FINDING GEODESICS ON SURFACES

TEAM 2: JONGMIN BAEK, ANAND DEOPURKAR,
AND KATHERINE REDFIELD

Abstract. We discuss different methods to compute geodesic paths on surfaces using methods from graph theory and numerical analysis.

1. Introduction

This paper focuses on the problem of computing geodesics on smooth surfaces. In the first few sections we assume we are given an approximate path to start from when attempting to compute a geodesic between two points. In section 2 we attempt to compute the geodesic between two points iteratively using the midpoints of an approximate path between them. In section 3 we explore a similar method, gradient descent, to iteratively update the path approximating the geodesic. In section 4 we compute geodesics by numerically solving the system of differential equations governing them. In section 5 we model the surface as a finite graph and apply graph search methods to find an approximate geodesic. In section 6 we test the methods on different surfaces and compare the results.

The following terminology will be used regularly throughout this paper. In these definitions, I refers to the closed interval $[0, 1]$. A *surface* is a smooth function from an open set $U \subset \mathbb{R}^2$ to $\mathbb{R}^3$ where $I^2 \subset U$. A path γ is a piecewise smooth map defined on I. We will sometime refer to a sequence of points as a path, this refers to the path constructed by connecting consecutive points. If γ is a path from I to I^2 and S is a surface, then $S \circ \gamma$ is a path from I to $\mathbb{R}^3$. The *arc length* of the path defined by $S \circ \gamma$ is denoted $L_s(\gamma)$, where

$$L_s(\gamma) = \int_0^1 \left\| \frac{d(S \circ \gamma)(t)}{dt} \right\| dt$$

. A path γ is a *geodesic* of a surface S if $L_s(\gamma)$ is locally minimized.

Date: November 16, 2007.

2. Midpoint Method

In this section our goal is to find an intuitive method for iteratively computing a geodesic from a point p to another point q on a surface S when we are given an approximate path γ to start from. We start by considering this problem in $\mathbb{R}^2$ in hopes of discovering a method which can be easily generalized to the problem of computing a geodesic on any surface.

2.1. Geodesics on a Plane.

To begin, we attempt to compute a geodesic from a point p to another point q, for $p, q \in \mathbb{R}^2$. We know that this path should be the straight line between p and q.

The *midpoint method* of computing a geodesic between two distinct points p and q in $\mathbb{R}^2$ can be described as follows:

(1) Start with a given approximation $\gamma_1 = p_1, \ldots, p_n$ made up of a finite set of points in $\mathbb{R}^2$ such that p_i and p_{i+1} are seperated by a small distance and the endpoints are $p_1 = p$ and $p_n = q$.
(2) Compute the midpoints of each segment, and connect them to get a new path. Place p_1 at the beginning of this path and p_n at the end of it. We call the resulting path γ_2.
(3) Iteratively perform step 2 to each new γ, until you have a path close enough to the geodesic.

Lemma 2.1. *Let γ be a path of length l_1 defined by a finite series of points in $\mathbb{R}^2$ that has distinct endpoints p and q. Let length l_2 of the path produced by an iteration of the midpoint method applied to γ. Then $l_2 \leq l_1$.*

This lemma follows directly from the triangle inequality.

Theorem 2.2. *Let γ be a path defined by a finite series of points in $\mathbb{R}^2$ that has endpoints p and q. Performing the midpoint method on this path will produce a series of points defining the straight line connecting p and q.*

Proof. Without loss of generality consider the path defined by the finite set of points $p_1, \ldots, p_n$ where $p_i = (x_i, y_i)$ and in particular $p_1 = (0, 0)$ and $p_n = (a, 0)$. The path resulting from m iterations of the midpoint method is made up of the points (x_i, y_i') where $y_i' = \frac{\sum_{j=0}^{\min(m,n-i)} y_{i+j}\binom{m}{j}}{2^m}$. Because the geodesic connecting p_1 and p_n should be a segment of the x-axis, we say that the error of this approximation is the magnitude of the y value furthest from 0. So since,

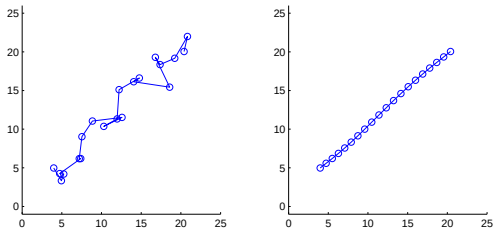
$$\lim_{m \longrightarrow \infty} \frac{\sum_{j=0}^{\min(m,n-i)} y_{i+j} \binom{m}{j}}{2^m} = 0$$

we know that by performing the midpoint method on our path, as the number of recursive calls increases the path returned approximates the straight line connecting $(0,0)$ and $(a,0)$ with an error that goes to 0.

In particular, we note that:

$$\begin{aligned} \left| \frac{\sum_{j=0}^{\min(m,n-i)} y_{i+j} \binom{m}{j}}{2^m} \right| &\leq \frac{\sum_{j=0}^{\min(m,n-i)} \left| y_{i+j} \binom{m}{j} \right|}{2^m} \\ &\leq \sum_{j=0}^{min(m,n-i)} |y_{i+j}| \cdot \frac{\binom{m}{m/2}}{2^m} \\ &\leq \sum_{j=1}^{n} |y_j| \cdot \frac{\binom{m}{m/2}}{2^m} \end{aligned}$$

So since $\sum_{j=1}^{n} |y_j|$ is just a constant, the midpoint method's x-th iteration produces an approximation of the geodesic whose error diminishes as $(\binom{x}{x/2}/2^x)$, which goes to 0. □

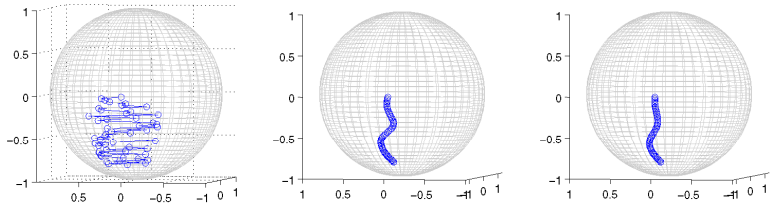


(a) Initial Path (b) Iteration 100

FIGURE 1. Midpoint Method on `plane`.

2.2. **Generalizing the Midpoint Method to $\mathbb{R}^3$.** While this midpoint method works well in $\mathbb{R}^2$, it does not work on a curved surface S because the midpoint m of any two consecutive points on our path is not guaranteed to be on S. To remedy this problem, we calculate the point m' on S closest to m in space and use that point in our path approximation instead of m. For simple surfaces such as spheres and toruses, m' can be calculated easily. On a random surface, m'

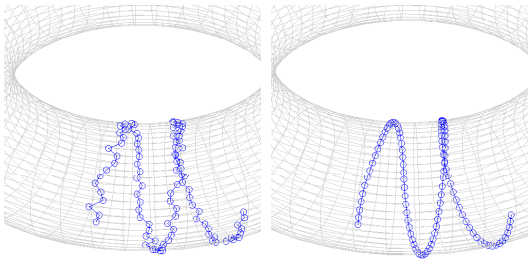
can be much more difficult to find so we use the built-in malab function `fminsearch` to compute it. Now we can iterate with the midpoint method as before to compte the geodesic on an arbitrary surface. Based on our test cases, this method continues to be effective.



(a) Initial Path (b) Iteration 100 (c) Iteration 200

FIGURE 2. Midpoint Method on `sphere`.

Consider the case of a sphere. The geodesic between two points on a sphere is always a segment of a *great circle*. Recall that a great circle on a sphere is one whose diameter passes through the center of the sphere. In the scope of our testing on spheres, the midpoint method consistently produced paths which approached the geodesic; see Figure 2 for an example. After 200 iterations, the path produced by the midpoint method appears much closer to the geodesic than the initial path provided.



(a) Initial Path (b) Iteration 100

FIGURE 3. Midpoint Method on `torus`.

It is important to note that a geodesic is not necessarily the global shortest path between p and q. Take as an example the torus in Figure 3. While the path our algorithm produced approachs a geodesic, it is clearly not the global shortest path between p and q over our surface. Because the original path wrapped around the torus, so does our final

path. This method has the same effect as looping a string around the torus following the original path given, and then pulling the string taut. The string will find the shortest path in a neighborhood of the original path, but the structure of the surface prevents it from moving to the global shortest path from its starting point.

Overall, the midpoint method generalizes rather intuitively to be used in calculating geodesics over curved surfaces, and will often find a good approximation of a geodesic after a relatively small number of iterations.

3. Gradient Descent

In this section we again attempt to iteratively compute a geodesic from an approximate path using a method similar to that described in Section 2.2. Now, instead of examining the midpoint of each segment of the path, we examine the middle point of each consecutive sequence of three points defined by the path and attempt to improve the path's approximation of the geodesic by altering points one by one.

Let us consider a simple model of iterative refinement of a path defined by three consecutive points. We call this sequence p_i, p_{i+1}, p_{i+2} in I^2. This section's method involves first fixing p_i and p_{i+2}, and then selecting a new middle point u from a neighborhood of the original p_{i+1} which allows the path to better approximate the geodesic connecting p_i and p_{i+2}.

The lower bound on the arc length of the geodesic connecting $S(p_i)$ and $S(p_{i+2})$ where S is a surface is the Euclidean distance $\|S(p_i) - S(p_{i+2})\|$. Assuming that this path must also contain p_{i+1}, the lower bound becomes $LB(p_{i+1})$ where

$$LB(x) = \|S(p_i) - S(x)\| + \|S(x) - S(p_{i+2})\|.$$

If the surface S is locally planar, and the points in the sequence are close enough, this lower bound will actually approximate the minimal geodesic length. Therefore, it would give us a better approximation of the minimal geodesic length if we decreased this lower bound on our arc length by choosing u that minimizes $LB(u)$.

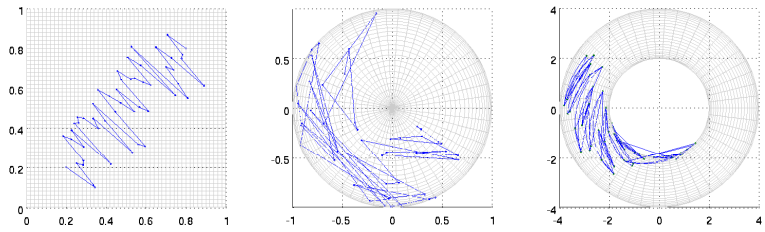
To minimize $LB(u)$ with respect to u, we employ *gradient descent*; that is, an iterative method in which u is updated in the direction opposing the gradient ∇LB at the current value of u, using some weight $\alpha > 0$:

$$u \quad \longleftarrow \quad u - \alpha \nabla LB(u).$$

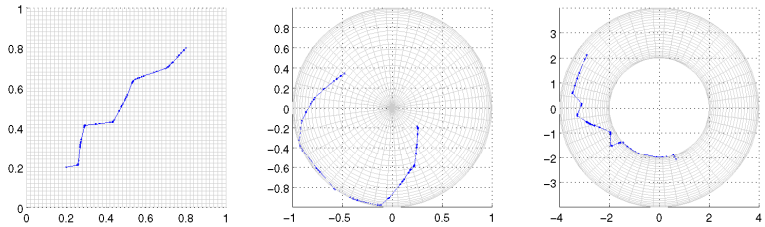
In our implementation, α is chosen such that it minimizes the following sum:

$$\|S(u - \alpha \nabla LB(u)) - S(p_i)\| + \|S(u - \alpha \nabla LB(u)) - S(p_{i+2})\|.$$

To extend this method from our short path to a path of arbitrary length, we consider three consecutive points at a time and update the middle point of each of these subsequences p_i, p_{i+1}, p_{i+2} of the path as we go.



(a) Initial Path on tt plane (b) Initial Path on tt sphere (c) Initial Path on tt torus



(d) Iteration 500 (e) Iteration 500 (f) Iteration 500

FIGURE 4. Gradient Descent on `plane`, `sphere` and `torus`.

We empirically observed that the gradient descent is effective only when $\|S(p_i) - S(p_{i+1})\|$ and $\|S(p_{i+1}) - S(p_{i+2})\|$ are comparable. Otherwise, the update is very slow. Hence, it is beneficial to drop p_{i+1} from the sequence if one leg becomes too small, and instead add a point to the part of the sequence that contains a long leg. In Figure 4 we see the results of using the gradient descent method on a random path over a plane, sphere and torus after 500 iterations. In general, while each individual iteration of the gradient descent method is much faster than the midpoint method, it overall takes many more iterations than the midpoint method to compute similarly accurate approximations of a geodesic on any given surface.

4. Differential Equation Approach

In this section we formulate the problem of finding geodesics on a surface as the problem of solving a system of ordinary differential equations. Although we treat the case of surfaces in $\mathbb{R}^3$ in this paper, we can easily generalize it to arbitrary submanifolds of $\mathbb{R}^N$.

4.1. **The Geodesic Equation.** Consider a smooth surface $M \subset \mathbb{R}^3$ given by a parametric map

$$S : U \to \mathbb{R}^3,$$

where $U \subset \mathbb{R}^2$ is an open set. We assume that S is a smooth immersion. In other words, we assume that dS_x is injective for all $x \in U$. A path $\gamma : I \to U$ gives a path $S \circ \gamma : I \to M$ on the surface. The length of this path is given by

$$L_S(\gamma) = \int_0^1 \left\| \frac{d(S \circ \gamma)}{dt}(t) \right\| dt.$$

By the chain rule, we obtain

$$\begin{aligned} L_S(\gamma) &= \int_0^1 \|dS_{\gamma(t)} \cdot d\gamma_t\| dt \\ &= \int_0^1 \sqrt{d\gamma_t^T \cdot dS_{\gamma(t)}^T dS_{\gamma(t)} \cdot d\gamma_t}\ dt \end{aligned} \tag{1}$$

Define an inner product on the tangent space to M at $p = S(u)$ by

$$\begin{aligned} g_u(v, w) &= \langle\, dS_u v, dS_u w \rangle \\ &= v^T dS_u^T dS_u w \end{aligned}$$

Using this inner product (1) can be rewritten as

$$L_S(\gamma) = \int_0^1 \sqrt{g_{\gamma(t)}(\dot\gamma(t), \dot\gamma(t))} \quad dt.$$

For $u \in U$, we let G_u be the 2 by 2 matrix $d\alpha_x^T d\alpha_x$ corresponding to the inner product g_u. Observe that, G_u is invertible since it corresponds to a positive definite (and hence nondegenerate) inner product. Denote the entries of G by g_{ij} and the entries of G^{-1} by g^{ij}.

For $1 \le i, j, k \le 2$, define the *Christoffel symbols* Γ_{ij}^k as follows

$$\Gamma_{ij}^k = \frac{1}{2} \sum_{l=1}^{n} g^{kl} \left(\frac{\partial g_{jl}}{\partial x_i} + \frac{\partial g_{li}}{\partial x_j} - \frac{\partial g_{ij}}{\partial x_l} \right). \tag{2}$$

The system of differential equations satisfied by geodesic paths can be given in terms of the Christoffel symbols.

Theorem 4.1. *Let $\gamma : I \to U$ be a smooth map. Denote the coordinates of γ by x_1 and x_2. Then the path $S \circ \gamma$ is a geodesic on M if γ_i satisfy the differential equations*

$$\ddot{x_k} + \sum_{i,j} \Gamma_{ij}^k \dot{x_i} \dot{x_j} = 0. \tag{3}$$

Example 4.2. *Let us calculate the Christoffel symbols and the equations* (3) *for a sphere. Recall that the unit sphere in $\mathbb{R}^3$ can be given paratemrically using the spherical coordinates by*

$$\alpha : (0, 2\pi) \times (0, \pi) \to \mathbb{R}^3$$
$$\alpha : (\theta, \phi) \mapsto (\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi)$$

Then the Jacobian matrix $d\alpha$ and the inner product matrix G are given by

$$d\alpha = \begin{pmatrix} -\sin\theta\sin\phi & \cos\theta\cos\phi \\ \cos\theta\sin\phi & \sin\theta\cos\phi \\ 0 & \sin\phi \end{pmatrix} \qquad G = \begin{pmatrix} \sin^2\phi & 0 \\ 0 & 1 \end{pmatrix}$$

The Christoffel symbols are

$$\Gamma^1 = \begin{pmatrix} 0 & \cot\phi \\ \cot\phi & 0 \end{pmatrix} \qquad \Gamma^2 = \begin{pmatrix} -\sin\phi\cos\phi & 0 \\ 0 & 0 \end{pmatrix}$$

Hence, a path $\gamma : I \to (0, 2\pi) \times (0, \pi)$ represents a geodesic on the sphere if the coordinates (θ, ϕ) of γ satisfy the differential equations given by

$$\ddot{\theta} + 2\dot{\theta}\dot{\phi}\cot\phi = 0$$
$$\ddot{\phi} - \dot{\theta}^2 \sin\phi\cos\phi = 0.$$

Having obtained the differential equation for geodesics, we turn to the numerical solution of the differential equation. Assume the setup described at the beginning of this section. Given two points p and q in U, the task is to find a geodesic on M joining $S(p)$ to $S(q)$. Equivalently, we want to find a path $\gamma : I \to U$ whose coordinates satisfy the differential equations (3). Namely, for $1 \leq k \leq 2$ we must have,

$$\ddot{x_k} + \sum_{i,j} \Gamma_{ij}^k \dot{x_i} \dot{x_j} = 0.$$

We solve the system using the method of finite differences. We choose a large positive integer N and subdivide the interval I into N equal parts in order to replace derivatives by finite differences. For notational simplicity, define $\epsilon = 1/N$. Furthermore, let $x^p = \gamma(p\epsilon)$ and x_j^p be the

jth coordinate of x^p for $1 \leq j \leq 2$ and $1 \leq p \leq N$. For brevity, we let $\Delta x_k^p = x_k^{p+1} - x_k^{p-1}$. Observe that

$$\dot{x_k}(p\epsilon) = \frac{1}{2\epsilon}\Delta x_k^p + O(\epsilon^2), \quad \ddot{x_k}(p\epsilon) = \frac{1}{\epsilon^2}(x_k^{p+1} + x_k^{p-1} - 2x_k^p) + O(\epsilon^2)$$

We approximate the first derivative as $\dot{x_k}(p\epsilon) \approx \Delta x_k^p/2\epsilon$ and the second derivative as $\ddot{x_k}(p\epsilon) \approx (x_k^{p+1} + x_k^{p-1} - 2x_k^p)/\epsilon^2$. Substituting in (3), we obtain the system

$$x_k^p = \frac{x_k^{p+1} + x_k^{p-1}}{2} + \frac{1}{4}\sum_{k,j} \Gamma_{i,j}^k(x^p)\Delta x_i^p \Delta x_j^p \tag{4}$$

subject to the boundary conditions $x^0 = p$ and $x^N = q$. We discuss two iterative methods to solve (4): one using functional iteration and one using Newton-Raphson method. We approximate a path $\gamma : I \to U$ by the $N+1$ tuple $(\gamma(0), \ldots, \gamma(i\epsilon), \ldots, \gamma(1))$ of points in U. In both methods, we assume that we are given an initial path from p to q represented by the $N+1$ tuple

$$s = (s^0, \ldots, s^N),$$

where each s^p is a point in U for $1 \leq p \leq N$. Starting with this initial path, we find a sequence of paths in U (represented as $N+1$ tuples of points in U) that converges to a solution of (4).

4.2. **Functional Iteration.** We phrase the problem of solving (4) as the problem of finding a fixed point of a map $F : U^{N+1} \to \mathbb{R}^{2\times(N+1)}$. We denote elements of $\mathbb{R}^{2\times(N+1)}$ by $(x^0, \ldots, x^N)$ where $x^p = [x_1^p, x_2^p]^T$ for $0 \leq p \leq N$, where $x_1^p, x_2^p \in \mathbb{R}$. Define the map F by

$$F(x^0, \ldots, x^N) = (y^0, \ldots, y^N),$$

where

$$y^0 = x^0, \quad y^N = x^N$$

$$y_k^p = \frac{x^{p+1} + x^{p-1}}{2} + \frac{1}{4}\sum_{1\leq i,j\leq 2} \Gamma_{i,j}^k(x^p) \cdot (x_i^{p+1} - x_i^{p-1})(x_j^{p+1} - x_j^{p-1}), \tag{5}$$

$$\text{for } 1 \leq k \leq 2 \text{ and } 1 \leq p \leq N-1$$

We can find a fixed point of F by iterating F starting with the given point s of $\mathbb{R}^{2\times(N+1)}$. In other words, we define a sequence $\{x(i)\}_{i\in\mathbb{N}}$ of elements of $\mathbb{R}^{2\times(N+1)}$ by

$$x(0) = s$$
$$x(i+1) = F(x(i)) \quad \text{for } i \geq 1.$$

We conjecture that the sequence is well defined and convergent if $\max_{i,p} |s_i^{p+1} - s_i^p|$ is sufficiently small and s is sufficiently close to a fixed point of F. In other words, the sequence converges if successive points in the initial sequence $(s^0, \ldots, s^N)$ are close enough and s approximates a path which is sufficiently close to a geodesic. If the sequence $x(i)$ converges, it is clear that the limiting value is a fixed point of F. Furthermore, observe that $x(i)^0 = s^0$ and $x(i)^N = s^N$ for all $i \in \mathbb{N}$. Hence, the limiting value approximates a path in U from p to q, and its image under S approximates a geodesic joining $\alpha(p)$ and $\alpha(q)$.

Observe that for a plane, all the Christoffel symbols $\Gamma_{i,j}^k$ are zero. Hence, the functional iteration method gives the midpoint method of finding geodesics.

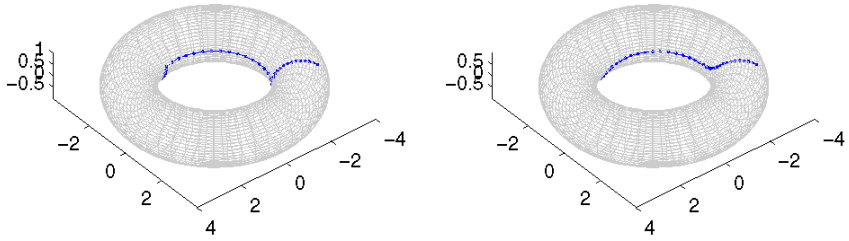
Assuming that applying the given parametrization function S takes constant time, let us calculate the running time of each iteration of the functional iteration method. Using the notation of (5), we see that each iteration involves calculating new y_k^p for $1 \leq k \leq 2$ and $1 \leq p \leq N$. For a given k and p, calculating new y_k^p involves calculating the Christoffel symbols at x^p. This can be done in constant time by calculating the entries $g_{ij}(x^p)$ of G_{x^p}, the entries of $G_{x^p}^{-1}$, the derivatives $\frac{\partial g_{ij}}{\partial x_l}$ and using (2). Derivatives can be computed numerically, or symbolically if the parametrization S admits symbolic treatment. Thus, each y_k^p can be computed in constant time, leading to $\Theta(N)$ time for each iteration.

Figure 6 and Figure 5 show the paths obtained by applying the function iteration method to an initial path on a plane and a torus.

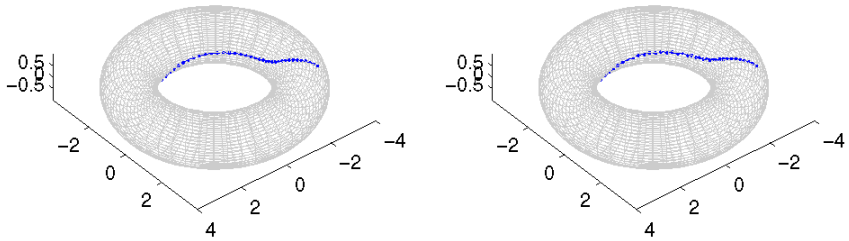
4.3. **Newton-Raphson Method.** We can rephrase the question of finding a fixed point of F as the question of finding a zero of $F(x) - x$. We can then find a zero by Newton-Raphson method, starting with the initial guess s. However, the function $F(x) - x$ has singular Jacobian; hence we have to make a slight adjustment to F to use the Newton-Raphson algorithm.

Recall that in the sequence $x(i) = (x^0(i), \ldots, x^N(i))$ defined above, the extreme values $x^0(i)$ and $x^N(i)$ stay constant at s^0 and s^N respectively. Hence, we can restrict our iteration to the intermediate values. In other words, fix $x^0 = s^0$, $x^N = s^N$ and define $H : U^{N-1} \to \mathbb{R}^{2\times(N-1)}$ by

$$H(x^1, \ldots, x^{N-1}) = (y^1, \ldots, y^{N-1}),$$



(a) Initial path (b) Iteration 25



(c) Iteration 100 (d) Iteration 250

FIGURE 5. Functional Iteration on a Torus

where

$$y_k^p = \frac{x^{p+1} + x^{p-1}}{2} + \frac{1}{4} \sum_{1 \le i,j \le 2} \Gamma_{i,j}^k(x^p) \cdot (x_i^{p+1} - x_i^{p-1})(x_j^{p+1} - x_j^{p-1}),$$

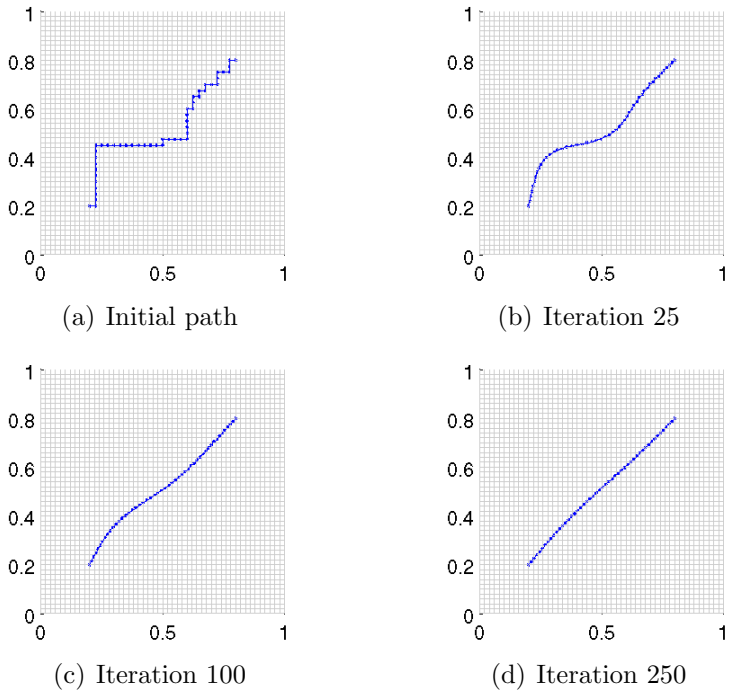
$$\text{for } 1 \le k \le 2 \text{ and } 1 \le p \le N-1$$

Thus, if $(x^1, \ldots, x^{N-1})$ is a zero of $H(x) - x$, then $(x^0, x^1, \ldots, x^{N-1}, x^N)$ is a solution of (4). We iterate using Newton-Raphson algorithm to find a zero of $H(x) - x$, starting with the initial guess $(s^1, \ldots, s^{N-1})$. In other words, we define the sequence $\{y(i)\}$ of points of U by

$$\begin{aligned} y(0) &= (s^1, \ldots, s^{N-1}) \\ y(i+1) &= y(i) - (dH_{y(i)} - I)^{-1} \cdot (H(y(i)) - y(i)), \quad \text{for } i \ge 1. \end{aligned} \tag{6}$$

We conjecture that if $\max_{i,p} |s_i^{p+1} - s_i^p|$ is sufficiently small and s is close to a solution of (4) then the sequence defined by (6) is well defined and converges to a solution of (4).

As we did for the function iteration method, let us compute the number of steps required for each iteration of the Newton-Raphson method. Observe that each iteration $y(i) \mapsto y(i+1)$ involves the computation of $H(y(i))$, which can be done in $\Theta(n)$ time as described



(a) Initial path (b) Iteration 25

(c) Iteration 100 (d) Iteration 250

FIGURE 6. Functional Iteration on a Plane

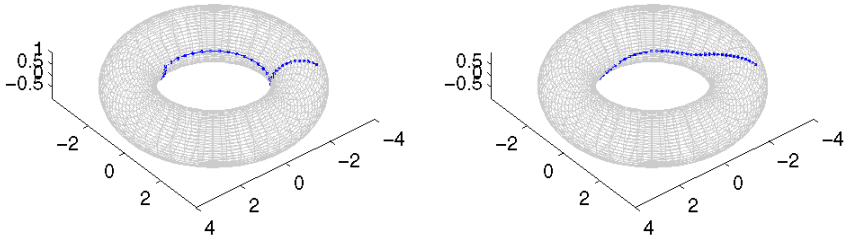
before. However, we also need to compute the Jacobian $dH_{y(i)}$ and the inverse of $(dH_{y(i)} - I$. Since $dH_{y(i)}$ is a $(2N-4)\times(2N-4)$ matrix, this would usually take $\Theta(N^3)$ time. However, we only need to calculate $(dH_{y(i)} - I)^{-1} \cdot (H(y(i)) - y(i))$, which is equivalent to solving for Z the system

$$(dH_{y(i)} - I)Z = H(y(i)) - y(i). \tag{7}$$

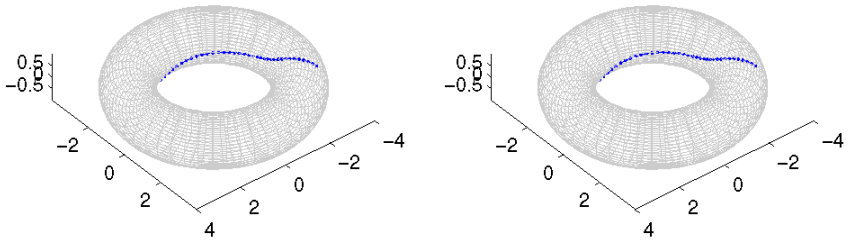
Furthermore, observe that $y(i+1)^p$ depends only on $y(i)^{p\pm1}$. Hence $dH_{y(i)} - I$ has $\Theta(N)$ nonzero entries, which are located on or near the diagonal. In other words, $dH_{y(i)} - I$ is a *banded* matrix, for which (7) can be solved in $\Theta(N)$ steps. Thus, each iteration of the Newton-Raphson method can be done in $\Theta(N)$ steps.

Figure 7 and Figure 8 show the results of applying the Newton-Raphson method to an initial path on a torus and a plane respectively.

Finally, we remark that although calculating the Christoffel symbols $\Gamma^k_{i,j}$ at a point is a constant time operation, numerically calculating them can be computationally quite intensive in practice. Both the methods outlined in this section run much faster if we can compute $\Gamma^k_{i,j}$ symbolically once and for all and evaluate these symbolic expressions at particular points. For example, in our implementation, one iteration

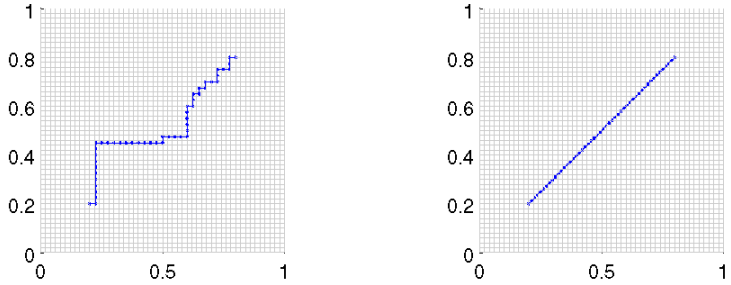


(a) Initial path (b) Iteration 1



(c) Iteration 2 (d) Iteration 3

FIGURE 7. Newton-Raphson on a Torus



(a) Initial path (b) Iteration 1

FIGURE 8. Newton-Raphson on a Plane

of Newton-Raphson for a sphere using symbolic $\Gamma_{i,j}^k$ took 1 second, whereas one iteration using numerically computed $\Gamma_{i,j}^k$ took 69 seconds (for $N = 26$).

5. Generating An Initial Path Using Graph Theory

So far we have considered various ways of iteratively refining a path to a geodesic. In order to employ these methods, however, one requires an approximate shortest path to begin with. This section explores a potential method for generating such path using graph theory.

Just as a path can be approximated by a sequence of points, a surface can also be approximated as a mesh of regularly-spaced points. For instance, computer-generated visualizations of surfaces necessarily involve a discrete approximation. If a mesh can successfully approximate a surface, then we can treat the problem of finding a geodesic as the problem of finding the shortest path between two vertices on a graph. As we will show, the shortest path on the mesh approximation of the surface is not too far from the minimal geodesic.

5.1. Induced-Mesh Approximation.

First we review the definition of a weighted graph and consider how to construct a weighted graph that approximates a given surface S.

Definition 5.1. *A* weighted graph G *is an ordered pair* (V, E) *corresponding to the set of vertices and the set of edges, along with a weight function* $w : E \to \mathbb{R}$ *specifying the weight of each edge.*

A natural way to create a graph on a given surface S is to pick regularly-spaced points in I^2 as vertices, and to join each point to its four immediate neighbors, where the cost of each edge $e = \{v_1, v_2\}$ is the Euclidean distance between $S(v_1)$ and $S(v_2)$.

Definition 5.2. *For* $n \in \mathbb{Z}$ *greater than 1, an* induced mesh *on a surface* S *is the weighted graph* $G = (V, E)$ *where*

$$V = \left\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\right\}^2 \subset I^2,$$

$$E = \{\{v_1, v_2\} \mid v_1 \text{ and } v_2 \text{ are adjacent.}\},$$

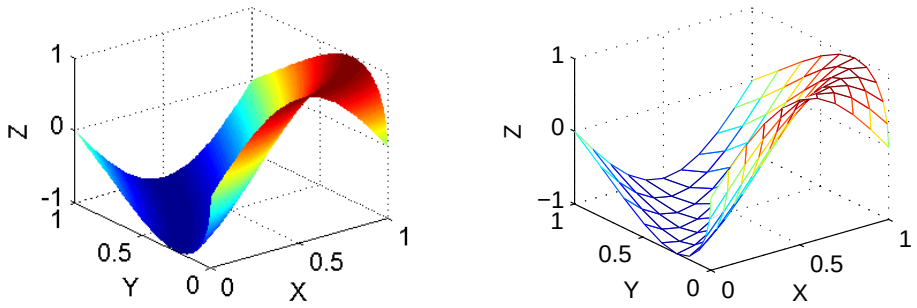
with

$$w : \{v_1, v_2\} \mapsto \|S(v_1) - S(v_2)\|, \quad \text{if } \{v_1, v_2\} \in E.$$

Here two vertices are adjacent if they are neighbor lattice points in I^2*. We denote the induced mesh by* $M_n(S)$*, and call* n *the* size *of the induced mesh.*

One way to characterize the induced mesh is to realize that E is the approximation of the metric induced on I^2 by S. We are estimating the minimal geodesic distance between v_1 and v_2 by the Euclidean distance, which is reasonable as long as S is locally planar and v_1, v_2 are close. As

an example, Figure 9 compares a particular surface S with an induced mesh on S.



(a) The surface S. (b) The induced mesh $M_{10}(S)$.

FIGURE 9. Comparison of the surface $S(u, v) = (u, v^2, \sin(\pi(u - v)))$ with an induced mesh on it.

Definition 5.3. *Let $G = (V, E)$ be a weighted graph with w as its weight function. Then a* path *between $s, d \in V$ is the sequence of vertices $(v_1, v_2, \ldots, v_k)$ where $v_1 = s$ and $v_k = d$ and $\{v_i, v_{i+1}\} \in E$ for all $i = 1, \ldots, k - 1$. The sequence is a* shortest path *in case it minimizes the sum*

$$\sum_{i=1}^{k-1} w(v_i, v_{i+1}).$$

Hence, the shortest path on an induced mesh is equivalent to the physically shortest passage on the mesh along the gridlines.

There are several known algorithms that find the shortest path on a weighted graph. *Dijkstra's Algorithm* is a very efficient such algorithm. Its pseudocode is given in the appendix.

Theorem 5.4. [CL, Theorem 24.6] *Let $G = (V, E)$ be a weighted graph with $w : E \to \mathbb{R}$ as its weight function. Given two points in V such that a path exists between the two, Dijkstra's Algorithm finds the shortest path between them in asymptotic runtime of $O(|V|^2 + |E|)$.*

In fact, Dijkstra's Algorithm can be executed in $O((|E|+|V|) \log |V|)$ when the graph is sparse, using a binary heap and an adjacency matrix. In case of $M_n(S)$, we have $|E| = 2n(n + 1)$ and $|V| = (n + 1)^2$.

Remark 5.5. *Given a surface S, Dijkstra's Algorithm can find the shortest path on $M_n(S)$ in asymptotic runtime of $O(n^2 \log n)$.*

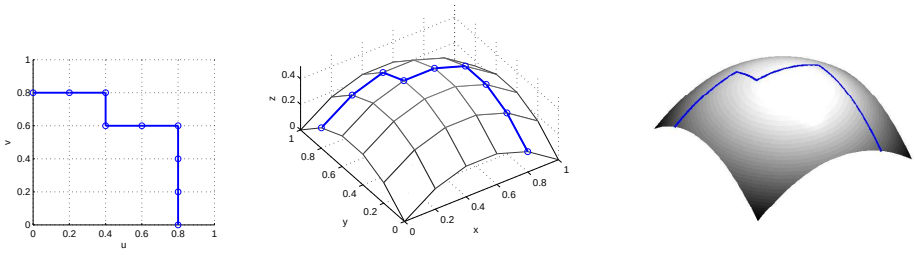
There also exists a generalization of Dijkstra's Algorithm, called *A-Star*, in case a lower bound on the distance to d can be given at each vertex v. In the case of an induced mesh on S, the Euclidean distance $\|S(d) - S(v)\|$ can serve as the lower bound. A-Star has the same asymptotic runtime as Dijkstra, but often runs faster.

A-star returns a sequence of vertices $(v_1, \ldots, v_k)$ that resides in I^2. To translate the sequence into a path on I^2, a simple linear interpolation will suffice, as illustrated in the following definition.

Definition 5.6. *Let $(v_1, \ldots, v_k)$ be a sequence of points in I^2. Then the* linear interpolation *of the sequence is the piecewise function $\gamma : I \to I^2$ where*

$$\gamma : t \mapsto \begin{cases} v_1(1-tk) + v_2(tk) & \text{if } 0 \le t \le 1/k \\ v_2(2-tk) + v_3(tk-1) & \text{if } 1/k \le t \le 2/k \\ \vdots & \vdots \\ v_i(i-tk) + v_{i+1}(tk-i+1) & \text{if } (i-1)/k \le t \le i/k \\ \vdots & \vdots \\ v_{k-1}(k-1-tk) + v_k(tk-k+1) & \text{if } (k-1)/k \le t \le 1 \end{cases}$$

In other words, we linearly interpolate between each pair of consecutive vertices. Note that the composition $\beta = S \circ \gamma$ constitutes a path on the surface S. Figure 10 provides an example.



(a) A path on a 10-by-10 lattice on I^2.

(b) The same path on the induced mesh on S.

(c) Composition of S with the linear interpolation of the path.

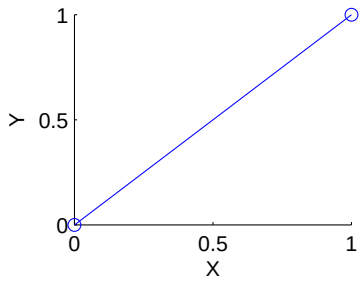
FIGURE 10. Translation of a sequence of points in I^2 into a path on S.

Therefore, the following recipe yields an approximate shortest path on a given surface S.

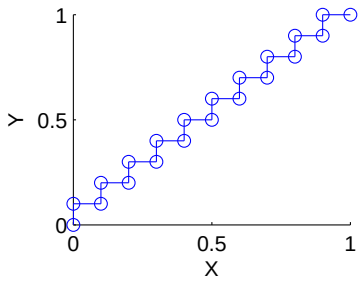
(1) Generate an induced mesh $M_n(S)$.

(2) Execute A-star to generate a sequence of points between two vertices.
(3) Construct a linear interpolation γ of the sequence in I^2.
(4) Return $\beta = S \circ \gamma$.

5.2. **Convergence.** The path generated by the above recipe begs the question of accuracy. It differs from the minimal geodesic since the path must always travel along the gridlines of the mesh, and each piecewise leg of the path is not itself a minimal geodesic. Figure 11 illustrates this point: whereas the geodesic joining $(0,0)$ and $(1,1)$ in I^2 has arc length $\sqrt{2}$, a linear interpolation of the sequence returned by A-star will have arc length of 2. Nonetheless, we may still prove that the arc length of this path is not too far from that of the minimal geodesic.



(a) The geodesic joining $(0,0)$ and $(1,1)$ in I^2.



(b) The linear interpolation of one shortest path on the induced mesh of size 10.

FIGURE 11. Comparison of the geodesic joining $(0,0)$ and $(1,1)$ in $\mathbb{R}^2$ and the approximation via the method in Section 5.1.

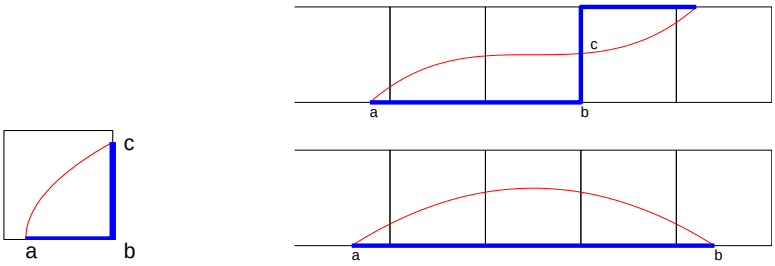
Throughout this section, we fix two points $p_1, p_2 \in I^2$, let S be a surface, and let $\tilde{\gamma} : I \to I^2$ be the path between p_1 and p_2 that minimizes $L_S(\cdot)$, parametrized by the arc length of its composition with S. Take an induced mesh $M_n(S)$.

Ideally we want to compare $S \circ \tilde{\gamma}$ with $\beta = S \circ \gamma$, the approximate path constructed by our recipe in Section 5.1. However, since we do not know a priori the path returned by A-star, we instead generate a different path β as described below:

Consider all squares on the grid of $M_n(S)$ that $S \circ \tilde{\gamma}$ passes through. The path enters through one of the four edges, and leaves through another. There are two cases:

- These two edges are the same or adjacent.

- These two edges are parallel. In this case, we keep attaching squares through which $S \circ \tilde{\gamma}$ enters and leaves through parallel edges, until $S \circ \tilde{\gamma}$ leaves through an edge adjacent to the entrance edge. See Figure 12(b).



(a) Adjacent entrance and exit edges.

(b) Non-adjacent entrance and exit edges. In this case, we add squares on either side until the path exits a square through a non-parallel edge.

FIGURE 12. Construction of an approximate path along the mesh. The red curves indicate $\tilde{\gamma}$, and the blue lines indicate our construction.

In the first case, join the two entrace and exit points along the grid, choosing the direction (clockwise or counterclockwise) to minimize the length in I^2. If the entrance and exit points lie on the same edge, we simply connect them; otherwise, the path will contain the vertex shared by the two edges, as seen in Figure 12(a).

In the second case, the entrance and exit points of $\tilde{\gamma}$ for the k consecutive squares may or may not lie on the same side. See Figure 12(b). If they do, simply connect them; if not, make a transition along one of the parallel edges.

Now we join all these lines we have generated for all squares $\tilde{\gamma}$ passes through. These lines form a piecewise-straight, continuous path on I^2. Furthermore, it is a valid path on $M_n(S)$, since it entirely consists of gridlines; each edge can never be partially covered, because if it were, then we must be backtracking, and we can simply remove the repetition. Now we have successfully obtained a path in I^2 along the induced mesh, and denote it by γ. Let β be a parametrization of $S \circ \gamma$ by its arc length.

We compare $S \circ \tilde{\gamma}$ with β in each of the square(s) we considered above. For now, assume the first case (Figure 12(a).) The entrance and exit points are a and c lying in I^2, with b being the corner vertex. We examine the following arc lengths:

- L_1, the arc length of the minimal geodesic between $S(a)$ and $S(c)$ on the surface.
- L_2, the Euclidean distance between $S(a)$ and $S(c)$.
- L_3, the sum of the Euclidean distances between $S(a)$ and $S(b)$, and between $S(b)$ and $S(c)$.
- L_4, the arc length of the composition of S with the linear interpolation of $\{a, b, c\}$.

L_1 corresponds to the arc length of the portion of $S \circ \tilde{\gamma}$ contained with in the square, whereas L_4 corresponds to the arc length of the portion of β containined within the square. We remind the reader that these lengths are the arc lengths of the part contained *within a single square*, not the length of the entire geodesic.

Lemma 5.7. *L_1 and L_4 are at most*

$$\frac{2}{n}\sqrt{G_x^2 + G_y^2 + G_z^2},$$

where G_x is the maximum of $\|\nabla S_x\|$ in I^2, and G_y, G_z are analogously defined.

Proof. Take a simple interpolation between the endpoints of $S \circ \tilde{\gamma}$ inside the square of side length $1/n$ by connecting them linearly inside I^2 and sending this line in I^2 through S. Then the arc length is bounded above by

$$\frac{\sqrt{2}}{n}\sqrt{G_x^2 + G_y^2 + G_z^2}.$$

To see this, note that the rate of increase in arc length relative to a unit step in I^2 (in an arbitrary direction) is at most $\sqrt{G_x^2 + G_y^2 + G_z^2}$. Since we move a total distance of at most $\frac{\sqrt{2}}{n}$ inside I^2 (because we are inside a small square, and the most we can move in a single direction is along the main diagonal), the total arc length must be bounded above as such. Because L_1 is shorter than this interpolation, the lemma for L_1 follows. As for L_4, since it takes a right-angled turn in I^2, we move a total distance of at most $\frac{2}{n}$ inside I^2, which indicates that the arc length is at most $\frac{2}{n}\sqrt{G_x^2 + G_y^2 + G_z^2}$ as desired. □

Lemma 5.8. *Let $\delta : I \to S(I^2)$ be a path on the surface S parametrized by arc length. Fix $0 \leq t_1 < t_2 \leq 1$. Then the arc length of δ between*

$t = t_1$ and $t = t_2$ differs from the straightline distance $\|\delta(t_2) - \delta(t_1)\|$ by at most

$$(t_2 - t_1)^2\sqrt{H_x^2 + H_y^2 + H_z^2},$$

where H_x is the maximum of $\|\nabla^2 S_x\|$ in I^2, and H_y, H_z are analogously defined.

Proof. The arc length of δ between $t = t_1$ and $t = t_2$ is,

$$L = \int_{t_1}^{t_2} \sqrt{\left(\frac{d\delta_x}{dt}\right)^2 + \left(\frac{d\delta_y}{dt}\right)^2 + \left(\frac{d\delta_z}{dt}\right)^2} dt. \tag{8}$$

Let $\Delta t, \Delta x, \Delta y, \Delta z$ be the differences $t_2 - t_1, x(t_2) - x(t_1), y(t_2) - y(t_1), z(t_2) - z(t_1)$, respectively. Then, by Mean Value Theorem, there exists some $t' \in [t_1, t_2]$ such that

$$\frac{d\delta_x}{dt}(t') = \frac{\Delta x}{\Delta t}.$$

This yields, for $r \in [t_1, t_2]$,

$$\begin{aligned}
\left|\frac{d\delta_x}{dt}(r)\right| &= \left|\frac{d\delta_x}{dt}(t') + \int_{t'}^{r} \frac{d^2\delta_x}{dt^2} dt\right| \\
&\leq \left|\frac{d\delta_x}{dt}(t')\right| + \int_{t'}^{r} \left|\frac{d^2\delta_x}{dt^2}\right| dt \\
&\leq \left|\frac{d\delta_x}{dt}(t')\right| + \int_{t'}^{r} H_x dt \\
&\leq \left|\frac{d\delta_x}{dt}(t')\right| + \Delta t H_x.
\end{aligned}$$

Therefore, for the same values of r,

$$\left(\frac{d\delta_x}{dt}(r)\right)^2 \leq \left(\left|\frac{\Delta x}{\Delta t}\right| + \Delta t H_x.\right)^2.$$

Similar inequalities hold for $\left(\frac{d\delta_y}{dt}(r)\right)^2$ and $\left(\frac{d\delta_z}{dt}(r)\right)^2$. Plugging these back into Equation (8), we obtain

$$L \leq \int_{t_1}^{t_2} \left[\sqrt{\left(\left|\frac{\Delta x}{\Delta t}\right| + \Delta t H_x\right)^2 + \ldots}\right] dt.$$

Since the integrand does not depend on t, we get

$$L \leq \sqrt{(|\Delta x| + \Delta t^2 H_x)^2 + (|\Delta y| + \Delta t^2 H_y)^2 + (|\Delta z| + \Delta t^2 H_z)^2}.$$

By the triangle inequality, we obtain

$$L \leq \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} + \Delta t^2 \sqrt{H_x^2 + H_y^2 + H_z^2}$$
$$= \|\delta(t_2) - \delta(t_1)\| + (t_2 - t_1)^2 \sqrt{H_x^2 + H_y^2 + H_z^2}.$$

Since the arc length of the minimal geodesic is bounded below by the Euclidean distance between the endpoints, namely $\|\delta(t_2) - \delta(t_1)\|$, the lemma follows. □

Theorem 5.9.

$$L_2 = L_1 + O\left(\frac{1}{n^2}\right),$$

and

$$L_4 = L_3 + O\left(\frac{1}{n^2}\right).$$

Proof. By Lemma 5.7, the length of L_1 is $O(1/n)$. Then, in application of Lemma 5.8 to L_1, the difference $t_2 - t_1$ must also be $O(1/n)$ because $S \circ \tilde{\gamma}$ is parametrized by arc length. Then it follows that the difference between L_2 and L_1 is

$$O(1/n)^2 \sqrt{H_x^2 + H_y^2 + H_z^2} = O(1/n)^2,$$

as desired. The statement for $L_4 - L_3$ follows similarly. □

Lemma 5.10.

$$L_3 \leq \sqrt{2} L_2.$$

Proof. Note that L_2 takes straight lines whereas L_3 travels along the grid. In other words, L_2 is the hypotenuses of a triangle of which the legs form L_3. Because the ratio of the sum of the legs to the hypotenuse is at most $\sqrt{2}$, the lemma follows. □

So far we have assumed the case in which $\tilde{\gamma}$ enters and leaves a square on $M_n(S)$ through adjacent edges. In the other case, shown in Figure 12(b), analogous results to Lemma 5.10 and Theorem 5.9 can also be obtained. We omit their proof for brevity, but the reader should be able to convince himself of the omitted fact.

Theorem 5.11. *As n approaches ∞, the arc length of the path generated by the recipe in Section 5.1 converges to at most $\sqrt{2} L_S(\tilde{\gamma})$.*

Proof. Combining Theorem 5.9 with Lemma 5.10 yields the relation

$$L_4 - \sqrt{2} L_1 = O\left(\frac{1}{n^2}\right).$$

Summing over all squares, we obtain that $L_S(\gamma)$ is less than the sum of $\sqrt{2} L_S(\tilde{\gamma})$ with an $O(1/n)$ term, since the number of squares through

which the path passes is $O(n)$. This quantity converges to zero as n approaches ∞, as desired. Now, since the recipe returns an optimal path with smaller arc length than β, the same inequality must still hold. □

Theorem 5.11 tells us that the recipe in Section 5.1 is a reasonable initialization for the aforementioned iterative methods, such as midpoint method, gradient descent, and Newton-Raphson.

From Figure 11, we see that the bound in Theorem 5.11 is tight in the sense that replacing $\sqrt{2}$ with a smaller constant will invalidate the theorem. The constant $\sqrt{2}$ seems to arise from the structure of the mesh, however, rather than the methods in the proof of the theorem. We conjecture that augmenting the mesh structure by adding diagonals in each square, et cetera, should reduce this constant.

6. Comparison Study

In this section, we apply the methods discussed in the previous sections to obtain geodesic path on different surfaces. We compare the results after different number of iterations and summarize our findings.

6.1. **Methodology.** We constructed four surfaces using `Matlab` and selected two points on each. The parametrizations of the surfaces are given in Table 1 of the appendix. For each surface, an initial path was generated by A-star on an induced mesh of size 40; another initial path was generated by connecting the two points with a sinusoidal wave of two full periods in I^2. For each surface-path pair, three methods were used to iteratively refine the path into a geodesic: 1) midpoint method, 2) gradient descent, and 3) Newton-Raphson. The Newton-Raphson method was used only for the initial path constructed from A-star, as the sinusoidal path is too far from geodesic and the method does not converge.

Figures 14 through 16 in the Appendix display the results on the four surfaces. Part (a) is the approximate path generated by A-star; Part (b)-(d) are the resulting paths after some number of iterations of each of the three methods. The three methods are abbreviated as MP (**M**id**p**oint search), GD (**G**radient **D**escent) and NR (**N**ewton-**R**aphson), and the number of iterations used is given in parentheses. Part (e)-(g) show the decrease in arc length over iterations for each method. Part (h)-(l) are analogous results from initializing with a sinusoidal path.

6.2. **Results.** In general, the Newton-Raphson method using difference equations converged in the fewest iterations, stabilizing in a couple iterations at most. We emphasize that to generate Figures 13 through 16, the Newton-Raphson method ran for 5 iterations, whereas midpoint search and gradient descent required 200 iterations to get reasonably close to the actual geodesic. However, each iteration of these methods consumed very little time, whereas the Newton-Raphson method suffers from the bottleneck of calculating Christoffel symbols. In case the parametrization of the surface is known and tractable, one can precompute the Christoffel symbols symbolically, rather than numerically, to conserve time. In that case, Newton-Raphson has a comparable runtime per iteration. See the discussion at the end of Section 4.

The Newton-Raphson method also required the initial path to be reliable and each consecutive pair of points to be very close. For instance, the Newton-Raphson method would diverge when initialized with the sinusoidal paths given in part (h) of Figures 13 through 16. On the other hand, the other two numerical methods did not require the initial path to be very reliable, and functioned satisfactorily when initialized with suboptimal paths.

Between the midpoint search and gradient descent, midpoint search performs slightly better, especially when the surface is not very planar or the initial path is wavy. We remark that one can use these two methods in conjunction—by picking the Euclidean midpoint and finding the closest point that lies on the surface, and then improving this point by running gradient descent.

7. Primary Authors

Section 1 through Section 3 were primarily authored by Katherine Redfield; Section 4 was primarily authored by Anand Deopurkar. Sections 5 and 6, along with the appendix, were primarily authored by Jongmin Baek.

8. Appendix

Program 1 Midpoint Search

```
// Iteratively refines a given path
// S: a surface I^2--->R^3
// pts: a sequence of points in I^2
// n: the number of iterations
function Midpoint_Search(S, pts, n)
  for n iterations
    for i=1 to length of pts
     // Map the points onto S
      pts2(i) = S(pts(i))
    for i=1 to length of pts
     // Take midpoints
      midpts(i) = (pts2(i) + pts2(i+1)) * 0.5
    for i=1 to length of midpts
     // find the closest point on the surface to midpts(i)
     pts(i) = argmin_p ||S(p)-S(midpts(i))||
     // argmin can be done in matlab using fminsearch.
```

Program 2 Gradient Descent

```
// Iteratively refines a given path
// S: a surface I^2--->R^3
// pts: a sequence of points in I^2
// n: the number of iterations
function Gradient_Descent(S, pts, n)
  for n iterations
    for i=1 to length of pts - 1
     // Take midpoints
      midpts(i) = (pts(i) + pts(i+1)) * 0.5
    for i=1 to lengths of midpts
     // Perform gradient descent, starting from midpts(i)
     f(p) := ||S(pts(i)-p)||+||S(pts(i+2)-p||
     grad = argmax_g (df(p)/dx,df(p)/dy).*g
                evaluated at p=midpts(i)
     beta = argmin_b f(p-b*grad)
     // argmin can be done using fminbnd in matlab
     pts(i) = midpts(i) - beta * grad
```

Program 3 Function Iteration and Newton Raphson

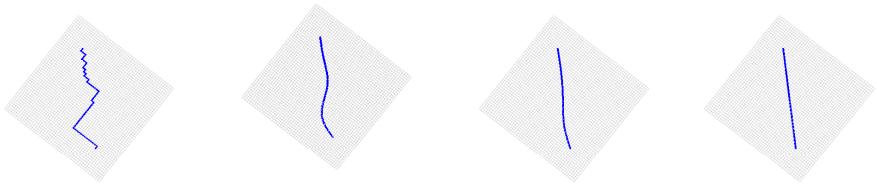
```
\\Computes one iteration of the functional iteration
\\method
\\     S = Map from I^2 --> I^3
\\     pts = Sequence of N-2 points in I^2
\\     init = Initial point (in I^2)
\\     final = Final point (in I^2)
function Iter(S, init, final, pts)
   x(0)   := init
   x(N-1) := final
   x(i)   := pts(i) for 0 < i < N - 1
   for p = 1 to N-2
     for k = 1,2
       y(p,k) = (x(p+1,k) + x(p-1,k))/2 +
                1/4 * Sum [Gamma(k,i,j)(x(p)) *
                           (x(p+1,i) - x(p-1,i)) *
                           (x(p+1,j) - x(p-1,j))]
        //Sum taken over all i,j in {1,2}
        //Calculating Gamma is straightforward. All the
        //derivatives encountered can be computed
        //numerically.
   return y

\\Computes one iteration of the Newton Raphson method
\\     S, pts, init, final are as in Iter
function NRIter(S,init, final, pts)
  \\Look at pts as a 2 x (N-2) array
   y  := Iter(S,init,final,pts)
   dH := Jacobian of H(X) -> Iter(S,init,final,X) at X=pts
         \\This can be computed numerically.
         \\Looking at pts as 2N-4 tuple, dH would be a
         \\2N-4 by 2N-4 matrix.
   Z  := Inverse(dH - I) * (y - pts)
         \\Can be obtained by solving (dH - I)Z = y - pts
         \\Use a method that exploits that dH - I is
         \\bounded for optimal performance.
         \\Look at Z as a 2 by N - 2 array.
   return (pts - Z)
```

Program 4 Dijkstra's Algorithm

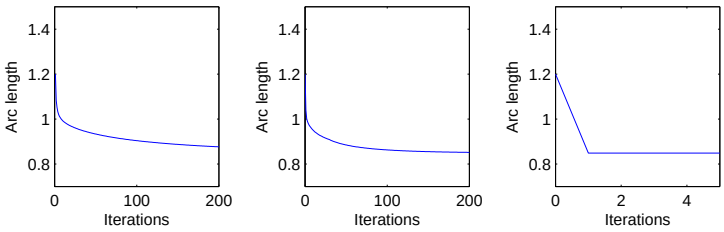
```
// Finds the shortest path between s and d in the graph G
function Dijkstra(G=(V,E,w),s,d)
  for each v in V
    dist[v] = infinity
    prev[v] = null
  dist[s] = 0
  while V not empty and u not equal to d
    // remove the vertex u with the least value of dist[u]
    u = extract_min(V)
    for each neighbor v of u
      alt = dist[u] + w(u,v)
      if alt < dist[v]
        dist[v] = alt
        prev[v] = u
```

Name	Parametrization
`plane`	$S(u,v) = (u,v,0)$.
`sphere`	$S(u,v) = (\cos(\phi)\cos(\theta), \cos(\phi)\sin(\theta), \sin(\phi))$ where $\phi = \pi(x-1/2)$ and $\theta = \pi(y-1/2)$.
`torus`	$S(u,v) = (3\cos(\theta) + \cos(\phi)\cos(\theta), 3\sin(\theta) + \cos(\phi)\sin(\theta), \sin(\phi))$ where $\theta = 2u\pi$ and $\phi = 2v\pi$.
`saddle`	$S(u,v) = (x, y, x^2 - y^2)$ where $x = 2u-1, y = 2v-1$.

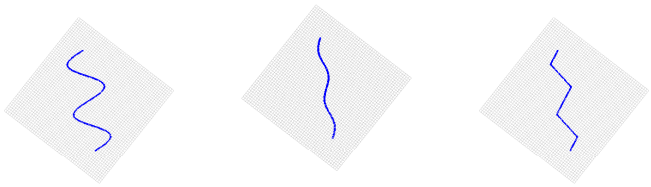
TABLE 1. Parametrizations of the surfaces used in Section 6.



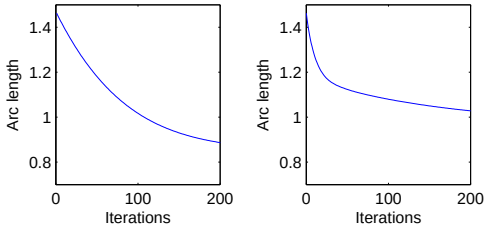
(a) Initial Path (b) MP(200) (c) GD(200) (d) NR(5)



(e) MP (f) GD (g) NR

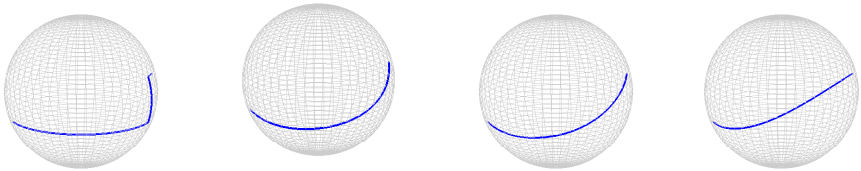


(h) Initial Path (i) MP(200) (j) GD(200)

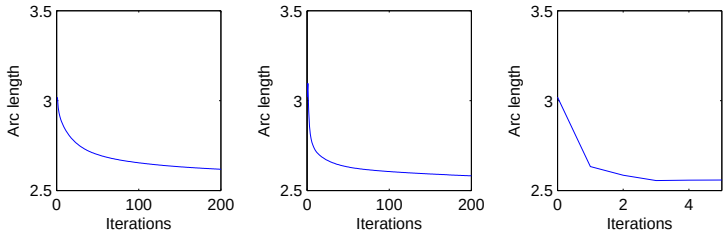


(k) MP (l) GD

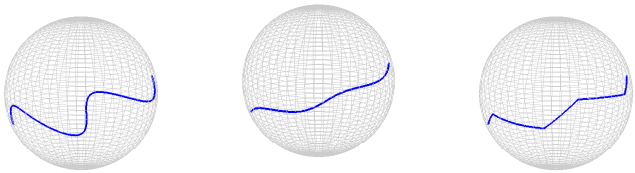
FIGURE 13. Result on `plane`.



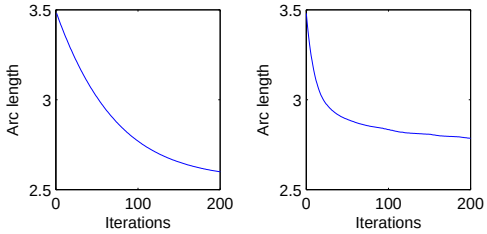
(a) Initial Path (b) MP(200) (c) GD(200) (d) NR(5)



(e) MP (f) GD (g) NR

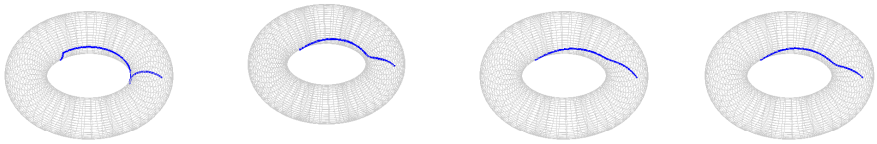


(h) Initial Path (i) MP(200) (j) GD(200)

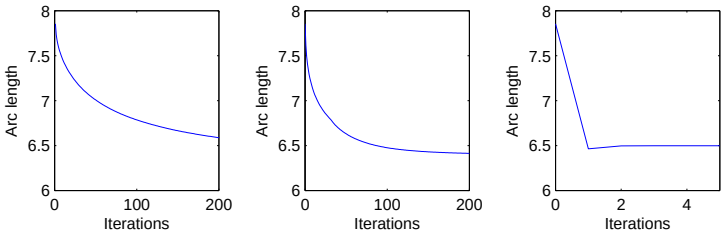


(k) MP (l) GD

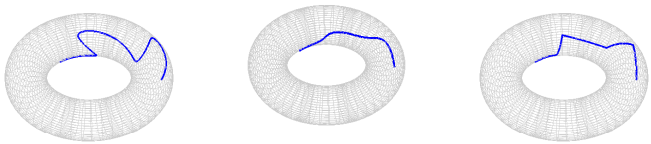
FIGURE 14. Result on `sphere`.



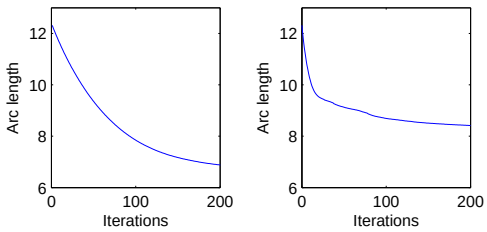
(a) Initial Path (b) MP(200) (c) GD(200) (d) NR(5)



(e) MP (f) GD (g) NR

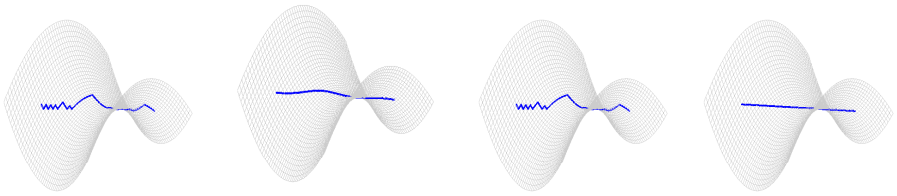


(h) Initial Path (i) MP(200) (j) GD(200)

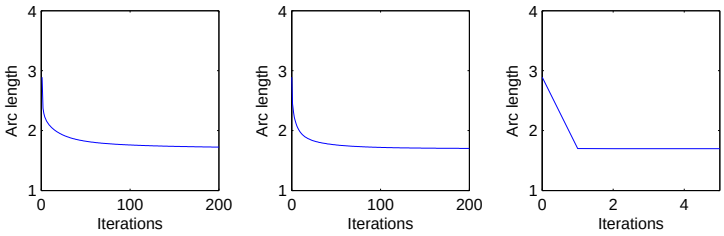


(k) MP (l) GD

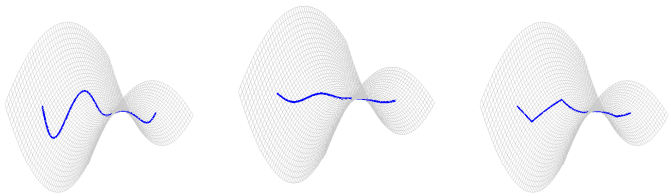
FIGURE 15. Result on `torus`.



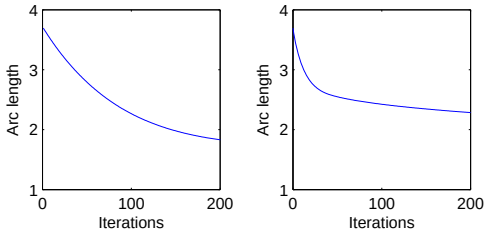
(a) Initial Path (b) MP(200) (c) GD(200) (d) NR(5)



(e) MP (f) GD (g) NR



(h) Initial Path (i) MP(200) (j) GD(200)



(k) MP (l) GD

FIGURE 16. Result on `saddle`.

References

[CL] T. E. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: "Introduction to Algorithms." MIT Press, Cambridge 1990.