious solutions obtained from finite element or finite difference methods often originate from inconsistent discretizations of the operators div, curl, and grad, and that addressing this inconsistency pays off numerically. Similarly, *Computational Electromagnetism* has also identified the issue of field discretization as the main reason for spurious modes in numerical results. An excellent treatment of the discretization of the Maxwell's equations resulted [Bossavit 1998], with a clear relationship to the differential case. Finally, recent developments in *Discrete Lagrangian Mechanics* have demonstrated the efficacy of a proper discretization of the Lagrangian of a dynamical system, rather than the discretization of its derived Euler-Lagrange equations: with a discrete Lagrangian, one can ensure that the integration scheme satisfies an exact discrete least-action principle, preserving all the momenta directly for arbitrary orders of accuracy [Marsden and West 2001]. Respecting the defining geometric properties of both the fields and the governing equations is a common link between all these recent approaches.

1.6 Advantages of Discrete Differential Modeling

The reader will have most probably understood our bias by now: we believe that the systematic construction, inspired by Exterior Calculus, of **differential, yet readily discretizable computational foundations** is a crucial ingredient for numerical fidelity. Because many of the standard tools used in differential geometry have discrete combinatorial analogs, the *discrete versions of forms or manifolds* will be formally identical to (and should partake of the same properties as) the continuum models. Additionally, such an approach should clearly maintain the separation of the topological (*metric-independent*) and geometrical (*metric-dependent*) components of the quantities involved, keeping the geometric picture (*i.e.*, intrinsic structure) intact.

A *discrete differential modeling approach to computations* will also be often much simpler to define and develop than its continuous counterpart. For example, the discrete notion of a differential form will be implemented simply as values on mesh elements. Likewise, the discrete notion of orientation will be more straightforward than its continuous counterpart: while the differential definition of orientation uses the notion of equivalence class of atlases determined by the sign of the Jacobian, the orientation of a mesh edge will be one of two directions; a triangle will be oriented clockwise or counterclockwise; a volume will have a direction as a right-handed helix or a left-handed one; no notion of atlas (a collection of consistent coordinate charts on a manifold) will be required.

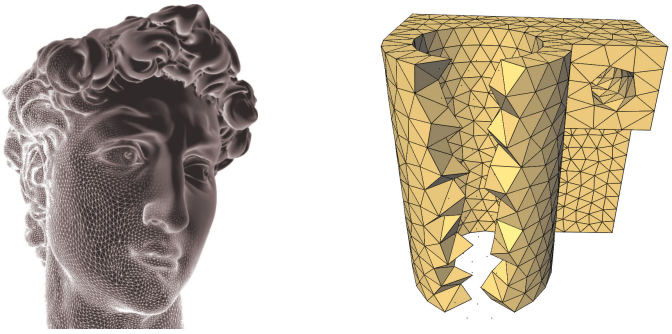


Figure 1: *Typical 2D and 3D meshes: although the David' head appears smooth, its surface is made of a triangle mesh; tetrahedral meshes (such as this mechanical part, with a cutaway view) are some typical examples of irregular meshes on which computations are performed. David's head mesh is courtesy of Marc Levoy, Stanford.*

1.7 Goal of This Chapter

Given these premises, this chapter was written with several purposes in mind. First, we wish to demonstrate that the foundations on which powerful methods of computations can be built are quite approachable—and are not as abstract as the reader may fear: the ideas involved are very intuitive as a side effect of the simplicity of the underlying geometric principles.

Second, we wish to help bridge the gap between applied fields and theoretical fields: we have tried to render the theoretical bases of our exposition accessible to computer scientists, and the concrete implementation insights understandable by non-specialists. For this very reason, the reader should not consider this introductory exposition as a definite source of knowledge: it should instead be considered as a portal to better, more focused work on related subjects. We only hope that we will ease our readers into foundational concepts that can be undoubtedly and fruitfully applied to all sorts of computations—be it for graphics or simulation.

With these goals in mind, we will describe the background needed to develop a principled, geometry-based approach to computational modeling that gets around the apparent mismatch between differential and discrete modeling.

2 Relevance of Forms for Integration

The evaluation of differential quantities on a discrete space (mesh) is a nontrivial problem. For instance, consider a piecewise-linear 2-dimensional surface embedded in a three-dimensional Euclidean space, *i.e.*, a triangle mesh. Celebrated quantities such as the Gaussian and mean curvatures are delicate to define on it. More

precisely, the Gaussian curvature can be easily proven to be zero everywhere *except* on vertices, where it is a Dirac delta function. Likewise, the mean curvature can only be defined in the distributional sense, as a Dirac delta function on edges. However, through local *integrations*, one can easily manipulate these quantities numerically: if a careful choice of non-overlapping regions is made, the delta functions can be properly integrated, rendering the computations relatively simple as shown, for example, in [Meyer et al. 2002; Hildebrandt and Polthier 2004]. Note that the process of integration to suppress discontinuity is, in spirit, equivalent to the idea of weak form used in the Finite Element method.

This idea of integrated value has predated in some cases the equivalent differential statements: for instance, it was long known that the genus of a surface can be calculated through a cell decomposition of the surface via the Euler characteristic. The actual Gauss-Bonnet theorem was, however, derived later on. Now, if one tries to discretize the Gaussian curvature of a piecewise-linear surface in an arbitrary way, it is not likely that its integral over the surface equals the desired Euler characteristic, while its discrete version, defined on vertices (or, more precisely, on the dual of each vertex), naturally preserves this topological invariant.

2.1 From Integration to Differential Forms

Integration is obviously a linear operation, since for any disjoint sets A and B,

$$\int_{A\cup B} = \int_A + \int_B .$$

Moreover, the integration of a smooth function over a subset of measure zero is always zero; for example, an area integral of (a lower dimensional object such as) a curve or a point is equal to zero. Finally, integration is *objective* (*i.e.*, relevant) only if its evaluation is invariant under change of coordinate systems. These three properties combined directly imply that the integrand (*i.e.*, the whole expression after the integral sign) has to be *antisymmetric*. That is, the basic building blocks of any type of integration are **differential forms**. Chances are, the reader is already very well acquainted with forms, maybe without even knowing it.

2.1.1 An Intuitive Definition

A differential form (also denoted as exterior[1] differential form) is, informally, an integrand, *i.e.*, a quantity that can be integrated. It is the dx in $\int dx$ and the $dx\ dy$ in $\iint dx\, dy$. More precisely, consider a smooth function $F(x)$ over an interval in $\mathbb{R}$. Now, define $f(x)$ to be its derivative, that is,

$$f(x) \ = \ \frac{dF}{dx} \ ,$$

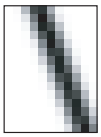
Rewriting this last equation (using slightly abusive notations for simplicity) yields $dF = f(x)dx$, which leads to:

$$\int_a^b dF = \int_a^b f(x)dx = F(b) - F(a). \qquad (1)$$

This last equation is known as the Newton-Leibnitz formula, or the first fundamental theorem of calculus. The integrand $f(x)dx$ is called a *1-form*, because it can only be integrated over any 1-dimensional (1D) real interval. Similarly, for a function $G(x, y, z)$, we have:

$$dG = \frac{\partial G}{\partial x}dx + \frac{\partial G}{\partial y}dy + \frac{\partial G}{\partial z}dz \ ,$$

which can be integrated over any 1D curve in $\mathbb{R}^3$, and is also a 1-form. More generally, *a k-form can be described as an entity ready (or designed, if you prefer) to be integrated on a kD (sub)region*. Note that forms are valued zero on (sub)regions that are of higher or lower order dimension than the original space; for example, 4-forms are zero on $\mathbb{R}^3$. These differential forms are extensively used in mathematics, physics and engineering, as we already hinted at the fact in Section 1.4 that most of our measurements of the world are of integral nature: even digital pictures are made out of local area integrals of the incident light over each of the sensors of a camera to provide a set of values at each pixel on the final image (see inset). The importance of this notion of forms in science is also evidenced by the fact that operations like gradient, divergence, and curl can all be expressed in terms of forms only, as well as fundamental theorems like Green's or Stokes.



2.1.2 A Formal Definition

For concreteness, consider the n-dimensional Euclidean space $\mathbb{R}^n$, $n \in \mathbb{N}$ and let $\mathcal{M}$ be an open region $\mathcal{M} \subset \mathbb{R}^n$; $\mathcal{M}$ is also called an *n-manifold*. The vector space $T_x\mathcal{M}$ consists of all the (tangent) vectors at a point $x \in \mathcal{M}$ and can be identified with $\mathbb{R}^n$ itself. A k-form ω^k is a rank-k, anti-symmetric, tensor field over $\mathcal{M}$. That is, at each point $x \in \mathcal{M}$, it is a multi-linear map that takes k tangent vectors as input and returns a real number:

$$\omega^k : T_xM \ldots \times T_xM \longrightarrow \mathbb{R}$$

which *changes sign when you switch two variables* (hence the term antisymmetric). Any k-form naturally induces a k-form on a submanifold, through restriction of the linear map to the domain that is the product of tangent spaces of the submanifold.

Comments on the Notion of Pseudo-forms There is a closely related concept named pseudo-form. Pseudo-forms change sign when we change the orientation of coordinate systems, just like pseudo-vectors. As a result, the integration of a pseudo-form does not change sign when the orientation of the manifold is changed. Unlike k-forms, a pseudo-k-form induces a pseudo-k-form on a submanifold *only* if a transverse direction is given. For example, fluid flux is sometimes called a pseudo-2-form: indeed, given a transverse direction, we know how much flux is going through a piece of surface; it does not depend on the orientation of the surface itself. Vorticity is, however, a true 2-form: given an orientation of the surface, the integration gives us the circulation around that surface boundary induced by the surface orientation. It does *not* depend on the transverse direction of the surface. But if we have an orientation of the ambient space, we can always associate transverse direction with internal orientation of the submanifold. Thus, in our case, we may treat pseudo-forms simply as forms because we can consistently choose a representative from the equivalence class.

2.2 The Differential Structure

Differential forms are the building blocks of a whole calculus. To manipulate these basic blocks, Exterior Calculus defines seven operators:

- d: the exterior derivative, that extends the notion of the differential of a function to differential forms;
- $\star$: the Hodge star, that transforms k-forms into (n-k)-forms;
- $\wedge$: the wedge product, that extends the notion of exterior product to forms;
- $\sharp$ and $\flat$: the sharp and flat operators, that, given a metric, transforms a 1-form into a vector and vice-versa;

[1] The word "exterior" is used as the exterior algebra is basically builds out of an *outer* product.

- ◇ i_X: the interior product with respect to a vector field X (also called contraction operator), a concept dual to the exterior product;
- ◇ $\mathcal{L}_X$: the Lie derivative with respect to a vector field X, that extends the notion of directional derivative.

In this chapter, we will restrict our discussions to the first three operators, to provide the most basic tools necessary in computational modeling.

2.3 A Taste of Exterior Calculus in $\mathbb{R}^3$

To give the reader a taste of the relative simplicity of Exterior Calculus, we provide a list of equivalences (in the continuous world!) between traditional operations and their Exterior Calculus counterpart in the special case of $\mathbb{R}^3$. We will suppose that we have the usual Euclidean metric. Then, forms are actually quite simple to conceive:

0-form $\Leftrightarrow$ scalar field
1-form $\Leftrightarrow$ vector
2-form $\Leftrightarrow$ vector
3-form $\Leftrightarrow$ scalar field

To be clear, we will add a superscript on the forms to indicate their rank. Then applying forms to vector fields amounts to:

1-form: $u^1(v) \Leftrightarrow u \cdot v$.
2-form: $u^2(v, w) \Leftrightarrow u \cdot (v \times w)$.
3-form: $f^3(u, v, w) \Leftrightarrow fu \cdot (v \times w)$.

Furthermore, the usual operations like gradient, curl, divergence and cross product can all be expressed in terms of the basic exterior calculus operators. For example:

$d^0 f = \nabla f,\ d^1 u = \nabla \times u,\ d^2 u = \nabla \cdot u;$
$*^0 f = f,\ *^1 u = u,\ *^2 u = u,\ *^3 f = f;$
$*^0 d^2 *^1 u^1 = \nabla \cdot u,\ *^1 d^1 *^2 u^2 = \nabla \times u,\ *^2 d^0 *^3 f = \nabla f;$
$f^0 \wedge u = fu,\ u^1 \wedge v^1 = u \times v,\ u^1 \wedge v^2 = u^2 \wedge v^1 = u \cdot v;$
$i_v u^1 = u \cdot v,\ i_v u^2 = u \times v,\ i_v f^3 = fv.$

Now that we have established the relevance of differential forms even in the most basic vector operations, time has come to turn our attention to make this concept of forms readily usable for computational purposes.

3 Discrete Differential Forms

Finding a discrete counterpart to the notion of differential forms is a delicate matter. If one was to represent differential forms using their coordinate values and approximate the exterior derivative using finite differences, basic theorems such as Stokes theorem would not hold numerically. The main objective of this section is therefore to present a proper discretization of the forms on what is known as simplicial complexes. We will show how this discrete geometric structure, well suited for computational purposes, is designed to preserve all the fundamental differential properties. For simplicity, we restrict the discussion to forms on 2D surfaces or 3D regions embedded in $\mathbb{R}^3$, but the construction is applicable to general manifolds in arbitrary spaces. In fact, the only necessary assumption is that the embedding space must be a vector space, a natural condition in practice.

3.1 Simplicial Complexes and Discrete Manifolds

For the interested reader, the notions we introduce in this section are defined formally in much more details (for the general case of k-dimensional spaces) in references such as [Munkres 1984] or [Hatcher 2004].

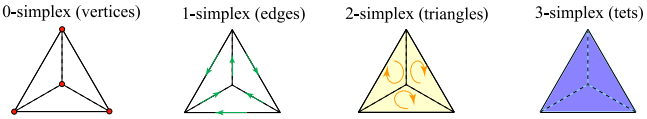


Figure 2: *A 1-simplex is a line segment, the convex hull of two points. A 2-simplex is a triangle, i.e., the convex hull of three distinct points. A 3-simplex is a tetrahedron, as it is the convex hull of four points.*

3.1.1 Notion of Simplex

A **k-simplex** is the generic term to describe the simplest mesh element of dimension k—hence the name. By way of motivation, consider a three-dimensional mesh in space. This mesh is made of a series of adjacent tetrahedra (denoted *tets* for simplicity throughout). The vertices of the tets are said to form a 0-simplex. Similarly, the line segments or edges form a 1-simplex, the triangles or faces form a 2-simplex, and the tets a 3-simplex. Note that we can define these simplices in a top-down manner too: faces (2-simplex) can be thought of as boundaries of tets (3-simplices), edges (1-simplices) as boundaries of faces, and vertices (0-simplices) as boundaries of edges.

The definition of a simplex can be made more abstract as a series of k-tuples (referring to the vertices they are built upon). However, for the type of applications that we are targeting in this chapter, we will often not make any distinction between an abstract simplex and its topological realization (connectivity) or geometrical realization (positions in space) .

Formally, a k-simplex σ_k is the non-degenerate convex hull of k+1 geometrically distinct points $v_0, \dots v_k \in \mathbb{R}^n$ with $n \geq k$. In other words, it is the intersection of all convex sets containing $(v_0, \dots v_k)$; namely:

$$\sigma_k = \{x \in \mathbb{R}^n | x = \sum_{i=0}^{k} \alpha^i \, \mathrm{v}_i \ \text{ with } \ \alpha^i \geq 0 \text{ and } \sum_{i=0}^{k} \alpha^i = 1\}.$$

The entities $v_0, \dots v_k$ are called the *vertices* and k is called the dimension of the k-simplex., which we will denote as:

$$\sigma_k = \{v_0 v_1 ... v_k\} \ .$$

3.1.2 Orientation of a Simplex

Note that all orderings of the $k+1$ vertices of a k-simplex can be divided into two equivalent classes, *i.e.*, two orderings differing by an even permutation. Such a class of ordering is called an *orientation*. In the present work, we always assume that local orientations are *given* for each simplex; that is, each element of the mesh has been given a particular orientation. For example, an edge $\sigma_1 = \{v_0 v_1\}$ on Figure 2 has an arrow indicating its default orientation. If the opposite orientation is needed, we will denote it as $\{v_1 v_0\}$, or, equivalently, by $-\{v_0 v_1\}$. For more details and examples, the reader is referred to [Munkres 1984; Hirani 2003].

3.1.3 Boundary of a Simplex

Any (k-1)-simplex spanned by a subset of $\{v_0, \dots v_k\}$ is called a *(k-1)-face* of σ_k. That is, a *(k-1)-face* is simply a (k-1)-simplex whose k vertices are all from the k+1 vertices of the k-simplex. The union of the (k-1)-faces is what is called the *boundary* of the k-simplex. One should be careful here: because of the default orientation of the simplices, the formal *signed* sum of the (k-1)-faces defines the boundary of the k-simplex. Therefore, the boundary operator takes a k-simplex and gives the sum of all its (k-1)-faces

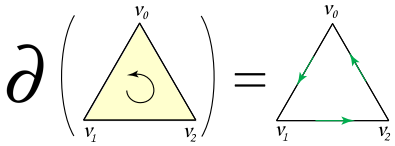


Figure 3: *The boundary operator ∂ applied to a triangle (a 2-simplex) is equal to the signed sum of the edges (i.e., the 1-faces of the 2-simplex).*

with 1 or -1 as coefficients depending on whether their respective orientations match or not, see Figure 4.

To remove possible mistakes in orientation, we can define the *boundary operator* as follows:

$$\partial\{v_0 v_1 ... v_k\} = \sum_{j=0}^{k} (-1)^j \{v_0, ..., \widehat{v_j}, ..., v_k\}, \qquad (2)$$

where $\widehat{v_j}$ indicates that v_j is missing from the sequence, see Figure 3. Clearly, each k-simplex has k+1 (k-1)-faces. For this statement to be valid even for $k = 0$, the empty set $\emptyset$ is usually defined as a (-1)-simplex face of every 0-simplex. The reader is invited to verify this definition on the triangle $\{v_0, v_1, v_2\}$ in Figure 3:

$$\partial\{v_0, v_1, v_2\} = \{v_1, v_2\} - \{v_0, v_2\} + \{v_0, v_1\}$$

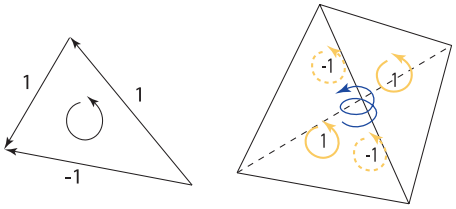
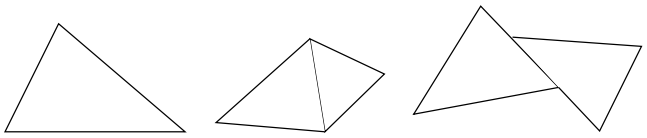


Figure 4: *Boundary operator applied to a triangle (left), and a tetrahedron (right). Orientations of the simplices are indicated with arrows.*

3.1.4 Simplicial Complex

A simplicial complex is a collection $\mathcal{K}$ of simplices, which satisfies the following two simple conditions:

- every face of each simplex in $\mathcal{K}$ is in $\mathcal{K}$;
- the intersection of any two simplices in $\mathcal{K}$ is either empty, or a entire common face.



Simplicial complexes *Not a simplicial complex*

Computer graphics makes heavy use of what is called *realizations* of simplicial complexes. Loosely speaking, a realization of a simplicial complex is an embedding of this complex into the underlying space $\mathbb{R}^n$. Triangle meshes in 2D and tet meshes in 3D are examples of such simplicial complexes (see Figure 1). Notice that polygonal meshes can be easily triangulated, thus can be easily turned into simplicial complexes. One can also use the notion of *cell complex* by allowing the elements of $\mathcal{K}$ to be non-simplicial; we will restrict our explanations to the simpler case of simplicial complexes for simplicity.

3.1.5 Discrete Manifolds

An n-dimensional discrete manifold $\mathcal{M}$ is an n-dimensional simplicial complex that satisfies the following condition: for each simplex, the union of all the incident n-simplices forms an n-dimensional ball (*i.e.*, a disk in 2D, a ball in 3D, etc), or half a ball if the simplex is on the boundary. As a consequence, each (n-1)-simplex has exactly two adjacent n-simplices—or only one if it is on a boundary.

Basically, the notion of discrete manifold corresponds to the usual Computer Graphics acceptation of "manifold mesh". For example in 2D, discrete manifolds cannot have isolated edges (also called sticks or hanging edges) or isolated vertices, and each of their edges is adjacent to 2 triangles (except for the boundary; in that case, the edge is adjacent to only one triangle). A surface mesh in 3D cannot have a "fin", *i.e.*, a edge with more than two adjacent triangles. To put it differently, infinitesimally-small, imaginary *inhabitants* of a n-dimensional discrete manifolds would consider themselves living in $\mathbb{R}^n$ as any small neighborhood of this manifold is isomorphic to $\mathbb{R}^n$.

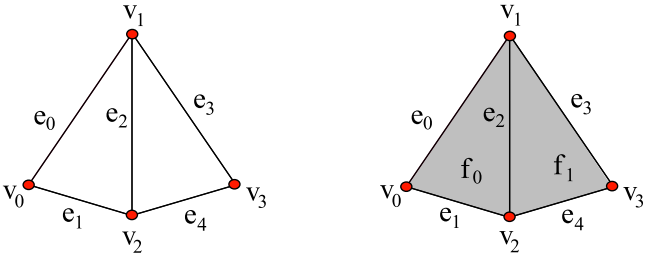


Figure 5: *(a) A simplicial complex consisting of all vertices $\{v_0, v_1, v_2, v_3\}$ and edges $\{e_0, e_1, e_2, e_3, e_4\}$. This simplicial complex is not a discrete manifold because the neighborhoods of the vertices v_1 and v_2 (or of any points along an edge) are not 1D balls. (b) If we add the triangles f_0 and f_1 to the simplicial complex, it becomes a 2-manifold with one boundary.*

3.2 Notion of Chains

We have already encountered the notion of chain, without mentioning it. Recall that the boundary operator takes each k-simplex and gives the *signed* sum of all its (k-1)-faces. We say that the boundary of a k-simplex produces a *(k-1)-chain*. The following definition is more precise and general.

3.2.1 Definition

A k-**chain** of an oriented simplicial complex $\mathcal{K}$ is a set of values, one for *each* k-*simplex* of $\mathcal{K}$. That is, a k-chain c can then be thought of as a linear combination of all the k-simplices in $\mathcal{K}$:

$$c = \sum_{\sigma \in \mathcal{K}} c(\sigma) \cdot \sigma, \qquad (3)$$

where $c(\sigma) \in \mathbb{R}$. More formally, a chain forms a free abelian group generated by the k-simplices, *i.e.*, a mapping from the collection of all k-simplices in $\mathcal{K}$ to $\mathbb{R}$. We will denote the group of all k-chains as $\mathcal{C}_k$.

3.2.2 Implementation of Chains

Let the set of all k-simplices in $\mathcal{K}$ be denoted $\mathcal{K}^k$, and let its cardinality be denoted as $|\mathcal{K}^k|$. A k-chain can simply be stored as a *vector* (or array) of dimension $|\mathcal{K}^k|$, *i.e.*, one number for each k-simplex $\sigma_k \in \mathcal{K}^k$.

3.2.3 Boundary Operator on Chains

We mentioned that the boundary operator ∂ was returning a particular type of chain, namely, a chain with coefficients equal to either 0,

1, or -1. Therefore, it should not be surprising that we can extend the notion of boundary to act also on k-chains, simply by linearity:

$$\partial \sum_k c_k \sigma_k = \sum_k c_k \partial \sigma_k.$$

That is, from one set of values assigned to all simplices of a complex, one can deduce another set of values derived by weighting the boundaries of each simplex by the original value stored on it. This operation is very natural, and can thus be implemented easily as explained next.

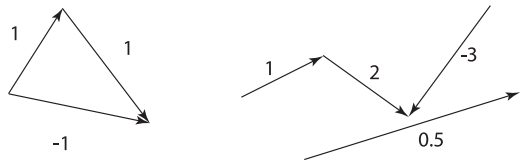


Figure 6: *(a) An example of* 1*-chain being the boundary of a face (*2*-simplex); (b) a second example of* 1*-chain with 4 non-zero coefficients.*

3.2.4 Implementation of the Boundary Operator

Since the boundary operator is a linear mapping from the space of k-simplices to the space of (k-1)-simplices, it can simply be represented by a *matrix* of dimension $|\mathcal{K}^{k-1}| \times |\mathcal{K}^k|$. The reader can convince herself that this matrix is sparse, as only immediate neighbors are involved in the boundary operator. Similarly, this matrix contains only the values 0, 1, and -1. Notice than in 3D, there are three non-trivial boundary operators ∂_k (∂_1 is the boundary operator on edges, ∂_2 on triangles, ∂_3 on tets). However, the operator needed for a particular operation is obvious from the type of the argument: if the boundary of a tet is needed, the operator ∂_3 is the only one that makes sense to apply; in other words, the boundary of a k-simplex σ_k is found by invoking $\partial_k \sigma_k$. Thanks to this context-dependence, we can simplify the notation and remove the superscript when there is no ambiguity.

3.3 Notion of Cochains

A k-cochain ω is the *dual* of a k-chain, that is to say, ω is a linear mapping that takes k-chains to $\mathbb{R}$. One writes:

$$\begin{array}{rccc} \omega: & \mathcal{C}_k & \rightarrow & \mathbb{R} \\ & c & \rightarrow & \omega(c), \end{array} \tag{4}$$

which reads as: a k-cochain ω operates on a k-chain c to give a scalar in $\mathbb{R}$. Since a chain is a linear combination of simplices, a cochain returns a linear combination of the values of that cochain on each simplex involved.

Clearly, a co-chain also corresponds to one value per simplex (since all the k-simplices form a basis for the vector space $\mathcal{C}_k$, and we only need to know the mapping of vectors in this basis to determine a linear mapping), and hence the notion of duality of chains and co-chains is appropriate. But contrary to a chain, a k-cochain is *evaluated* on each simplex of the dimension k. In other words, a k-cochain can be thought of as a *field* that can be evaluated on each k-simplex of an oriented simplicial complex $\mathcal{K}$.

3.3.1 Implementation of Cochains

The numerical representation of cochains follows from that of chains by duality. Recall that a k-chain can be represented as a vector c_k of length equal to the number of k-simplices in $\mathcal{M}$. Similarly, one may represent ω by a vector ω^k of the same size as c_k.

Now, remember that ω operates on c to give a scalar in $\mathbb{R}$. The linear operation $\omega(c)$ translates into an inner product $\omega^k \cdot c_k$. More specifically, one may continue to think of c_k as a *column vector* so that the $\mathbb{R}$-valued linear mapping ω can be represented by a *row vector* $(\omega^k)^t$, and $\omega(c)$ becomes simply the matrix multiplication of the row vector $(\omega^k)^t$ with the column vector c_k. The evaluation of a cochain is therefore trivial to implement.

3.4 Discrete Forms as Co-Chains

The attentive reader will have noticed by now: *k-cochains are discrete analogs to differential forms*. Indeed, a continuous k-form was defined as a linear mapping from k-dimensional sets to $\mathbb{R}$, as we can only integrate a k-form on a k-(sub)manifold. Note now that a kD set, when one has only a mesh to work with, is simply a *chain*. And a linear mapping from a chain to a real number is what we called a cochain: *a cochain is therefore a natural discrete counterpart of a form*.

For instance a 0-form can be evaluated at each point, a 1-form can be evaluated on each curve, a 2-form can be evaluated on each surface, etc. Now if we *restrict* integration to take place only on the k-submanifold which is the sum of the k-simplices in the triangulation, we get a k-cochain; thus k-cochains are a discretization of k-forms. One can further map a continuous k-form to a k-cochain. To do this, first integrate the k-form on each k-simplex and assign the resulting value to that simplex to obtain a k-cochain on the k-simplicial complex. This k-cochain is a discrete representation of the original k-form.

3.4.1 Evaluation of a Form on a Chain

We can now naturally extend the notion of evaluation of a differential form ω on an arbitrary chain simply by linearity:

$$\int_{\sum_i c_i \sigma_i} \omega = \sum_i c_i \int_{\sigma_i} \omega. \tag{5}$$

As mentioned above, the integration of ω on each k-simplex σ_k provides a discretization of ω or, in other words, a mapping from the k-form ω to a k-cochain represented by:

$$\omega[i] = \int_{\sigma_i} \omega.$$

However convenient this chain/cochain standpoint is, in practical applications, one often needs a point-wise value for a k-form or to evaluate the integration on a particular k-submanifold. How do we get these values from a k-cochain? We will cover this issue of *form interpolation* in Section 6.

4 Operations on Chains and Cochains

4.1 Discrete Exterior Derivative

In the present discrete setting where the discrete differential forms are defined as cochains, defining a discrete exterior derivative can be done very elegantly: Stokes' theorem, mentioned early on in Section 2, can be used to *define* the exterior derivative d. Traditionally, this theorem states a vector identity equivalent to the well-known curl, divergence, Green's, and Ostrogradsky's theorems. Written in terms of forms, the identity becomes quite simple: it states that d applied to an arbitrary form ω is evaluated on an arbitrary simplex σ as follows:

$$\int_\sigma d\omega = \int_{\partial\sigma} \omega. \tag{6}$$

You surely recognize the usual property that an integral over a k-dimensional set is turned into a boundary integral (*i.e.*, over a set of dimension k-1). With this simple equation relating the evaluation of $d\omega$ on a simplex σ to the evaluation of ω on the boundary of this simplex, the exterior derivative is *readily defined*: each time you encounter an exterior derivative of a form, replace any evaluation over a simplex σ by a direct evaluation of the form itself over the boundary of σ. Obviously, Stokes' theorem will be enforced by construction!

4.1.1 Coboundary Operator

The operator d is called the *adjoint* of the boundary operator ∂: if we denote the integral sign as a pairing, *i.e.*, with the convention that $\int_\sigma \omega = [\omega, \sigma]$, then applying d on the left hand side of this operator is equivalent to applying ∂ on the right hand: $[d\omega, \sigma] = [\omega, \partial\sigma]$. For this very reason, d is sometimes called the coboundary operator.

Finally, by linearity of integration, we can write a more general expression of Stokes' theorem, now extended to arbitrary chains as follows:

$$\int_{\sum_i c_i\sigma_i} d\omega = \int_{\partial\left(\sum_i c_i\sigma_i\right)} \omega = \int_{\sum_i c_i\partial\sigma_i} \omega = \sum_i c_i \int_{\partial\sigma_i} \omega$$

Consider the example shown in Figure 7. The discrete exterior derivative of the 1-form, defined as numbers on edges, is a 2-form represented by numbers on oriented faces. The orientation of the 1-forms may be opposite to that induced on the edges by the orientation of the faces. In this case, the values on the edges change sign. For instance, the 2-form associated with the d of the 1-forms surrounding the oriented shaded triangle takes the value $\omega = 2 - 1 - 0.75 = 0.25$.

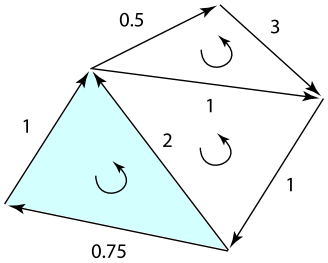


Figure 7: *Given a* 1*-form as numbers on oriented edges, its discrete exterior derivative is a* 2*-form. In particular, this* 2*-form is valued 0.25 on the oriented shaded triangle.*

4.1.2 Implementation of Exterior Derivative

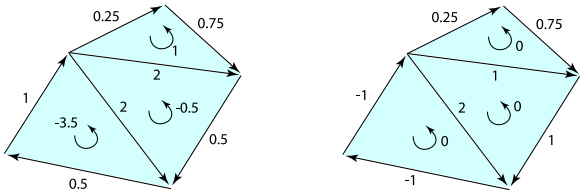
Since we use vectors of dimension $|\mathcal{K}^k|$ to represent a k-cochain, the operator d can be represented by a matrix of dimension $|\mathcal{K}^{k+1}| \times |\mathcal{K}^k|$. Furthermore, this matrix has a trivial expression. Indeed, using the matrix notation introduced earlier, we have:

$$\int_{\partial c} \omega = \omega^t(\partial c) = (\omega^t\partial)c = (\partial^t\omega)^t c = \int_c d\omega.$$

Thus, the matrix d is simply equal to ∂^t. This should not come as a surprise, since we previously discussed that d is simply the adjoint of ∂. Note that extreme care should be used when boundaries are present. However, and without digging too much into the details, it turns out that even for discrete manifolds *with* boundaries, the previous statement is valid. Implementing the exterior derivative while preserving Stokes' theorem is therefore a trivial matter in practice. Notice that just like for the boundary operator, there is actually more than one matrix for the exterior derivative operator: there is one per simplex dimension. But again, the context is sufficient to actually know which matrix is needed. A brute force approach that gets rid of these multiple matrices is to use a notion of super-chain, *i.e.*, a vector storing *all* simplices, ordered from dimension 0 to the dimension of the space: in this case, the exterior derivative can be defined as a single, large sparse matrix that contains these previous matrices as blocks along the diagonal. We will not use this approach, as it makes the exposition less intuitive in general.

4.2 Exact/Closed Forms and Poincaré Lemma

A k-form ω is called exact if there is a (k-1)-form α such that $\omega = d\alpha$, and it is called closed if $d\omega = 0$.



(a) *(b)*

Figure 8: *(a) The 2-form on the oriented shaded triangles defined by the exterior derivative* d *of the 1-form on the oriented edges is called an* exact 2-form*; (b) The 1-form on the oriented edges whose derivative* d *is identically zero is called a* closed 1-form*.*

It is worth noting here that every exact form is closed, as will be seen in Section 4.3. Moreover, it is well-known in the continuous setting that a closed form on a smooth contractible (sub)-manifold is locally exact (to be more accurate: exact over any disc-like region). This result is called the *Poincaré lemma*. The discrete analogue to this lemma can be stated as follows: given a closed k-cochain ω on a star-shaped complex, that is to say, $d\omega = 0$, there exits a (k-1)-cochain α such that $\omega = d\alpha$. For a formal statement and proof of this discrete version, see [Desbrun et al. 2004].

4.3 Introducing the deRham Complex

The boundary of a boundary is the empty set. That is, the boundary operator applied twice to a k-simplex is zero. Indeed, it is easy to verify that $\partial\,\partial\sigma_k = 0$, since each ($k$-2)-simplex will appear exactly twice in this chain with different signs and, hence, cancel out (try it at home!). From the linearity of ∂, one can readily conclude that the property $\partial\,\partial = 0$ is true for all k-chains since the k-simplices form a basis. Similarly, one has that the discrete exterior derivative satisfies $d\ d = \partial^t\partial^t = (\partial\,\partial)^t = 0$, analogously to the exterior derivative of differential forms (notice that this last equality corresponds to the equality of mixed partial derivatives, which in turn is responsible for identities like $\nabla \times \nabla = 0$ and $\nabla \cdot \nabla \times = 0$ in $\mathbb{R}^3$).

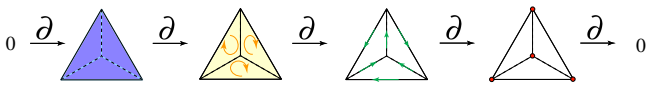


Figure 9: *The chain complex of a tetrahedron with the boundary operator: from the tet, to its triangles, to their edges, and to their vertices.*

4.3.1 Chain Complex

In general, a *chain complex* is a sequence of linear spaces, connected with a linear operator D that satisfies the property $D\,D = 0$. Hence, the boundary operator ∂ (resp., the coboundary operator d) makes the spaces of chains (resp., cochains) into a chain complex, as shown in Figures 9 and 13.

When the spaces involved are the spaces of **differential** forms, and the operator is the exterior derivative d, this chain complex is called the *deRham complex*. By analogy, the chain complex for the spaces of **discrete** forms and for the coboundary operator is called the discrete deRham complex (or sometimes, the cochain complex).

4.3.2 Examples

Consider the 2D simplicial complex in Figure 10(a) and choose the oriented basis of the i-dimensional simplices ($i = 0$ for vertices, $i = 1$ for edges and $i = 2$ for the face) as suggested by the ordering in the figure.

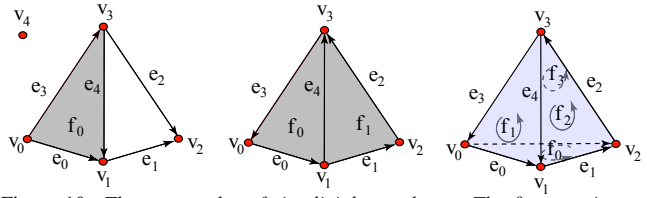


Figure 10: *Three examples of simplicial complexes. The first one is not manifold. The two others are.*

One gets $\partial(f_0) = e_0 - e_4 - e_3$, which can be identified with the vector $(1, 0, 0, -1, -1)$ representing the coefficient in front of each simplex. By repeating similar calculations for all simplices, one can readily conclude that the boundary operator ∂ is given by:

$$\partial_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \\ -1 \end{pmatrix}, \quad \partial_1 = \begin{pmatrix} -1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

That is, the chain complex under the boundary operator ∂ can be written as:

$$0 \longrightarrow \mathcal{C}_2 \xrightarrow{\partial_2} \mathcal{C}_1 \xrightarrow{\partial_1} \mathcal{C}_0 \longrightarrow 0$$

where $\mathcal{C}_i$, $i = 0, 1, 2$, denote the spaces of i-chains.

Consider now the domain to be the mesh shown in Figure 10(b). The exterior derivative operator, or the coboundary operator, can be expressed as:

$$d^0 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 \end{pmatrix}, \qquad d^1 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & -1 \end{pmatrix}.$$

It is worth noting that, since d is adjoint to ∂ by definition, the coboundary operator d induces a cochain complex:

$$0 \longleftarrow \mathcal{C}^2 \xleftarrow{d^1} \mathcal{C}^1 \xleftarrow{d^0} \mathcal{C}^0 \longleftarrow 0$$

where $\mathcal{C}^i$, $i = 0, 1, 2$, denote the spaces of i-cochains.

Finally, suppose the domain is the tetrahedron in Figure 10(c), then the exterior derivative operators are:

$$d^0 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}, d^1 = \begin{pmatrix} 1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}, d^2 = \begin{pmatrix} -1 & 1 & 1 & -1 \end{pmatrix}.$$

4.4 Notion of Homology and Cohomology

Homology is a concept dating back to Poincaré that focuses on studying the topological properties of a space. Loosely speaking, homology does so by counting the number of holes. In our case, since we assume that our space is a simplicial complex (*i.e.*, triangulated), we will only deal with *simplicial homology*, a simpler, more straightforward type of homology that can be seen as a discrete version of the continuous definition (in other words, it is equivalent to the continuous one if the domain is triangulated). As we are about to see, the notion of discrete forms is intimately linked with these topological notions. In fact, we will see that (co)homology is the study of the relationship between *closed* and *exact* (co)chains.

4.4.1 Simplicial Homology

A fundamental problem in topology is that of determining, for two spaces, whether they are topologically equivalent. That is, we wish to know if one space can be morphed into the other without having to puncture it. For instance, a sphere-shaped tet mesh is not topologically equivalent to a torus-shaped tet mesh as one cannot alter the sphere-shaped mesh (*i.e.*, deform, refine, or coarsen it locally) to make it look like a torus.

The key idea of homology is to define *invariants* (*i.e.*, quantities that cannot change by continuous deformation) that characterize topological spaces. The simplest invariant is the number of connected components that a simplicial complex has: obviously, two simplicial complexes with different numbers of pieces cannot be continuously deformed into each other! Roughly speaking, homology groups are an extension of this idea to define more subtle invariants than the number of connected components. In general, one can say that homology is a way to *define* the notion of holes/voids/tunnels/components of an object in any dimension.

Cycles and their Equivalence Classes Generalizing the previous example to other invariants is elegantly done using the notion of *cycles*. A cycle is simply a *closed k-chain*; that is, a linear combination of k-simplices so that the boundary of this chain (see Section 3.2) is the empty set. Any set of vertices is a closed chain; any set of 1D loops are too; etc. Equivalently, a k-cycle is any k-chain that belongs to $\mathrm{Ker}\ \partial_k$, by definition.

On this set of all k-cycles, one can define *equivalence classes*. We will say that a k-cycle is *homologous* to another k-cycle (*i.e.*, in the same equivalence class than the other) when these two chains differ by a boundary of a (k+1)-chain (*i.e.*, by an *exact chain*). Notice that this exact chain is, by definition (see Section 4.2), in the image of ∂_{k+1}, *i.e.*, $\mathrm{Im}\ \partial_{k+1}$. To get a better understanding of this notion of equivalence class, the reader is invited to look at Figure 11: the 1-chains L_1 and L_3 are part of the same equivalent class as their difference is indeed the boundary of a well-defined 2D cycle—a rubber-band shape in this case. Notice that as a consequence, L_1 can be deformed into L_3 without having to tear the loop apart. however, L_2 is not of the class, and thus cannot be deformed into L_3; there's no 2-cycle that corresponds to their difference.

4.4.2 Homology Groups

Let us now use these definition on the simple case of the 0^{th} homology group $\mathcal{H}_0$.

Homology Group $\mathcal{H}_0$ The boundary of any vertex is $\emptyset$. Thus, any linear combination of vertices is a 0-cycle by definition. Now if two vertices v_0 and v_1 are connected by an edge, $v_1 - v_0$ (*i.e.*, the difference of two cycles) is the boundary of this edge. Thus, by our previous definition, two vertices linked by an edge are *homologous* as their difference *is* the boundary of this edge. By the same reasoning, any two vertices taken from the *same* connected component are, also, homologous, since there exists a chain of edges in between. Consequently, we can pick only one vertex per connected component to form a basis of this homology group. Its dimension, β_0, is therefore simply the number of connected components. The

basis elements of that group are called *generators*, since they generate the whole homology group.

Homology Group $\mathcal{H}_1$ Let us proceed similarly for the 1^{st} homology class: we now have to consider 1-cycles (linear combinations of 1D loops). Again, one can easily conceive that there are different *types* of such cycles, and it is therefore possible to separate all possible cycles into different equivalence classes. For instance, the loop L_1 in Figure 11 is topologically distinct from the curve L_2: one is around a hole and the other is not, so the difference between the two is *not* the boundary of a 2-chain. Conversely, L_1 *is* in the same class as curve L_3 since they differ by one connected area. Thus, in this figure, the 1^{st} homology group is a 1-dimensional group, and L_1 (or L_3, equivalently) is its unique generator. The reader is invited to apply this simple idea on the triangulated torus, to find two loops as generators of $\mathcal{H}_1$.

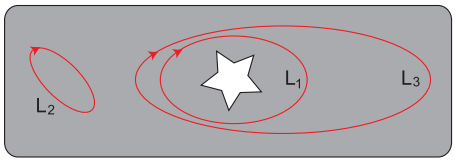


Figure 11: *Example of Homology Classes: the cycles L_1 and L_2 are topologically distinct as one encloses a hole while the other does not; L_1 and L_3 are however in the same equivalence class.*

Formal Definition of Homology Groups We are now ready to generalize this construction to all homology groups. Remember that we have a series of k-chain spaces:

$$C_n \xrightarrow{\partial_n} C_{n-1} \ldots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0$$

with the property that $\partial\ \partial$ is the empty set. This direct implies that image of $\mathcal{C}_j$ is always in the kernel of ∂_{j+1}—such a series is called a *chain complex*. Now, the homology groups $\{\mathcal{H}_k\}_{k=0..n}$ of a chain complex based on ∂ are defined as the following quotient spaces:

$$\mathcal{H}_k = \text{Ker } \partial_k / \text{Im } \partial_{k+1}.$$

The reader is invited to check that this definition is *exactly* what we did for the 0^{th} and 1^{st} homology groups—and it is now valid for any order: indeed, we use the fact that closed chains (belonging to Ker ∂) are homologous *iff* their difference is in Im ∂, and this is exactly what this quotient vector space is.

Example Consider the example in Figure 10(a). Geometrically, $\mathcal{H}_0$ is nontrivial because the simplicial complex σ is disconnected (it is easy to see $\{v_0, v_4\}$ form a basis for $\mathcal{H}_0$), while $\mathcal{H}_1$ is nontrivial since the cycle $(e_1 - e_2 + e_4)$ is not the boundary of any 2-chain of σ ($\{(e_1 - e_2 + e_4)\}$ is indeed a basis for this 1D space $\mathcal{H}_1$).

Link to Betti Numbers The dimension of the k-th cohomology group is called k-th Betti number; $\beta_k = \dim \mathcal{H}_k$. For a 3D simplicial complex embedded in $\mathbb{R}^3$, these numbers have very straightforward meanings. β_0 is the number of connected components, β_1 is the number of tunnels, β_2 is the number of voids, while β_3 is the number of 4D holes, which is 0 in the Euclidean (flat 3D) case. Finally, note that $\sum_{k=0..n}(-1)^k \beta_k$, where β_k is the k-th Betti number, gives us the well-known Euler characteristics.

4.4.3 Cohomology Groups

The definition of homology groups is much more general than what we just reviewed. In fact, the reader can take the formal definition in the previous section, replace all occurrences of chain by cochain, of ∂ by d, and reverse the direction of the operator between spaces (see Section 4.3.2): this will also define equivalence classes. Because cochains are dual of chains, and d is the adjoint of ∂, these equivalence classes define what are actually denoted as *cohomology groups*: the cohomology groups of the deRham complex for the coboundary operator are simply the quotient spaces Ker d/Im d. Finally, note that the homology and cohomology groups are not only dual notions, but they are also isomorphic; therefore, the cardinalities of their basis are equal.

4.4.4 Calculation of the Cohomology Basis

One usual way to calculate a cohomology basis is to calculate a Smith Normal Form to obtain the homology basis first (possibly using progressive mesh [Gu and Yau 2003]), with a worst case complexity is $O(n^3)$, and then find the corresponding cohomology basis derived from this homology basis. We provide an alternative method here with worst case complexity also equal to $O(n^3)$. The advantage of our method is that it directly calculates the cohomology basis.

Our algorithm is a modified version of an algorithm in [Edelsbrunner et al. 2000], although they did not use it for the same purpose[2]. We will use $row\#(.)$ to refer to the row number of the *last non-zero coefficient* in a particular column.

The procedure is as follows:

1. Transform d^k (size $|\mathcal{K}^{k+1}| \times |\mathcal{K}^k|$) in the following manner:

```
// For each column of d^k
for(i = 0; i < |σ^k|; i++)
    // Reduce column i
    repeat
        p ← row#(d^k[i])
        find j < i such as p==row#(d^k[j])
        make d^k[i][p] zero by adding to d^k[i] a multiple of d^k[j]
    until j_not_found or column i is all zeros
```

   In the end of this procedure, we get $D^k = d^k\ N^k$, whose non-zero column vectors are linearly independent of each other and with different $row\#(.)$, and N^k is a non-singular upper triangular matrix.

2. Construct $K^k = \{N_i^k \mid D_i^k = 0\}$ (where N_i^k and D_i^k are column vectors of matrices N^k and D^k respectively).
   K^k is a basis for kernel of d^k.

3. Construct $I^k = \{N_i^k | \exists j \text{ such that } i = row\#(D_j^{(k-1)})\}$

4. Construct $P^k = K^k - I^k$
   P^k is a basis for cohomology basis.

Short proof of correctness: First thing to notice is that N_i^k's are all linearly independent because N^k is nonsingular. For any non-zero linear combination of vectors in P^k , $row\#(.)$ of it (say i) equals to the max of $row\#(.)$ of vectors with non-zero coefficients. But i is not $row\#(.)$ of any $D_i^{(k-1)}$ (and thus any linear combination of them) by definition of P^k. Therefore, we know that the linear combination is not in the image space of d^{k-1} (since the range of d^{k-1} is the same as D^{k-1}, by construction). Thus, P^k spans a subspace of $\text{Ker}(d^k)/\text{Im}(d^{k-1})$ of dimension $\text{Card}(P^k)$.

One can also prove that I^k is a subset of K^k. Pick such an N_i^k with $i = row\#(D_j^{(k-1)})$. We have: $d^k\ D_j^{(k-1)} = 0$ (since $d^k \circ d^{k-1} = 0$). Now $row\#(\tau \equiv (N^k)^{-1}\ {d^{(k-1)}}_j) = i$ (the inverse of an upper triangular matrix is also an upper triangular matrix). So conse-

[2]Thanks to David Cohen-Steiner for pointing us to the similarities

quently, $0 = d^k\ d^{(k-1)}{}_j = D^k(N^k)^{-1}\ d^{(k-1)}{}_j = D^k\tau$ means that $D^k_i = 0$ because the columns of D^k are linearly independent or 0. Therefore, $\mathrm{Card}(P^k) = \mathrm{Card}(K^k) - \mathrm{Card}(I^k) = Dim(\mathrm{Ker}(d^k)) - Dim(\mathrm{Im}(d^{(k-1)}))$, and we conclude that, P^k spans $\mathrm{Ker}(d^k)/\mathrm{Im}(d^{k-1})$ as expected. ∎

4.4.5 Example

Consider the 2D simplicial complex in Figure 10(a) again. We will show an example of running the same procedure described above to compute homology basis. The only difference with the previous algorithm is that we use ∂ instead of d, since we compute the homology basis instead of the cohomology basis.

1. Compute the $D^K = \partial_k N^k$'s and N^k's: D^2 is trivial, as it is the same as ∂_2.

$$D^1 = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad N^1 = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

2. Contruct the K's:

$$K^0 = \{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}\}$$

$$= \{v_0, v_1, v_2, v_3, v_4\}$$

(N^0 is the identity)

$$K^1 = \{\begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \\ 1 \end{pmatrix}, \} = \{(-e_0 - e_1 + e_2 + e_3), (e_1 - e_2 + e_4)\}$$

3. Construct the I's:

$$\begin{aligned} I^0 &= \{v_1\ (1 = row\#(D_0^1)), \\ &\quad v_2\ (2 = row\#(D_1^1), \\ &\quad v_3\ (3 = row\#(D_2^1))\} \\ I^1 &= \{(e_1 - e_2 + e_4)\ (4 = row\#(D_0^2)\} \end{aligned}$$

4. Consequently, the homology basis is:

$$P^0 = \{v_0, v_1, v_2, v_3, v_4\} - \{v_1, v_2, v_3\} = \{v_0, v_4\}$$

$$P^1 = \{(-e_0 - e_1 + e_2 + e_3)\}$$

This result confirms the basis we gave in the example of Section 4.4.2 (Note that $-(-e_0 - e_1 + e_2 + e_3) - (e_1 - e_2 + e_4) = e_0 - e_4 - e_3 = \partial f_0$, thus $(-e_0 - e_1 + e_2 + e_3)$ spans the same homology space as $(e_1 - e_2 + e_4)$).

4.5 Dual Mesh and its Exterior Derivative

Let us introduce the notion of *dual mesh* of triangulated manifolds, as we will see that it is one of the key components of our discrete calculus. The main idea is to associate to each *primal* k-simplex a *dual* $(n$-$k)$-cell. For example, consider the tetrahedral mesh in Figure 13, we associate a dual 3-cell to each primal vertex (0-simplex), a dual polygon (2-cell) to each primal edge (1-simplex), a dual edge (1-cell) to each primal face (2-simplex), and a dual vertex (0-cell) to the primal tet (3-simplex). By construction, the number of dual $(n$-$k)$-cells is equal to that of primal k-simplices. The collection of dual cells is called a *cell complex*, which need not be a simplicial complex in general.

Yet, this dual complex inherits several properties and operations from the primal simplicial complex. Most important is the notion of *incidence*. For instance, if two primal edges are on a same primal face, then the corresponding dual faces are incident, that is, they share a common dual edge (which is the dual of the primal common face). As a result of this incidence property, one may easily derive a boundary operator on the dual cell complex and, consequently, a discrete exterior derivative! The reader is invited to verify that this exterior derivative on the dual mesh can be simply written as the opposite of a primal one *transposed*:

$$d^{n-k}_{Dual} = (-1)^k (d^{k-1}_{Primal})^t. \tag{7}$$

The added negative sign appears as the orientation on the dual is induced from the primal orientation, and must therefore be properly accounted for. Once again, an implementation can overload the definition of this operator d when used on dual forms using this previous equation. In the remainder of our chapter, we will be using d as a contextual operator to keep the notations a simple as possible. Because we have defined a proper exterior derivative on the dual mesh (still satisfying $d \circ d = 0$), this dual cell complex also carries the structure of a chain complex. The structure on the dual complex may be linked to that of the primal complex using the Hodge star (a metric-dependent operator), as we will discuss in Section 5.

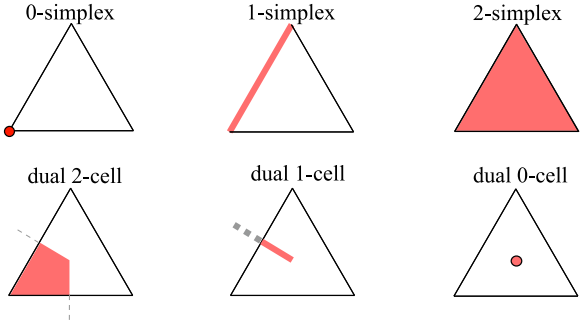


Figure 12: *A 2-dimensional example of primal and dual mesh elements. On the top row, we see the primal mesh (a triangle) with a representative of each simplicial complex being highlighted. The bottom row shows the corresponding circumcentric dual cells (restricted to the triangle).*

4.5.1 Dualization: The $*$ Operator

For simplicity, we use the circumcentric (or Voronoi) duality to construct the dual cell complex. The circumcenter of a k-simplex is defined as the center of the k-circumsphere, which is the unique k-sphere that has all $k+1$ vertices of the k-simplex on its surface. In Figure 12, we show examples of circumcentric dual cells of a 2D mesh. The dual 0-cell associated with the triangular face is the circumcenter of the triangle. The dual 1-cell associated with one of the primal edges is the line segment that joins the circumcenter of the triangle to the circumcenter of that edge, while the dual 2-cell associated with a primal vertex is corner wedge made of the convex hull of the circumcenter of the triangle, the two centers of the adjacent edges, and the vertex itself (see Figure 12, bottom left). Thereafter, we will denote as $*$ the operation of *duality*; that is, a primal simplex σ will have its dual called $*\sigma$ with the orientation induced by the primal orientation and the manifold's orientation. For a formal definition, we refer the reader to [Hirani 2003] for instance. It is also worth noting that other notions of duality such as the barycentric duality may be employed. For further details on dual cell (or "block") decompositions , see [Munkres 1984].

4.5.2 Wedge Product

In the continuous setting, the wedge product $\wedge$ is an operation used to construct higher degree forms from lower degree ones; it is the

antisymmetric part of the tensor product. For example, let α and β be 1-forms on a subset $\mathcal{R} \subset \mathbb{R}^3$, their wedge product $\alpha \wedge \beta$ is a 2-form on $\mathcal{R}$. In this case, one can relate the wedge product to the cross product of vector fields on $\mathcal{R}$. Indeed, if one considers the vector representations of α and β, the vector proxy to $\alpha \wedge \beta$ is the cross product of the two vectors. Similarly, the wedge product of a 1-form γ with the 2-form $\omega = \alpha \wedge \beta$ is a 3-form $\mu = \alpha \wedge \omega$ (also called volume-form) on $\mathcal{R}$ which is analogous to the scalar triple product of three vectors.

A discrete treatment of the wedge operator can be found in [Hirani 2003]. In this work, we only need to introduce the notion of a discrete *primal-dual wedge product*: given a primal k-cochain γ and a dual $(n\text{-}k)$-cochain ω, the discrete wedge product $\gamma \wedge \omega$ is an n-form (or a volume-form). For instance, in the example depicted in the inset, the wedge product of the primal 1-cochain with the dual 1-cochain is a 2-form associated with the diamond region defined by the convex hull of the union between the primal *and* dual edge (see inset).

5 Metric-Dependent Operators on Forms

Notice that up to now, we did *not* assume that a metric was available, *i.e.*, we never required anything to be *measured*. However, such a metric is necessary for many purposes. For instance, simulating the behavior of objects around us requires measurements of various parameters in order to be able to model laws of motion, and compare the numerical results of simulations. Consequently, a certain number of operations on forms can only be defined once a metric is know, as we shall see in this section.

5.1 Notion of Metric and Inner Product

A *metric* is, roughly speaking, a nonnegative function that describes the "distance" between neighboring points of a given space. For example, the Euclidean metric assigns to any two points in the Euclidean space $\mathbb{R}^3$, say $\mathbf{X} = (x_1, x_2, x_3)$ and $\mathbf{Y} = (y_1, y_2, y_3)$, the number:

$$d(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X} - \mathbf{Y}\|^2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

defining the "standard" distance between any two points in $\mathbb{R}^3$. This metric then allows one to measure length, area, and volume. The Euclidean metric can be expressed as the following quadratic form:

$$g^{\text{Euclid}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Indeed, the reader can readily verify that this matrix g satisfies: $d^2(\mathbf{X}, \mathbf{Y}) = (\mathbf{X} - \mathbf{Y})^t g (\mathbf{X} - \mathbf{Y})$. Notice also that this metric *induces an inner product* of vectors. Indeed, for two vectors $\mathbf{u}$ and $\mathbf{v}$, we can use the matrix g to define:

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^t g \, \mathbf{v}.$$

Once again, the reader is invited to verify that this equality does correspond to the traditional dot product when g is the Euclidean metric. Notice that on a non-flat manifold, subtraction of two points is only possible for points infinitesimally close to each other, thus the metric is actually defined pointwise for the tangent space at each point: it does not have to be constant. Finally, notice that a volume form can be induced from a metric by defining $\mu^n = \sqrt{det(g)} dx^1 \wedge \ldots \wedge dx^n$.

5.2 Discrete Metric

In the discrete setting presented in this paper, we only need to measure length, area, volume of the simplices and dual cells. We therefore do not have a full-blown notion of a metric, only a *discrete metric*. Obviously, if one were to use a finer mesh, more information on the metric would be available: having more values of length, area, and volume in a neighborhood provides a better approximation of the real, continuous metric.

5.3 The Differential Hodge Star

Let us go back for a minute to the differential case to explain a new concept. Recall that the metric defines an inner product for vectors. This notion also extends to forms: given a metric, one can define the product of two k-forms $\in \Omega^k(\mathcal{M})$ which will measure, in a way, the projection of one onto the other. A formal definition can be found in [Abraham et al. 1988]. Given this inner product denoted $\langle \ , \ \rangle$, we can introduce an operator $\star$, called the *Hodge star*, that maps a k-form to a complementary $(n\text{-}k)$-form:

$$\star : \ \Omega^k(\mathcal{M}) \to \Omega^{n-k}(\mathcal{M}),$$

and is defined to satisfy the following equality:

$$\alpha \wedge \star\beta = \langle \alpha, \beta \rangle \ \mu^n$$

for any pair of k-forms α and β (recall that μ^n is the volume form induced by the metric g). However, notice that the wedge product is very special here: it is the product of k-form and a $(n\text{-}k)$-form, two complementary forms. This fact will drastically simplify the discrete counterpart of the Hodge star, as we now cover.

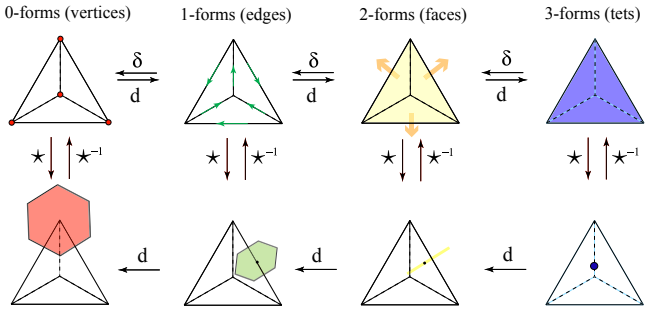


Figure 13: *On the first line, the 'primal' chain complex is depicted and on the second line we see the dual chain complex (i.e., cells, faces, edges and vertices of the Voronoi cells of each vertex of the primal mesh).*

5.4 Discrete Hodge Star

In the discrete setting, the Hodge star becomes easier: we only need to define how to go from a *primal* k-cochain to a *dual* $(n\text{-}k)$-cochain, and vice-versa. By definition of the dual mesh, k-chains and dual $(n\text{-}k)$-chains are represented by vectors of the same dimension. Similarly to the discrete exterior derivative (coboundary) operator, we may use a matrix (this time of size $|\mathcal{K}^k| \times |\mathcal{K}^k|$) to represent the Hodge star. Now the question is: what should the coefficients of this matrix be?

For numerical purposes we want it to be symmetric, positive definite, and sometimes, even diagonal for faster computations. One such diagonal Hodge star can be defined with the diagonal elements as the ratio of sizes of a k-simplex and its dual $(n\text{-}k)$-simplex. In other words, we can define the discrete Hodge star through the following simple rule:

$$\frac{1}{|\sigma^k|} \int_{\sigma^k} \omega = \frac{1}{|*\sigma^k|} \int_{*\sigma^k} \star\omega \qquad (8)$$

Therefore, any primal value of a k-form can be easily *transferred* to the dual mesh through proper scaling—and vice-versa; to be precise, we have:

$$\star_k \star_{n-k} = (-1)^{k(n-k)} \mathrm{Id}, \tag{9}$$

which means that $\star$ on the dual mesh is the inverse of the $\star$ on the primal *up to a sign* (the result of antisymmetry of wedge product, which happens to be positive for any k-form when $n = 3$).

So we must use the inverse of the Hodge star to go from a dual $(n-k)$-cochain to a k-cochain. We will, however, use indistinguishably $\star$ to mean either the star or its inverse, as there is no ambiguity once we know whether the operator is applied to a primal or a dual form: this is also a context-dependent operator.

Implementation Based on Eq. (8), the inner product of forms α^k and β^k at the diamond-shaped region formed by each k-simplex and its dual $(n-k)$-simplex is simply the product of the value of α at that k-simplex and value of $\star\beta$ at that dual $(n-k)$-simplex. Therefore, the sum over the whole space gives the following inner product (which involves only linear algebra matrix and vector multiplications)

$$\langle \alpha^k, \beta^k \rangle = \alpha^t \star \beta. \tag{10}$$

where the Hodge star matrix has, as its only non-zero coefficients, the following diagonal terms:

$$(\star_k)_{qq} = |(*\sigma)_q| / |(\sigma_q)|.$$

Notice that this definition of the inner product, when $\alpha = \beta$, induces the definition of the norm of k-forms.

Again, there are three different Hodge stars in $\mathbb{R}^3$, one for each simplex dimension. But as we discussed for all the other operators, the dimension of the form on which this operator is applied disambiguates which star is meant. So we will not encumber our notation with unnecessary indices, and will only use the symbol $\star$ for any of the three stars implied.

The development of an accurate, yet fast to compute, Hodge star is still an active research topic. However, this topic is beyond the scope of the current paper chapter and will be addressed in a future publication.

5.5 Discrete Codifferential Operator δ

We already have a linear operator d which maps a k-form to a $k+1$-form, but we do not have a linear operator which maps a k-form to a $(k-1)$-form. Having defined a discrete Hodge star, one can now create such an *adjoint* operator δ for the discrete exterior derivative d. Here, adjoint is meant with respect to the inner product of forms; that is, this operator δ satisfies:

$$\langle d\alpha, \beta \rangle \;=\; \langle \alpha, \delta\beta \rangle \qquad \forall \alpha \in \Omega^{k-1}(M), \beta \in \Omega^k(M)$$

For a smooth, compact manifold without boundary, one can prove that $(-1)^{n(k-1)+1} \star d \star$ satisfies the above condition [Abraham et al. 1988]. Let us try to use the same definition in the discrete setting; *i.e.*, we wish to define the discrete δ applied to k-forms by the relation:

$$\delta \equiv (-1)^{n(k-1)+1} \star d \star, \tag{11}$$

Beware that we use the notation d to mean the context-depedent exterior derivative. If you apply δ to a primal k-form, then the exterior derivative will be applied to a *dual* $(n-k)$*-form*, and thus, Equation 7 should be used. Once this is well understood, it is quite straightforward to verify that the following series of equalities:

$$\begin{aligned}
\langle d\alpha, \beta \rangle &\overset{Eq.\ (10)}{=} && (d\alpha)^t \star \beta = \alpha^t d^t \star \beta \\
&\overset{Eq.\ (9)\ w/\ k \leftrightarrow k-1}{=} && \alpha^t (-1)^{(k-1)(n-(k-1))} \star \star d^t \star \beta \\
&= && \alpha^t (-1)^{n(k-1)+1} \star \star (-1)^k d^t \star \beta \\
&\overset{Eq.\ (11)}{=} && \langle \alpha, \delta\beta \rangle
\end{aligned}$$

holds on our discrete manifold. So indeed, the discrete d and δ are also adjoint, in a similar fashion in the discrete setting as they were in the continuous sense. For this reason, δ is called the *codifferential operator*.

Implementation of the Codifferential Operator Thanks to this easily-proven adjointness, the implementation of the discrete codifferential operator is a trivial matter: it is simply the product of three matrices, mimicking exactly the differential definition mentioned in Eq. (11).

5.6 Exercise: Laplacian Operator

At this point, the reader is invited to perform a little exercise. Let us first state that the Laplacian Δ of a form is defined as: $\Delta = \delta d + d\delta$. Now, applied to a 0-form, notice that the latter term disappears. Question: in 2D, what *is* the Laplacian of a function f at a vertex i? The answer is actually known: it is the now famous *cotangent formula* [Pinkall and Polthier 1993], since the ratio of primal and dual edge sizes leads to such a trigonometric equality.

6 Interpolation of Discrete Forms

In Section 3.4, we argued that k-cochains are discretizations of k-forms. This representation of discrete forms on chains, although very convenient in many applications, is not sufficient to fulfill certain demands such as obtaining a point-wise value of the k-form. As a remedy, one can use an interpolation of these chains to the rest of space. For simplicity, these interpolation functions can be taken to be linear (by linear, we mean with respect to the coordinates of the vertices).

6.1 Interpolating 0-forms

It is quite obvious to linearly interpolate discrete 0-forms (as 0-cochains) to the whole space: we can use the usual vertex-based linear interpolation basis, often referred to as the *hat function* in the Finite Element literature. This basis function will be denoted as φ_i for each vertex v_i. By definition, φ_i satisfies:

$$\varphi_i = 1 \quad \text{at } v_i, \qquad \varphi_i = 0 \quad \text{at } v_j \neq v_i$$

while φ_i linearly goes to zero in the one-ring neighborhood of v_i. The reader may be aware that these functions are, within each simplex, *barycentric coordinates*, introduced by Möbius in 1827 as mass points to define a *coordinate-free geometry*.

With these basis functions, one can easily check that if we denote a vertex v_j by σ_j, we have:

$$\int_{v_j} \varphi_{v_i} = \int_{\sigma_j} \varphi_{\sigma_i} = \int_{\sigma_j} \varphi_i = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Therefore, these interpolating functions represent a basis of 0-cochains, that exactly corresponds to the dual of the natural basis of 0-chains.

6.2 Interpolating 1-forms

We would like to be able to extend the previous interpolation technique to 1-forms now. Fortunately, there is an existing method to do just that: the *Whitney* 1-form (used first in [Whitney 1957]) associated with an edge σ_{ij} between v_i and v_j is defined as:

$$\varphi_{\sigma_{ij}} = \varphi_i d\varphi_j - \varphi_j d\varphi_i.$$

A direct computation can verify that:

$$\int_{\sigma_{kl}} \varphi_{\sigma_{ij}} = \begin{cases} 1 & \text{if } i=k \text{ and } j=l, \\ -1 & \text{if } i=l \text{ and } j=k, \\ 0 & \text{otherwise.} \end{cases}$$

Indeed, it is easy to see that the integral is 0 when we are not integrating it on edge e_{ij}, because at least one of the vertex (say, i) is not on the edge, thus, $\varphi_i = 0$ and $d\varphi_i = 0$ on the edge. However, along the edge σ_{ij}, we have $\varphi_i + \varphi_j = 1$, therefore:

$$\int_{\sigma_{ij}} \varphi_{\sigma_{ij}} = \int_{\varphi_i=1}^{\varphi_i=0} (\varphi_i d(1-\varphi_i) - (1-\varphi_i)d\varphi_i) = \int_{\varphi_i=1}^{\varphi_i=0} (-d\varphi_i) = 1.$$

We thus have defined a correct basis for 1-cochains.

6.3 Interpolating with Whitney k-Forms

One can extend these 1-form basis functions to arbitrary k-simplices. In fact, Whitney k-forms are defined similarly:

$$\varphi_{\sigma_{i_0,i_1,\ldots,i_k}} = k! \sum_{j=0\ldots k} (-1)^j \varphi_{i_j} d\varphi_{i_0} \wedge \ldots \wedge \widehat{d\varphi_{i_j}} \ldots \wedge d\varphi_{i_k}$$

where $\widehat{d\varphi_{i_p}}$ means that $d\varphi_{i_p}$ is excluded from the product. Notice how this definition exactly matches the case of vertex and edge bases, and extends easily to higher dimensional simplices.

Remark If a metric is defined (for instance, the Euclidean metric), we can simply identify $d\varphi$ with $\nabla\varphi$ for the real calculation. This corresponds to the notion of *sharp* ($\sharp$), but we will not develop this point other than for pointing out the following remark: the traditional gradient of a linear function f in 2D, known to be constant per triangle, can indeed be re-written à la Whitney:

$$\nabla f = \sum_i f_i \nabla\varphi_i \overset{\varphi_i+\varphi_j+\varphi_k=1}{=} \sum_{i,j,i\neq j} (f_i - f_j)(\varphi_i \nabla\varphi_j - \varphi_j \nabla\varphi_i).$$

The values $(f_i - f_j)$ are the edge values associated with the gradient, *i.e.*, the values of the one-form df.

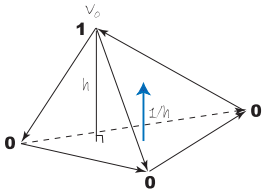


Figure 14: $\nabla\varphi$ *for the vertex on top*

Basis of Forms The integration of the Whitney form φ_{σ_k} associated with the k-simplex σ_k will be 1 on that particular simplex, and 0 on all others. Indeed, it is a simple exercise to see that the integration of φ_{σ_k} is 0 on a different k-simplex, because there is *at least one* vertex of this simplex v_j that does not belong to σ_k, so its hat function φ_j is valued 0 everywhere on σ_k. Since φ_j or $d\varphi_j$ appears in every term, the integral of φ_{σ_k} is 0. To see that the integral is 1 on the simplex itself, we can use Stokes' theorem (as our discrete forms satisfy it exactly on simplices): first, suppose $k < n$, and pick a $k+1$-simplex, such that the k-simplex σ_k is a *face* of it. Since it is 0 on other faces, the integral of the Whitney form is equal to the integral of $d\varphi_{\sigma_k} = (k+1)! d\varphi_{i_0} \wedge \ldots \wedge \varphi i_k$ on the $k+1$-simplex, if we use φ_{i_j} as a local reference frame for the integration, $\int_{\sigma^{k+1}} d\varphi_{i_0} \wedge \ldots \wedge \varphi_{i_k}$ is simply the volume of a standard simplex, which is $\frac{1}{(k+1)!}$, thus the integral is 1. The case when $k = n$ is essentially the same as $k = n-1$.

This means that these Whitney forms are forming a basis of their respective form spaces. In a way, these bases are an extension of the Finite Element bases defined on nodes, or of the Finite Volume elements that are constant per tet.

Note finally that the Whitney forms are not continuous; however, they *are* continuous *along the direction of the k-simplex* (*i.e.*, tangential continuity for 1-forms, and normal continuity for 2-forms); this is the only condition needed to make the integration well defined. In a way, this property is the *least* we can ask them to be. We would lose generality if we were to add any other condition! The interested reader is referred to [Bossavit 1998] for a more thorough discussion on these Whitney bases and their relations to the notion of weak form used in the Finite Element Method.

7 Application to Hodge Decomposition

We now go through a first application of the discrete exterior calculus we have defined up to now. As we will see, the discrete case is often much simpler that its continuous counterpart; yet it captures the same properties.

7.1 Introducing the Hodge Decomposition

It is convenient in some applications to use the Helmholtz-Hodge decomposition theorem to decompose a given continuous vector field or differential form (defined on a smooth manifold $\mathcal{M}$) into components that are mutually orthogonal (in $\mathcal{L}^2$ sense), and easier to compute (see [Abraham et al. 1988] for details). In fluid mechanics for example, the velocity field is generally decomposed into a part that is the gradient of a potential function and a part that is the curl of a stream vector potential (see Section 8.3 for further details), as the latter one is the incompressible part of the flow. When applied to k-forms, this decomposition is known as the *Hodge decomposition for forms* and can be stated as follows:

Given a manifold $\mathcal{M}$ and a k-form ω^k on $\mathcal{M}$ with appropriate boundary conditions, ω^k can be decomposed into the sum of the exterior derivative of a (k-1)-form α^{k-1}, the codifferential of a (k+1)-form β^{k+1}, and a harmonic k-form h^k:

$$\omega^k = d\alpha^{k-1} + \delta\beta^{k+1} + h^k.$$

Here, we use the term *harmonic* to mean that h^k satisfies the equation $\Delta h^k = 0$, where Δ is the Laplacian operator defined as $\Delta = d\delta + \delta d$. The proof of this theorem is mathematically involved and requires the use of elliptic operator theory and similar tools, as well as a careful study of the boundary conditions to ensure uniqueness. The discrete analog that we propose has a very simple and straightforward proof as shown below.

7.2 Discrete Hodge Decomposition

In the discrete setting, the discrete operators such as the exterior derivative and the codifferential can be expressed using matrix representation. This allows one to easily manipulate these operators using tools from linear algebra. In particular, the discrete version of the Hodge decomposition theorem becomes a simple exercise in linear algebra. Note that we will assume a boundaryless domain for simplicity (the generalization to domains with boundary is conceptually as simple).

Theorem 7.1 *Let $\mathcal{K}$ be a discrete manifold and let $\Omega^k(\mathcal{K})$ be the space of discrete Whitney k-forms on $\mathcal{K}$. Consider the linear operator $d^k : W^k \to W^{k+1}$, such that $d^{k+1} \circ d^k = 0$, and a discrete Hodge star which is represented as a symmetric, positive definite matrix. Furthermore, define the codifferential (the adjoint of the operator d) as done in Section 5.5; namely, let $\delta^{k+1} = (-1)^{n(k-1)+1}(\star^k)^{-1}(d^k)^t\star^{k+1}$. In this case, the following orthogonal decomposition holds for all k:*

$$\Omega^k(\mathcal{K}) = d\Omega^{k-1}(\mathcal{K}) \oplus \delta\Omega^{k+1}(\mathcal{K}) \oplus \mathcal{H}^k(\mathcal{K})$$

where $\oplus$ means orthogonal sum, and $\mathcal{H}^k(\mathcal{K})$ is the space of harmonic k-forms on $\mathcal{K}$, that is, $\mathcal{H}^k(\mathcal{K}) = \{h \mid \Delta^k h = 0\}$.

Proof For notational convenience, we will omit the superscript of the operators when the rank is obvious. We first prove that the three component spaces are orthogonal. Clearly, using the facts that the Laplacian operator Δ is equal to $d\delta$+δd and that d and δ are adjoint operators, one has that $\forall h \in \mathcal{H}^k$:

$$\begin{aligned}\langle \Delta h, h\rangle = 0 \;&\Rightarrow\; \langle d\delta h, h\rangle + \langle \delta dh, h\rangle = \langle dh, dh\rangle + \langle \delta h, \delta h\rangle = 0\\ &\Rightarrow\; dh = 0 \text{ and } \delta h = 0\end{aligned}$$

Also, $\forall\alpha$ and $\beta \in \Omega^k(\mathcal{K})$, one has:

$$\langle d\alpha, \delta\beta\rangle = \langle dd\alpha, \beta\rangle = 0$$

and

$$\langle d\alpha, h\rangle = \langle \alpha, \delta h\rangle = 0 \qquad \langle h, \delta\beta\rangle = \langle dh, \beta\rangle = 0$$

Now, any k-form that is perpendicular to $d\Omega^{k-1}(\mathcal{K})$ and $\delta\Omega^{k+1}(\mathcal{K})$ must be in $\mathcal{H}^k(\mathcal{K})$, because this means $dh = 0$ and $\delta h = 0$, so $\Delta h = d\delta h + \delta dh = 0$.

Alternatively, we can prove that:

$$\Omega^k(\mathcal{K}) = \Delta\Omega^k(\mathcal{K}) \oplus \mathcal{H}^k(\mathcal{K}).$$

By analogy to the previous argument, it is easy to show that $\Delta\Omega^k$ is orthogonal to $\mathcal{H}^k$. Additionally, the dimension of these two spaces sum up to the dimension of Ω^k, which means the decomposition is complete. ∎

Note that the reader can find a similar proof given in Appendix B of [Frankel 2004], where it is used for Kirchhoff's Circuit Laws. There, Frankel does not mention that we can actually use cochains as the discretization of forms, and his operations using a "metric" of cochains can be interpreted as a Hodge star.

Implementation of the Discrete Hodge Decomposition

Before we discuss how to numerically implement the discrete Hodge decomposition, we prove a useful result (that has a continuous analog).

Lemma 7.2 *In the discrete setting, one can find exactly one harmonic cochain from each cohomology equivalence class.*

Proof It is can be readily shown that the bases of harmonic cochains and the cohomology groups both have the dimension equal to $dim(\text{Ker } d^k) - dim(\text{Im } d^{k-1})$. To this end, recall that a cohomology basis is defined as is $\text{Ker}(d^k)/\text{Im}(d^{k-1})$ and has dimension $dim(\text{Ker } d^k) - dim(\text{Im } d^{k-1})$. Now, in order to see that the space of harmonic cochains has this same dimension, simply note that: $\text{Ker}(d^k) = d\Omega^{k-1} \oplus \mathcal{H}^k$.

Now, the equation $\delta(\omega + df) = 0$ has a solution for each ω in one cohomology equivalence class. We know that the cochains forming different cohomology groups are linearly independent, hence, we conclude that these harmonic cochains span $\mathcal{H}^k$. ∎

By virtue of the above lemma, the implementation of the Hodge decomposition is simply recursive in the rank of the form (i.e., cochain). The case of 0-forms is trivial: fix one vertex to a constant, and solve the Poisson equation for 0-forms. Now suppose that we have a decomposition working for $(k-1)$-forms, and we look for the decomposition of k-forms. Our approach is to get the harmonic component h^k first, so that we only need to solve a Poisson equation for the rest:

$$\Delta\omega^k = f^k - h^k \tag{12}$$

One is left with the problem of finding a basis of harmonic forms. Since we are given a Hodge star operator, we will use it to define the metric on the space of cochains. This metric allows us to define a basis for harmonic k-form (the dimension of this harmonic space is generally small, since it is the k-th Betti number β_k). First, one needs to calculate the cohomology basis $\{P_i\}$ based on the algorithm in Section 4.4.4. Once we have $\{P_i\}$, we solve one special decomposition of (k-1)-forms by first computing the forms f_i satisfying:

$$\Delta f_i = -\delta P_i \tag{13}$$

Now $H^k = P_i + df_i$ gives us the forms in basis for harmonic k-form space. After normalization, we have the basis to calculate the projection $h^k = HH^t f^k$, where we assemble all H^k into a matrix H. This completes the procedure of calculating the decomposition.

A non-singular matrix is often preferable when it comes to solve a linear system efficiently; we can change the Laplacian matrix slightly to make the Poisson equation satisfy this requirement. First, we can get an orthonormal basis for harmonic form space (the dimension is β^k). Now for basis e^j (column vector with j-th element equal to 1, and 0 everywhere else), take the distance of e^j to the harmonic space $|e^j - HH^t e^j|$; notice that this can be done in constant time. Now take out the j-th column and j-th row of Δ if e^j has the smallest distance from harmonic space, and repeat the step for β^k times. We are left with a non-singular matrix, and the solution to the new linear system is a solution to the original Poisson equation.

8 Others Applications

8.1 Form-based Proof of Tutte's Theorem

The notion of forms as convenient, intrinsic substitutes for vector fields has been used to provide a concise proof of the celebrated *Tutte's Embedding Theorem*. This important result in graph theory states that if one fixes the boundary of a 3-connected graph (i.e., a typical polygonal mesh) to a convex domain in the plane and ensures that every non-boundary vertex is a *strict convex combination* of its neighbors, then one obtains a planar straight-line embedding of the graph. In other words, this embedding procedure will not result in fold-overs. A significantly shorter alternative to the original proof of this theorem was proposed by Gortler, Gotsman, and Thurston [Gortler et al. 2006], using discrete 1-forms on edges. We now present a sketch of their approach, using a formulation more in line with the terms we used in this paper.

A Tutte embedding assigns to each vertex v_i of a graph G some 2D coordinates $\mathbf{X}(v_i) = (x(v_i), y(v_i))$. By definition, each interior vertex v_i satisfies a linear condition on its coordinates of the form: $\mathbf{X}(v_i) = \sum_{v_j \in \mathcal{N}(i)} w_{ij} \mathbf{X}(v_j)$, where $\mathcal{N}(i)$ is the set of 1-ring neighbors of vertex v_i. These coefficients w_{ij} are all non-negative due to the condition of strict convex combination mentioned above. Now, for a given Tutte embedding, one can construct a 0-form $z(v) = \alpha x(v) + \beta y(v)$ for any pair of positive coefficients α and β. Notice that this 0-form satisfies the same convex combination condition: $z(v_i) = \sum_{v_j \in \mathcal{N}(i)} w_{ij} z(v_j)$. As they are non negative, one can identify these coefficients w_{ij} to the diagonal Hodge star of primal 1-forms (see Section 7) defined by a particular metric. Therefore, the relationship $0 = \sum_{v_j \in \mathcal{N}(i)} w_{ij}(z(v_j) - z(v_i))$ is equivalent to: $d \star dz = 0$. There are two immediate conclusions:

- the 1-form $\omega = dz$ is closed (since it is the exterior derivative of a 0-form), and
- it is also co-closed since $\delta\omega = (\star d \star) dz = \star(d \star dz) = 0$.

To use the previously defined 1-form ω to prove Tutte's theorem, Gortler *et al.* then invoke the usual definition of index of vector fields, *i.e.*, the number of revolutions that the direction of the vector fields does along any small curve around this vertex. This concept is one of the oldest in Algebraic Topology, initially stated by Poincaré and then developed by Hopf and Morse in the continuous case. Its discrete counterpart was first proposed by Banchoff, and used for instance in [Lazarus and Verroust 1999]. A discrete Poincare-Hopf index theorem also holds, stating that the sum of all indices must be equal to 2 for a genus-0 patch. The final argument uses the link between (co)closed forms and their indices. Indeed, because we found a closed *and* coclosed form ω, it can be easily shown that these two properties induce that the index of each face must be less or equal to zero, as well as the index of each vertex. Because the boundary of the patch is convex, only two vertices on the boundary have index 1. Since all the indices must sum to 2 and each interior index must be less than zero, we can conclude that *each interior index is zero*. Because this argument is valid for every positive pair (α, β), one can easily deduce that each interior face is convex and each vertex is a "wheel"; thus, injectivity can be guaranteed.

This rather elegant proof demonstrates how discrete forms and their obvious links to Algebraic Topology can be quite powerful in a variety of applications. We also point the interested reader to other papers, such as [Mercat 2001; Gu and Yau 2003], for which special discrete Hodge stars are defined to satisfy a discrete definition of conformality: there are also very interesting research on this particular topic, once again using the calculus of exterior forms.

8.2 Electromagnetism with Forms

Electromagnetism can be formulated very elegantly using differential forms. For a detailed exposition of the geometric structure in E&M, we refer the reader to [Bossavit 1998] and [Warnick et al. 1997]. In this approach, the electric field E is represented by a 1-form as the integral of E along a path traced by a test charge q, and is equal to the electromotive force experienced by that charge. The electric displacement L as well as the current density J are represented by 2-forms. The charge distribution ρ is a 3-form. The magnetic field B is represented by a 2-form since it is measured as a flux. whereas the magnetic field intensity H is a 1-form.

With these conventions, Maxwell's equations can be rewritten as follows:

$$\partial_t B + dE = 0, \qquad -\partial_t L + dH = J, \qquad dL = \rho, \qquad (14)$$

subject to the *constitutive* equations:

$$L = \epsilon E, \qquad H = \mu B, \qquad (15)$$

where ϵ is the permittivity, and μ is the permeability. The constitutive relations (15) are very similar to the Hodge star operator that transforms a k-form to an $(n\text{-}k)$-form. Here, ϵ operates on the electric field E (1-form) to yield the electric displacement L (2-form) while μ transforms the magnetic field B (2-form) into the magnetic field intensity H (1-form). To this end, one may think of both ϵ and μ as Hodge star operators induced from appropriately chosen metrics. Note that the balance laws in (14) are metric-independent.

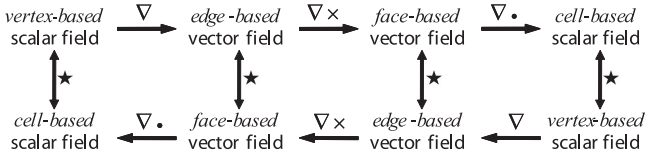
As the reader can guess, one can readily discretize this representation of the physical quantities E, L, ... and the associated system of equations (14-15) using the tools presented in this chapter. The resulting numerical algorithm preserves *exactly* the geometric structure of the system, see [Bossavit 1998].

8.3 Fluids

The geometric structure of Fluid Mechanics, specifically Euler's equations for inviscid fluids, has been investigated (see [Marsden and Weinstein 1983] and references therein). In this geometric framework, vorticity is represented as a two-form (an area-form) and Euler's equations can be written as vorticity advection. Roughly speaking, vorticity measures the rotation of a fluid parcel; we say the fluid parcel has vorticity when it spins as it moves along its path. Vorticity advection means that the vorticity (as a two-form) moves dynamically as if it is pushed forward by the fluid flow. The integral of the vorticity on a given bounded domain is equal, by Stokes theorem, to the circulation around the loop enclosing the domain. This quantity as the loop is advected by the fluid is conserved in the absence of external forcing, as well as the total energy of the fluid. Inspired by this geometric viewpoint and in light of the present development of Discrete Exterior Calculus, we have proposed a discrete differential approach to fluid mechanics and an integration scheme that satisfy the properties of conservation of circulation, see [Elcott et al. 2005] for further details.

9 Conclusions

In this chapter, we have provided an introduction to discrete differential forms and explained how they can be extremely useful in computational science. A convenient Discrete Exterior Calculus solely based on values stored on a discrete manifold has been derived. In the common 3D case, this calculus for scalar and vector fields can be summarized by the following schematic graph:



We have also given a discrete version of the Hodge decomposition, useful for a number of computations in various fields. This geometric approach to computations is particularly novel, thus many details need to be explored and proven superior to the current approaches. In order to work towards this goal, more work needs to be done to further demonstrate that this idea of forms as fundamental readily-discretizable elements of differential equations can be successfully used in various other contexts where predictive power is crucial.

Acknowledgments

The authors wish to first thank Jerrold E. Marsden for his tremendous support along the way. Peter Schröder and Herbert Edelsbrunner helped us with a thorough proofreading of this document. We also wish to acknowledge the support of Alain Bossavit, Anil

Hirani, Melvin Leok, David Cohen-Steiner, Sharif Elcott, Pierre Alliez, and Eitan Grinspun.

References

ABRAHAM, R., MARSDEN, J., AND RATIU, T., Eds. 1988. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences Vol. 75, Springer.

BJÖRNER, A., AND WELKER, V. 1995. The homology of "k-equal" manifolds and related partition lattices. *Advances in Math. 110*, 277–313.

BOBENKO, A., AND SEILER, R., Eds. 1999. *Discrete Integrable Geometry and Physics*. Clarendon Press.

BOSSAVIT, A. 1998. *Computational Electromagnetism*. Academic Press, Boston.

BURKE, W. L. 1985. *Applied Differential Geometry*. Cambridge University Press.

CARROLL, S. 2003. *Spacetime and Geometry: An Introduction to General Relativity*. Pearson Education.

CARTAN, É. 1945. *Les Systèmes Differentiels Exterieurs et leurs Applications Géometriques*. Hermann, Paris.

DESBRUN, M., LEOK, M., AND MARSDEN, J. E. 2004. Discrete Poincaré Lemma . *Appl. Num. Math.*.

DIMAKIS, A., AND MÜLLER-HOISSEN, F. 1994. Discrete Differential Calculus, Graphs, Topologies, and Gauge Theory. *Journal of Mathematical Physics 35*, 6703–6735.

DORAN, C., AND LASENBY, A., Eds. 2003. *Geometric Algebra for Physicists*. Cambridge University Press.

EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN, A. 2000. Topological persistence and simplification. In *IEEE Symposium on Foundations of Computer Science*, 454–463.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2005. Discrete, vorticity-preserving, and stable simplicial fluids. In *Chapter 9 in ACM SIGGRAPH Lecture Notes on Discrete Differential Geometry*.

FLANDERS, H. 1990. *Differential Forms and Applications to Physical Sciences*. Dover Publications.

FLANDERS, H., Ed. 2001. *Geometric Methods for Computational Electromagnetics*. EMW Publishing, Cambridge Mass.

FORMAN, R. Bochner's Method for Cell Complexes and Combinatorial Ricci Curvature. *J. of Discrete and Computational Geom. 29*, 3, 323–374.

FRANKEL, T. 2004. *The Geometry of Physics*. Second Edition. Cambridge University Press, United Kingdom.

GORTLER, S., GOTSMAN, C., AND THURSTON, D. 2006. One-Forms on Meshes and Applications to 3D Mesh Parameterization. *Computer Aided Geometric Design 23*, 2, 83–112.

GROSS, P. W., AND KOTIUGA, R. 2004. *Electromagnetic Theory and Computation: A Topological Approach*. Cambridge University Press.

GU, X., AND YAU, S.-T. 2003. Global conformal surface parameterization. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, Eurographics Association, 127–137.

HARRISON, J. 2005. Ravello Lecture Notes on Geometric Calculus – Part I. Tech. rep., UC Berkeley.

HATCHER, A. 2004. *Algebraic Topology*. Cambridge University Press.

HILDEBRANDT, K., AND POLTHIER, K. 2004. Anisotropic filtering of non-linear surface features. In *Computer Graphics Forum*, M.-P. Cani and M. Slater, Eds., vol. 23. Proc. Eurographics 2004.

HIRANI, A. N. 2003. *Discrete Exterior Calculus*. PhD thesis, Caltech.

HYMAN, J. M., AND SHASHKOV, M. 1997. Natural Discretizations for the Divergence, Gradient, and Curl. *International Journal of Computers and Mathematics with Applications 33*.

KANSO, E., ARROYO, M., DESBRUN, M., MARSDEN, J. E., AND TONG, Y. 2005. On the Geometric Character of Continuum Mechanics. *preprint*.

LAZARUS, F., AND VERROUST, A. 1999. Level Set Diagrams of Polyhedral Objects. In *Proceedings of the 5^{th} ACM Symposium on Solid Modeling and Applications*, 130–140.

LOVELOCK, D., AND RUND, H. 1993. *Tensors, Differential Forms, and Variational Principles*. Dover Publications.

MARSDEN, J. E., AND WEINSTEIN, A. 1983. Coadjoint orbits, vortices and Clebsch variables for incompressible fluids. *Physica D 7*, 305–323.

MARSDEN, J. E., AND WEST, M. 2001. Discrete Mechanics and Variational Integrators. *Acta Numerica*.

MERCAT, C. 2001. Discrete Riemann Surfaces and the Ising Model. *Commun. Math. Phys. 218*, 1, 177–216.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2002. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Proceedings of VisMath*.

MORITA, S. 2001. *Geometry of Differential Forms*. Translations of Mathematical Monographs, Vol. 201. Am. Math. Soc.

MUNKRES, J. R. 1984. *Elements of Algebraic Topology*. Addison-Wesley, Menlo Park, CA.

PINKALL, U., AND POLTHIER, K. 1993. Computing Discrete Minimal Surfaces. *Experimental Mathematics 2*, 1, 15–36.

RAMASWAMY, V., AND SHAPIRO, V. 2004. Combinatorial Laws for Physically Meaningful Design. *J. of Computing and Information Science in Engineering 4*, 1, 3–10.

SCHREIBER, U. 2003. On Superstrings in Schrödinger Representation. *Preprint*.

SHARPE, R. W. 1997. *Differential Geometry:Cartan's Generalization of Klein's Erlangen Programme*. Springer-Verlag, NY.

WARNICK, K. F., SELFRIDGE, R. H., AND ARNOLD, D. V. 1997. Teaching Electromagnetic Field Theory Using Differential Forms. *IEEE Trans. on Education 40*, 1, 53–68.

WHITNEY, H. 1957. *Geometric Integration Theory*. Princeton Press, Princeton.

ZAPATRIN, R. 1996. Polyhedral Representations of Discrete Differential Manifolds. *Preprint*.

Further Reading

Despite a large number of theoretical books, we are aware of only a few books with a truly "applied flavor", in line with this chapter. For applications based on this exterior calculus or other geometric algebras, see [Bossavit 1998; Flanders 2001; Bobenko and Seiler 1999; Doran and Lasenby 2003; Gross and Kotiuga 2004; Ramaswamy and Shapiro 2004]. The reader interested in the application of differential forms to E&M is further referred to [Warnick et al. 1997], for applications in fluid mechanics see [Marsden and Weinstein 1983], and in elasticity see [Kanso et al. 2005] and [Frankel 2004]. The reader is also invited to check out current developments of variants of DEC, for instance, in [Dimakis and Müller-Hoissen 1994; Schreiber 2003; Zapatrin 1996; Harrison 2005].

Finally, the interested reader can find additional material on the following websites: Graphics and Applied Geometry at Caltech:
http://multires.caltech.edu/pubs/
http://www.geometry.caltech.edu/
Computational E&M (Alain Bossavit):
http://www.lgep.supelec.fr/mse/perso/ab/bossavit.html
Discrete Vector Fields and Combinatorial Topology (R. Forman):
http://math.rice.edu/∼forman/
Discrete Mechanics at Caltech (Jerrold E. Marsden):
http://www.cds.caltech.edu/∼marsden/

Chapter 8: Building Your Own DEC at Home

Sharif Elcott
Caltech

Peter Schröder
Caltech

1 Overview

The methods of Discrete Exterior Calculus (DEC) have given birth to many new algorithms applicable to areas such as fluid simulation, deformable body simulation, and others. Despite the (possibly intimidating) mathematical theory that went into deriving these algorithms, in the end they lead to simple, elegant, and straightforward implementations. However, readers interested in implementing them should note that the algorithms presume the existence of a suitable simplicial complex data structure. Such a data structure needs to support local traversal of elements, adjacency information for all dimensions of simplices, a notion of a *dual mesh*, and all simplices must be *oriented*. Unfortunately, most publicly available tetrahedral mesh libraries provide only *unoriented* representations with little more than vertex-tet adjacency information (while we need vertex-edge, edge-triangle, edge-tet, *etc.*). For those eager to implement and build on the algorithms presented in this course without having to worry about these details, we provide an implementation of a DEC-friendly tetrahedral mesh data structure in C++. This chapter documents the ideas behind the implementation.

1.1 Motivation

Extending a classic pointer-based mesh data structure to 3D is unwieldy, error-prone, and difficult to debug. We instead take a more abstract set-oriented view in the design of our data structure, by turning to the formal definition of an abstract simplicial complex. This gives our implementation the following desirable properties:

- We treat the mesh as a graph and perform all of our operations combinatorially.
- There is no cumbersome pointer-hopping typical of most mesh data structures.
- The design easily generalizes to arbitrary dimension.
- The final result is very compact and simple to implement.

In effect we are taking advantage of the fact that during assembly of all the necessary structures one can use high level, abstract data structures. That way formal definitions can be turned into code almost verbatim. While these data structures (*e.g.*, sets and maps) may not be the most efficient for *computation*, an approach which uses them during assembly is far less error prone. Once everything has been assembled it can be turned easily into more efficient packed representations (*e.g.*, compressed row storage format sparse matrices) with their more favorable performance during the actual computations which occur, *e.g.*, in physical simulation.

1.2 Outline

We will begin with a few definitions in Section 2, and see how these translate into our tuple-based representation in Section 3. The boundary operator, described in Section 4, facilitates mesh traversal and implements the discrete exterior derivative. We show how everything is put together in Section 5. Finally, we discuss our implementation of the DEC operators in Section 6.

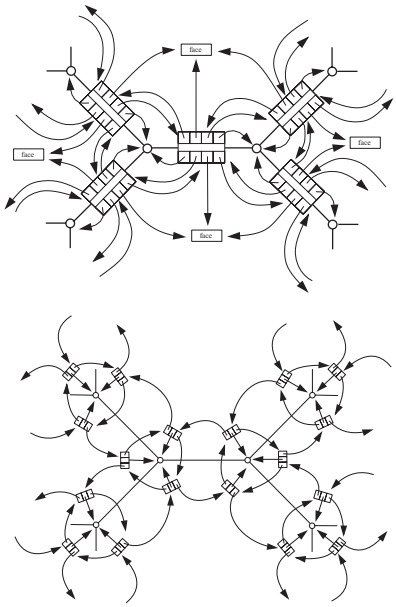


Figure 1: *Some typical examples of 2D mesh representations (from [Joy et al. 2002]; used with permission). Such pointer-based data structures become quite difficult to manage once they are extended to 3D.*

2 Definitions

We begin by recalling the basic definitions of the objects we are dealing with. The focus here is on the rigorous mathematical definitions in a form which then readily translates into high level algorithms. The underlying concepts are simply what we all know informally as *meshes* in either two (triangle) or three (tet) dimensions.

Simplices A *simplex* is a general term for an element of the mesh, identified by its dimension. 0-simplices are vertices, 1-simplices are edges, 2-simplices are triangles, and 3-simplices are tetrahedra.

Abstract Simplicial Complex This structure encodes all the relationships between vertices, edges, triangles, and tets. Since we are only dealing with combinatorics here the atomic element out of which everything is built are the integers $0 \leq i < n$ referencing the underlying vertices. For now they do not yet have point positions in space. Formally, an abstract simplicial complex is a *set of subsets* of the integers $0 \leq i < n$, such that if a subset is contained in the complex then so are all its subsets. For example, a 3D complex is a collection of tetrahedra (4-tuples), triangles (3-tuples), edges (2-tuples), and vertices (singletons), such that if a tetrahedron is present in the complex then so must be its triangles, edges, and

vertices. All our simplicial complexes will be proper three or two manifolds, possibly with boundary and may be of arbitrary topology (*e.g.*, containing voids and tunnels).

Manifold The DEC operators that we build on are defined only on meshes which represent manifolds. Practically speaking this means that in a 3D simplicial complex all triangles must have two incident tets only (for a boundary triangle there is only one incident tet). Every edge must have a set of tets incident on it which form a single "ring" which is either open (at the boundary) or closed (in the interior). Finally for vertices it must be true that all incident tets form a topological sphere (or hemisphere at the boundary). These properties should be asserted upon reading the input. For example, for triangles which bound tets one must assert that each such triangle occurs in at most two tets. For an edge the "ring" property of incident tets can be checked as follows. Start with one incident tet and jump across a shared triangle to the next tet incident on the edge. If this walk leads back to the original tet *and* all tets incident on the edge can thusly be visited, the edge passes the test. (For boundary edges such a walk starts at one boundary tet and ends at another.) The test for vertices is more complex. Consider all tets incident on the given vertex. Using the tet/tet adjacency across shared triangles one can build the adjacency graph of all such tets. This graph must be a topological sphere (or hemisphere if the vertex is on the boundary).

Since we need everything to be properly oriented we will only allow *orientable* manifolds (*i.e.*, no Möbius strips or Klein bottles).

Regularity To make life easier on ourselves we also require the simplicial complex to be *strongly regular*. This means that simplices must not have identifications on their boundaries. For example, edges are not allowed to begin and end in the same vertex. Similarly, the edges bounding a triangle must not be identified nor do we allow edges or triangles bounding a tet to be identified. In practice this is rarely an issue since the underlying geometry would need to be quite contorted for this to occur. Strictly speaking though such identifications are possible in more general, abstract settings without violating the manifold property.

Embedding It is often useful to distinguish between the *topology* (neighbor relationships) and the *geometry* (point positions) of the mesh. A great deal of the operations performed on our mesh can be carried out using only topological information, *i.e.*, without regard to the embedding. The embedding of the complex is given by a map $p : [0,n) \mapsto (x,y,z) \in \mathbb{R}^3$ on the vertices (which is extended piecewise linearly to the interior of all simplices). For example, when we visualize a mesh as being composed of piecewise linear triangles (for 2D meshes) or piecewise linear tets, we are dealing with the geometry. Most of the algorithms we describe below do not need to make reference to this embedding. When implementing these algorithms it is useful to only think in terms of combinatorics. There is only one stage where we care about the geometry: the computation of metric dependent quantities needed in the definition of the Hodge star.

3 Simplex Representation

Ignoring orientations for a moment, each k-simplex is represented as a $(k+1)$-tuple identifying the vertices that bound the simplex. In this view a tet is simply a 4-tuple of integers, a triangle is a 3-tuple of integers, an edge is a 2-tuple, and a vertex is a singleton. Note that all permutations of a given tuple refer to the same simplex. For example, (i,j,k) and (j,i,k) are different *aliases* for the same triangle. In order to remove ambiguities, we must designate one *representative* alias as the representation of the simplex in our data structures. We do this by using the *sorted* permutation of the tuple. Thus each simplex (tuple) is stored in our data structures as its canonical (sorted) representative. Then if we, for example, need to check whether two simplices are in fact the same we only need to compare their representatives element by element.

All this information is stored in lists we designate **V**, **E**, **F**, and **T**. They contain one representative for every vertex, edge, triangle, and tet, respectively, in the mesh.

3.1 Forms

The objects of computation in an algorithm using DEC are forms. Formally, a differential k-form is a quantity that can be integrated over a k dimensional domain. For example, consider the expression $\int f(x)dx$ (x being a scalar). The integrand $f(x)dx$ is called a 1-*form*, because it can be integrated over any 1-dimensional interval. Similarly, the dA in $\int\int dA$ would be a 2-form.

Discrete differential forms are dealt with by storing the results of the integrals themselves, instead of the integrands. That is, discrete k-forms associate one value with each k-simplex, representing the integral of the form over that simplex. With this representation we can recover the integral over any k-dimensional chain (the union of some number of k-simplices) by summing the value on each simplex (using the linearity of the integral).

Since all we have to do is to associate one value with each simplex, for our purposes forms are simply vectors of real numbers where the size of the vector is determined by the number of simplices of the appropriate dimension. 0-forms are vectors of size $|\mathbf{V}|$, 1-forms are vectors of size $|\mathbf{E}|$, 2-forms are vectors of size $|\mathbf{F}|$, and 3-forms are vectors of size $|\mathbf{T}|$. Such a vector representation requires that we assign an index to each simplex. We use the position of a simplex in its respective list (**V**, **E**, **F**, or **T**) as its index into the form vectors.

3.2 Orientation

Because the vectors of values we store represent integrals of the associated k-form over the underlying simplices, we must keep track of orientation. For example, reversing the bounds of integration on $\int_a^b f(x)dx$ flips the sign of the resulting value. To manage this we need an *intrinsic orientation* for each simplex. It is with respect to this orientation that the values stored in the form vectors receive the appropriate sign. For example, suppose we have a 1-form f with value f_{ij} assigned to edge $e = (i,j)$; that is, the real number f_{ij} is the integral of the 1-form f over the line segment (p_i, p_j). If we query the value of this form on the edge (j,i) we should get $-f_{ij}$.

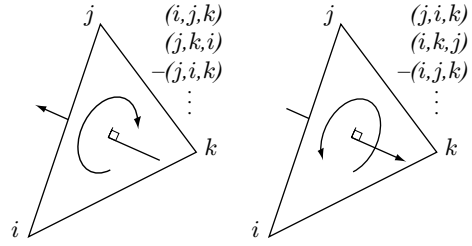


Figure 2: *All permutations of a triple (i,j,k) refer to the same triangle, and the sign of the permutation determines the orientation.*

Hence every tuple must be given a sign indicating whether it agrees $(+)$ or disagrees $(-)$ with the intrinsic orientation of the simplex. Given a set of integers representing a simplex, there are two equivalence classes of orderings of the given tuple: the even and odd permutations of the integers in question. These two equivalence classes correspond to the two possible orientations of the simplex (see Fig. 2).

Note that assigning a sign to any one alias (*i.e.*, the representative) implicitly assigns a sign to all other aliases. Let us assume for a moment that the sign of all representatives is known. Then the sign S of an arbitrary tuple t, with representative r, is

$$\mathbf{S}(t) = \begin{cases} \mathbf{S}(r) & \text{if } t \text{ is in the same equivalence class as } r \\ -\mathbf{S}(r) & \text{if } t \text{ is in the opposite equivalence class.} \end{cases}$$

More formally, let P be the permutation that permutes t into r (*i.e.*, $r = P(t)$). Then

$$\mathbf{S}(t) = \mathbf{S}(P)\mathbf{S}(P(t)).$$

(Here $\mathbf{S}(P)$ denotes the sign of the permutation P with $+1$ for even and -1 for odd permutations.)

All that remains, then, is to choose an intrinsic orientation for each simplex and set the sign of the representative alias accordingly. In general the assignment of orientations is arbitrary, as long as it is consistent. For all subsimplices we choose the representative to be positively oriented, so that the right-hand-side of the above expression reduces to $\mathbf{S}(P)$. For top-level simplices (tets in 3D, triangles in 2D), we use the convention that a positive volume corresponds to a positively oriented simplex. We therefore require a volume form which, together with an assignment of points to vertices, will allow us to orient all tets. Recall that a volume form accepts three (for 3D; two for 2D) vectors and returns either a positive or negative number (assuming the vectors are linearly independent). So the sign of a 4-tuple is:

$$\mathbf{S}(i_0,i_1,i_2,i_3) = \mathbf{S}(\mathrm{Vol}(p_{i_1} - p_{i_0}, p_{i_2} - p_{i_0}, p_{i_3} - p_{i_0})).$$

4 The Boundary Operator

The *faces* of a k-simplex are the $(k-1)$-simplices that are incident on it, *i.e.*, the subset of one lower dimension. Every k-simplex has $k+1$ faces. Each face corresponds to removing one integer from the tuple, and the relative orientation of the face is $(-1)^i$ where i is the index of the integer that was removed. To clarify:

- The faces of a tet $+(t_0,t_1,t_2,t_3)$ are $-(t_0,t_1,t_2)$, $+(t_0,t_1,t_3)$, $-(t_0,t_2,t_3)$, and $+(t_1,t_2,t_3)$.
- The faces of a triangle $+(f_0,f_1,f_2)$ are $+(f_0,f_1)$, $-(f_0,f_2)$, and $+(f_1,f_2)$.
- The faces of an edge $+(e_0,e_1)$ are $-(e_0)$ and $+(e_1)$.

We can now define the boundary operator ∂ which maps simplices to their their faces. Given the set of tets $\mathbf{T}$ we define $\partial^3 : \mathbf{T} \to \mathbf{F}^4$ as

$$\begin{aligned} \partial^3(+(i_0,i_1,i_2,i_3)) \quad = \quad & \{-(i_0,i_1,i_2),+(i_0,i_1,i_3), \\ & -(i_0,i_2,i_3),+(i_1,i_2,i_3)\}. \end{aligned}$$

Similarly for $\partial^2 : \mathbf{F} \to \mathbf{E}^3$ (which maps each triangle to its three edges) and $\partial^1 : \mathbf{E} \to \mathbf{V}^2$ (which maps each edge to its two vertices).

We represent these operators as sparse adjacency matrices (or, equivalently, signed adjacency lists), containing elements of type $+1$ and -1 only. So ∂^3 is implemented as a matrix of size $|\mathbf{F}| \times |\mathbf{T}|$ with 4 non-zero elements per column, ∂^2 an $|\mathbf{E}| \times |\mathbf{F}|$ matrix with 3 non-zero elements per column, and ∂^1 a $|\mathbf{V}| \times |\mathbf{E}|$ matrix with 2 non-zero elements per column (one $+1$ and one -1). The transposes of these matrices are known as the *coboundary* operators, and they map simplices to their *cofaces*—neighbor simplices of one higher dimension. For example, $(\partial^2)^T$ maps an edge to the "pinwheel" of triangles incident on that edge.

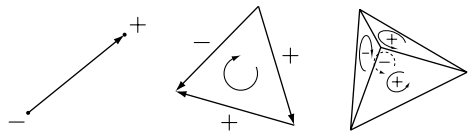


Figure 3: *The boundary operator identifies the faces of a simplex as well as their* relative *orientations. In this illustration, arrows indicate intrinsic orientations and signs indicate the relative orientation of a face to a parent.*

These matrices allow us to iterate over the faces or cofaces of any simplex, by walking down the columns or across the rows, respectively. In order to traverse neighbors that are more than one dimension removed (*i.e.*, the tets adjacent to an edge or the faces adjacent to a vertex) we simply concatenate the appropriate matrices, but without the signs. (If we kept the signs in the matrix multiplication any such consecutive product would simply return the zero matrix reflecting the fact that the boundary of a boundary is always empty.)

5 Construction

Although we still need a few auxiliary wrapper and iterator data structures to provide an interface to the mesh elements, the simplex lists and boundary matrices contain the entirety of the topological data of the mesh. All that remains, then, is to fill in this data.

We read in our mesh as a list of (x,y,z) vertex positions and a list of 4-tuples specifying the tets. Reading the mesh in this format eliminates the possibility of many non-manifold scenarios; for example, there cannot be an isolated edge that does not belong to a tet. We assume that all integers in the range $[0,n)$ appear at least once in the tet list (this eliminates isolated vertices), and no integer outside of this range is present.

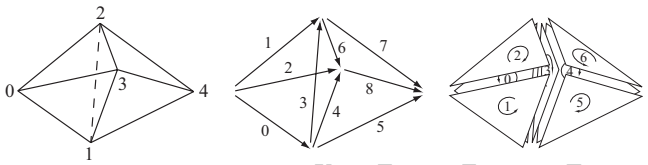
Once $\mathbf{T}$ is read in, building $\mathbf{E}$ and $\mathbf{F}$ is trivial; for each tuple in $\mathbf{T}$, append all subsets of size 2 and 3 to $\mathbf{E}$ and $\mathbf{F}$ respectively. We must be sure to avoid duplicates, either by using a unique associative container, or by sorting the list afterward and removing duplicates. Then the boundary operator matrices are constructed as follows:

```
for each simplex s
    construct a tuple for each face f of s
        as described in Section 4
    determine the index i of f by locating
        its representative
    set the entry of the appropriate matrix
        at row i, column s to S(f)
```

Figure 4 shows a complete example of a mesh and its associated data structure.

6 DEC Operators

Now we discuss the implementation of the two most commonly used DEC operators: the exterior derivative and the Hodge star. As we will see, in the end these also amount to nothing more than sparse matrices that can be applied to our form vectors.



$$\mathbf{V}=\begin{bmatrix}[0]\\[1]\\[2]\\[3]\\[4]\end{bmatrix}\quad \mathbf{E}=\begin{bmatrix}[0,1]\\[0,2]\\[0,3]\\[1,2]\\[1,3]\\[1,4]\\[2,3]\\[2,4]\\[3,4]\end{bmatrix}\quad \mathbf{F}=\begin{bmatrix}[0,1,2]\\[0,1,3]\\[0,2,3]\\[1,2,3]\\[1,2,4]\\[1,3,4]\\[2,3,4]\end{bmatrix}\quad \mathbf{T}=\begin{bmatrix}[0,1,2,3]\\[1,2,3,4]\end{bmatrix}$$

$$\partial^3=\begin{pmatrix}-1&\\1&\\-1&\\1&-1\\&1\\&-1\\&1\end{pmatrix}$$

$$\partial^2=\begin{pmatrix}1&1&&&&&&\\-1&&1&&&&&\\&-1&-1&&&&&\\1&&&1&1&&&\\&1&&-1&&1&&\\&&&&-1&-1&&\\&&1&1&&&1&\\&&&&1&&-1&\\&&&&&1&1&\end{pmatrix}$$

$$\partial^1=\begin{pmatrix}-1&-1&-1&&&&&&\\1&&&-1&-1&-1&&&\\&1&&1&&&-1&-1&\\&&1&&1&&1&&-1\\&&&&&1&&1&1\end{pmatrix}$$

Figure 4: *A simple mesh and all associated data structures.*

6.1 Exterior Derivative

As we have seen earlier in the course, the discrete exterior derivative is defined using Stokes' theorem such that

$$\int_\sigma d\omega = \int_{\partial\sigma}\omega$$

where ω is a k-form, and σ is a $(k+1)$-simplex. In words, this equation states that the evaluation of $d\omega$ on a simplex is equal to the evaluation of ω on the boundary of that simplex.

Let us try to understand this theorem with a few examples. Consider a 0-form f, *i.e.*, a function giving values at vertices. With that, df is a 1-form which can be integrated along an edge (say with endpoints denoted a and b) and Stokes' theorem states the well known fact

$$\int_{[a,b]} df = f(b)-f(a).$$

The right hand side is simply the evaluation of the 0-form f on the boundary of the edge (*i.e.*, its endpoints), with appropriate signs indicating the orientation of the edge.

What about triangles? If f is a 1-form (one value per edge), then df is a 2-form that can be evaluated on a triangle abc as

$$\begin{aligned}\int_{\Delta abc} df &= \int_{\partial(\Delta abc)} f\\ &= \int_{[a,b]} f+\int_{[b,c]} f+\int_{[c,a]} f\\ &= f_{ab}+f_{bc}+f_{ca}\end{aligned}$$

using the subscript notation from Section 3.2. Again, the right hand side is simply the evaluation of the 1-form f on the boundary of the triangle—its three edges.

We can restate the general form of the theorem for our discrete forms as

$$d\omega_\sigma = \sum_{s\in\partial\sigma}\omega_s$$

Written this way, it is easy to see that this can be implemented as the multiplication of a form vector by the coboundary matrix ∂^T.

6.2 The Dual Mesh and the Hodge Star

Every complex has a dual. The dual of a simplicial complex is a *cell complex* where primal k-simplices correspond to dual $(n-k)$-cells. So in our case there are $|\mathbf{V}|$ dual polyhedra, $|\mathbf{E}|$ dual polygons, $|\mathbf{F}|$ dual edges, and $|\mathbf{T}|$ dual vertices, corresponding to primal vertices, edges, triangles, and tetrahedra, respectively (see Fig. 5). Note that, since every dual cell is co-located with a primal simplex and the cardinality is the same, in the code there is no explicit representation for the dual mesh. Where appropriate, dual cells are queried through the corresponding primal simplex index.

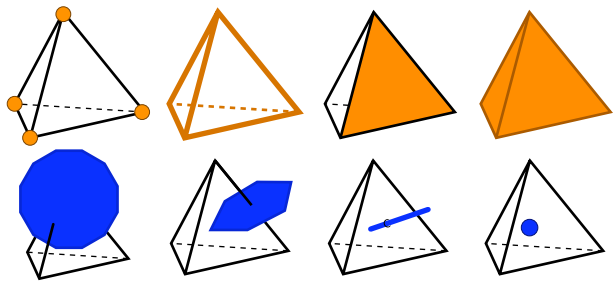


Figure 5: *There is one dual polyhedron for every primal vertex, one dual polygon for every primal edge, one dual edge for every primal triangle, and one dual vertex for every primal tetrahedron.*

The operator that transforms a primal k-form into a dual $(n-k)$-form is known as the *Hodge star*. There are many different kinds of Hodge stars, the simplest of which is the *diagonal Hodge star*.

We again attempt to motivate the definition with some intuition. When transferring a quantity from a primal simplex to a dual cell, the quantities must "agree" somehow. Since these are integral values, simply setting the value on the dual to be equal to the value on the primal does not make sense, as the domain of integration is unrelated. Instead, we require that the *integral density* be equal. So, if ω denotes the evaluation of a form on a primal k-simplex σ, then $\star\omega$ is the value on the dual $(n-k)$-cell $\tilde{\sigma}$ such that

$$\frac{\omega}{\mathrm{Vol}(\sigma)} = \frac{\star\omega}{\mathrm{Vol}(\tilde{\sigma})}$$

allowing us to define $\star$ as

$$\star = \frac{\mathrm{Vol}(dual)}{\mathrm{Vol}(primal)}.$$

In effect the diagonal Hodge star requires that the averages of the integrand over the respective domains agree.

This is represented as a diagonal matrix so that, again, application of the operator becomes a simple matrix-vector multiplication. Note that when transforming quantities from the dual to the primal, the inverse of this matrix is used. Since the matrix is diagonal we only store the diagonal entries. There are as many of these as there are simplices of the appropriate dimension. Consequently the diagonal Hodge star can be represented with vectors of length $|\mathbf{V}|$, $|\mathbf{E}|$, $|\mathbf{F}|$, and $|\mathbf{T}|$ respectively.

6.2.1 Calculating Dual Volumes

So far the entire implementation has been in terms of the combinatorics of the mesh, but when constructing the Hodge star we must finally introduce the geometry. After all, the purpose of the Hodge star is to capture the metric. The volumes of the primal simplices are straightforward: 1 for vertices, length for edges, area for triangles, and volume for tetrahedra. The dual volumes are similarly defined, but in order to avoid constructing the graph of the dual mesh explicitly, we calculate the dual volumes as follows.

If we use the circumcentric realization of the dual mesh (*i.e.*, dual vertices are at the circumcenters of the associated tets), we can exploit the following facts when calculating the dual volumes.[1]

- A dual edge (dual of a primal triangle t) is linear, is normal to t, and is collinear with the circumcenter of t (though the line segment need not necessarily pass through t).
- A dual polygon (dual of a primal edge e) is planar, is orthogonal to e, and is coplanar with the center of e (though it need not intersect e).
- A dual cell (dual of a primal vertex v) is the convex intersection of the half-spaces defined by the perpendicular bisectors of the edges incident on v.

Just as with primal vertices, the volume of a dual vertex is defined to be 1. For the others, we can conceptually decompose each cell into pieces bounded by lower dimensional cells, and sum the volumes of the pieces. For example, a dual polyhedron can be seen as the union of some number of pyramids, where the base of each pyramid is a dual polygon and the apex is the primal vertex. Similarly, a dual polygon can be seen as a union of triangles with dual edges at the bases, and dual edges can be seen as a union of (two) line segments with dual vertices at the bases. The following pseudocode illustrates how the volumes are calculated.

```
vec3 C( Simplex s );  // gives the circumcenter of s

// Initialize all dual volumes to 0.

// Dual edges
for each primal triangle f
    for each primal tet t_f incident on f
        b ← t_f.dualVolume  // 1
        h ← ||C(f) − C(t_f)||
        f.dualVolume ← f.dualVolume + (1/1) bh

// Dual polygons
for each primal edge e
    for each primal triangle f_e incident on e
        b ← f_e.dualVolume
        h ← ||C(e) − C(f_e)||
        e.dualVolume ← e.dualVolume + (1/2) bh

// Dual polyhedra
for each primal vertex v
    for each primal edge e_v incident on v
        b ← e_v.dualVolume
        h ← ||C(v) − C(e_v)||
        v.dualVolume ← v.dualVolume + (1/3) bh
```

Note that, even when dealing with the geometry of the mesh, this part of the implementation still generalizes trivially to arbitrary dimension.

[1] Circumcentric duals may only be used if the mesh satisfies the Delaunay criterion. If it does not, a barycentric dual mesh may be used. However, care must be taken if a barycentric dual mesh is used, as dual edges are no longer straight lines (they are piecewise linear), dual faces are no longer planar, and dual cells are no longer necessarily convex.

7 Summary

All the machinery discussed above can be summarized as follows:

- k-forms as well as the Hodge star are represented as vectors of length $|\mathbf{V}|$, $|\mathbf{E}|$, $|\mathbf{F}|$, and $|\mathbf{T}|$;
- the discrete exterior derivative is represented as (transposes of) sparse adjacency matrices containing only entries of the form $+1$ and -1 (and many zeros); the adjacency matrices are of dimension $|\mathbf{V}| \times |\mathbf{E}|$ (boundary of edges), $|\mathbf{E}| \times |\mathbf{F}|$ (boundary of triangles), and $|\mathbf{F}| \times |\mathbf{T}|$ (boundary of tets).

In computations these matrices then play the role of operators such as grad, curl, and div and can be composed to construct operators such as the Laplacian (and many others).

While the initial setup of these matrices is best accomplished with associative containers, their final form can be realized with standard sparse matrix representations. Examples include a compressed row storage format, a vector of linked lists (one linked list for each row), or a two dimensional linked list (in effect, storing the matrix and its transpose simultaneously) allowing fast traversal of either rows or columns. The associative containers store integer tuples together with orientation signs. For these we suggest the use of sorted integer tuples (the canonical representatives of each simplex). Appropriate comparison operators needed by the container data structures simply perform lexicographic comparisons.

And that's all there is to it!

Acknowledgments This work was supported in part through a James Irvine Fellowship to the first author, NSF (DMS-0220905, DMS-0138458, ACI-0219979), DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Alias, and Pixar.

References

JOY, K. I., LEGAKIS, J., AND MACCRACKEN, R. 2002. *Data Structures for Multiresolution Representation of Unstructured Meshes*. Springer-Verlag, Heidelberg, Germany.

Stable, Circulation-Preserving, Simplicial Fluids

Sharif Elcott Yiying Tong Eva Kanso Peter Schröder Mathieu Desbrun
Caltech

Figure 1: *Discrete Fluids: we present a novel geometric integration scheme for fluids applicable to tetrahedral meshes of arbitrary domains. Aside from resolving the boundaries precisely, our approach also provides an accurate treatment of vorticity through a discrete preservation of Kelvin's circulation theorem. Here, a hot smoke cloud rises inside a bunny shaped domain of* $7K$ *vertices (*$32K$ *tets—the equivalent complexity of a regular* 19^3 *grid), significantly reducing the computational cost for such an intricate boundary compared to regular grid-based techniques (*0.47*s/frame on a Pentium 4, 3GHz).*

Abstract

Visual quality, low computational cost, and numerical stability are foremost goals in computer animation. An important ingredient in achieving these goals is the conservation of fundamental motion invariants. For example, rigid and deformable body simulation benefits greatly from conservation of linear and angular momenta. In the case of fluids, however, none of the current techniques focuses on conserving invariants, and consequently, they often introduce a visually disturbing numerical diffusion of *vorticity*. Visually just as important is the resolution of complex simulation domains. Doing so with regular (even if adaptive) grid techniques can be computationally delicate. In this chapter, we propose a novel technique for the simulation of fluid flows. It is designed to respect the defining differential properties, *i.e.*, the *conservation of circulation* along arbitrary loops as they are transported by the flow. Consequently, our method offers several new and desirable properties: (1) arbitrary simplicial meshes (triangles in 2D, tetrahedra in 3D) can be used to define the fluid domain; (2) the computations are efficient due to discrete operators with small support; (3) the method is stable for arbitrarily large time steps; (4) it preserves *discrete circulation* avoiding numerical diffusion of vorticity; and (5) its implementation is straightforward.

1 Introduction

Conservation of motion invariants at the discrete computational level, *e.g.*, linear and angular momenta in solid mechanics, is now widely recognized as being an important ingredient in the construction of numerically stable simulations with a high degree of visual realism [Marsden and West 2001]. Much of the progress in this direction has been enabled by a deeper understanding of the underlying geometric structures and how they can be preserved as we go from continuous models to discrete computational realizations. So far, advances of this type have yet to deeply impact fluid flow simulations. Current methods in fluid simulation are rarely designed to conserve *defining physical properties*. Consider, for example, the need in many methods to continually project the numerically updated velocity field onto the set of divergence free velocity fields; or the need to continually reinject vorticity lost due to numerical dissipation as a simulation progresses. In this chapter, we introduce a novel, geometry-based algorithm for fluid simulation which, among other desirable properties, exactly preserves vorticity at a discrete level.

1.1 Previous Work

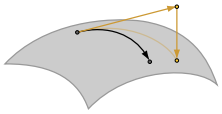
Fluid Mechanics has been studied extensively in the scientific community both mathematically and computationally. The physical behavior of incompressible fluids is usually modeled by the Navier Stokes (NS) equations for viscous fluids and by the Euler equations for inviscid (non-viscous) fluids. Numerical approaches in computational fluid dynamics typically discretize the governing equations through Finite Volumes (FV), Finite Elements (FE) or Finite Differences (FD) methods. We will not attempt to review the many methods proposed (an excellent survey can be found in [Langtangen et al. 2002]) and instead focus on approaches used for fluids in computer graphics. Some of the first fluid simulation techniques used in the movie industry were based on Vortex Blobs [Yaeger et al. 1986] and Finite Differences [Foster and Metaxas 1997]. To circumvent the ill-conditioning of these iterative approaches for large time steps and achieve unconditional stability, Jos Stam [1999; 2001] introduced to the graphics community the *method of characteristics* for fluid advection and the Helmholtz-Hodge decomposition to ensure the divergence-free nature of the fluid motion [Chorin and Marsden 1979]. The resulting algorithm, called *Stable Fluids*, is an extremely successful semi-Lagrangian approach based on a regular grid Eulerian space discretization, that led to many refinements and extensions which have contributed to the enhanced visual impact of fluid animations. Among others, these include the use of staggered grids and monotonic cubic interpolation [Fedkiw et al. 2001]; improvements in the handling of interfaces [Foster and Fedkiw 2001]; extensions to curved surfaces [Stam 2003; Tong et al. 2003; Shi and Yu 2004] and visco-elastic objects [Goktekin et al. 2004]; and goal oriented control of fluid motion [Treuille et al. 2003; McNamara et al. 2004; Pighin et al. 2004; Shi and Yu 2005].

1.2 Shortcomings of Stable Fluids

However, the Stable Fluids technique is not without drawbacks. Chief among them is the difficulty of accomodating complex domain boundaries with regular grids, as addressed recently with hybrid meshes [Feldman et al. 2005]. Local adaptivity [Losasso et al. 2004] can greatly help, but the associated octree structures require significant overhead. Note that regular partitions of space (adaptive or not) can suffer from preferred direction sampling, leading to visual artifacts similar to aliasing in rendering.

Additionally, the different variants of the original Stable Fluids algorithm [Stam 1999] are all based on a class of discretization ap-

proaches known in Computational Fluid Dynamics as fractional step methods. In order to numerically solve the Euler equations over a time step, they proceed in two stages. They first update the velocity field assuming the fluid is inviscid and disregard the divergence-free constraint. Then, the resulting velocity is projected onto the closest divergence-free flow (in the $\mathcal{L}^2$ sense) through an exact Helmholtz-Hodge decomposition. Despite the simplicity of this fractional integration, one of its consequences is excessive numerical diffusion: advecting velocity before reprojecting onto a divergence-free field creates major energy loss. While this shortcoming is not a major issue in graphics as the visual impact of such a loss is not always significant, another consequence of the fractional integration is an *excessive diffusion of vorticity*. This last issue affects the motion significantly, since the presence of vortices in liquids and volutes in smoke is one of the most important visual cues of our perception of fluidity. Vorticity confinement [Steinhoff and Underhill 1994; Fedkiw et al. 2001] was designed to counteract this diffusion through local reinjection of vorticity. Unfortunately, it is hard to control how much can safely be added back without affecting stability or plausibility of the results. Adding vortex particles can further reduce this loss [Guendelman et al. 2005], but adds computational cost to the Stable Fluids method without suppressing the loss completely. One can understand this seemingly inevitable numerical diffusion through the following geometric argument: the solutions of Euler equations are *geodesic* (*i.e.*, shortest) paths *on* the manifold of all possible divergence-free flows; thus, advecting the fluid *out* of the manifold is not a proper substitute to this intrinsic constrained minimization, even if the post re-projection is, in itself, exact. For a more detailed numerical analysis of this flaw, see [Chang et al. 2002].



1.3 Towards a Better SetUp

In this chapter we show that a careful setup of *discrete differential quantities*, designed to respect structural relationships such as vector calculus identities, leads in a direct manner to a numerical simulation method which respects the defining *geometric structure* of the fluid equations. A key ingredient in this approach is the location of physical quantities on the appropriate geometric structures (*i.e.*, vertices, edges, faces, or cells). This greatly simplifies the implementation as all variables are *intrinsic*. It also ensures that the approach works for curved manifolds without any changes. With the help of a *discrete calculus on simplicial complexes* we construct a novel integration scheme which employs intrinsically divergence-free variables. This removes the need to enforce the usual divergence-free constraint through a numerically lossy projection step. Our time integration method *by construction* preserves circulation and consequently vorticity. It accomplishes this while being simple, numerically efficient, and unconditionally stable, achieving high visual quality even for large time steps. Although our approach shares the same algorithmic structure as Stable Fluids based methods (use of backward path tracing and sparse linear systems), it fundamentally differs from previous techniques on the following points:

- **our technique is based on a classical *vorticity* formulation of the Navier-Stokes and Euler equations**; unlike most vorticity-based methods in CFD and CG, our approach is *Eulerian* as we work only with a fixed mesh and *not* a Lagrangian representation involving vorticity particles (or similar devices);
- we use an unconditionally-stable, semi-Lagrangian backward advection strategy for vertices just like Stable Fluids; **in contrast to Stable Fluids however we do not point-sample velocity, but rather compute *integrals of vorticity***; this simple change removes the need to enforce incompressibility of the velocity field through projection;
- **our strategy exactly preserves circulation along discrete loops in the mesh**; capturing this *geometric property* of fluid dynamics guarantees that vorticity does not get dissipated as is typically the case in Stable Fluids; consequently no vorticity confinement (or other post processes) are required to maintain this important element of visual realism;
- **our method has multiple advantages from an implementation point of view**: it handles arbitrary meshes (regular grids, hybrid meshes [Feldman et al. 2005], and even cell complexes) with non-trivial topology; the operators involved have very small support leading to very sparse linear systems; all quantities are stored intrinsically (scalars on edges and faces) without reference to global or local coordinate frames; the computational cost is comparable to previous approaches.

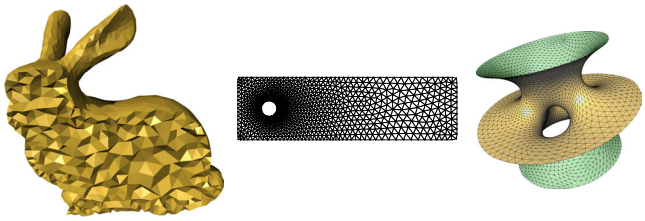


Figure 2: *Domain Mesh: our fluid simulator uses a simplicial mesh to discretize the equations of motion; (left) the domain mesh (shown as a cutaway view) used in Fig. 1; (middle) a coarser version of the flat 2D mesh used in Fig. 8; (right) the curved triangle mesh used in Fig. 10.*

The organization of this chapter is as follows. In Section 2, we motivate our approach through a brief overview of the theory and computational algorithms for Fluid Mechanics. We present a novel discretization of fluid mechanics and a circulation-preserving integration algorithm in Section 2.2. The computational machinery required by our approach is developed in Section 3, while the algorithmic details are given in Section 4. Several numerical examples are shown and discussed in Section 5.

2 Of Motion & Flow Representation

Before going into the details of our algorithm we discuss the underlying fluid equations with their relevant properties and how these can be captured over discretized domains.

2.1 Geometry of Fluid Motion

Euler Fluids Consider an inviscid, incompressible, and homogeneous fluid on a domain $\mathcal{D}$ in 2 or 3D. The *Euler equations*, governing the motion of this fluid (with no external forces for now), can be written as:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla p \,, \\ div(\mathbf{u}) &= 0 \,, \quad \mathbf{u} \parallel \partial\mathcal{D} \,. \end{aligned} \tag{1}$$

We assume unit density ($\rho = 1$) and use $\mathbf{u}$ to denote the fluid velocity, p the pressure, and $\partial\mathcal{D}$ the boundary of the fluid region $\mathcal{D}$. The pressure term in Eq. (1) can be dropped easily by rewriting the Euler equations in terms of *vorticity*. Recall that traditional vector calculus defines vorticity as the curl of the velocity field, $\boldsymbol{\omega} = \nabla \times \mathbf{u}$. Taking the curl ($\nabla\times$) of Eq.(1), we obtain

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} + \mathcal{L}_{\mathbf{u}}\boldsymbol{\omega} &= \mathbf{0} \,, \\ \boldsymbol{\omega} = \nabla \times \mathbf{u} \,, \quad div(\mathbf{u}) &= 0 \,, \quad \mathbf{u} \parallel \partial\mathcal{D} \,. \end{aligned} \tag{2}$$

where $\mathcal{L}_{\mathbf{u}}\boldsymbol{\omega}$ is the Lie derivative, equal in our case to $\mathbf{u} \cdot \nabla\boldsymbol{\omega} - \boldsymbol{\omega} \cdot \nabla\mathbf{u}$. In this form, this vorticity-based equation states that *vorticity*

is simply advected along the fluid flow. [1] Note that Equation (2) is equivalent to the more familiar $\frac{D\boldsymbol{\omega}}{Dt} = \boldsymbol{\omega} \cdot \nabla \mathbf{u}$, and therefore already includes the vortex stretching term appearing in *Lagrangian* approaches. Roughly speaking, vorticity measures the local spin of a fluid parcel. Therefore, vorticity advection means that this local spin moves along with the flow.

Since the integral of vorticity on a given bounded surface equals (by Stokes' theorem) the *circulation* around the bounding loop of the surface, one can explain the geometric nature of an ideal fluid flow in particularly simple terms: **the circulation around any closed loop $\mathcal{C}$ is conserved throughout the motion of this loop in the fluid**. This key result is known as Kelvin's circulation theorem, and is usually written as:

$$\Gamma(t) \;=\; \oint_{\mathcal{C}(t)} \mathbf{u} \cdot \mathbf{dl} \;=\; \text{constant}\,, \tag{3}$$

where $\Gamma(t)$ is the circulation of the velocity on the loop $\mathcal{C}$ at time t as it gets advected in the fluid. This will be the key to our time integration algorithm.

Viscous Fluids In contrast to ideal fluids, incompressible *viscous* fluids generate very different fluid behaviors. They can be modelled by the *Navier-Stokes* equations (compare with Eq. (1)):

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \;&=\; -\nabla p + \nu \Delta \mathbf{u}\,, \\ div(\mathbf{u}) \;&=\; 0\,, \quad \mathbf{u}|_{\partial \mathcal{D}} = \mathbf{0}\,. \end{aligned} \tag{4}$$

where Δ denotes the Laplace operator, and ν is the *kinematic viscosity*. Note that different types of boundary conditions can be added depending on the chosen model. Despite the apparent similarity between these two models for fluid flows, the added diffusion term dampens the motion, resulting in a slow decay of circulation. This diffusion also implies that the velocity of a viscous fluid at the boundary of a domain must be null, whereas an inviscid fluid could have a non-zero tangential component on the boundary. Here again, one can avoid the pressure term by taking the curl of the equations (compare with Eq. (2)):

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} + \mathcal{L}_{\mathbf{u}} \boldsymbol{\omega} \;&=\; \nu \Delta \boldsymbol{\omega}\,, \\ \boldsymbol{\omega} \;=\; \nabla \times \mathbf{u}\,, \quad div(\mathbf{u}) \;&=\; 0\,, \quad \mathbf{u}|_{\partial \mathcal{D}} = \mathbf{0}\,. \end{aligned} \tag{5}$$

2.2 Discrete Setup and Time Integration

For a discrete time and space numerical simulation of Eqs. (2) and (5) we need a discretized geometry of the domain (given as a simplicial mesh for instance), appropriate discrete analogs of velocity $\mathbf{u}$ and vorticity fields $\boldsymbol{\omega}$, along with the operators which act on them. We will present our choices before describing the geometry-based time integration approach.

2.2.1 Space Discretization

We discretize the spatial domain in which the flow takes place using a locally oriented simplicial complex, *i.e.*, either a tet mesh for 3D domains or a triangle mesh for 2D domains, and refer to this discrete domain as $\mathcal{M}$ (see Figure 2). The domain may have non-trivial topology, *e.g.*, it can contain tunnels and voids (3D) or holes (2D), but is assumed to be compact. To ensure good numerical properties in the subsequent simulation we require the simplices of $\mathcal{M}$ to be well shaped (aspect ratios bounded away from zero). This assumption is quite common since many numerical error estimates depend heavily on the element quality. We use meshes generated with the method of [Alliez et al. 2005]. Collectively we refer to the sets of vertices, edges, triangles, and tets as V, E, F, and T.

We will also need a *dual mesh*. It associates with each original simplex (vertex, edge, triangle, tet, respectively) its dual (dual cell, dual face, dual edge, and dual vertex, respectively) (see Fig. 3). The geometric realization of the dual mesh uses tet circumcenters as dual vertices and Voronoi cells as dual cells; dual edges are line segments connecting dual vertices across shared tet faces and dual faces are the faces of the Voronoi cells. Notice that storing values on primal simplices or on their associated dual cells is conceptually equivalent, since both sets have the same cardinality. We will see in Section 3 that corresponding primal and dual quantities are related through a simple (diagonal) linear operator.

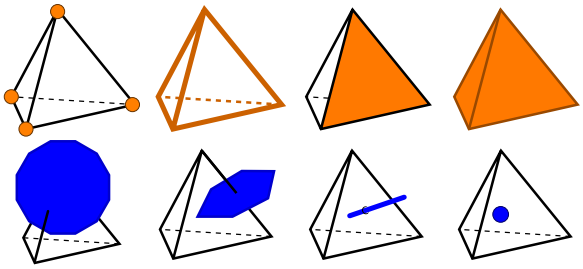


Figure 3: *Primal and Dual Cells: the simplices of our mesh are vertices, edges, triangles and tets (up); their circumcentric duals are dual cells, dual faces, dual edges and dual vertices (bottom).*

2.2.2 Intrinsic Discretization of Physical Quantities

In order to faithfully capture the geometric structure of fluid mechanics on the discrete mesh, we define the usual physical quantities, such as velocity and vorticity, through **integral values over the elements of the mesh $\mathcal{M}$**. This is the sharpest departure from traditional numerical techniques in CFD: we not only use values at nodes and tets (as in FEM and FVM), but also allow association (and storage) of field values at *any* appropriate simplex. Depending on whether a given quantity is defined pointwise, or as a line, area or volume density, the corresponding discrete representation will "live" at the associated 0, 1, 2, and 3 dimensional mesh elements. With this in mind we use a simple discretization of the physical quantities of fluid mechanics based on fluxes associated to faces.

Velocity as Discrete Flux To encode a coordinate free (intrinsic) representation of velocity on the mesh we use *flux*, *i.e.*, the mass of fluid transported across a given surface area per unit time. Note that this makes flux an *integrated*, not pointwise, quantity. On the discrete mesh, fluxes are associated with the triangles of the tet mesh. Thus fluid velocity $\mathbf{u}$ is treated as a 2-form and represented as a vector U of values on faces (size $|F|$). This coordinate-free point of view, also used in [Feldman et al. 2005], is reminiscent of the staggered grid method used in [Fedkiw et al. 2001] and other non-collocated grid techniques (see [Goktekin et al. 2004]). In the staggered grid approach one does not store the x, y, z components of a vector at nodes but rather associates them with the corresponding grid faces. We may therefore think of the idea of storing fluxes on the triangles of our tet mesh as a way of *extending* the idea of staggered grids to the more general simplical mesh setting. This was previously exploited in [Bossavit and Kettunen 1999] in the context of E&M computations. It also makes the usual no-transfer boundary conditions easy to encode: boundary faces experience no flux

[1] Note that this is a more canonical characterization of the motion than the velocity-based one: the latter was also an advection but under the additional *constraint* of keeping the velocity field divergence-free, which is the reason for the gradient of pressure.

across them. Encoding this boundary condition for velocity vectors stored at vertices is far more cumbersome.

Divergence as Net Flux on Tets Given the incompressibility of the fluid, the velocity field must be divergence-free ($\nabla \cdot \mathbf{u} = 0$). In the discrete setting, the integral of the divergence over a tet becomes particularly simple. According to the generalized Stokes' theorem this integral equals the sum of the fluxes on all four faces: the discrete notion of divergence is therefore simply the net flux for each tet. Divergence can therefore be stored as a 3-form, *i.e.*, as a value associated to each tet (a vector of cardinality $|T|$). The notion of incompressibility becomes particularly intuitive: whatever gets in each tet must be compensated by whatever comes out (see Fig. 4).

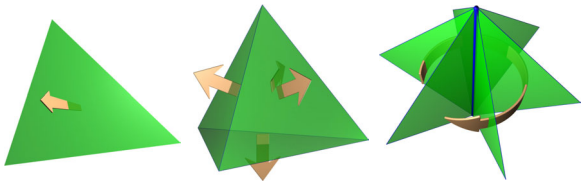


Figure 4: *Discrete Physical Quantities: in our geometric discretization, fluid flux lives on faces (left), divergence lives on tets (middle), and vorticity lives on edges (right).*

Vorticity as Flux Spin Finally we need to define vorticity on the mesh. To see the physical intuition behind our definition, consider an edge in the mesh. It has a number of faces incident on it, akin to a paddle wheel (see Figure 4). The flux on each face contributes a torque to the edge. The sum of all these, when going around an edge, is the net torque that would "spin" the edge. We can thus give a physical definition of vorticity as a weighted sum of fluxes on all faces incident to a given edge. This quantity is now associated with primal edges—or, equivalently, dual faces—and is thus represented by a vector Ω of size $|E|$.

In Section 3, we will see that these physical intuitions can be derived formally from simple algebraic relationships.

2.3 Geometric Integration of Fluid Motion

Since we are using the vorticity formulation of the fluid equations (Eqs. (2) or (5)) the time integration algorithm must update the discrete vorticity variables which are stored on each primal edge. We have seen that the fluid equations state that vorticity is advected by the velocity field. The fundamental idea of our geometric integration algorithm is thus to ensure that Kelvin's theorem holds in the discrete setting: the circulation around any loop in the fluid remains constant as the loop is advected. This can be achieved by backtracking loops: for any given loop at the current time, determine its backtracked image in the velocity field ("where did it come from?") and compute the circulation around the backtracked loop. This value is then assigned as the circulation around the original loop at the present time, *i.e.*, circulation is properly advected *by construction* (see Figure 5 for a depiction of this loop advection idea).

Since we store vorticity on primal edges, a natural choice for these loops are the bounding loops of the dual faces associated to each primal edge (see Figure 3). Notice that these loops are polylines formed by sequences of dual vertices around a given primal edge. Consequently an efficient implementation of this idea requires only that we backtrack *dual vertices* in the velocity field. Once these positions are known *all* backtracked dual loops associated to *all* primal edges are known. These Voronoi loops can indeed generate any discrete, dual loop: the sum of adjacent loops is a larger, outer loop as the interior edges cancel out due to opposite orientation as sketched in Fig. 5(right). The evaluation of circulation around these backtracked loops will be quite straightforward. Invoking Stokes' theorem, the integral of vorticity over a dual face equals the circulation around its boundary. With this observation we have achieved our goal of updating vorticities and, *by design*, ensured a discrete version of Kelvin's theorem. The algorithmic details of this simple geometric approach to time integration of the equations of motion for fluids will be given in Section 4.

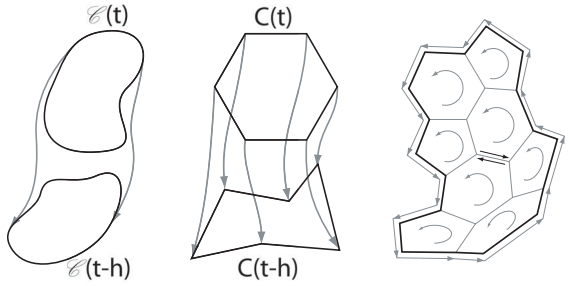


Figure 5: *Kelvin's Theorem: (left) in the continuous setting, the circulation on any loop being advected by the flow is constant. (middle) our discrete integration scheme enforces this property on each Voronoi loop, (right) thus on any discrete loop.*

3 Computational Machinery

Now that we have described our choices of spatial and physical discretizations, along with a way to use them to integrate the fluid's motion, we must manipulate the numerics involved in a principled manner to guarantee proper physical behavior. In this section, we point out that the intuitive definition of our physical quantities living at the different simplices of a mesh can be made precise through the definition of a *discrete differential structure*. Consequently, the basic operators to go from fluxes to the divergence, curl, or Laplacian of the velocity field can be *formally* defined through the use of discrete differential forms. We will mostly focus on presenting the practical implementation of the few operators we need; more importantly, we will show that this implementation reduces to *simple linear algebra* with very sparse matrices.

3.1 Discrete Differential Structure

Integrals and Forms In Section 2.2, we have opted for manipulating the physical quantities in the form of a line, surface, and volume integral computed directly on our meshed domain to render the setup entirely intrinsic, *i.e.*, with no need for vector fields to be stored with respect to arbitrary coordinate frames. Such an integral represents the evaluation of a mathematical entity called a *differential form* . In the continuous three-dimensional setting, a 0-form is simply a function on that 3D space. A 1-form, or line-form, is a quantity that can be *evaluated through integration over a curve*. Thus a 1-form can be thought of as a proxy for a vector field, and its integral over a curve becomes the circulation of this vector field. A 2-form, or area-form, is to be *integrated over a surface*, that is, it can be viewed as a proxy for a vector perpendicular to that surface (and its evaluation becomes the flux of that vector field through the surface); finally, a 3-form, or volume-form, is to be *integrated over a volume* and can be viewed as a proxy for a function. Classic calculus and vector calculus can then be substituted with a special calculus involving only differential forms, called *exterior calculus*—the basis of Hodge theory and modern differential geometry (for a comprehensive discussion, see, for example, [Abraham et al. 1988]).

Discrete Forms and Their Representation However, in our framework, the continuous domain is replaced (or approximated) by a mesh, the only structure we can work with. Therefore, the integrated physical values we store on the mesh corresponds to *discrete differential k-forms* [Desbrun et al. 2006]. A discrete differential k-form, $k = 0, 1, 2$, or 3, is the evaluation (*i.e.*, the integral) of the differential k-form on all k-cells, or k-simplices. In practice, discrete k-forms can simply be considered as *vectors of numbers*

according to the simplices they live on: 0-forms live on vertices, and are expressed as a vector of length $|V|$; correspondingly, 1-forms live on edges (length $|E|$), 2-forms live on faces (length $|F|$), and 3-forms live on tets (length $|T|$). Dual forms, *i.e.*, forms that we will evaluate on the dual mesh, are treated similarly. The reader should now realize that in our discretization of physical quantities, the notion of flux that we described is thus a primal 2-form (integrated over faces), while its vorticity is a dual 2-form (integrated over dual faces), and its divergence becomes a primal 3-form (integrated over tets).

Discrete Differential Calculus on Simplicial Meshes These discrete forms can now be used to build the tools of calculus through Discrete Exterior Calculus (DEC), a coherent calculus mimicking the continuous setting that only uses discrete combinatorial and geometric operations [Munkres 1984; Hirani 2003; Desbrun et al. 2006]. At the core of its construction is the definition of a discrete d operator (analog of the continuous exterior derivative), and a discrete Hodge star, which will allow us to move values from the primal mesh to the dual mesh and vice-versa. For a more comprehensive introduction to DEC and the use of discrete differential forms, we refer the interested readers to [Desbrun et al. 2006].

3.2 Two Basic Operators

The computations involved in our approach only require the definition of two basic operators: one is the exterior derivative d, necessary to compute derivatives, like gradients, divergences, or curls; the other is the Hodge star, to transfer values from primal simplices to dual simplices.

Exterior Derivative d Given an oriented mesh, we implement our first operator by simply assembling the *incidence matrices* of the mesh. These will act on the vectors of our discrete forms and implement the discrete exterior derivative operator d as explained in more details in Appendix A. For our 3D implementation, there are three sparse matrices involved, which contain only entries of type 0, $+1$, and -1. Care is required in assembling these incidence matrices, as the orientation must be taken into account in a consistent manner [Elcott and Schröder 2006]. The first one is d_0, the incidence matrix of vertices and edges ($|E|$ rows and $|V|$ columns). Each row contains a single $+1$ and -1 for the end points of the given edge (and zero otherwise). The sign is determined from the orientation of the edge. The second matrix similarly encodes the incidence relations of edges and faces ($|F|$ rows and $|E|$ columns), with appropriate $+1$ and -1 entries according to the orientation of edges as one moves around a face. More generally d_k is the incidence matrix of k-cells on $k+1$-cells.

A simple debugging sanity check (necessary but not sufficient) is to compute consecutive products: d_0 followed by d_1 must be a matrix of zeros; d_1 followed by d_2 must similarly give a zero matrix. This reflects the fact that the boundary of any boundary is the empty set. It also corresponds to the calculus fact that curl of grad is zero as is divergence of curl (see [Desbrun et al. 2006]).

Hodge Star The second operator we need will allow us to transfer quantities back and forth between the primal and dual mesh. We can project a primal k-form to a conceptually-equivalent dual $(3-k)$-form with the *Hodge star*. We will denote $\star_0$ (resp., $\star_1, \star_2, \star_3$) the Hodge star taking a 0-form (resp., 1-form, 2-form, and 3-form) to a dual 3-form (resp., dual 2-form, dual 1-form, dual 0-form). In this chaper we will use what is known as the *diagonal Hodge star* [Bossavit 1998]. This operator simply scales whatever quantity that is stored on mesh cells by the volumes of the corresponding dual and primal cells: let $\mathrm{vol}(.)$ denote the volume of a cell (*i.e.*, 1 for vertices, length for edges, area for triangles, and volume for tets), then

$$(\star_k)_{ii} = \mathrm{vol}(\tilde{\sigma}_i)/\,\mathrm{vol}(\sigma_i)$$

where σ_i is any primal k-simplex, and $\tilde{\sigma}_i$ is its dual. These linear operators, describing the local metric, are diagonal and can be stored as vectors. Conveniently, the inverse matrices going from dual to primal quantities are trivial to compute for this diagonal Hodge star.

Overloading Operators Note that both the d_k and the $\star_k$ operators are *typed*: the subscript k is *implicitly determined* by the dimension of the argument. For example, the velocity field $\mathbf{u}$ is a 2-form stored as a vector U of cardinality $|F|$. Consequently the expression dU implies use of the $|T| \times |F|$-sized matrix d_2. In the implementation this is accomplished with operator overloading (in the sense of C++). We will take advantage of this from now on and drop the dimension subscripts.

3.3 Offline Matrix Setup

With these overloads of d and $\star$ in place, we can now set up the only two matrices (C and L) that will be used during simulation. They respectively represent the discrete analogs of the curl and Laplace operators [Desbrun et al. 2006].

Curl Since we store fluxes on faces and gather them in a vector U, the *circulation* of the vector field $\mathbf{u}$ can be derived as values on dual edges through $\star U$. *Vorticity*, typically a 2-form in fluid mechanics [Marsden and Wenstein 1983], is easily computed by then summing this circulation along the dual edges that form the boundary of a dual face. In other words, $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ becomes, in terms of our discrete operators, simply $\Omega = d^T \star U$. We therefore create a matrix C offline as $d^T\star$, *i.e.*, the composition of an incidence matrix with a diagonal matrix.

Laplacian The last matrix we need to define is the discrete Laplacian. The discrete analog of $\Delta\phi = (\nabla\nabla\cdot - \nabla\times\nabla\times)\phi = \boldsymbol{\omega}$ is simply $(\star d \star^{-1} d^T \star + d^T \star d)\,\Phi = \Omega$ as explained in Appendix B. This last matrix, a simple composition of incidence and diagonal matrices, is precomputed and stored as L for later use.

4 Implementation

To facilitate a direct implementation of our integration scheme, we provide pseudocode (Figure 6) along with implementation notes which provide details for specific steps and how these relate to the machinery developed in earlier sections.

```
//Load mesh and build incidence matrices
C ← d^T ⋆
L ← ⋆d⋆^{-1} d^T ⋆ + d^T ⋆ d

//Time stepping h
loop
    //Advect Vorticities
    for each dual vertex (tet circumcenter) c_i
        ĉ_i ← PathTraceBackwards(c_i);
        v_i ← InterpolateVelocityField(ĉ_i);
    for each dual face f
        Ω_f ← 0
        for each dual edge (i, j) on the boundary of f
            Ω_f ← Ω_f + ½(v_i + v_j) · (ĉ_i − ĉ_j);

    //Add forces
    Ω ← Ω + h C F

    //Add diffusion for Navier-Stokes
    Ψ ← SolveCG( (⋆ − ν h L)Ψ = Ω );
    Ω ← ⋆ Ψ

    //Convert vorticities back to fluxes
    Φ ← SolveCG( L Φ = Ω );
    U ← dΦ;
```

Figure 6: *Pseudocode of our fluid motion integrator.*

4.1 External Body Forces

The use of external body forces, like buoyancy, gravity, or stirring, is common practice to create interesting motions. Incorporating external forces into Eq. (4) is straightforward, resulting in:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \;=\; -\nabla p + \nu \Delta \mathbf{u} + \mathbf{f} \,.$$

Again, taking the curl of this equation allows us to recast this equation in terms of vorticity:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathcal{L}_{\mathbf{u}} \boldsymbol{\omega} \;=\; \nu \Delta \boldsymbol{\omega} + \nabla \times \mathbf{f} \,. \tag{6}$$

Thus, we note that an external force influences the vorticity only through the force's curl (the $\nabla \cdot \mathbf{f}$ term is compensated for by the pressure term keeping the fluid divergence-free). Thus, if we express our forces through the vector F of their resulting fluxes in each face, we can directly add the forces to the domain by incrementing Ω by the circulation of F over the time step h, *i.e.*:

$$\Omega \leftarrow \Omega + h \; C \; F.$$

4.2 Adding Diffusion

If we desire to simulate a viscous fluid, we must add the diffusion term present in Eq. (5). Note that previous methods were sometimes omitting this term because their numerical dissipation was already creating (uncontrolled) diffusion. In our case, however, this diffusion needs to be properly handled if viscosity is desired. This is easily done through an unconditionally-stable implicit integration as done in Stable Fluids (*i.e.*, we also use a fractional step approach). Using the discrete Laplacian in Eq. (8) and the current vorticity Ω, we simply solve for the diffused vorticity Ω' using the following linear system:

$$(\star - \nu h L) \;\star^{-1} \Omega' = \Omega.$$

4.3 Interpolation of Velocity

In order to perform the backtracking of dual vertices we must first define a velocity field over the entire domain using the data we have on primal faces (fluxes). This can be done by computing a unique velocity vector for each dual vertex and then using barycentric interpolation of these vectors over each dual Voronoi cell [Warren et al. 2004], defining a continuous velocity field over the entire domain. This velocity field can be used to backtrack dual vertices as well as transport particles or dyes (*e.g.*, for visualization purposes) with standard methods.

To see that such a vector, one for each dual vertex, is well defined consider the following argument. The flux on a face corresponds under duality (via the Hodge star) to a circulation along the dual edge of this face. Now, there is a linear relation between fluxes per tet due to the incompressibility condition (fluxes must sum to zero). This translates directly to a linear condition on the four circulations at each tet too. Thus, there is a unique vector (with three components) at the dual vertex whose projection along the dual edges is consistent with the observed circulations.

Relation to k-form Basis Functions The standard method to interpolate k-form data in a piecewise linear fashion over simplicial complexes is based on Whitney forms [Bossavit 1998]. In the case of primal 2-forms (fluxes) this results in a piecewise constant (per tet) velocity field. Our argument above, using a Voronoi cell based generalized barycentric interpolation of dual 1-forms (circulation), in fact *extends* the Whitney form machinery to the dual setting. This is a novel contribution which may be useful in other computational applications of discrete forms. We note that the generalized barycentric coordinates have linear accuracy [Warren et al. 2004], an important requirement in many settings.

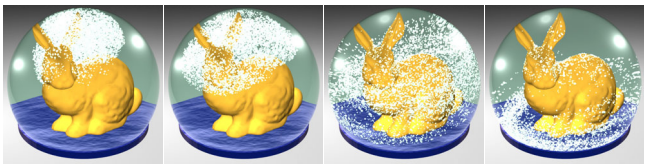


Figure 7: *Bunny Snow Globe: the snow in the globe is advected by the inner fluid, initially stirred by a vortex to simulate a spin of the globe.*

4.4 Handling Boundaries

The algorithm as described above does not constrain the boundaries, thus achieving "open" boundary conditions. No-transfer boundary conditions are easily imposed by setting the fluxes through the boundary triangles to zero. Non-zero flux boundary conditions (*i.e.*, forced fluxes through the boundary as in the case of Fig. 8) are more subtle. First, remark that all these boundary fluxes *must* sum to zero; otherwise, we would have little chance of getting a divergence-free fluid in the domain. Since the total divergence is zero, there exists a harmonic velocity field satisfying exactly these conditions. This is a consequence of the Helmholtz-Hodge decomposition theorem with normal boundary conditions [Chorin and Marsden 1979]. Thus, this harmonic part $\mathbf{h}$ can be computed *once and for all* through a Poisson equation using the same setup as described in Appendix B. This precomputed velocity field allows us to deal very elegantly with these boundary conditions: we simply perform the same algorithm as we described by setting all boundary conditions to zero (with the exception of backtracking which takes the precomputed velocity into account), and *reinject* the harmonic part at the end of each time step (*i.e.*, add $\mathbf{h}$ to the current velocity field).

Viscous Fluids near Boundaries The Voronoi cells at the boundaries are slightly different from the usual, interior ones, since boundary vertices do not have a full 1-ring of tets. In the case of NS equations, this has no significant consequence: we set the velocity on the boundary to zero, resulting also in a zero circulation on the dual edges on the boundary. The rest of the algorithm can be used as is.

Inviscid Fluids near Boundaries For Euler equations, however, the tangential velocity at the boundary is not explicitly stored anywhere. Consequently, the boundary Voronoi faces need an additional variable to remedy this lack of information. We store in these dual faces the current integral vorticity. From this additional information given at time $t = 0$, we can *deduce* at each later time step the missing circulation on the boundary: since the circulation over the inside dual edges is known, and since the total integral must sum to the vorticity (Stokes' theorem), a simple substraction is all that is needed to update this missing circulation.

4.5 Handling Arbitrary Topology

Although the problem of arbitrary domain topology (*e.g.*, when the first Betti number is not zero) is rarely discussed in CFD or in our field, it is important nonetheless. In the absence of external forces, the circulation along each loop (of winding number 1) around a tunnel is constant in time. So once again, we precalculate a constant harmonic field based on the initial circulation around each tunnel, and simply *add it* to the current velocity field for advection purposes. This procedure serves two purposes: first, notice that we now automatically enforce the discrete equivalent of Kelvin's theorem on *any* (shrinkable or non-shrinkable) loop; second, arbitrary topologies are accurately handled very efficiently.

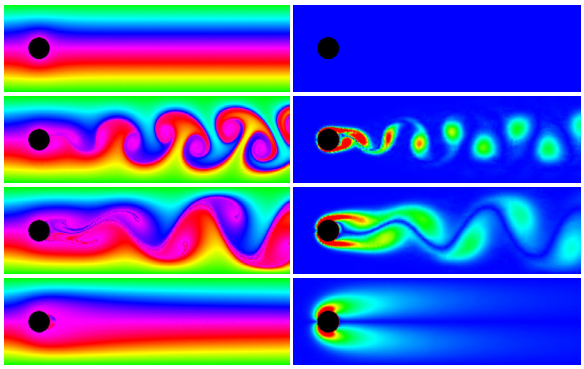
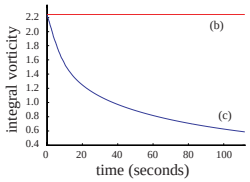


Figure 8: *Obstacle Course: in the usual experiment of a flow passing around a disk, the viscosity as well as the velocity can significantly affect the flow appearance; (left) our simulation results for increasing viscosity and same left boundary flux; (right) the vorticity magnitude (shown in false colors) of the same frame. Notice how the usual irrotational flow is obtained (top) for zero viscosity, while the von Karman vortex street appears as viscosity is introduced.*

5 Results and Discussion

We have tested our method on some of the usual "obstacle courses" in CFD. We start with the widely studied example of a flow past a disk (see Fig. 9). Starting with zero vorticity, it is well known that in the case of an inviscid fluid, the flow remains irrotational for all time. By construction, our method does respect this physical behavior since circulation is preserved for Euler equations. We then increase the viscosity of the fluid incrementally, and observe the formation of a vortex wake behind the obstacle, in agreement with physical experiments. As evidenced by the vorticity plots, vortices are shed from the boundary layer as a result of the adherence of the fluid to the obstacle, thanks to our proper treatment of the boundary conditions.

The behavior of vortex interactions observed in existing experimental results was compared to numerical results based on our novel model and those obtained from the semi-Lagrangian advection method. It is known from theory that two like-signed vortices with a finite vorticity core will merge when their distance of separation is smaller than some critical value. This behavior is captured by the experimental data and shown in the first series of snapshots of Fig. 9. As the next row of snapshots indicates, the numerical results that our model generated present striking similarities to the experimental data. In the last row, we see that a traditional semi-Lagrangian advection followed by re-projection misses most of the fine structures of this phenomenon. This can be attributed to the loss of total integral vorticity as evidenced in this inset; in comparison our technique preserves this integral exactly.



We have also considered the flow on curved surfaces in 3D with complex topology, as depicted in Fig. 10. We were able to easily extend our implementation of two-dimensional flows to this curved case thanks to the intrinsic nature of our approach.

We consider a smoke cloud surrounded by air, filling the body of a bunny as an example of flow in a domain with complex boundary. The buoyancy drives the air flow which, in turn, advects the smoke cloud in the three-dimensional domain bounded by the bunny mesh as shown in Fig. 1. In the last simulation, we show a snow globe with a bunny inside (see Fig. 7). We emulate the flow due to an initial spin of the globe using a swirl described as a vorticity field.

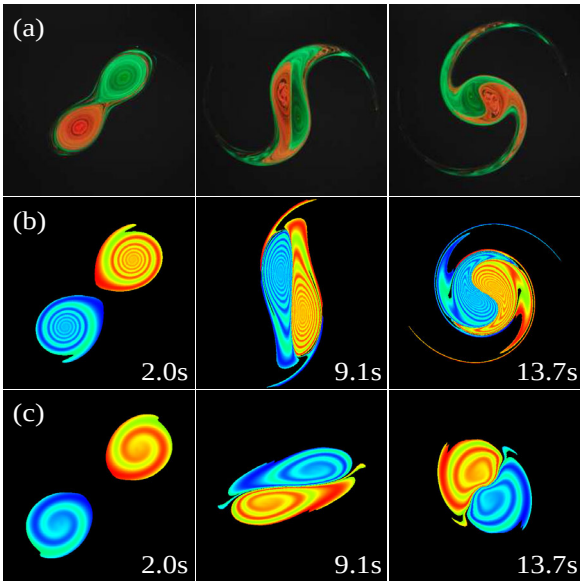


Figure 9: *Two Merging Vortices: discrete fluid simulations are compared with a real life experiment (courtesy of Dr. Trieling, Eindhoven University; see* `http://www.fluid.tue.nl/WDY/vort/index.html`*) where two vortices (colored in red and green) merge slowly due to their interaction (a); while our method faithfully captures the merging phenomenon (b), a traditional semi-lagrangian scheme does not capture the correct motion because of vorticity damping (c).*

The snow particles are transported by the flow as they fall down under the effect of gravity. Both examples took *less than half a second per frame* to compute, exemplifying the advantage of using tet meshes to resolve fine boundaries.

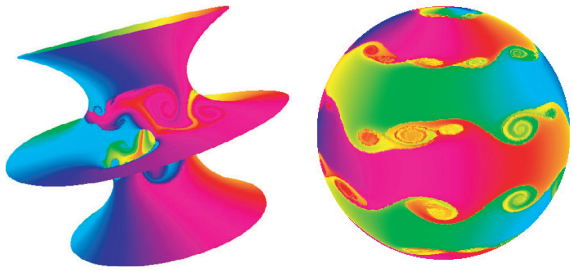


Figure 10: *Weather System on Planet Funky: the intrinsic nature of the variables used in our algorithm makes it amenable to the simulation of flows on arbitrary curved surfaces.*

6 Conclusion

In this chapter, we have introduced a novel theoretical approach to fluid dynamics, along with its practical implementation and various simulation results. We have carefully discretized the physics of flows to respect the most fundamental geometric structures that characterize their behavior. Among the several specific benefits that we demonstrated, the most important is the circulation preservation property of the integration scheme, as evidenced by our numerical examples. The discrete quantities we used are intrinsic, allowing us to go to curved manifolds with no additional complication. Finally, the machinery employed in our approach can be used on any simplicial complex. However, the same methodology also applies directly to more general spatial partitionings, and in particular, to regular grids and hybrid meshes [Feldman et al. 2005]—rendering our approach widely applicable to existing fluid simulators.

For future work, a rigorous comparison of the current method with standard approaches should be undertaken. Using Bjerknes' circulation theorem for compressible flows may also be an interesting

avenue. Additionally, we limited ourselves to the investigation of our scheme without focusing on the separate issue of order of accuracy. Coming up with an integration scheme that is higher-order accurate will be the object of further investigation, as it requires a better (denser) Hodge star. Finally, we note that our geometric approach bears interesting similarities with the work of Chang *et al.* [2002], in which they propose a purely algebraic approach to remedy the shortcomings of the traditional fractional step approach. Using their numerical analysis on our approach may provide a simple way to study the accuracy of our scheme.

References

ABRAHAM, R., MARSDEN, J., AND RATIU, T., Eds. 1988. *Manifolds, Tensor Analysis, and Applications*, vol. 75 of *Applied Mathematical Sciences*. Springer.

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. 2005. Variational tetrahedral meshing. *ACM Transactions on Graphics 24*, 3, 617–625.

BOSSAVIT, A., AND KETTUNEN, L. 1999. Yee-like schemes on a tetrahedral mesh. *Int. J. Num. Modelling: Electr. Networks, Dev. and Fields 12* (July), 129–142.

BOSSAVIT, A. 1998. *Computational Electromagnetism*. Academic Press, Boston.

CHANG, W., GIRALDO, F., AND PEROT, B. 2002. Analysis of an exact fractional step method. *Journal of Computational Physics 180*, 3 (Nov.), 183–199.

CHORIN, A., AND MARSDEN, J. 1979. *A Mathematical Introduction to Fluid Mechanics*, 3rd edition ed. Springer-Verlag.

DESBRUN, M., KANSO, E., AND TONG, Y. 2006. Discrete differential forms for computational sciences. In *Discrete Differential Geometry*, E. Grinspun, P. Schröder, and M. Desbrun, Eds., Course Notes. ACM SIGGRAPH.

ELCOTT, S., AND SCHRÖDER, P. 2006. Building your own dec at home. In *Discrete Differential Geometry*, E. Grinspun, P. Schröder, and M. Desbrun, Eds., Course Notes. ACM SIGGRAPH.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. *ACM Transactions on Graphics 24*, 3, 904–909.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 23–30.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *Proceedings of SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 181–188.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics 23*, 3 (Aug.), 463–468.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shell. *ACM Transactions on Graphics 24*, 3, 973–981.

HIRANI, A. 2003. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology.

LANGTANGEN, H.-P., MARDAL, K.-A., AND WINTER, R. 2002. Numerical methods for incompressible viscous flow. *Advances in Water Resources 25*, 8-12 (Aug-Dec), 1125–1146.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics 23*, 3 (Aug.), 457–462.

MARSDEN, J. E., AND WENSTEIN, A. 1983. Coadjoint orbits, vortices and Clebsch variables for incompressible fluids. *Physica D 7*, 305–323.

MARSDEN, J. E., AND WEST, M. 2001. Discrete mechanics and variational integrators. *Acta Numerica 10*, 357–515.

MCNAMARA, A., TREUILLE, A., POPOVIC, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Transactions on Graphics 23*, 3 (Aug.), 449–456.

MUNKRES, J. R. 1984. *Elements of Algebraic Topology*. Addison-Wesley.

PIGHIN, F., COHEN, J. M., AND SHAH, M. 2004. Modeling and Editing Flows using Advected Radial Basis Functions. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 223–232.

SHI, L., AND YU, Y. 2004. Inviscid and Incompressible Fluid Simulation on Triangle Meshes. *Journal of Computer Animation and Virtual Worlds 15*, 3-4 (June), 173–181.

SHI, L., AND YU, Y. 2005. Controllable smoke animation with guiding objects. *ACM Transactions on Graphics 24*, 1, 140–164.

STAM, J. 1999. Stable fluids. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 121–128.

STAM, J. 2001. A simple fluid solver based on the fft. *Journal of Graphics Tools 6*, 2, 43–52.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics 22*, 3 (July), 724–731.

STEINHOFF, J., AND UNDERHILL, D. 1994. Modification of the euler equations for *Vorticity Confinement*: Applications to the computation of interacting vortex rings. *Physics of Fluids 6*, 8 (Aug.), 2738–2744.

TONG, Y., LOMBEYDA, S., HIRANI, A. N., AND DESBRUN, M. 2003. Discrete multiscale vector field decomposition. *ACM Transactions on Graphics 22*, 3, 445–452.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics 22*, 3 (July), 716–723.

WARREN, J., SCHAEFER, S., HIRANI, A., AND DESBRUN, M., 2004. Barycentric coordinates for convex sets. Preprint.

YAEGER, L., UPSON, C., AND MYERS, R. 1986. Combining physical and visual simulation - creation of the planet jupiter for the film 2010. *Computer Graphics (Proceedings of ACM SIGGRAPH) 20*, 4, 85–93.

A Discrete Exterior Derivative

A thorough explanation of the discrete exterior derivative of discrete forms is out of the scope of this chapter, and we refer the reader to existing tutorials contained in these course notes [Desbrun et al. 2006]. However, the reader should be aware that this operator is simply implemented via the use of *incidence matrices*.

Indeed, a key ingredient to defining the discrete version of the exterior derivative d is Stokes' theorem:

$$\int_\sigma d\alpha = \int_{\partial\sigma} \alpha,$$

where σ denotes a $(k+1)$-cell and α is a k-form. Stokes' theorem states that the integral of $d\alpha$ (a $(k+1)$-form) over a $(k+1)$-cell equals the integral of the k-form α over the *boundary* of the $(k+1)$-cell (*i.e.*, a k-cell). Stokes' theorem can thus be used as a way to *define* the d operator in terms of the boundary operator ∂. Or, said differently, once we have the boundary operator, the operator d follows immediately if we wish Stokes' theorem to hold on the simplicial complex.

To use a very simple example, consider a 0-form f, *i.e.*, a function giving values at vertices. With that, df is a 1-form which can be integrated along an edge (say with end points denoted a and b) and Stokes' theorem states the well known fact:

$$\int_{[a,b]} df = f(b) - f(a).$$

The right hand side is simply the evaluation of the 0-form f on the boundary of the edge, *i.e.*, its endpoints (with appropriate signs indicating the orientation of the edge). Actually, one can define a hierarchy of these operators that mimic the operators given in the continuous setting (up to an application of the Hodge star) by the gradient (∇), curl ($\nabla\times$), and divergence ($\nabla\cdot$), namely,

- d_0: maps 0-forms to 1-forms and corresponds to the **Gradient**;
- d_1: maps 1-forms (values on edges) to 2-forms (values on faces). The value on a given face is simply the sum (by linearity of the integral) of the 1-form values on the boundary (edges) of the face with the signs chosen according to the local orientation. d_1 corresponds to the **Curl**;
- d_2: maps 2-forms to 3-forms and corresponds to the **Divergence**.

Since the boundary of any mesh element can be directly read from the incidence matrices of the mesh, the exterior derivative is trivial to implement once the mesh is known as it depends only on its connectivity [Elcott and Schröder 2006].

B Recovery of Velocity

We have seen in Section 4 how the vorticity ω can be derived directly from the set of all face fluxes as $d^T \star U = \omega$. However, during the simulation, we will also need to recover flux *from* vorticity. For this we employ the Helmholtz-Hodge decomposition theorem, stating that any vector field $\mathbf{u}$ can be decomposed into three components (given appropriate boundary conditions)

$$\mathbf{u} = \nabla \times \phi + \nabla\boldsymbol{\psi} + \mathbf{h}.$$

When represented in terms of discrete forms this reads as:

$$U = d\Phi + \star^{-1} d^T \star \Psi + H \tag{7}$$

For the case of incompressible fluids (*i.e.*, with zero divergence), two of the three components are sufficient to describe the velocity field: the curl of a vector potential and a harmonic field. To see this apply d to both sides of Eq. 7:

$$dU = 0 = dd\Phi + d \star^{-1} d^T \star \Psi + dH.$$

Since $dd = 0$ and d of a harmonic form always vanishes, we find that $d \star^{-1} d^T \star \Psi = 0$ to begin with. Since Ψ is a 3-form $d \star^{-1} d^T \star \Psi = \Delta\Psi = 0$, *i.e.*, Ψ is harmonic which implies in particular that $\star^{-1} d^T \star \Psi = 0$, proving our claim that

$$U = d\Phi + H.$$

If the topology of the domain is trivial, we can furthermore ignore the harmonic part H (we discuss a full treatment of arbitrary topology in Section 4.5), leaving us with $U = d\Phi$.

Since our algorithm computes an updated Ω which is related to U as $d^T \star U$, we need to find a solution to

$$\Omega = d^T \star d\Phi,$$

where Ω is the known quantity, and $d\Phi$ the unknown. Unfortunately the kernel of $d^T \star d$ is not empty so we can not determine Φ directly from this equation. To pick a unique solution for Φ, we require additionally that $d^T \star \Phi = 0$. By doing so we pick a *particular* solution from the kernel of d^T. But if $d^T \star \Phi = 0$ then certainly $(\star d \star^{-1} d^T \star)\Phi = 0$ and we can add it to our equation for Ω arriving at

$$\Omega = (d^T \star d + \star d \star^{-1} d^T \star)\Phi. \tag{8}$$

This latter equation is simply a Poisson equation for Φ since

$$\star^{-1}\Omega = \Delta\Phi$$

which has a unique solution. Let $U = d\Phi$, and we have recovered U as desired.

The fact that Eq. 8 is indeed a Poisson problem follows from the definition of the Laplacian Δ in *differential calculus* as $d \star^{-1} d^T \star + \star^{-1} d^T \star d$. In the language of vector calculus this translates to $\Delta\phi = (\nabla\nabla\cdot - \nabla\times\nabla\times)\phi = \nabla\times\mathbf{u}$. Notice that the left-side matrix (that we will denote L) is *symmetric and sparse*, thus ideally suited for fast numerical solvers. Our linear operators (and, in particular, the discrete Laplacian) differ from another discrete Poisson setup on simplicial complexes proposed in [Tong et al. 2003]: the ones we use have smaller support, which results in sparser and better conditioned linear systems [Bossavit 1998]—an attractive feature in the context of numerical simulation.

An Algorithm for the Construction of Intrinsic Delaunay Triangulations with Applications to Digital Geometry Processing

Matthew Fisher
Caltech

Boris Springborn
TU Berlin

Alexander I. Bobenko
TU Berlin

Peter Schröder
Caltech

Abstract

The discrete *Laplace-Beltrami* operator plays a prominent role in many Digital Geometry Processing applications ranging from denoising to parameterization, editing, and physical simulation. The standard discretization uses the cotangents of the angles in the immersed mesh which leads to a variety of numerical problems. We advocate use of the *intrinsic* Laplace-Beltrami operator. It satisfies a local maximum principle, guaranteeing, *e.g.*, that no flipped triangles can occur in parameterizations. It also leads to better conditioned linear systems. The intrinsic Laplace-Beltrami operator is based on an *intrinsic* Delaunay triangulation of the surface. We give an incremental algorithm to construct such triangulations together with an overlay structure which captures the relationship between the extrinsic and intrinsic triangulations. Using a variety of example meshes we demonstrate the numerical benefits of the intrinsic Laplace-Beltrami operator.

1 Introduction

Delaunay triangulations of domains in $\mathbb{R}^2$ play an important role in many numerical computing applications because of the quality guarantees they make, such as: among all triangulations of the convex hull of a set of points in the plane the Delaunay triangulation maximizes the minimum angle. Many error estimators in finite element approaches to the solution of partial differential equations, *e.g.*, are directly related to the minimum angle and are more favorable if this minimum is maximized (for an extensive discussion see [Shewchuk 2002]). The construction of such triangulations for domains in $\mathbb{R}^2$ is now standard textbook material and a basic ingredient in many meshing algorithms. For immersed surfaces in $\mathbb{R}^3$, which are given as simplicial 2-manifold meshes (possibly with boundary), the picture is not nearly as clear. Algorithms which numerically manipulate such meshes are of great importance in Digital Geometry Processing applications. Examples include surface denoising [Desbrun et al. 1999], thin-shell simulation [Grinspun et al. 2003], construction of discrete minimal surfaces [Pinkall and Polthier 1993], surface parameterization [Desbrun et al. 2002; Lévy et al. 2002], computation of discrete conformal structures [Gu and Yau 2003; Mercat 2001], and geometric modeling [Botsch and Kobbelt 2004b], among many others. Similar to the setting of domains in $\mathbb{R}^2$ one also finds that meshes which satisfy an *intrinsic* Delaunay criterion give rise to better numerical behavior in all the above geometry processing examples.

A classic algorithm to convert a given planar triangulation into a Delaunay triangulation involves edge flipping, whereby an edge which violates the local Delaunay criterion is flipped until no such edge remains. In the same vein, one can construct an *intrinsic Delaunay triangulation* (see Figure 1) of a simplicial surface in $\mathbb{R}^3$ by performing *intrinsic edge flips* (see Figure 2). Importantly, because the edge flips are intrinsic the shape of the original embedded mesh *does not change*. This notion of intrinsic Delaunay triangulation (iDT) takes into account only the intrinsic geometry of the mesh, *i.e.*, the mesh is considered as an abstract surface with a metric that is locally Euclidean except at isolated points (the vertices of the mesh) where it has cone-like singularities. The relevant data read off from the input mesh are the combinatorics of its edge graph as well as the length of each edge. With this data alone one may now

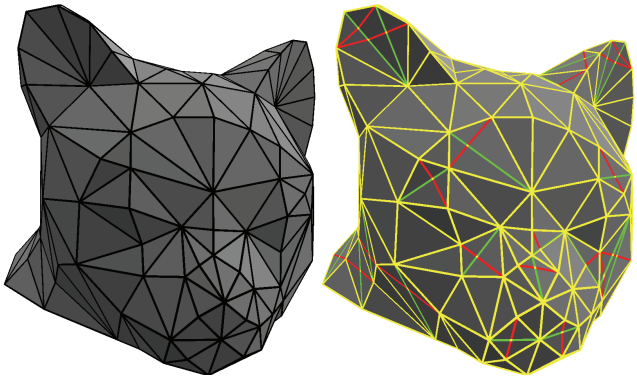


Figure 1: *Left: carrier of the (cat head) surface as defined by the original embedded mesh. Right: the intrinsic Delaunay triangulation: yellow edges are part of the original and intrinsic Delaunay triangulation; red edges resulted from flipping; and green edges denote original edges which are not part of the intrinsic Delaunay triangulation. Note that the red edges are geodesic lines on the original surface.*

ask of each interior edge whether it satisfies the local Delaunay condition since the associated predicate can be based solely on the observed edge lengths and local combinatorics (and thus entirely on intrinsic data). The intrinsic flip algorithm proceeds as follows: While there is an edge that violates the local Delaunay criterion, perform a combinatorial flip on it and update its length. Note that this procedure does not change the intrinsic geometry of the input mesh at all. One may visualize the iDT of such a mesh as a graph drawn on the original simplicial surface, as shown in Figure 1. It is not hard to see that the intrinsic flip algorithm terminates, thus producing an intrinsic triangulation with all interior edges satisfying the local Delaunay criterion [Indermitte et al. 2001]. Recently, Bobenko and Springborn [2005] have shown that as in the planar case the iDT is essentially unique and satisfies a *global* empty circumcircle property. A technical difficulty that one encounters when dealing with iDTs is that they are not necessarily *regular* triangulations. A triangulation is called *regular* if each triangle is incident with three different edges and three different vertices. It is called *strongly regular* if it is regular and the intersection of two triangles is either empty or *one* edge or *one* vertex, and if the intersection of two edges is either empty or *one* vertex. The usual definition of the term *triangulation* implies strong regularity. This is too narrow for our purposes. For example, edges of an iDT may form loops (see Figure 3). Therefore, we do not require triangulations to be regular.

With an iDT in hand one may define an *intrinsic Laplace Beltrami* (iLB) operator [Bobenko and Springborn 2005]. In contrast to the *extrinsic Laplace Beltrami* (eLB) operator, which is based on the original triangulation, the iLB operator has many favorable numerical properties. For example, in the construction of discrete harmonic parameterizations one can guarantee injectivity of the computed parameterization because a discrete maximum principle holds for the iLB while this is generally not the case for the eLB. (These issues with the eLB have been the cause for many proposals to nu-

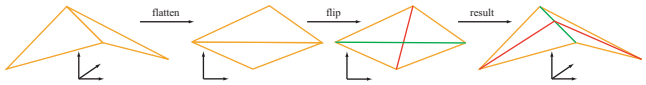


Figure 2: *Given a pair of adjacent triangles in $\mathbb{R}^3$ their intrinsic geometry can be visualized by isometrically mapping them to the plane. If the edge is not Delaunay it is flipped inside the original surface (original edge: green; flipped edge: red). The flipped edge is a geodesic segment in this surface and the surface remains unchanged.*

merically "fix" the eLB.) More generally, the iLB is numerically better conditioned than the eLB leading to more efficient applications in particular when higher powers of the iLB are required. The iDT is also useful in other settings. For example, Circle Patterns [Kharevych et al. 2006], used in the construction of discrete conformal maps, require the intrinsic Delaunay property of each edge. In that setting the use of iDTs leads to results with far lower quasi-conformal distortion (see Figure 8).

Contributions In this paper we describe an algorithm for the construction of iDTs for immersed simplicial 2-manifold meshes (possibly with boundary) of arbitrary topology. A first version of the algorithm performs intrinsic edge flips only and requires little more than a standard edge based data structure. In some applications one also requires knowledge of the original (extrinsic) edges crossed by the intrinsic (geodesic) triangulation edges. This requires the maintenance of an overlay data structure. We show that this structure can be built incrementally during flipping and depends *only* on combinatorial predicates. It is well known that edge flipping can have a long running time (edge flipping in planar triangulations, *e.g.*, takes $\Theta(n^2)$ time). Empirically we find the algorithm to run quite fast however, and its runtime to be easily dominated by subsequent numerical computing tasks. We present statistics for a variety of input meshes and show, among other observations, that the condition number of the associated Laplace-Beltrami operator can be noticeably reduced when using the iDT. Coupled with the guarantee of satisfying a discrete maximum principle, this results in more robust and efficient numerical behavior for a host of DGP applications.

2 Intrinsic Delaunay Triangulations

We begin this section with a brief recall of the relevant intrinsic geometry of piecewise linearly immersed simplicial meshes, before describing the edge flipping algorithm without (Section 2.2) and with (Section 2.3) overlay maintenance.

2.1 Piecewise Flat Surfaces

A *piecewise flat surface* (PF surface) is a 2-dimensional manifold equipped with a metric such that every interior point of the manifold has a neighborhood that is isometric to a neighborhood in the plane, *except* for a number of isolated points, the *cone points*. Each cone point has a neighborhood that is isometric to a neighborhood of the apex of a Euclidean cone. Small circles of radius r around a cone point have length αr with $\alpha \neq 2\pi$. The number α is the *cone angle* of the cone point (which may be smaller or larger than 2π). Its *Gaussian curvature* is $2\pi - \alpha$. We always assume that the boundary of a PF surface, if present, is piecewise geodesic.

A concrete way to construct PF surfaces is through gluing polygons: take a set of planar polygons together with a partial isometric edge pairing, *i.e.*, a set of pairs of different polygon edges such that the edges in each pair have the same length and every edge is contained in at most one pair. Gluing the polygons along the paired edges one obtains a PF surface. The unpaired edges make up the boundary. We emphasize that the notion of a PF surface belongs to the *intrinsic geometry of surfaces*: here we are not interested in whether or how a PF surface can be isometrically embedded in $\mathbb{R}^3$.

When we speak of gluing polygons together, we mean the abstract construction of identifying corresponding points along the edges.

A possible representation which describes a *triangulation of a PF surface* consists of a graph structure to describe an abstract surface triangulation (which need not be regular) together with a labeling of the edges by positive numbers signifying edge length. The only constraint on the edge lengths is that they must satisfy the triangle inequalities for each face. For if that is the case, one can construct all the triangles and glue them to obtain a PF surface. All cone points are vertices of the triangulation. Such a triangulation is a *Delaunay triangulation of a PF surface* if for each interior edge the local Delaunay criterion is satisfied: the sum of the opposite angles in the adjacent triangles is less than π. For more on Delaunay triangulations of PF surfaces see [Bobenko and Springborn 2005] and references therein.

Note: Since a Delaunay triangulation of a PF surface is not necessarily regular, it is essential that the data structure which is used to represent the abstract surface triangulation can represent non-regular triangulations. For example, winged-edge, half-edge, or quad-edge data structures may be used. A data structure based on triangle-vertex incidences is not suited.

2.2 Intrinsic Delaunay Flipping Algorithm

Any 3D surface mesh with flat faces is intrinsically a PF surface. In general, every vertex of such a mesh is a cone vertex. Suppose an immersed surface (in $\mathbb{R}^3$) is given in the form of a 2-manifold triangle mesh. More precisely, we have a 2-manifold complex $K = (V, E, T)$ of vertices, edges, and triangles together with the point positions $P(v_i) = p_i \in \mathbb{R}^3$ (one for each vertex $v_i \in V$). Piecewise linear (PL) interpolation over each triangle then defines the *carrier* of the surface. For this surface we seek an iDT. This triangulation depends only on the complex K and metric data associated with each edge $e_{ij} \in E$. This metric data is the Euclidean length $L(e_{ij}) = l_{ij} = \|p_i - p_j\|$ of each embedded edge $e_{ij} \in E$. The pair (K, L) constitutes the input to the iDT algorithm which returns a complex with corresponding *intrinsic* lengths $(\tilde{K}, \tilde{L})$. Note that once L is computed P will play no further role in the algorithm.

For each edge in $\tilde{E}$ we will also report all edges in E crossed by it (if any). Note that an edge in $\tilde{E}$ may cross a given edge in E multiple times.

While K is generally strongly regular, we do not require regularity. In fact, the output of the algorithm will in general not be regular (see Figure 3).

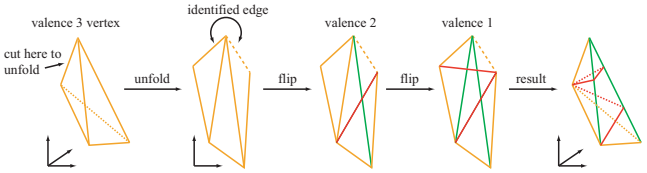


Figure 3: *Even if the input mesh is strongly regular it is quite easy to arrive at non-regular configurations through intrinsic edge flipping. Here a vertex of valence 3 is reduced to a valence 1 vertex through two intrinsic flips (original edges: green; flipped edges: red).*

The transformation of (K, L) into $(\tilde{K}, \tilde{L})$ is straightforward and based on the classic edge flipping algorithm:

Require: (K, L)
Ensure: $(\tilde{K}, \tilde{L})$ is intrinsic Delaunay
  $\forall e \in E$: Mark(e)
  Stack $s \leftarrow E$
  while !Empty(s) **do**

```
    e_ij ← Pop(s) and Unmark(e_ij)
    if !Delaunay(e_ij) then
      e_kl ← Flip(e_ij) and compute l_kl
      for all e ∈ {e_kj, e_jl, e_li, e_ik} do
        if !Mark(e) then
          Mark(e) and Push(s, e)
        end if
      end for
    end if
  end while
  return (K̃, L̃) ← (K, L)
```

The predicate Delaunay(e_{ij}) returns true for boundary edges. For interior edges it checks whether the edge e_{ij} is locally Delaunay using only the edge lengths of the two adjacent triangles t_{ijk} and t_{lij}. The edge is locally Delaunay iff $\alpha_{ij}^k + \alpha_{ij}^l \leq \pi$, where α_{ij}^k and α_{ij}^l are the angles at k respectively l opposite the edge e_{ij}. Using the Cosine Theorem one may calculate $\alpha_{ij}^k + \alpha_{ij}^l$ as

$$\alpha_{ij}^k + \alpha_{ij}^l = \cos^{-1}\left(\frac{l_{kj}^2 + l_{ik}^2 - l_{ij}^2}{2 l_{kj} l_{ik}}\right) + \cos^{-1}\left(\frac{l_{jl}^2 + l_{li}^2 - l_{ij}^2}{2 l_{jl} l_{li}}\right).$$

Alternatively one may compute the cotan weight of the LB operator

$$w_{ij} = \tfrac{1}{2}\left(\cot \alpha_{ij}^k + \cot \alpha_{ij}^k\right)$$

directly from the edge lengths. Using $l_{ijk}^{\pm\pm\pm} = \pm l_{ij} \pm l_{jk} \pm l_{ki}$ as shorthand for signed sums of edge lengths around a triangle t_{ijk} we get

$$\tfrac{1}{2}\cot \alpha_{ij}^k = \frac{l_{ijk}^{+++} l_{ijk}^{-++} - l_{ijk}^{+-+} l_{ijk}^{++-}}{4\sqrt{l_{ijk}^{+++} l_{ijk}^{-++} l_{ijk}^{+-+} l_{ijk}^{++-}}}$$

and the analogous formula for $\frac{1}{2}\cot \alpha_{ij}^l$. This expression follows from the half-angle theorem

$$\tan \frac{\alpha_{ij}^k}{2} = \sqrt{\frac{l_{ijk}^{+-+} l_{ijk}^{++-}}{l_{ijk}^{+++} l_{ijk}^{-++}}}$$

using $\tan(2x) = 2\tan x/(1 - \tan^2 x)$. The edge e_{ij} is locally Delaunay iff $w_{ij} \geq 0$. This latter implementation of the Delaunay predicate avoids inverse trigonometric functions. Also, one may reuse the final weights w_{ij} to calculate the iLB operator (see Section 3).

The fact that this algorithm terminates is shown in [Indermitte et al. 2001]. The proof is essentially the same as for planar triangulations. Infinite loops cannot occur because a suitable function on the set of triangulations decreases with every edge flip. In the case of PF surfaces an added complication results from the fact that the number of triangulations on a fixed vertex set may be infinite. So an additional requirement for a "suitable function" is that it is unbounded on any infinite set of triangulations. That the output $(\tilde{K}, \tilde{L})$ is globally intrinsic Delaunay, *i.e.*, that the local Delaunay property implies a global empty circumcircle property, requires some more work [Bobenko and Springborn 2005].

2.3 Incremental Overlay

The information contained in $(\tilde{K}, \tilde{L})$ is sufficient to assemble the iLB operator and perform computations with it. In some applications the resulting values (at vertices) are to be interpolated across the carrier surface. Recall that the carrier surface is defined through PL interpolation with respect to the input triangulation. However, the data arising from a computation using the iLB operator is most naturally PL interpolated with respect to the iDT. To be able to perform both types of interpolation simultaneously, as is required in texture mapping, for example, one needs a graph structure representing the overlay of both triangulations K and $\tilde{K}$. We describe an incremental algorithm which maintains this overlay during flipping.

We maintain a graph structure (*e.g.*, a half-edge data structure) which describes, at each stage of the flip algorithm, the overlay of the original triangulation and the current one. The vertices of the original and of the current triangulation (they have the same vertex set) are also vertices of the overlay graph. Additionally the overlay graph contains vertices corresponding to points where an edge of the original triangulation is crossed by an edge of the current triangulation. These we will not call vertices but *crossings*. The overlay graph structure distinguishes between vertices and crossings so that we can tell them apart. The edges of the overlay graph we will call *segments* because they are segments of edges of the original and current triangulations. Each *segment* is labeled red, green, or yellow: green if it is part of an edge that belongs to the original triangulation but not the current one (we will also call such an *edge* a green edge), red if it is part of an edge that belongs to the current triangulation but not the original one (a red edge), and yellow if it is part of an edge that belongs to both the original and current triangulation (a yellow edge). Note that from this overlay graph structure one can reconstruct both the original and the current triangulation. Green (red) *edges* correspond to sequences of more than one green (red) *segment*, because an edge of the current triangulation that does not belong to the original triangulation must cross an original edge (and vice versa). A yellow edge on the other hand corresponds to a single yellow segment, because such an edge cannot cross any other edges. Each crossing has four adjacent segments which are alternatingly colored green and red.

Initially, the original triangulation is also the current one, so it is also the overlay graph, with all edges being yellow segments. Performing a flip now requires updates of the overlay graph. We will see that this requires only combinatorial information. After the new combinatorics are established, the edge lengths can be updated independently.

Updating the Overlay Topology We illustrate this using the examples shown in Figure 4. First, consider the flip shown on the left. The horizontal red edge (consisting of four segments) which is to be flipped, is removed. This requires merging of pairs of green segments (previously crossed by the red edge) incident to red/green crossings. This leaves us with a red quadrilateral (formed by the two triangles incident on the edge) with five green segments on its interior. Inserting the (vertical) flipped red edge will lead to some new crossings with green segments. Importantly, we can tell which of these green segments will be crossed and in what order: The green segments that are crossed are those that are incident to the left as well as right boundary of the quadrilateral. (In this accounting the top and bottom vertices do not belong to either boundary.) This eliminates two of the five segments from further consideration. The remaining segments are incident to the boundaries *in the same order* as they are crossed by the flipped red (vertical) edge. This is so because there are no vertices in the interior of the quadrilateral and crossings cannot occur between green segments. So we know which green segments to split and how to insert the flipped red edge. *None of these considerations required any coordinates or lengths.*

The flip shown on the right of Figure 4 illustrates the special case when a yellow edge is flipped, or, vice versa, a red edge is flipped *onto* a green edge. The procedure is the same as described above, except that flipping a yellow edge does not lead to its removal but rather only to a color change (making it green). Reversely, if a red

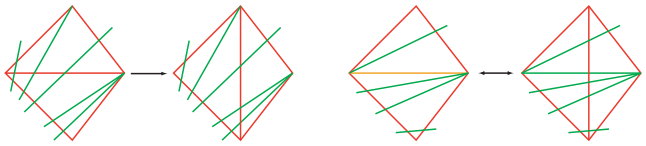


Figure 4: *Maintaining the overlay topology. Left: the common case when a red edge is flipped and remains red. Right: the special cases when a yellow edge is flipped, or, reversely, a red edge is flipped onto a green edge. Each case also illustrates a variety of possible green/red crossings.*

edge is flipped onto a green edge, the result is a yellow edge.

In either example, some or all of the four boundary edges of the quadrilateral might be yellow instead of red. Such edges do not contain any crossings. The procedure remains the same. This concludes the description of all possible cases.

Updating Edge Lengths First note that during the algorithm, lengths of segments are not required, only lengths of edges. If actual crossing intersection points, *e.g.*, in terms of segment lengths, are required, one can find these by laying out the quadrilateral hinge of a given edge. We may think of segment lengths as barycentric coordinates with respect to their containing edge (and its length). Assuming that all segment lengths are known before an edge flip, one can lay out an isometric copy of the red quadrilateral in the plane and obtain coordinates for the four vertices and all crossings of green segments with the boundary. These can be used to obtain coordinates for the new crossings of green edges with the flipped edge, and thus to calculate the new segment lengths. The layout process must be able to cope with identified edges/segments or vertices, since the triangulation is in general not regular. In particular a given vertex in the mesh may have multiple different (u, v) coordinates in the layout.

3 IDT in Applications

The discrete Laplace Beltrami operator is central in applications ranging from denoising/smoothing (intrinsic mean curvature flow [Desbrun et al. 1999]), to editing (using the bi-Laplace-Beltrami operator [Botsch and Kobbelt 2004b]), texture mapping (using a pair of discrete harmonic functions [Desbrun et al. 2002]), and construction of discrete minimal surfaces [Pinkall and Polthier 1993], to give but a few references (see also the references in [Grinspun et al. 2005]). In all of these cases one needs to solve (sequences of) linear problems involving a system matrix Δ with off-diagonal entries

$$\Delta_{ij} = -(\cot \alpha_{ij}^k + \cot \alpha_{ij}^l)$$

for edge e_{ij} where α_{ij}^k is the angle opposite to edge e_{ij} in triangle t_{ijk} (and similarly for α_{ij}^l). The diagonal Δ_{ii} holds the negative sum of the off-diagonal entries in that row. (We ignore here scaling factors which arise in various applications.) Δ is symmetric and positive definite, given appropriate boundary conditions. These are typically given as desired values (Dirichlet data) or cross boundary derivatives (Neumann data). Non-Delaunay edges in the mesh give rise to negative cotan weights, which leads to a loss of the local (discrete) maximum principle. One symptom of this is the loss of injectivity ("flipped triangles") when computing texture maps. Such issues are entirely avoided if we use the iLB operator, *i.e.*, Δ_{ij} depends on the iDT. Note that the *number* of non-zero entries in the matrix is the same for both eLB and iLB, thus having no impact on the operations count in a single application of the matrix to a vector.

Table 1 gives some statistics of the edge flipping algorithm for a number of representative meshes. Comparing the total number of flips with the number of iDT edges which cross exactly two original triangles ("simple") we see that this simplest case is by far the most common. The other extreme is the iDT edge consisting of the most segments, *i.e.*, crossing the longest ("lgst.") chain of original triangles. These are generally also short with the notable exception of the Camel where the longest chain of segments is length 22. An aggregate measure is the total number of crossings generated for all iDT edges: Max Planck (7167), Bunny (2434), Camel (25043), Horse (4027), and Feline (13317). Here the Camel stands out with more than half as many crossings as original vertices. More typical are cases such as Horse and Feline. At the extreme are high quality remeshes such as those of Botsch and Kobbelt [2004a] with rarely more than a small handful of non-Delaunay edges.

Model	V	flips	simple	lgst.	κ_i/κ_e
Cat hd.	131	45	40	3	0.8114
Bny hd.	741	380	275	6	0.6139
Bty. Frk.	1042	560	364	12	0.1841
Hygeia	8268	4017	2861	6	0.1053
Planck	25445	6417	5584	5	0.7746
Bunny	34834	2365	2283	4	0.0758
Camel	40240	17074	12618	22	0.7218
Horse	48485	3581	3127	7	0.6734
Feline	49864	12178	10767	7	0.5746

Table 1: *Statistics for some representative meshes. Number of: vertices; edge flips; flipped edges crossing only two original triangles; maximal number of segments in an iDT edge; and condition number improvement as ratio for iLB and eLB). All runs were well under 1 second on a 2GHz Athlon.*

Figure 5 shows a histogram of coefficients for the iLB versus eLB operator for the Hygeia model (other models have similar histograms). Obviously there are no negative coefficients anymore, but we also see a noticeable decrease in the number of large coefficients.

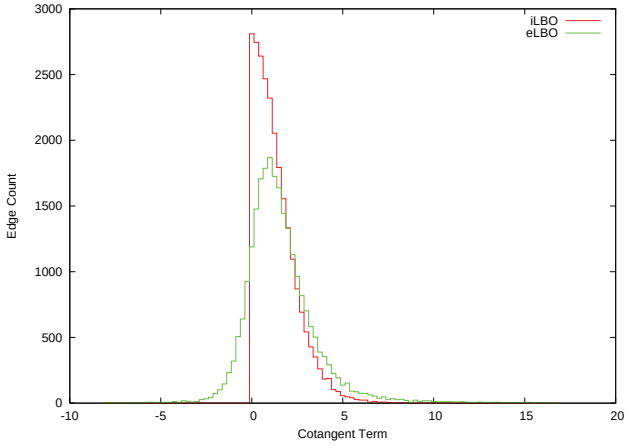


Figure 5: *Histogram of coefficients in the eLB versus iLB operator for the Hygeia model.*

For numerical considerations the most relevant figure of merit is the condition number of the Laplace-Beltrami operator, which we evaluate by considering the ratio of iLB (κ_i) to eLB condition number (κ_e). Generally we find an improvement between 20% and 40%, due to the largest eigenvalue being decreased (as predicted by theory [Rippa 1990]). Notable outliers are the "Beautiful Freak" dataset [Desbrun et al. 2002], which was specifically engineered to be challenging (80% improvement) and the Bunny (over 90% improvement).

The numerical improvements in the iLB operator over the eLB are

also noticeable in the quality of discrete harmonic parameterizations [Desbrun et al. 2002; Lévy et al. 2002; Mercat 2001; Gu and Yau 2003]. In this case the overlay graph is required to properly interpolate the induced texture mapping functions. Figure 6 shows the original triangulation (left column) and iDT (right column) for the "Beautiful Freak" dataset when mapped to the texture plane using Dirichlet (disk) boundary conditions. The resulting checker board pattern when mapped onto the surface is shown below. (Note that because of the Dirichlet boundary conditions we do not expect the resulting texture to be conformal.) The distortion in each triangle

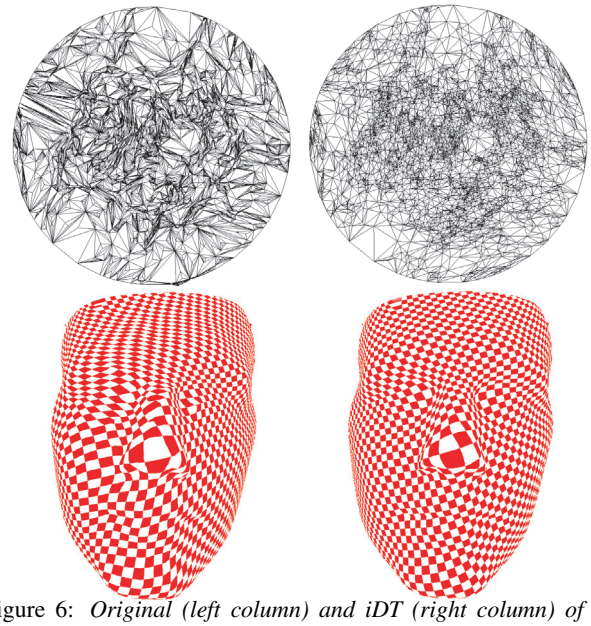


Figure 6: *Original (left column) and iDT (right column) of the Beautiful Freak dataset (Dirichlet boundary conditions). Texture plane image and resulting checker board mapping onto the surface.*

can be visualized by plotting the ratio of largest to smallest singular value of the Jacobian. In the case of conformal parameterizations this ratio can be as low as unity. Figure 7 compares the results for original triangulation (left column) and iDT (right column) with Dirichlet (top) boundary conditions to the disk (see Figure 6) and natural boundaries (bottom). As expected the distortion is overall lower for natural boundaries, but even in that case there is still a marked difference between original triangulation and iDT.

The lowering of distortion is also noticeable in other parameterization applications. Kharevych *et al.* [Kharevych et al. 2006] used circle patterns to compute discrete conformal mappings for embedded meshes. Briefly, in this approach discrete conformal mappings for triangle meshes (of arbitrary topology) are computed via triangle circumcircles and the angles they make with one another at shared edges. A requirement of the underlying theory is that all these angles be in $[0, \pi]$. While this can be enforced through clipping illegal values to the nearest legal value, a better approach is to change the combinatorics of the triangulation so that it is iDT. Figure 8 compares the distortion for two datasets when using the original triangulation (left column) and iDT (right column).

4 Conclusion

Given a triangle mesh annotated with lengths for every edge (assuming these lengths satisfy the triangle inequality) an intrinsic Delaunay triangulation is well defined and can be found with the help of a simple and low cost edge flipping algorithm. If explicit representations of the iDT edge crossings are required, an overlay graph can be incrementally maintained, using only combinatorial considerations. For applications which require a discrete Laplace-

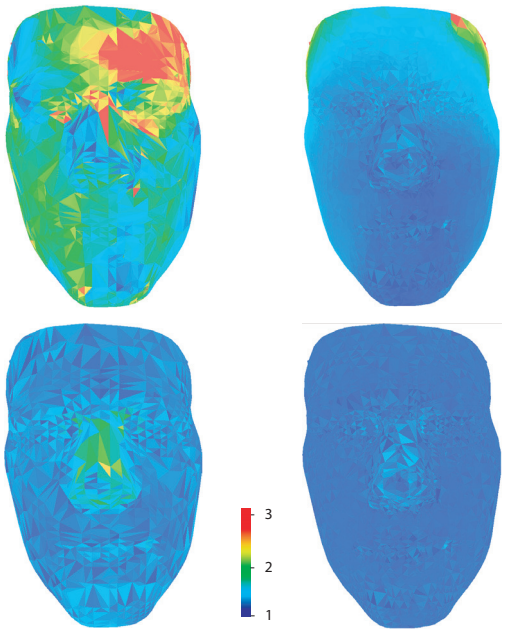


Figure 7: *Comparison of distortion (ratio of large to small singular values of Jacobian) between original triangulation (left column) and iDT (right column) for Dirichlet (top row) and natural (bottom row) boundary conditions.*

Beltrami operator (*e.g.*, denoising, parameterization, editing, simulation), defining it over the iDT has numerical advantages from improved condition numbers to lower error in the computations, and a reduced need for special case handling.

Clearly an (embedded) mesh which is intrinsically Delaunay to begin with is most desirable. It would be interesting to explore this as a constraint in remeshing algorithms, be they for static, or dynamically deforming, surfaces. Finally, we mention the challenge of making the algorithm robust. While we have not experienced any problems in our experiments, correct execution (and termination) of the implementation of the algorithm may require more than simple floating point arithmetic.

Acknowledgments This work was supported in part by NSF (CCF-0528101), DFG Research Center MATHEON "Mathematics for Key Technologies", DOE (W-7405-ENG-48/B341492), the Caltech Center for Mathematics of Information, nVidia, Alias, and Pixar. Special thanks to Mathieu Desbrun, Yiying Tong, Liliya Kharevych, Herbert Edelsbrunner, and Cici Koenig.

References

BOBENKO, A. I., AND SPRINGBORN, B. A., 2005. A discrete Laplace-Beltrami operator for simplicial surfaces. Preprint http://www.arxiv.org/math/0503219.

BOTSCH, M., AND KOBBELT, L. 2004. A Remeshing Approach to Multiresolution Modeling. In *Symp. on Geom. Proc.*, 185–192.

BOTSCH, M., AND KOBBELT, L. 2004. An Intuitive Framework for Real-Time Freeform Modeling. *ACM Trans. on Graphics 23*, 3, 630–634.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. 1999. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *Comp. Graphics (Proc. of SIGGRAPH)*, 317–324.

Figure 8: *Two examples (Hygeia, genus zero; cut camel with free boundaries) comparing distortion for original triangulation (left column) and iDT (right column). (Used with permission from [Kharevych et al. 2006].)*

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic Parameterizations of Surface Meshes. *Comp. Graph. Forum (Proc. of EG) 21*, 3, 209–218.

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete Shells. In *Symp. on Comp. Anim.*, 62–67.

GRINSPUN, E., SCHRÖDER, P., AND DESBRUN, M., Eds. 2005. *Discrete Differential Geometry*. Course Notes. ACM SIGGRAPH.

GU, X., AND YAU, S.-T. 2003. Global Conformal Surface Parameterization. In *Symp. on Geom. Proc.*, 127–137.

INDERMITTE, C., LIEBLING, T. M., TROYANOV, M., AND CLÉMENÇON, H. 2001. Voronoi Diagrams on Piecewise Flat Surfaces and an Application to Biological Growth. *Th. Comp. Sci. 263*, 263–274.

KHAREVYCH, L., SPRINGBORN, B., AND SCHRÖDER, P. 2006. Discrete Conformal Mappings via Circle Patterns. *ACM Trans. on Graphics*.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Trans. on Graphics 21*, 3, 362–371.

MERCAT, C. 2001. Discrete Riemann Surfaces and the Ising Model. *Comm. in Math. Physics 218*, 1, 177–216.

PINKALL, U., AND POLTHIER, K. 1993. Computing Discrete Minimal Surfaces and Their Conjugates. *Exp. Math. 2*, 1, 15–36.

RIPPA, S. 1990. Minimal Roughness Property of the Delaunay Triangulation. *CAGD 7*, 6, 489–497.

SHEWCHUK, J. 2002. What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Proc. of Intl. Meshing Roundtable*, 115–126.

Discrete Geometric Mechanics for Variational Time Integrators

Ari Stern Mathieu Desbrun

Caltech

Abstract

In this chapter, we present a *geometric*—instead of a traditional numerical-analytic—approach to the problem of time integration. Geometry at its most abstract is the study of symmetries and their associated invariants. Variational approaches based on such notions are commonly used in geometric modeling and discrete differential geometry. Here we will treat mechanics in a similar way. Indeed, the very essence of a mechanical system is characterized by its *symmetries* and *invariants*. Thus preserving these symmetries and invariants (*e.g.*, certain momenta) into the discrete computational setting is of paramount importance if one wants discrete time integration to properly capture the underlying continuous motion. Motivated by the well-known variational and geometric nature of most dynamical systems, we review the use of *discrete variational principles* as a way to derive robust, and accurate time integrators.

1 Introduction

Prediction is difficult, especially of the future.

—Mark Twain

Time Evolution of Dynamical Systems Time evolving phenomena such as the swinging of a clock pendulum, the bouncing of a soft ball on the floor, or even biological systems and stock market indicators are often modeled (*i.e.*, studied and understood) as dynamical systems. Mathematical models of the evolution in time of these systems generally involve systems of differential equations. *Solving* a physical system means figuring out how to move the system forward in time from a set of initial conditions, allowing the computation of, for instance, the trajectory of the soft ball (*i.e.*, its position as a function of time) thrown onto the floor. Although this example can easily be solved analytically, direct solutions of the differential equations governing a system are generally hard or impossible—we need to resort to *numerical techniques* to find a discrete temporal description of a motion. Consequently, there has been a significant amount of research in applied mathematics on how to deal with some of the most useful systems of equations, leading to a plethora of numerical schemes with various properties, orders of accuracy, and levels of complexity of implementation (see [Press et al. 1992] for a general overview).

Accurate vs. Qualitative Integrators While it is unavoidable to make approximations in numerical algorithms (*i.e.*, to differ from the continuous equivalent), the matter becomes whether the numerics can provide satisfactory results. The notion of satisfactory is, however, objective-dependent. If simulation is used for the design of a plane wing through a series of tests over a wide range of situations, *qualitative* reproduction of the wing behavior may be preferable over absolute numerical *accuracy*. If, however, simulation is used to find the proper launch parameters for a satellite to be put at a particular orbit, accurate results are crucial. This apparent mismatch in objectives has been, historically, aggravated by the cultural gap existing between applied and theoretical communities. We will show that in fact, one does not have to ask for *either* predictability *or* accuracy: simple methods exist that guarantee *good statistical predictability* by respecting the geometric properties of the exact flow of the differential equations, while being *also* easily rendered arbitrarily accurate.

Animation, or Simulation? In Computer Animation, time integrators are crucial computational tools at the core of most physics-based animation techniques. Animating a rigid body for instance uses the principles of classical mechanics, involving second order differential equations. In their most rudimentary form, these principles express the *relationship between forces acting on the body and its acceleration* given by *Newton's laws of motion*. From these equations of motion, classical time integrators (such as fourth-order Runge-Kutta, implicit Euler, and more recently the Newmark scheme) have been methods of choice in practice [Parent 2001; Hauth et al. 2003] to result in motions with good visual behavior—arguably, the top priority in graphics. Nonetheless, allowing the equations of motion to be slightly violated is commonly used to better control the resulting animation [Barzel et al. 1996], as long as it still looks visually plausible. In other words, local accuracy can be tinkered with just as long as the motion is still "globally" right.

Goals In this chapter, we provide an introduction to geometric mechanics, first from a continuous, then from a discrete point of view. Departing sharply from traditional numerical-analytic expositions, we point out how respecting the geometry of mechanics is not only natural, but it provides simple and powerful foundations for the design of robust time integrators. In particular, we will introduce the notion of *variational integrators* as a class of solvers specifically designed to preserve this underlying physical structure, even for large time steps that would produce overdamped or diverging results with more traditional methods.

2 Geometric Approach to Mechanics

Dynamics as a Variational Problem Considering mechanics from a variational point of view goes back to Euler, Lagrange and Hamilton. The form of the variational principle most important for continuous mechanics is due to Hamilton, and is often called *Hamilton's principle* or the *least action principle*: it states that a dynamical system always finds an optimal course from one position to another—or, as P.L. Moreau de Maupertuis put it, "Nature is thrifty in all its actions". A more formal definition will be presented in Section 4.1, but one consequence is that we can recast the traditional way of thinking about an object accelerating in response to applied forces into a geometric viewpoint. There, the path followed by the object has *optimal geometric properties*—analog to the notion of geodesics on curved surfaces. This point of view is equivalent to Newton's laws in the context of classical mechanics, but is broad enough to encompass areas ranging to E&M and quantum mechanics.

Discrete Structure-Preserving Integrators Geometric integrators are a class of numerical time-stepping methods that exploit this geometric structure of mechanical systems [Hairer et al. 2002]. Of particular interest within this class, *variational integrators* [Marsden and West 2001] discretize the variational formulation of mechanics we mentioned above, providing a solution for most ordinary and partial differential equations that arise in mechanics. While the idea of discretizing variational formulations of mechanics is standard for elliptic problems using Galerkin Finite Element methods for instance, only recently has it been used to derive variational time-stepping algorithms for mechanical systems. This approach allows the construction of integrators with any order of accuracy [West 2003; Lew 2003], and can handle constraints as well as external forcing. Results have been shown to be equal or superior to all other types of integrators for simulations of a large range of physical phenomena [Kane et al. 2000], making this discrete-geometric framework both versatile and powerful.

Of particular interest in computer animation, the simplest variational integrator can be implemented by taking two consecutive positions $q_0 = q(t_0)$ and $q_1 = q(t_0+dt)$ of the system to compute the next position $q_2 = q(t_0 + 2dt)$. Repeating this process calculates an entire discrete (in time) trajectory. In this chapter, we describe the foundations necessary to derive such variational schemes based on geometric arguments.

3 A Motivating Example: The Pendulum

Before we delve into the details of what variational integrators are, let us first look at a simple example to exemplify how slight variations in the design of time integrators can result in widely different behaviors.

3.1 Setup and Conventions

Consider a simple pendulum of mass m and length L, swinging under the influence of the gravitational acceleration g. Let $q(t)$ represents the pendulum's angle with the vertical at time t. As this angle is the only degree of freedom for this simple example, we can express the equations of motion for this system based solely on q and its derivatives:

$$\ddot{q} = -\frac{g}{L}\sin q, \tag{1}$$

where we use the "dot" notation to represent derivatives with respect to time, *i.e.*:

$$\dot{q} := \frac{dq}{dt}, \qquad \text{and} \qquad \ddot{q} := \frac{d^2q}{dt^2}.$$

We can rewrite this equation as a system of two *coupled* first-order equations in the variables q and v:

$$\dot{q} = v \tag{2}$$

$$\dot{v} = -\frac{g}{L}\sin q \tag{3}$$

If the initial conditions $q(0)$ and $\dot{q}(0)$ are given, then we could theoretically solve this differential equation for q. Assume for a moment that we don't have access to the analytical solution to this problem (in fact, as in many cases, no such solution is known). We can only hope to *approximate* the solution using an integrator. To achieve this goal, we first discretize the problem. That is, we break up time into N equal steps of length h, so that we no longer have a continuous notion of time, but have instead a discrete set of times $t_k = kh$. Then, finding an approximation to the differential equation on our new discrete time domain is tantamount to solving for the values of the angles at the various times, *i.e.*, finding the values $q_k = q(t_k)$ for $k = 1, \ldots, N$.

Given this setup, how can we compute the q_k's? There are actually many choices, and the important point to realize is, not all of them perform equally well.

3.2 Three Numerical Schemes

Assuming that the time step h is small enough with respect to all other derivatives of q, we could leverage the well-known Taylor expansion:

$$q(t+h) = q(t) + h\dot{q}(t) + \mathcal{O}(h)\,.$$

Using this first order approximation, one can easily derive the following, straightforward update rules by applying Taylor expansion to both q and v:

$$\begin{cases} q_{k+1} = q_k + h\, v_k \\ v_{k+1} = v_k - h\, \dfrac{g}{L}\sin q_k \end{cases}$$

Given the previous values q_k, v_k, this method gives us an explicit formula to compute the next values in time q_{k+1}, v_{k+1}; this specific time integrator is called the ***explicit Euler method***. Repeating this procedure by setting $k := k + 1$ provides a way to compute the whole motion.

Alternatively, we could change the time integration procedure by evaluating the right hand sides of the former rules at the *next* time step, through:

$$\begin{cases} q_{k+1} = q_k + h\, v_{k+1} \\ v_{k+1} = v_k - h\, \dfrac{g}{L}\sin q_{k+1} \end{cases}$$

This method is no longer explicit, but *implicit*: one needs to use a (non-linear) solver to find the pair q_{k+1}, v_{k+1} that satisfy these equations, given the current values q_k and v_k. This time integrator is traditionally called the ***implicit Euler method***.

Finally, one could use a seemingly strange mix of the two, by first updating v_{k+1} explicitly, then q_{k+1} using the new value v_{k+1} (thus, still explicitly):

$$\begin{cases} v_{k+1} = v_k - h\, \dfrac{g}{L}\sin q_k \\ q_{k+1} = q_k + h\, v_{k+1} \end{cases}$$

Notice that the difference with the first scheme is rather minimal. However, this particular time integrator is known as the ***symplectic Euler method***.

These three methods are called *finite difference methods*, since they approximate the left-hand side derivatives of Eqs. (2-3) by taking the difference between consecutive values. Notice in particular that, while the implicit method is more computationally expensive, the two others involve the exact same amount of operations. Thus, their behavior should not be very different, right?

3.3 Comparing Integrators

Numerical tests of these three integrators reveal obvious differences in practice (to avoid going too much into sordid details of numerical analysis, we will stick to a fixed time step $h = 0.01$ for all experiments). First, one quickly realizes that the explicit Euler suffers from stability problems: the motion of the pendulum *amplifies* over time! An obvious consequence is that the pendulum's energy

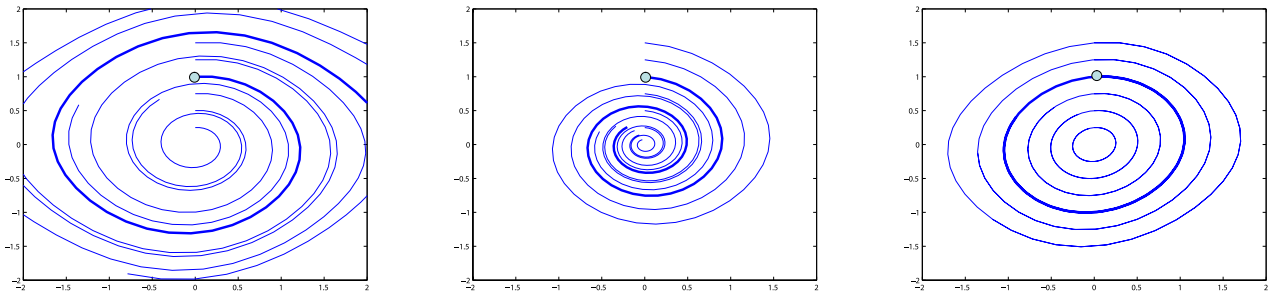


Figure 1: *Three integrators in phase space (q, p): (left) explicit, (middle) implicit, (right) symplectic. Six initial conditions are shown, with their respective trajectories; only the symplectic integrator captures the periodic nature of the pendulum. The bold trajectories correspond to the exact same initial condition.*

increases over time, rather than being conserved. Thus, in practice, the solution often "blows up" and becomes unstable as time progresses—not a great quality for a time integrator. Fortunately, the implicit Euler *is* stable: the amplitude of the pendulum's oscillations actually *decreases* over time, avoiding any chance of numerical divergence (see Fig. 2). However, this stability comes at a cost: the pendulum *loses* energy, causing the pendulum to slow down towards a stop, even if our original equations do not include any damping forces. Effectively, we resolved the stability issue through the introduction of ***numerical dissipation***—but we induced the opposite problem instead. The symplectic method, on the other hand, both is stable *and* oscillates with constant amplitudes. This is obviously a superior method for physical simulation, given that no additional numerical operations were needed to get the correct qualitative behavior!

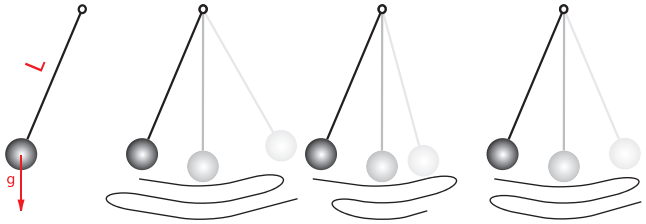


Figure 2: *The pendulum: for the equation of motion of a pendulum of length L and unit mass in a gravitation field g (left), our three integrators behave very differently: while the explicit Euler integrator exhibits amplifying oscillations, the implicit one dampens the motion, while the symplectic integrator perfectly captures the periodic nature of the pendulum.*

Now, if we are only solving for the position of the pendulum only at one particular time, it does not really matter which method we use: taking small enough time steps will guarantee arbitrarily good accuracy. However, if we wish our time integrator to be globally predictive, the least we can ask for is to get a pendulum that actually keeps on swinging. Even a simple animation of a grandfather clock or a child on a swing would look unrealistic if it seemed to gain or lose amplitude inexplicably. In other words, the behavior of energy over time is of key importance. But how do we know that an integrator will have these good properties ahead of time? Can we construct them for an arbitrary physical system? The answer, as we shall see, comes from the world of geometric mechanics and a concept called *symplecticity*.

4 Geometric Mechanics

In the familiar Newtonian view of mechanics, we begin by adding up the forces F on a body and writing the equations of motion using the famous second law,

$$F = ma, \tag{4}$$

where a represents the acceleration of the body. With geometric mechanics, however, we consider mechanics from a variational point of view. In this section, we review the basic foundations of *Lagrangian mechanics*, one of the two main flavors of geometric mechanics (we will only point to some connections with *Hamiltonian mechanics*).

4.1 Lagrangian Mechanics

Consider a finite-dimensional dynamical system parameterized by the *state variable* q, i.e., the vector containing all degrees of freedom of the system. In mechanics, a function of a position q and a velocity $\dot{q}$ called the Lagrangian function L is defined as the kinetic energy K (usually, only function of the velocity) minus the potential energy U of the system (usually, only function of the state variable):

$$L(q, \dot{q}) = K(\dot{q}) - U(q).$$

Variational Principle The *action functional* is then introduced as the integral of L along a path $q(t)$ for time $t \in [0, T]$:

$$S(q) = \int_0^T L(q, \dot{q})\, dt.$$

With this definition, the main result of Lagrangian dynamics, *Hamilton's principle*, can be expressed quite simply: this variational principle states that *the correct path of motion of a dynamical system is such that its action has a stationary value*, *i.e.*, the integral along the correct path has the same value to within first-order infinitesimal perturbations. As an "integral principle" this description encompasses the entire motion of a system between two fixed times (0 and T in our setup). In more ways than one, this principle is very similar to a statement on the *geometry* of the path $q(t)$: the action can be seen as the analog of a measure of "curvature", and the path is such that this curvature is extremized (*i.e.*, minimized or maximized).

Euler-Lagrange Equations How do we determine which path optimizes the action, then? The method is similar to optimizing an ordinary function. For example, given a function $f(x)$, we know that its critical points exist where the derivative $\nabla f(x) = 0$. Since q is a path, we cannot simply take a "derivative" with respect to q; instead, we take something called a ***variation***. A variation of the path q is written δq, and can be thought of as an infinitesimal

perturbation to the path at each point, with the important property that the perturbation is null at the endpoints of the path. Computing variations of the action induced by variations δq of the path $q(t)$ results in:

$$\delta S(q) = \delta \int_0^T L(q(t), \dot{q}(t))\, dt = \int_0^T \left[\frac{\partial L}{\partial q} \cdot \delta q + \frac{\partial L}{\partial \dot{q}} \cdot \delta \dot{q} \right] dt$$
$$= \int_0^T \left[\frac{\partial L}{\partial q} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) \right] \delta q \; dt \; + \; \left[\frac{\partial L}{\partial \dot{q}} \cdot \delta q \right]_0^T ,$$

where integration by parts is used in the last equality. When the endpoints of $q(t)$ are held fixed with respect to all variations $\delta q(t)$ (*i.e.*, $\delta q(0) = \delta q(T) = 0$), the rightmost term in the above equation vanishes. Therefore, the condition of stationary action for arbitrary variations δq with fixed endpoints stated in Hamilton's principle directly indicates that the remaining integrand in the previous equation must be zero for all time t, yielding what is known as the *Euler-Lagrange equations*:

$$\frac{\partial L}{\partial q} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) \; = \; 0. \tag{5}$$

For a given Lagrangian, this formula will give the *equations of motion* of the system.

Forced Systems To account for non-conservative forces or dissipation F, the least action principle is modified as follows:

$$\delta \int_0^T L(q(t), \dot{q}(t))\, dt + \int_0^T F(q(t), \dot{q}(t)) \cdot \delta q \; dt = 0 \, .$$

This is known as the *Lagrange-d'Alembert principle*.

Lagrangian vs. Hamiltonian Mechanics. Hamiltonian mechanics provides an alternative formulation, which is closely related to the Lagrangian. The reader may consult any book on mechanics for the relationships between the two descriptions. We simply point out here (as it will be useful later) that in the Hamiltonian formulation, the dynamics are described in *phase space*, i.e, the current state of a dynamical system is given as a pair (q, p), where q is the state variable, while p is the momentum, defined by $p = \partial L / \partial \dot{q}$.

4.2 Example

Let make the previous definitions more concrete by detailing a particularly simple example. Given a particle with mass M in a gravitational field, *i.e.*, in a potential field $V = Mg \cdot q$, the Lagrangian is written:

$$L(q, \dot{q}) = \frac{1}{2} \dot{q}^T M \dot{q} - Mg \cdot q.$$

Taking the variation of the action, one gets:

$$\delta \int_a^b \left(\frac{1}{2} \dot{q}^T M \dot{q} - Mg \; q \right) dt = \int_a^b \left(M\dot{q} \cdot \delta \dot{q} - Mg \cdot \delta q \right) dt.$$

Next, we integrate the $\delta \dot{q}$ term by parts; the boundary terms disappear, since $\delta q = 0$ at the endpoints.

$$= \int_a^b (-M\ddot{q} - Mg) \cdot \delta q \, dt = 0.$$

Since the integral equals 0 for any variation δq, the first term inside the integral must equal 0. Therefore, the Euler-Lagrange equations become:

$$M\ddot{q} = -Mg,$$

which are precisely the Newtonian equation of motion $F = ma$.

4.3 Symmetries and Invariants

Finally, we arrive at a crucial question: why exactly do physical systems conserve certain quantities? If we can answer this question *and* mimic the continuous dynamics in our discrete implementations, only then can we hope to get good numerical properties for our time integrators. This question is partially answered by ***Noether's theorem***, an extremely powerful theorem in physics which states that each symmetry of a system leads to a physical invariant (*i.e.*, a conserved quantity). For example, take the dynamics of an elastic object in the void. The Lagrangian can easily be shown to be translation invariant: translating all the mass particles of the elastic object would not change the value of the Lagrangian. Similarly, the Lagrangian is rotation-invariant as moving all the particles of the object by a global rotation has no reason to affect the Lagrangian either. This means that the system has a *translational and rotational symmetry*. Noether's theorem then states that the *linear and angular momenta* are preserved. These symmetries, if respected in the discrete setting, will provide equivalent discrete invariants in time integrators! In fact, we will see that these invariants can be preserved in time integrators at no extra computational cost by simply respecting the geometric, variational nature of dynamics.

4.4 Phase Space and Symplecticity

To visualize a dynamical system, we often plot its trajectories in ***phase space***. In its simplest version as in the one-dimensional pendulum case, it is in fact a phase plane where one axis represents the position q and the other axis represents either velocity $\dot{q}$ or, more usually, momentum $p = m\dot{q}$. Note that for higher dimensional systems, there is an additional axis corresponding to each additional position component q^i and its corresponding velocity $\dot{q}^i$ (or momentum p_i). The graphs that result from plotting the trajectories in phase space are called ***phase portraits***.

Going back to our motivating example of the pendulum, we can now more clearly see the qualities/flaws of the time integrators by looking at their respective phase portraits in Fig. 1. While the pendulum's phase portrait has a characteristic structure of nested, energy-preserving orbits (since the oscillations are periodic), this was not true for the two first discrete approximations: the trajectories of explicit Euler spiraled outwards (dramatically increasing magnitude of oscillations, thus energy), while those of implicit Euler spiraled inwards. Why did some of the phase portraits look better than others? How can we preserve the closedness of the orbits without making the time integrator more complicated?

One of the key features of Lagrangian flows (*i.e.*, motions) is that they are ***symplectic***. Formally, this means that the flow preserves the canonical two-form $\Omega = dq^i \wedge dp_i$. In the two-dimensional phase plane, this directly implies that *the area of any phase-space region is preserved under the flow* (see Liouville's theorem in classical mechanics). For example, let us take a given region of initial conditions in phase-space. If we advance all these states simultaneously, the regions deforms under the flow in a way that preserves the original area as shown in Fig. 3 a cat-head shaped region: this phenomenon is called *symplecticity*. However, as seen on this same figure, explicit and implicit Euler both fail the test of symplecticity. Because orbits spiral outward under explicit Euler, a region will *expand*, and its area will increase. Conversely, implicit Euler decreases the area inside the evolving region. Preserving this property of the flow in phase space for our time integrators (that is, having them be *symplectic in a discrete sense*) is key to ensure globally correct behavior!

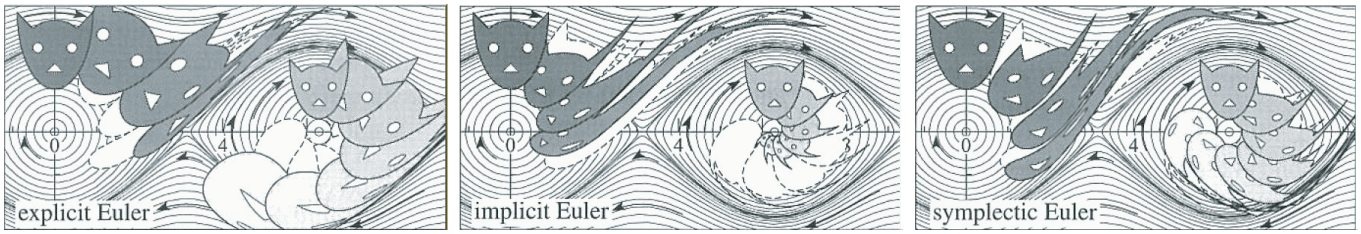


Figure 3: *Symplecticity [reproduced from [Hairer et al. 2002]]: while a continuous Lagrangian system is symplectic (that is to say, in this simple case, an area in phase space evolves along the flow without changing its area), discrete time integrators rarely share this property. From our three time integrators compared in Section 3, only the last one is symplectic. In the background, the reader will recognize the shape of the orbits obtained in Fig. 1(right).*

5 Discrete Geometric Mechanics

Having quickly reviewed classical Lagrangian mechanics in the continuous domain, we now explain how this geometric view of mechanics can elegantly be mimicked in the discrete setting.

5.1 General Idea

The driving idea behind discrete geometric mechanics is to leverage the variational nature of mechanics and to preserve this variational structure in the discrete setting. In fact, very few integrators have a variational nature: the explicit and implicit Euler methods discussed above are not variational, and not surprisingly, they both exhibited poor global behavior in the case of the pendulum. Instead of simply approximating the equations of motion to first (or higher) order as we did before, one can directly *discretize the variational principle* behind them. That is, if one designs a discrete equivalent of the Lagrangian, then discrete equations of motion can be easily derived from it by paralleling the derivations followed in continuous case. In essence, good numerical methods will come from discrete analogs to the Euler-Lagrange equations—equations that truly derive from a variational principle.

5.2 Discrete Lagrangian Dynamics

Setup The main idea is to discretize the least action principle directly rather than discretizing (5). To this end, a path $q(t)$ for $t \in [0,T]$ is replaced by a *discrete path* $q : \{t_0 = 0, t_1, \ldots, t_k, \ldots, t_N = T\}$ where $k, N \in \mathbb{N}$. Here, q_k is viewed as an approximation to $q(t_k)$.

Discrete Lagrangian The Lagrangian L is approximated on each time interval $[t_k, t_{k+1}]$ by a *discrete Lagrangian*[1] $L_d(q_k, q_{k+1}, h)$, with h being the time interval between two samples $h = t_{k+1} - t_k$ (chosen here to be constant for simplicity):

$$L_d(q_k, q_{k+1}) \approx \int_{t_k}^{t_{k+1}} L(q, \dot{q})\, dt.$$

Now, the right-hand side integral can be approximated through a one-point quadrature, i.e., by the length of the interval times the value of the integrand evaluated somewhere between q_k and q_{k+1} and with $\dot{q}$ replaced by $(q_{k+1} - q_k)/h$:

$$L_d(q_k, q_{k+1}, h) = h\, L\left((1-\alpha)q_k + \alpha q_{k+1}, \frac{q_{k+1} - q_k}{h}\right) \quad (6)$$

where $\alpha \in [0,1]$. For $\alpha = 1/2$, the quadrature is second-order accurate, while any other value leads to linear accuracy.

[1]This term could also be called an action, as it is a time integral of a Lagrangian; however, just like the term "discrete curvature" in CG refers to a small local integral of a continuous curvature, we prefer this naming convention.

Discrete Stationary Action Principle Given the discrete Lagrangian, the *discrete action functional* becomes simply a sum:

$$S_d := S_d(\{q_i\}_{i=0..N}) = \sum_{k=0}^{N-1} L_d(q_k, q_{k+1}) \approx \int_a^b L(q, \dot{q})\, dt = S(q).$$

Taking fixed-endpoint variations of this discrete action S_d, we obtain:

$$\delta S_d = \sum_{k=0}^{N-1} \Big[D_1 L_d(q_k, q_{k+1}) \cdot \delta q_k + D_2 L_d(q_k, q_{k+1}) \cdot \delta q_{k+1} \Big],$$

where $D_1 L$ (resp., $D_2 L$) denotes the partial derivative with respect to the first (resp., second) arguments of L. Reindexing the rightmost terms, and using the fixed endpoint condition $\delta q_0 = \delta q_N = 0$, one gets:

$$\delta S_d = \sum_{k=1}^{N-1} \Big[D_1 L_d(q_k, q_{k+1}) + D_2 L_d(q_{k-1}, q_k) \Big] \cdot \delta q_k.$$

Setting this variation equal to 0 and noting that each δq_k is arbitrary, we arrive at the ***discrete Euler-Lagrange (DEL) equations***

$$D_1 L_d(q_k, q_{k+1}) + D_2 L_d(q_{k-1}, q_k) = 0. \quad (7)$$

Notice that this condition only involves three consecutive positions. Therefore, for two given successive positions q_k and q_{k+1}, Eq. (7) *defines* q_{k+2}. That is, these equations of motion are actually the algorithm for an integrator! And since the DEL equations derive from the extremization of a discrete action, such an algorithm enforces the variational aspect of the motion numerically.

Link to Previous Numerical Schemes Let us go back to the pendulum case. For this system, the Lagragian (kinetic energy minus potential energy) is:

$$L(q, \dot{q}) = \frac{1}{2} L^2 \dot{q}^2 + gL\, \cos(q).$$

First, the user can convince her/himself that the Euler-Lagrange equation is indeed, Eq. (1) through a simple derivation. Second, it is also a simple (yet, interesting) exercise to verify that the symplectic Euler integrator used earlier results from the DEL equations just described, for the particular choice of $\alpha = 0$ in the quadrature rule defined in Eq. 6.

5.3 Update Rule in Phase Space

In mechanics, the initial conditions are typically specified as a position and a velocity or momentum rather than two positions, therefore it is beneficial to write (7) in a position-momentum form [West 2003]. To this end, define the momentum at time t_k to be:

$$p_k := D_2 L_d(q_{k-1}, q_k) = -D_1 L_d(q_k, q_{k+1})$$

where the second equality holds due to (7). The position-momentum form of the variational integrator discussed above is then given by:

$$p_k = -D_1 L_d(q_k, q_{k+1}) , \qquad p_{k+1} = D_2 L_d(q_k, q_{k+1}). \quad (8)$$

For (q_k, p_k) known, (8)(left) is an (often implicit) equation whose solution gives q_{k+1}. q_{k+1} is then substituted in (8)(right) to find p_{k+1}. This provides an update rule in phase space.

5.4 Adding Dissipation

In case of forcing and/or dissipation, the discrete action can be modified by adding the non-conservative force term and using the discrete Lagrange-d'Alembert principle [Marsden and West 2001]:

$$\delta S_d + \sum_{k=0}^{N} \left(F_d^-(q_k, q_{k+1}) \cdot \delta q_k + F_d^+(q_k, q_{k+1}) \cdot \delta q_{k+1} \right) = 0.$$

where $F_d^-(q_k, q_{k+1})$ and $F_d^+(q_k, q_{k+1})$ are discrete external forces acting respectively on the right of q_k and on the left of q_{k+1}. In other words, $F_d^-(q_k, q_{k+1}) \cdot \delta q_k + F_d^+(q_k, q_{k+1}) \cdot \delta q_{k+1}$ can be seen as a two-point quadrature of the continuous forcing term $\int_{t_k}^{t_{k+1}} F \cdot \delta q \, dt$. The *forced discrete Euler-Lagrange equations* can be expressed in a convenient, position-momentum form as follows:

$$p_k = -D_1 L_d(q_k, q_{k+1}) - F_d^-(q_k, q_{k+1}) ,$$
$$p_{k+1} = D_2 L_d(q_k, q_{k+1}) + F_d^+(q_k, q_{k+1}).$$

This variational treatment of energy decay, despite its simplicity, has also been proven superior to the usual time integration schemes that often add numerical viscosity to get stability [West 2003].

5.5 Last Words

Variational integrators often perform better than their non-variational counterparts because they preserve the *underlying geometry* of the physical system. This has two important consequences. First, the integrators are guaranteed to be symplectic, which in practice will result in excellent energy behavior, rather than perpetual damping or blowing up. Second, they are also guaranteed to preserve discrete momenta of the system. As a consequence, simulations and animations using these integrators usually have great physical and visual fidelity with low computational cost. ***Caveat:*** The reader may be misled into thinking that our scheme does not require the typical Courant-Friedrichs-Levy (CFL) condition (or equivalent) on the time step size. This is, of course, untrue: the same usual theoretical limitations of explicit schemes are still valid for symplectic explicit schemes. However, we can easily design symplectic implicit schemes that do not share this particular limitation, generally allowing for much larger time steps. Finally, we can make them of arbitrarily higher order by simply improving the quadrature rule used to convert the continuous Lagrangian into a discrete Lagrangian.

Aknowledgements The authors are most grateful to Yiying Tong, Jerrold E. Marsden, and Eva Kanso for their constant help.

References

BARZEL, R., HUGHES, J., AND WOOD, D. N. 1996. Plausible motion simulation for computer graphics animation. In *Proceedings of the EG Workshop on Computer Animation and Simulation*, 183–197.

HAIRER, E., LUBICH, C., AND WANNER, G. 2002. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer.

HAUTH, M., ETZMUSS, O., AND STRASSER, W. 2003. Analysis of Numerical Methods for the Simulation of Deformable Models. *The Visual Computer 19*, 7-8, 581–600.

KANE, C., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2000. Variational Integrators and the Newmark Algorithm for Conservative and Dissipative Mechanical Systems. *Int. J. Numer. Methods Engrg. 49*, 1295–1325.

LEW, A. 2003. *Variational Time Integrators in Computational Solid Mechanics*. Phd thesis, California Institute of Technology.

MARSDEN, J. E., AND WEST, M. 2001. Discrete Mechanics and Variational Integrators. *Acta Numerica*, 357–515.

PARENT, R. 2001. *Computer Animation: Algorithms and Techniques*. Series in Computer Graphics. Morgan Kaufmann.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., AND VETTERLING, W. T. 1992. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press.

WEST, M. 2003. *Variational Integrators*. Phd thesis, California Institute of Technology.