

Loki: Software for Computing Cut Loci

Robert Sinclair and Minoru Tanaka

CONTENTS

- 1. Introduction
 - 2. The Algorithm
 - 3. Use of the Software
 - 4. Kokkendorff's Conjecture
 - 5. Conclusion and Further Work
- Acknowledgements
References

The first quantitatively correct pictorial atlas of the cut locus of a nontrivially deformed standard torus in \mathbb{R}^3 given a non-symmetrically placed starting point is presented along with a description of the software tool Loki used to generate it. Loki can compute the cut locus from a point on a genus-1 two-dimensional Riemannian manifold defined either by a parametrization or its metric, these to be given in closed form. The algorithm computes a piecewise polynomial approximation to the exponential map and inverts this numerically, thus correctly taking into account the global nature of the problem. As an example of its use in motivating and guiding traditional mathematical research, we provide a preliminary conjecture based upon the output of this software and both a counterexample and a proof motivated by the conjecture.

1. INTRODUCTION

This report has been written with several different audiences in mind. Mathematicians familiar with global differential geometry can skip the Sections pertaining to compilation or algorithmic details, and may in fact wish to go directly to Section 4.1. On the other hand, those interested in using the software may wish to look at Section 3. first. We will refer to the new software package by its name (Loki) throughout.

The necessary mathematical background is given in [do Carmo, 76], Section 11.4 of [Berger and Gostiaux, 88], Section 2.1 of [Klingenberg, 82] and Section 2, Chapter 13 of [do Carmo, 92]. See also the classic works [Myers, 35] and [Myers, 36].

The cut locus from a point p on a surface V is the closure of the set of points that can be connected to p by at least two distinct shortest paths in V .

Loki is a software tool designed to compute a numerical approximation to the cut locus of a genus-1 two-dimensional Riemannian manifold defined either by a parametrization given in closed form, or by the functions E , F and G of the standard notation for the first fundamental form of the surface (see Equation (1–2) or

2000 AMS Subject Classification: Primary 53-04; Secondary 53C20

Keywords: Cut locus, computational global differential geometry

Section 10.4.1.1 of [Berger and Gostiaux, 88]) given in closed form.

It has been shown ([Gluck and Singer, 78] and [Gluck and Singer, 79]) that one can find a metric and a starting point such that the cut locus is non-triangulable for any smooth manifold of dimension greater than or equal to two (for example, the cut locus falls into infinitely many components if one removes one particular point from it). Fortunately, it is now known ([Itoh and Tanaka, 98]) that the Hausdorff dimension of the cut locus is in a direct relation to the smoothness of the metric, meaning that “nice” surfaces can be expected to have “simple” cut loci. Clearly this result is of vital importance in justifying the development of Loki.

To be more specific, it is known [Buchner, 78] (see also [Ozols, 74]) that generic cut loci in low dimensional manifolds are triangulable and structurally stable. In dimension 2, a generic cut locus is simple to describe: each point q has a neighborhood in the cut locus which is either (i) a straight line through q , (ii) a straight line starting at q , or (iii) three straight lines meeting at q to form a “Y”. See also [Bishop, 77], where it is shown that ordinary cut points of a point m are dense in the cut locus of m .

It is possible to find published papers giving specific examples, such as [Tsuji, 97]. [Bleecker, 81] explores the case of compact surfaces without conjugate points in detail. Bleecker derived (in his Ph.D. thesis in 1973) precise upper and lower bounds for the number of vertices in the cut locus. [Degen, 97] consists of a diagram showing the cut-locus of a solid ellipsoid (which is not the same as the cut-locus from a point on a surface, but related) and some comments (a theorem) as to how the diagram was constructed.

It is, however, a fact that it is often prohibitively difficult to compute the cut locus of a given surface using purely analytical methods — for example, if the surface lacks rotational symmetry. Furthermore, as pointed out in a recent review of the status of mathematical research on the cut locus [Berger, 00], inverse results (recognizing a surface on the basis of its cut locus from a point) are almost completely unknown.

In 1995, J. Rebel, under the supervision of Prof. S. Markvorsen and Dr. J. Gravesen of the Technical University of Denmark wrote software for the interactive approximation and visualization of cut loci on surfaces of revolution. At that time, computation of the cut locus by software without human intervention seemed unlikely. The user would generally have to sit and work with the program for hours to obtain a plot of a cut locus.

The conclusion of the thesis [Rebel, 95] begins

The computer program offers the necessary tools to visualize the Cut Locus on a surface of revolution, but it is not able to do that on its own. This was in no way the intention throughout the construction and seems also impossible to implement.

The thesis contains an appendix with plots of the cut loci for the standard torus

$$(u, v) \mapsto ((2 + \cos u) \cos v, (2 + \cos u) \sin v, \sin u) \quad (1-1)$$

for a sequence of starting points beginning on the outside equator $(3, 0, 0)$, and moving up perpendicularly to the equator to an angle of $\pi/2$ (corresponding to the point $(2, 1, 0)$). The uncertainty of these plots is large enough to be clearly visible.

Prof. Markvorsen and Dr. Gravesen encouraged the first author to develop a more accurate, automatic software tool which would facilitate numerical experimentation. Loki indeed improves upon this earlier package by being automatic (the user can go and do something else while it is computing) and of significantly higher accuracy, and by being able to treat any torus-like surface (not only surfaces of revolution).

It is obvious that the ability to compute shortest paths on a manifold is a prerequisite for being able to compute the cut locus. It is however important to realize that globally shortest (and not just locally shortest) paths are required. For this reason, iterative methods based upon improving one or two initial approximations (see, for example [Maekawa, 96]) are not adequate as the basis for a cut locus algorithm.

One may choose to base a cut locus computation on an algorithm for finding a curve between two points on a surface, such that each point on the curve is equally distant from the two points. Such a curve is an example of what is called a medial curve. If we take surfaces with periodic parametrizations, then the starting point for which the cut locus is to be computed has many preimages in parameter space.

One would construct medial curves in parameter space between pairs of such preimages of the starting point, and thus arrive at a subset of the cut locus (what would be missing are the parts proceeding from conjugate points). Work on medial curves on surfaces done to date [Rausch et al. 97] however does not address any of the global properties which are vital to an understanding of the cut locus, although it has led to an algorithm for computing

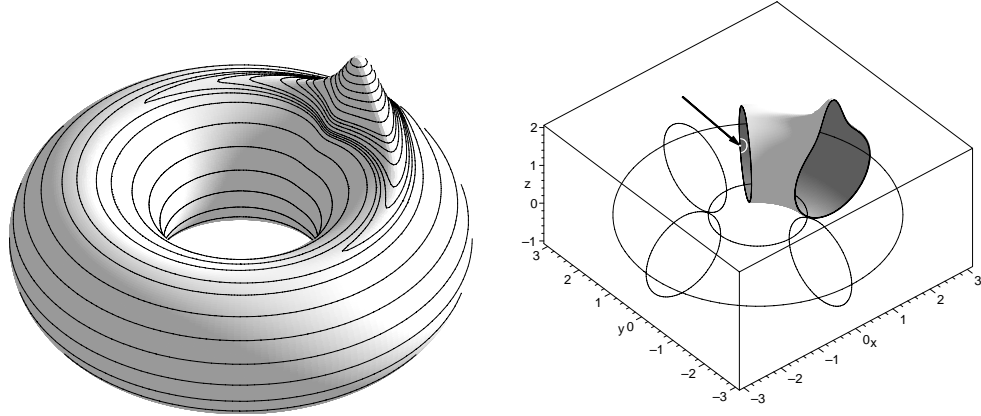


FIGURE 1. The nontrivial torus-like surface which will be used throughout the rest of this paper. Note the starting point below the equator and furthest from the viewer. It is pointed to by an arrow and marked by a white semicircle in the illustration on the right.

geodesic Voronoi diagrams on certain classes of surfaces [Kunze, et al. 97]. Voronoi diagrams (see [Aurenhammer, 1991] for an overview) are related to the cut locus since they indicate which points on a surface are closest to those of a set of source points, but encode less geometric information than the cut locus since they are insensitive to the position of conjugate points. To show how closely related the cut locus and a Voronoi diagram can be, consider the case of a compact two-dimensional Riemann surface with a 2π -periodic parametrization in both dimensions.

On such a surface, the Voronoi diagram for the vertices of a regular quadratic grid of side-length 2π in parameter space is, when projected onto the surface, a subset of the cut locus from the point which is the projection of any one of these vertices onto the surface. This Voronoi diagram is in fact equal to the cut locus on the same surface from the same starting point if conjugate points do not play any role.

It is precisely the role of conjugate points which makes computation of the cut locus based upon a triangulation of the surface essentially impossible. The reason is that each vertex acts as a weak obstacle, creating conjugate points all over the triangulated surface, and the approximated cut locus “grows hair” as a result. To be more precise, triangulation makes a surface less smooth, and we know from [Itoh and Tanaka, 98] that this influences the Hausdorff dimension of the cut locus. Of course one is able to compute approximations to distances and Voronoi diagrams on such surfaces [Kimmel and Sethian, 98, Kimmel and Sethian, 99, Barth and Sethian, 98].

Level set methods ([Sethian, 99], in particular Chapters 19 and 20), which are the basis of the aforementioned

work on triangulated surfaces, are in many ways complementary to Loki, which computes an approximation to the exponential map. Level set methods are designed for situations corresponding in one way or another to forest fires, where any given tree can only be burnt once. Exponential map methods are designed for situations corresponding in one way or another to echoes, where any given person can hear the same shout many times.

The point is that different physical phenomena require different treatments of wavefronts, and that different algorithms tend to complement each other, so that one may be more suited to a given computation, but both have their justification in the grand scheme of things. Level set methods use a viscosity term to smear out the sharp corners and self-intersections of a wavefront which define the cut locus, and for this reason they are not ideally suited to computing the cut locus. Exponential map methods do not involve any such smoothing, and this allows Loki to locate the cut locus with great precision.

1.1 Notation

Throughout this report, we will use the variable names u and v to refer to coordinates in \mathbb{R}^2 , which is the universal covering of torus-like surfaces, as in Equation (1–1). It is expected that the mapping to the surface will be 2π -periodic in both of these variables. Coordinates in \mathbb{R}^3 will be named x , y and z .

Surfaces which cannot be embedded in \mathbb{R}^3 can alternatively be defined by the three functions $E(u, v)$, $F(u, v)$ and $G(u, v)$, which define the metric given by

$$dl^2 = E(u, v) du^2 + 2F(u, v) du dv + G(u, v) dv^2. \quad (1-2)$$

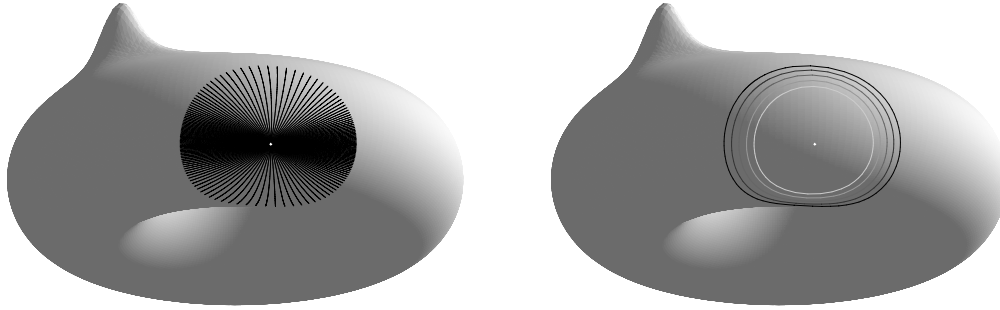


FIGURE 2. Geodesic curves and circles (on the left and right, respectively) emanating from the starting point on the surface of Figure 1.

Points on the surface can be parametrized by an initial angle from the starting point, s , and a distance from it, t . The angle s is measured in \mathbb{R}^2 using the standard Euclidean metric and u and v as coordinates.

The cut locus will sometimes be plotted in terms of (u, v) coordinates, sometimes in terms of (s, t) , and sometimes in terms of $(X, Y) \equiv (t \cdot \cos s, t \cdot \sin s)$, which we choose to call a polar representation.

1.2 A Nontrivial Torus-Like Surface

To provide a concrete example, we take some nontrivial surface and define it exactly. We have chosen to use the surface shown in Figure 1. It is a deformation of the standard torus given by

$$\begin{aligned} \hat{x}(u, v) &= (2 + \cos u) \cdot \cos v \\ \hat{y}(u, v) &= (2 + \cos u) \cdot \sin v \\ \hat{z}(u, v) &= \sin u. \end{aligned} \tag{1-3}$$

The deformation is produced by adding an offset centred at $(u_0, v_0) = (1.57, 0)$, with widths $(\Delta u, \Delta v) = (0.2, 0.2)$ and amplitude $d = -1$ in the direction of the normal (n_x, n_y, n_z) , where the offset function is analytic and of an appropriate periodicity:

$$\begin{aligned} x(u, v) &= \hat{x}(u, v) + f(u, v, 1.57, 0, 0.2, 0.2, -1) \\ &\times \frac{n_x(u, v)}{\sqrt{n_x^2(u, v) + n_y^2(u, v) + n_z^2(u, v)}} \end{aligned} \tag{1-4}$$

(and analogously for y and z) where

$$\begin{aligned} f(u, v, u_0, v_0, \Delta u, \Delta v, d) \\ = \frac{d\Delta u^2 \Delta v^2}{4 \left(\left[1 + \frac{1}{2}(\Delta u)^2 - \cos(u - u_0) \right] \cdot \left[1 + \frac{1}{2}(\Delta v)^2 - \cos(v - v_0) \right] \right)} \end{aligned} \tag{1-5}$$

and

$$n_x(u, v) = \frac{\partial \hat{y}(u, v)}{\partial u} \cdot \frac{\partial \hat{z}(u, v)}{\partial v} - \frac{\partial \hat{z}(u, v)}{\partial u} \cdot \frac{\partial \hat{y}(u, v)}{\partial v} \tag{1-6}$$

etc. The surface is given by the parametrization

$$(u, v) \mapsto (x(u, v), y(u, v), z(u, v)). \tag{1-7}$$

The starting point shown in Figure 1 has coordinates $(u_P, v_P) = (-0.5, 1)$. One can readily imagine geodesic curves of some small given length beginning at this starting point (see Figure 2). Figure 3 shows the development in “time” (distance) of these geodesics and geodesic circles. What is most interesting is the intersection of geodesics behind (further from the starting point) the bump on the torus, and the “swallowtail” feature in the associated geodesic circles. The points of first intersection of these circles define one segment of the cut locus. The first such point is a conjugate point with respect to the starting point.

Figure 4 illustrates the cut locus of this surface for the given starting point. Note the component on the side of the bump farthest from the starting point, and also the two components close to the outer equator. These “sideburns” also begin at conjugate points, due to the focusing action of the positive curvature along the outer equator.

Figures 5, 6 and 7 provide further representations of the same cut locus. They all include the same segments of geodesic circles as shown in the upper illustration in Figure 4. If one takes the outer boundary of Figure 7, one has the Voronoi diagram for the set of source points which are preimages in the universal covering of the starting point (u_P, v_P) : $\{(u_P + 2\pi i, v_P + 2\pi j) \mid i, j \in \mathbb{Z}\}$.

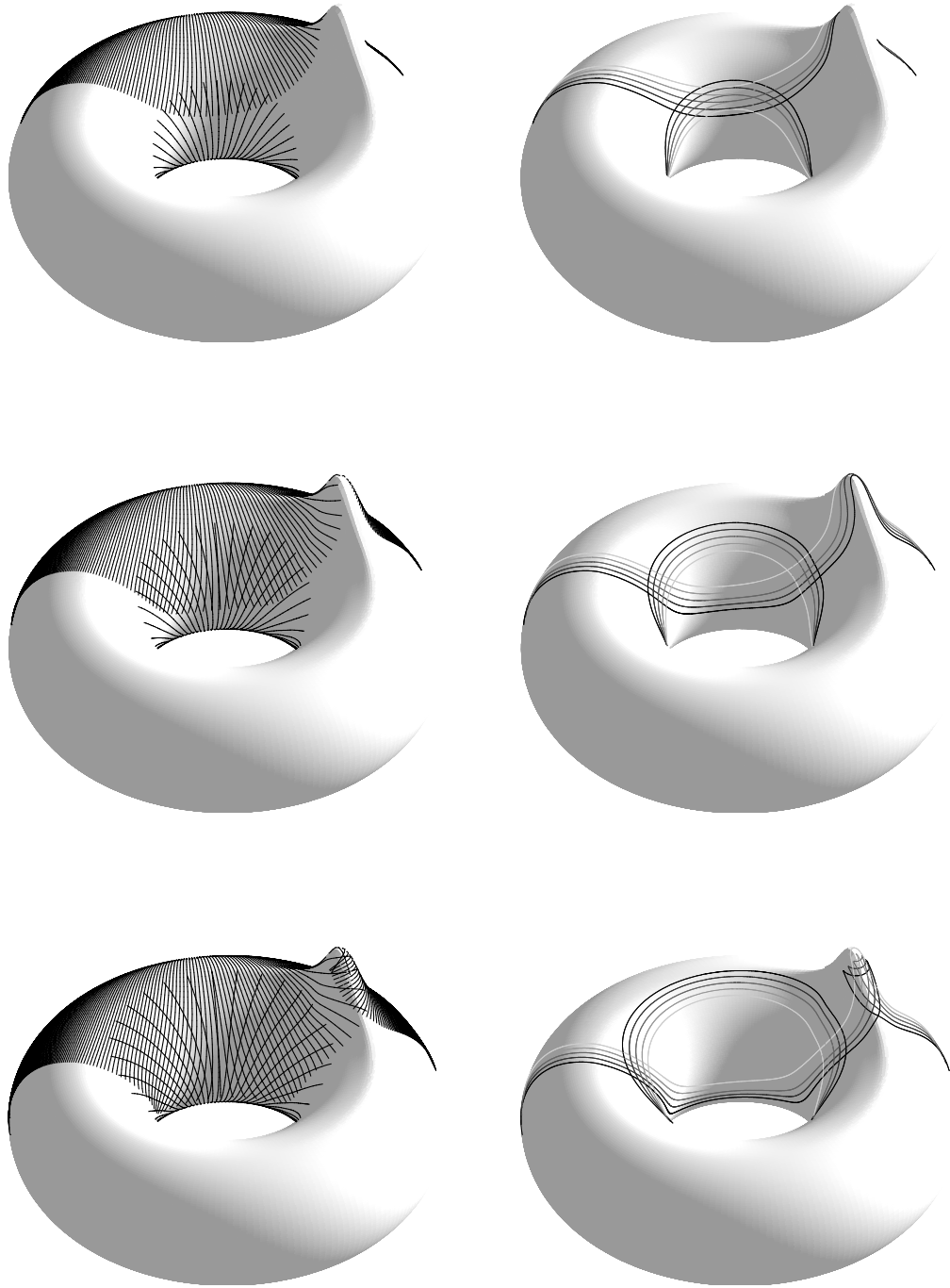


FIGURE 3. Geodesic curves and circles (on the left and right, respectively) emanating from a non-symmetrically placed starting point on the far side of the surface (see Figures 1 and 2). The propagation of geodesic circles with time is hinted at by the use of different grey tones.

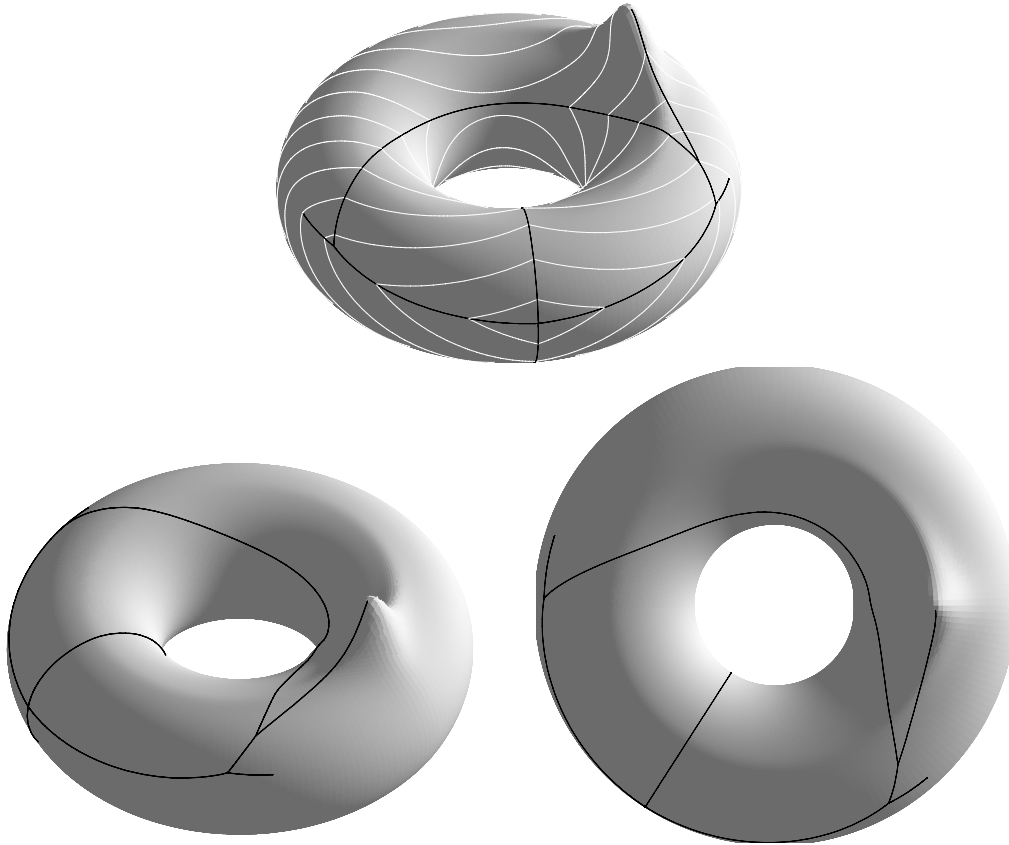


FIGURE 4. Several views of the cut locus of the surface from the starting point defined in the text (see Figure 1 and Equations (1-3) to (1-7)). The upper illustration includes geodesic circles in white, up to their intersection with the cut locus.

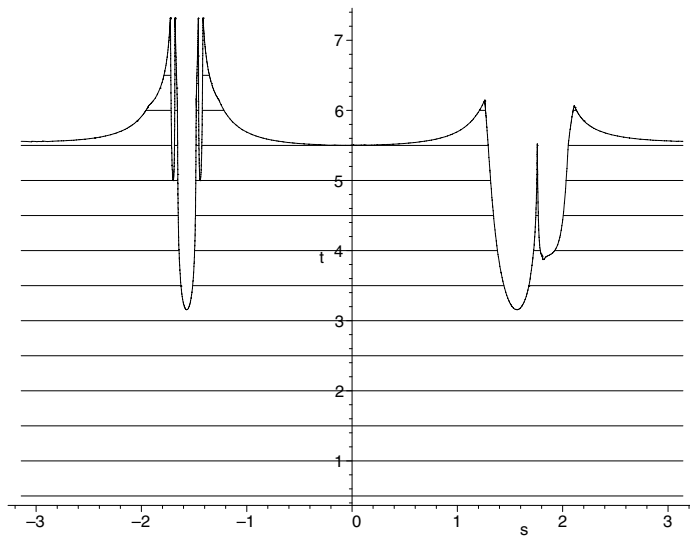


FIGURE 5. The cut locus of Figure 4 as a maximum distance (t) one can go in a given direction (at the angle s) from a given starting point before there are shorter paths to the same endpoint.

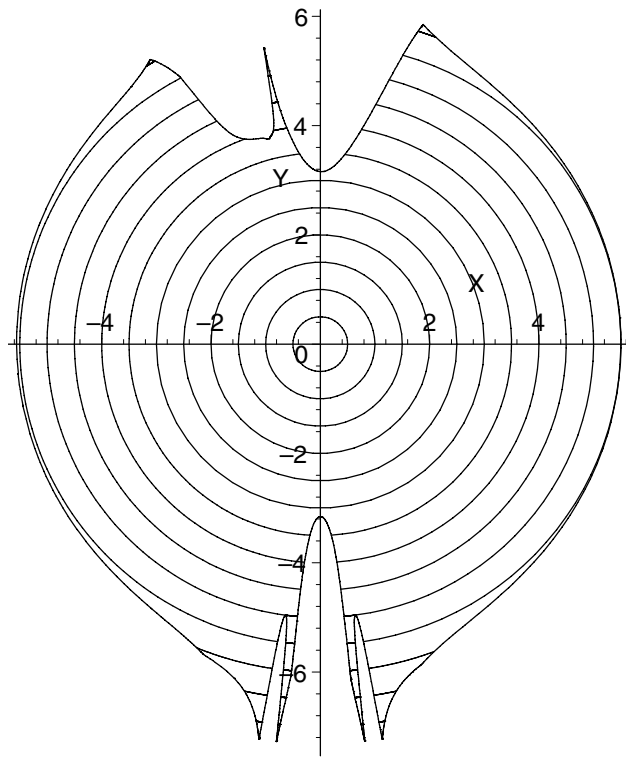


FIGURE 6. The cut locus of Figure 4 in a polar representation.

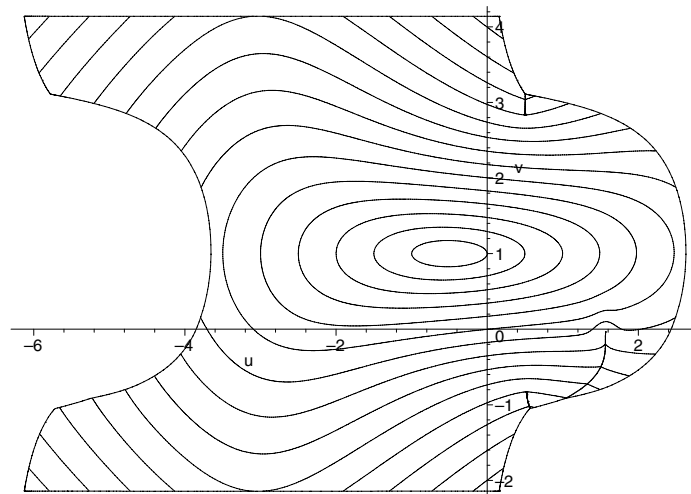


FIGURE 7. The cut locus of Figure 4 in terms of the 2π -periodic coordinates used to parametrize the surface.

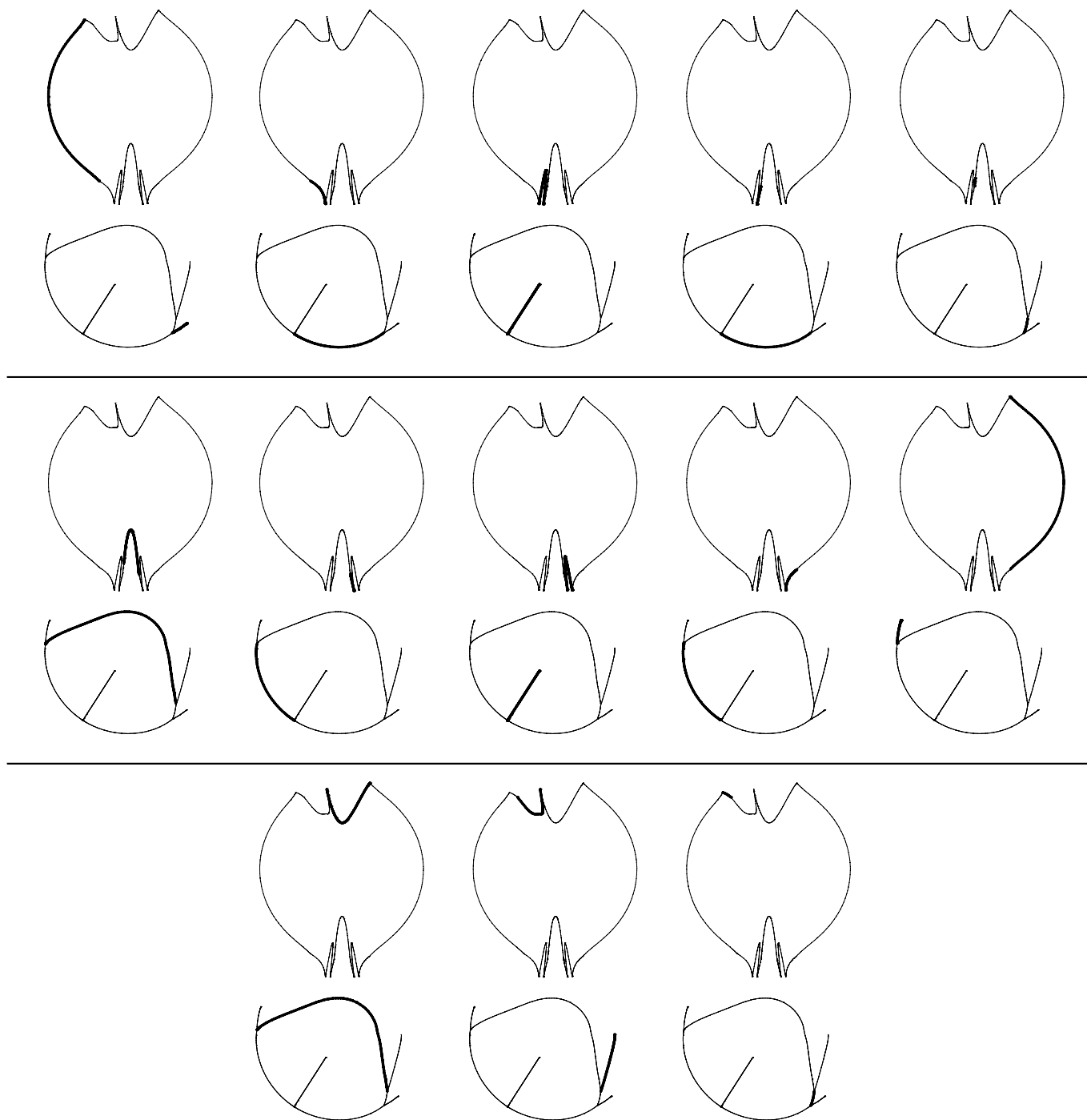


FIGURE 8. Pairs of representations of the cut locus of Figure 4, the upper of each pair in polar representation as in Figure 6, the lower a projection onto the (x, y) plain as in the lower right illustration of Figure 4. Each pair highlights a different segment of the cut locus, showing the complex relationship between the two representations.

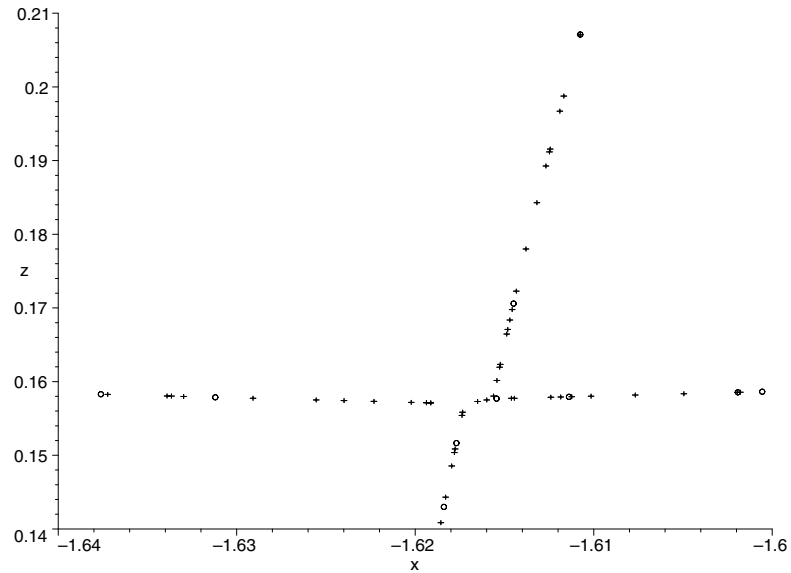


FIGURE 9. A close-up of a portion of the cut locus illustrated in Figure 4. The point of interest appears to be the intersection of four curves in Figure 4, but this close-up shows that it is in fact a pair of points, each being the intersection of three curves. This fact alone, and the angles at these points, may be useful in determining the geometry of the surface from the cut locus. The circles and crosses are data generated by Loki for different input tolerances (see Table 1).

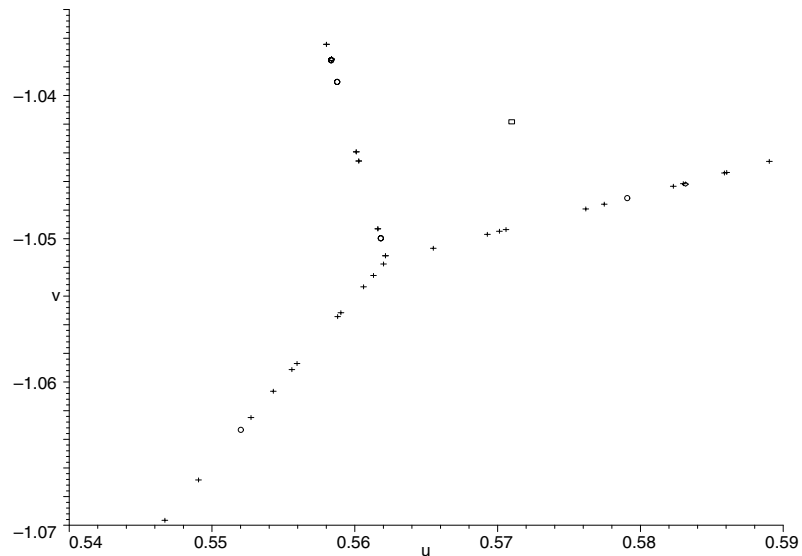


FIGURE 10. Here is a further close-up of the cut locus illustrated in Figure 4, this time viewed in (u, v) coordinates (c.f. Figure 7). Angles measured from such data may help identify the surface involved. The squares, circles and crosses are data generated by Loki for different input tolerances, the squares corresponding to the greatest tolerance (see Table 1).

2. THE ALGORITHM

The central idea behind the algorithm is to construct a piecewise polynomial approximation of the exponential map up to some given maximum distance from the starting point, and then to find the maximum distance one can follow geodesic curves from the starting point until they are no longer the shortest path to their endpoints. This is done by bisection, using numerical inversion of the approximation of the exponential map to find all paths of distance no greater than the given maximum distance leading from the starting point to any endpoint. Figure 17 illustrates the data on which bisection acts. The cut point is the first point of intersection between the diagonal ($T = t$) and the other curves.

The algorithm splits up into two well-defined parts: the construction of an approximation of the exponential map, and its inversion.

Loki has been written with differential geometers in mind, and always with an eye to allowing extensions or even restructuring. A tradeoff between efficiency and flexibility has been important at all stages of the project. It is for this reason that Loki has been written in C++ in an object-oriented style ([Stroustrup, 97]). It became clear at an early stage of the development that memory usage would be a problem, and this has had a significant influence on the development of the package.

2.1 Approximation of the Exponential Map

The aim is to construct a piecewise polynomial approximation of the exponential map up to some given maximum distance from the starting point. Since we can expect construction of this approximation to be costly, it makes sense to compute it only once, storing it as a data structure which is suited to numerical inversion.

In the present context, an exponential map is a map from angle-distance pairs to parametric coordinates in \mathbb{R}^2 . In the notation used here, it allows one to compute (u, v) from (s, t) . For obvious practical reasons, it will be restricted to some maximum distance from the starting point, t_{max} . Due to the fact that this mapping tends to cover points on the surface more than once, we cannot use the usual device of triangulating the surface using (u, v) coordinates and assigning an (s, t) pair to each node without significant modifications. One would have to assign a list of pairs to each node, and the question of grid adaptivity is difficult to cope with in a satisfactory manner if more than one set of values are sharing each grid point. A more natural data structure to use is one which follows the geodesic curves and circles more like a scarf

which can be wrapped around one's neck many times (Figure 15 gives some idea of what is meant). This scarf can be composed of many patches (see Figure 11), each being a local polynomial approximation of the mapping $(s, t) \mapsto (u, v)$. We have decided to base the algorithm on fourth-degree polynomials, these being the highest degree polynomials for which one has explicit formulæ for their roots, and four also being the order to which standard Runge-Kutta (RK45) is locally correct.

A patch is constructed by beginning with a fourth-degree polynomial approximation of a segment of a geodesic circle centred around the angle s_0 with distance t_0 from the starting point

$$\hat{u}_{t_0}(s) = \sum_{i=0}^4 \frac{\hat{u}_i}{i!} \cdot (s-s_0)^i$$

(and the same for v) and then “virtually” evolving this curve using standard Runge-Kutta out to a distance of t . The Christoffel symbols appearing in the geodesic equations are computed automatically by Loki. The idea is to use automatic differentiation ([Rall, 81]) by overloading all arithmetic operations appearing in the Runge-Kutta routine, but actually evaluating for $t = t_0$. What one obtains are Taylor coefficients for an expansion in $t - t_0$ about every point of the segment at t_0 :

$$u(s, t) = \sum_{i=0}^4 \sum_{j=0}^4 \frac{u_{i,j}}{i! j!} \cdot (s-s_0)^i \cdot (t-t_0)^j$$

(and the same for v). The point in using a numerical algorithm in combination with automatic differentiation is that automatic differentiation, although it generally reduces numerical error, is not immune to it. Applying automatic differentiation to a numerically stable algorithm is very simply an attempt to keep numerical error to a minimum.

We can then evaluate $\hat{u}_{t_1}(s) := u(s, t_1)$ and $\hat{v}_{t_1}(s) := v(s, t_1)$ (for a fixed value of t_1 greater than t_0) to obtain the next fourth-degree polynomial approximation of a segment of a geodesic circle centred around the angle s_0 with distance t_1 from the starting point, and the whole process can be repeated. In this way, each patch is connected to its predecessor, and the data structure resembles a set of streamers radiating out from the starting point.

The algorithm uses some heuristics to decide when a patch should be shortened (in t) or split (in s). These decisions would otherwise be a global problem in themselves, since splitting a patch results in two independent

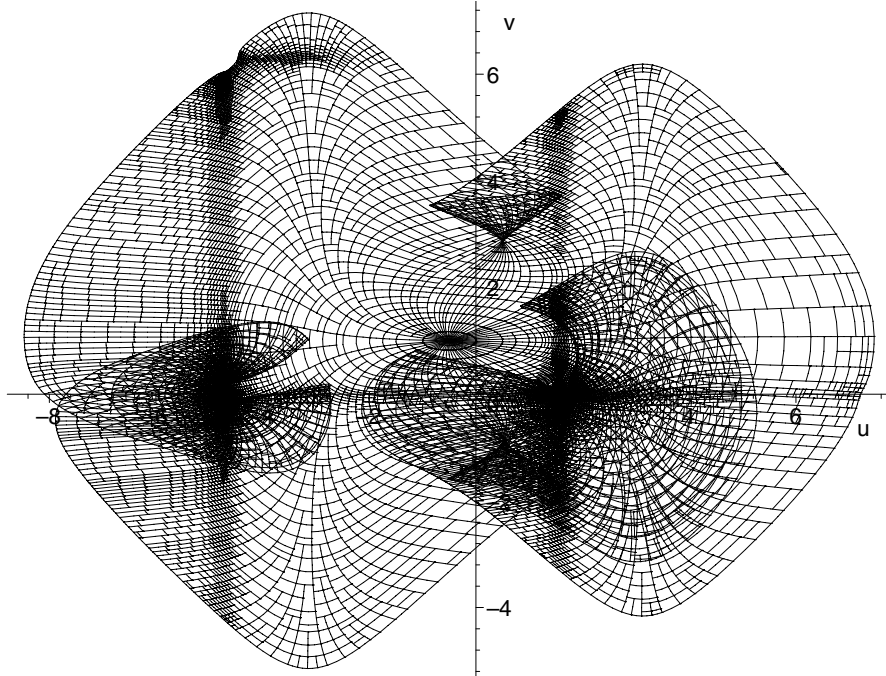


FIGURE 11. The data structure used for approximating the exponential map of a nontrivial surface with nontrivial starting point (see Figure 1). An input tolerance of 10^{-4} was used to generate this approximation. The product form of the offset used to define the bump on the surface (see Equation (1-5)) is responsible for the regular square grid artifact (“darkening”) of side-length 2π .

streamers from that point onwards (instead of just continuing with one). It is therefore a good idea to avoid splitting where possible.

At the same time, shortening cannot be repeated indefinitely. In regions of negative curvature (diverging geodesic curves), it is clearly an advantage to split rather than to shorten. The algorithm shortens until the patch’s length (in t) would be a small fraction of its width (in s). If the preceding patch was not split, then the current patch is. Otherwise it continues to shorten. In any case, the algorithm checks that the “speed” of the geodesic curves does not deviate from 1 by more than the input tolerance in the middle or at the corners of the patch, and that actually (rather than virtually) using Runge-Kutta up the middle and to the corners of the patch gives the same values as the Taylor expansions computed by automatic differentiation. Finally, it was found that rescaling the speed ($|dl/dt|$, see Equation (1-2)) of the geodesic curves to 1 (the correct value) at each such step helps to reduce error.

2.2 Inversion of the Exponential Map

Inversion of the exponential map corresponds to finding all the geodesic curves up to the given maximum length

which connect the starting point (u_p, v_p) with some other given point (U, V) on the surface. See Figure 12. The current implementation of Loki uses a fairly primitive, but robust, divide-and-conquer algorithm to solve the system

$$\begin{aligned} u(s, t) &= U \\ v(s, t) &= V \end{aligned}$$

on each patch of the data structure (Figure 11). This part of the algorithm is fairly standard, and will not be discussed further.

In particular, it does not seem that any great attention need be paid to the kinds of singularities generally encountered when plotting contour lines ([Taubin, 94]), primarily because the solution set here is actually an intersection of curves which tend to meet at an angle. Any improvements to the algorithm could increase efficiency but not accuracy. Since accuracy and reliability are the main goals, it was decided not to spend too much time investigating this point.

One important point to be made is that the periodicity of the parametrizations of torus-like surfaces requires that inversion be done for a set of preimages of the point being inverted. See Figures 13 and 14. Since there are actually infinitely many preimages, some finite set must

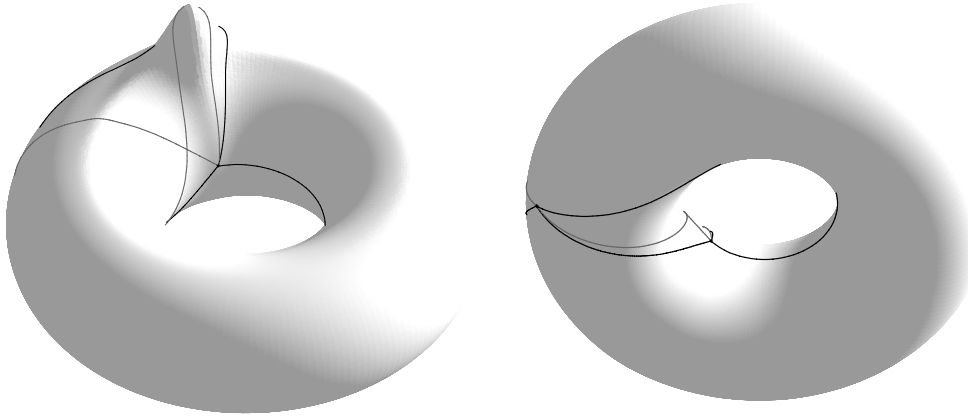


FIGURE 12. Geodesic curves joining two points on a nontrivial surface.

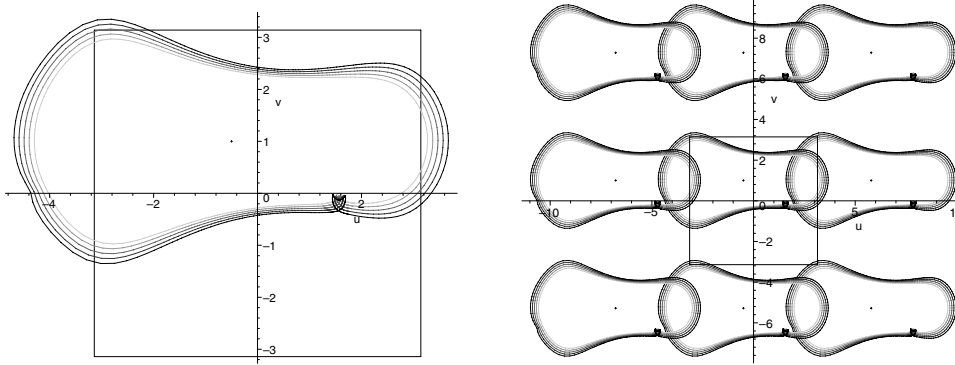


FIGURE 13. The algorithm actually works in (u, v) coordinates (a parametrization of the surface), but makes the necessary identifications appropriate for a torus by identifying points with coordinates differing by multiples of 2π in both dimensions. The geodesic circles plotted here are the same ones displayed at the bottom of Figure 3.

be chosen. This is done by the user by setting the variable `copies`. The set of preimages used in inverting is given by

$$\{(U + 2\pi i, V + 2\pi j) \mid i, j \in \mathbb{Z}; i, j \in [-\text{copies}, \text{copies}]\}.$$

During implementation of Loki, it was discovered that gaps between the various patches of the approximation of the exponential map (see Figure 15), and also overlap between patches, were causing serious problems even for small input tolerances. The reason is that the bisection algorithm being used to locate a cut point assumes that inverting the exponential map gives one all geodesic curves which pass through a given point. If one of these is missing, the bisection algorithm may take a step in the wrong direction.

The solution chosen here is to linearly interpolate between patches where gaps or overlaps can occur (at the edges of the streamers spoken of before). The linear in-

terpolation has the form

$$\tilde{u}(x, t) = x \cdot u(s_{gap-}, t) + (1 - x) \cdot v(s_{gap+}, t),$$

where

$$\tilde{u}(x, t) = \tilde{u}_0(x, t_0) + \sum_{i=1}^4 \frac{\tilde{u}_i(x, t_0)}{i!} (t - t_0)^i$$

(and also for \tilde{v}). Given an interval $[t_0, t_1]$ to work in, Loki determines whether there might be a solution to the system

$$\begin{aligned} \tilde{u}(x, t) &= U \\ \tilde{v}(x, t) &= V. \end{aligned}$$

If an $x \in [0, 1]$ and a $t \in [t_0, t_1]$ can be found, then the algorithm returns (s_{gap}, t) as the preimage of (u, v) .

The algorithm follows these steps to determine whether there could be a solution in the given interval (and uses the same strategy for v):

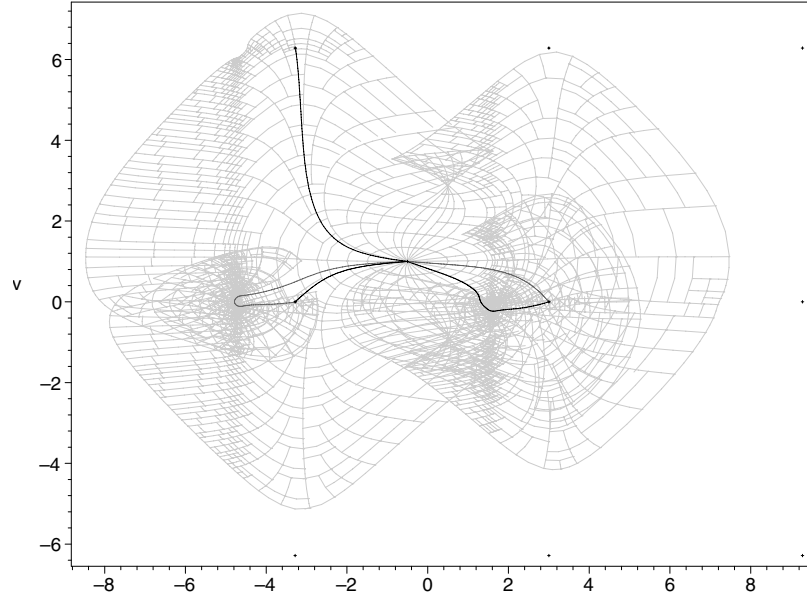


FIGURE 14. Computation of geodesics between the starting point and some other given point on a torus is accomplished by generating copies of the other point with (u, v) coordinates differing by multiples of 2π , and computing shortest paths of no more than the given maximum distance to all of these copies. In this way, only one exponential map stored in memory suffices. The paths shown here are those mapped onto the embedding of the surface in \mathbb{R}^3 in Figure 12.

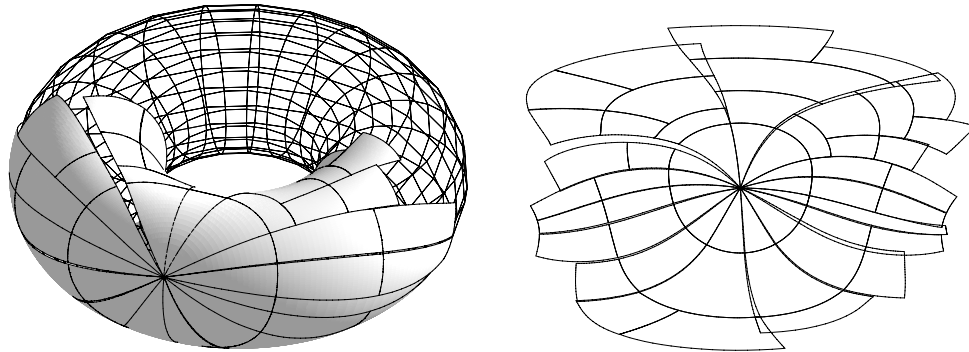


FIGURE 15. Part of the adaptive data structure approximating the exponential map for a standard torus. In this case the error tolerance has deliberately been set high, so that the gaps between patches are visible to the naked eye. These gaps remain a problem even for much smaller tolerances. The illustration on the right shows the same data structure in terms of (u, v) coordinates. It is clear that overlaps also occur. These are dealt with in the same manner as the gaps.

1. If $[\tilde{u}(0, t_0) - U] \cdot [\tilde{u}(1, t_0) - U] \leq 0$ or $[\tilde{u}(0, t_1) - U] \cdot [\tilde{u}(1, t_1) - U] \leq 0$ then there is a solution to $U = \tilde{u}(x, t)$.
2. If $|\tilde{u}_0(0, t_0) - U| > \sum_{i=1}^4 \frac{|\tilde{u}_i(0, t_0)|}{i!} (t_1 - t_0)^i$ or $|\tilde{u}_0(1, t_0) - U| > \sum_{i=1}^4 \frac{|\tilde{u}_i(1, t_0)|}{i!} (t_1 - t_0)^i$ then there is no solution.
3. If (using explicit formulæ for the roots) there is a $t \in [t_0, t_1]$ for which $U = \tilde{u}(0, t)$ or $U = \tilde{u}(1, t)$, then there is a solution.
4. Otherwise there is no solution.

This algorithm is applied to all the possible gaps/overlaps of the data structure used to approximate the exponential map. Figure 11 shows such a data structure, and Figure 16 the corresponding lines where gaps or overlaps must be checked.

In order to keep memory usage down, it was decided to recompute the linear interpolations \tilde{u} and \tilde{v} every time the exponential map is inverted. This is not as inefficient as it sounds. The choice of whether or not to recompute them is based upon the ratio of the time needed to compute them (which is short, because this is just a matter of

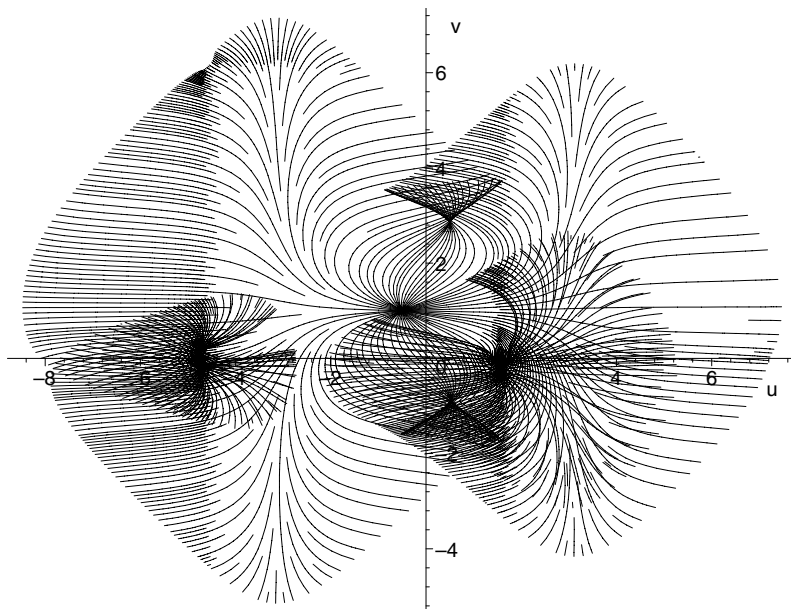


FIGURE 16. The pattern of gaps and overlaps in the data structure used to approximate the exponential map for a nontrivial surface.

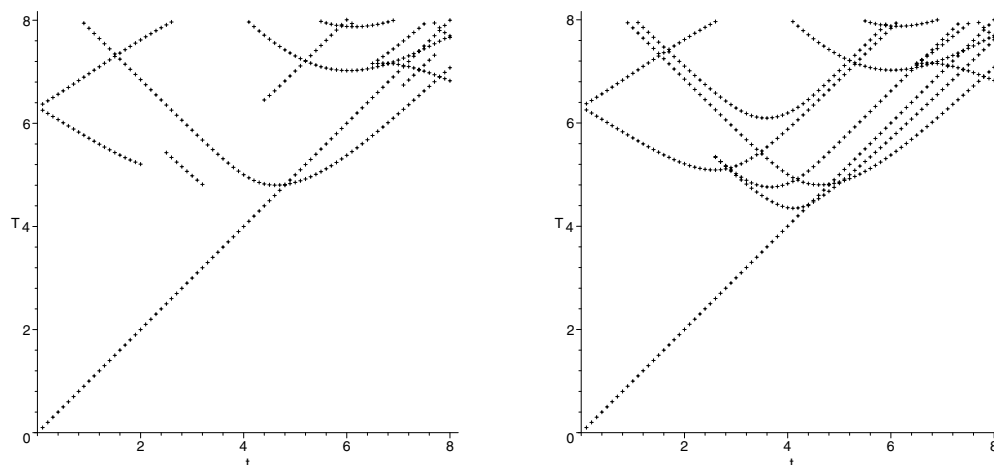


FIGURE 17. If one travels a distance t from the chosen starting point (u_P, v_P) along a geodesic curve with a given initial direction ($s = 2$), one will reach some unique point (Q) on the surface. On a compact surface there will always be other geodesics which also begin at (u_P, v_P) and pass through Q . If one compiles a list of the lengths (T) one must travel along these various geodesics from (u_P, v_P) to reach Q , then this list of distances can be plotted vertically as a function of t . The two figures presented here correspond to two different choices of input tolerance. On the left is 10^{-4} , on the right is 10^{-5} . Note that the larger input tolerance on the left results in the algorithm incorrectly deleting parts of some of the curves. This is because different geodesic curves appear to be identical at too low a resolution.

polynomial evaluation) and the time to look for solutions (which can be long, because one is searching for zeros in a nonlinear system).

Similarly many memory accesses are involved in both options, although choosing not to store the interpolations turns out to halve the memory requirements of the entire program.

2.3 Error Estimation

It should be pointed out that the computation of global geometric quantities is fraught with difficulties, and one of these is that no good method of estimating the error is available. Consider a general purpose numerical root-finder. While it is usually possible to give reliable local error estimates for roots found, it is essentially impossi-

Input tolerance	Symbol	Memory	CPU time	Est. error
10^{-3}	□	4M	2 hrs.	1.2951
10^{-4}	◇	8M	8 hrs.	0.3823
10^{-5}	○	19M	2 days 10 hrs.	0.0180
10^{-6}	+	45M	9 days 19 hrs.	0.0037

TABLE 1. Performance and accuracy of the algorithm as a function of input tolerance. The symbols given in the second column are those used in the various plots depicting output data for different values of the input tolerance.

ble (if the function is not analytic, for example) to make global statements such as “all roots have been found”, unless one goes to extraordinary lengths (like using interval arithmetic). It is precisely this point which we must not forget when computing the cut locus. While one can give local error estimates for the accuracy of computed geodesic curves, one cannot easily estimate the error in the number of geodesic curves found to pass through a given point. However, the cut locus is defined by precisely this number (i.e. the cut locus is the closure of all points having at least *two* shortest length curves back to the starting point). In practice, this means that one has no proof that the output will converge for all possible surfaces, but this does not prevent one from using Loki as a tool for experimentation.

The gaps/overlaps mentioned above are a good measure of the actual inaccuracy of the exponential map approximation. Their width (and the difference between their position and that which is obtained by using Runge-Kutta in a straightforward manner) is in fact used to produce the error estimate given in the output file. In this way, the input tolerance given by the user results in the error estimate given by the program. See Table 1 for examples of pairs of input tolerances and error estimates.

This error estimate is however nothing more than an estimate. There are in fact many nontrivial processes which result in error. Figure 17 illustrates one such process. When inverting the approximation to the exponential map, one must have some way of telling whether two geodesic curves, given the (s, t) coordinates of points on them, should be considered to be identical or distinct. This is not always clear, particularly if the lengths are very close. The decision is based upon computing the (u, v) coordinates of corresponding points along the length of the curves, and seeing if they deviate by more than the estimated error.

The variable `checks` controls the number of points checked. It is an integer. The algorithm only checks those pairs of geodesic curves for uniqueness which al-

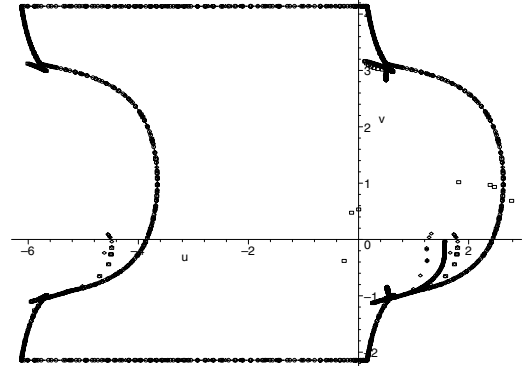


FIGURE 18. Output data in (u, v) coordinates (see also the cut locus plot in Figure 7). Squares, diamonds, circles and crosses correspond to ever decreasing input tolerance (see Table 1).

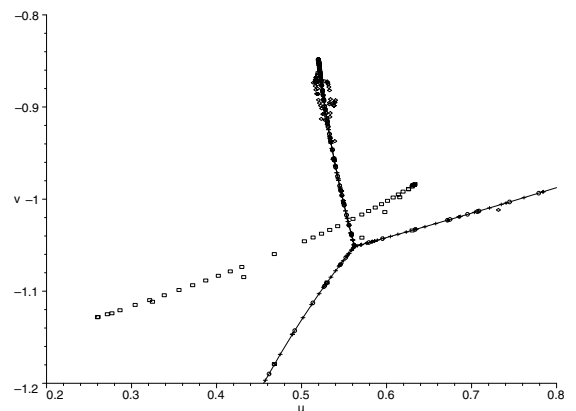


FIGURE 19. A close-up of Figure 18. Note that the squares, which are output data corresponding to the largest input tolerance (and therefore the least “trustworthy”) define an artifact (a curve) which disappears completely when the input tolerance is lowered.

ready have the same endpoints. The points which are checked are placed at multiples of $1/\text{checks}$ of the total length from one end. In other words, if `checks` is set to 2, then only one point, halfway between the endpoints, will be checked. The default is 4, meaning that three equispaced points are checked.

If the input tolerance has been set too large, one of the consequences is that geodesic curves which are distinct may be confused. As a result, the inversion algorithm will effectively ignore some solutions, and this will have consequences for the bisection routine responsible for determining the position of the cut point. The reader

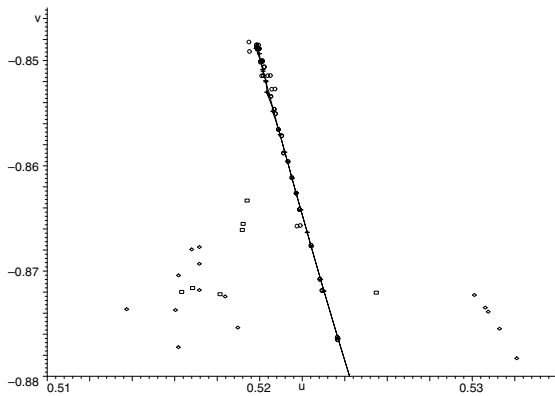


FIGURE 20. A close-up of Figure 19.

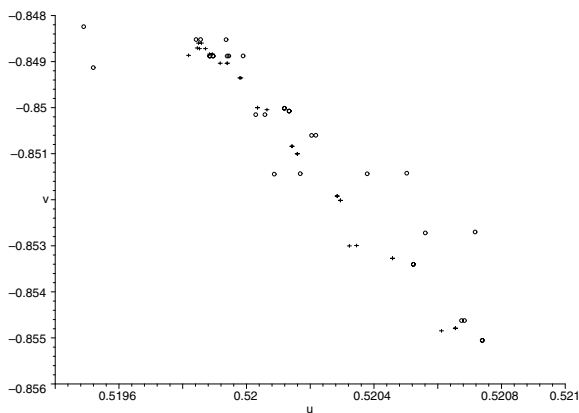


FIGURE 21. A close-up of Figure 20.

should now understand that the problem of error estimation is at least as difficult as that of computing the cut locus. The output error estimate is therefore only intended as a guideline.

Another variable (`copies`, see Section 2.2) also influences the error in a nontrivial way.

An indication of how the output converges on the cut locus can be seen in Figures 18 to 21, where the symbols of Table 1 have been used throughout.

2.4 Code Generation

This Section will only deal with technical issues relating to the automatic generation of large fragments of Loki's source code, and in particular with a software tool developed with more than just Loki in mind.

Experience with Loki has shown that it is advisable to demand IEEE-754 arithmetic [IEEE, 85], oth-

erwise machine-dependent floating point exceptions tend to cause the program to abort, and results may vary from architecture to architecture. When using `g++`, the option `-ffloat-store` can be recommended. In the case of the Compaq C++ compiler for Tru64 UNIX, the `-ieee` option should be used.

It is a well-known fact that compilers have difficulties allocating registers for very large procedures (see, for example, Section 5.2 of [Lawall, 98]). For this reason, a special automatic differentiation tool was written during the development of Loki. Its purpose is to generate C code for computing the many higher derivatives required (up to fourth order in two variables) for functions such as cosine or square root. The tool first generates an intermediate representation of the required code containing only unary and binary operations (this is the function code list spoken of in [Rall, 81]). It then optimizes by identifying multiple occurrences of the same operation. Finally, and this is the important point here, it performs “register allocation” at source code level by ensuring that a small number of variable names are reused as much as possible. For example, consider the code fragment

```
t1452=t3321+t42;
t1460=sqrt(t1452);
t1463=t1460+t42;
t1464=sqrt(t1463);
```

The result of the “register allocation” mentioned above could be to transform that code to

```
r10=t3321+r3;
r1=sqrt(r10);
r10=r1+r3;
t1464=sqrt(r10);
```

(the uncertainty is due to the fact that the optimizations are global in nature, so looking at a single code fragment in isolation does not allow one to conclude exactly how it will be transformed). Note that such transformations automatically lead to an improvement in data locality, which also tends to optimize cache usage.

On a SUN E6500 processor, using `g++` version 2.8.1 and with the options `-O3 -ffloat-store`, this “register allocation” results in an increase in speed of 42% (for computing derivatives of the square root operation). With the compiler options `-O1 -ffloat-store -fschedule-insns` (see [Lawall, 98]), an increase of 34% is achieved. Using the Compaq C++ compiler `cxx` version 6.2 on a DEC Alpha machine with options `-ieee -O4`, an increase of 20% is seen.

While this optimization tool has clearly been of some use in the development of Loki, one can expect that over time compilers will improve to the point where they are able to perform such optimizations themselves, so this tool will not be described any further here. Nonetheless, it did generate approximately two thirds of Loki's 12000 lines.

It should also be pointed out that some numerical difficulties were encountered in connection with automatic differentiation applied to division. In computing the higher order derivatives, powers of the constant term of the denominator are required, and this can lead to overflow. A good solution seems to be to rescale such that this constant term becomes one. That is, compute using

$$\frac{a}{b} = a \cdot \left(\frac{1}{b_0}\right) \cdot \left(\frac{1}{b_0} \cdot b\right)^{-1}$$

instead of simply dividing a by b .

3. USE OF THE SOFTWARE

The file `metric.cc` as defined in Figure 22 defines the standard torus given by Equation (1-1). This is a good sample file to start with when first using Loki. A suitable input file is shown in Figure 23. Once these two files have been set up appropriately in a directory containing the source code for Loki, it should be compiled and started.

```
x=(2.0+cos(u))*cos(v);
y=(2.0+cos(u))*sin(v);
z=sin(u);
```

FIGURE 22. Contents of the file `metric.cc` for the standard torus in \mathbb{R}^3 .

For example, if the input file of Figure 23 is called `input`, then the following commands will start Loki on the DEC alpha machines of the Department of Mathematics of the Technical University of Denmark:

```
cxx -ieee -O4 -o loki main.cc -lm
loki < input > output
```

On one of our DEC alpha machines, computation takes approximately twenty minutes.

The file `metric.cc` as defined in Figure 24 defines the same surface as Equations (1-3) to (1-7). Note in particular the appearance of the same numerical constants as in Equation (1-4). A corresponding input file is shown

```
tolerance = 1e-3
maximum distance = 7.5
u coordinate of starting point = -0.5
v coordinate of starting point = 1.0
```

FIGURE 23. A sample runtime input file.

```
x=(2.0+cos(u))*cos(v);
y=(2.0+cos(u))*sin(v);
z=sin(u);

BUMP(1.57, 0.0, 0.2, 0.2, -1.0);
```

FIGURE 24. Contents of the file `metric.cc` for a surface embedded in \mathbb{R}^3 .

in Figure 23. These two files may be used as a first non-trivial example (but one which will most probably take several hours) of the use of Loki.

One can also choose to define the surface in terms of its metric. See Equation (1-2) and Figure 25.

```
E=1.0+cos(u)*cos(u)
*(parameter*parameter-1.0);
F=0.0;
G=(2.0+cos(u))*(2.0+cos(u));
```

FIGURE 25. Contents of the file `metric.cc` for a surface defined purely in terms of its metric.

Selection of a starting point, a maximum distance for the exponential map, an error tolerance and a parameter value are made in an input file (standard input). As has already been pointed out, Figure 23 is an example of such a file.

It is a good idea to choose a large input tolerance (such as 10^{-4}) and a large maximum distance (this will of course depend upon which surface has been defined!) for the first computation. One can then use the output to adjust the maximum distance to be closer to the actual maximum distance to the cut locus (see Figure 5), and choose a smaller input tolerance. This should be done until the user is satisfied that some sort of convergence has been achieved.

Loki can also be used to compute distance matrices for sets of points on open or closed surfaces. The runtime input file should have the form shown in Figure 26, containing a line stating that a distance matrix is required,

```

tolerance = 1e-4
maximum distance = 0.7
copies = 0
compute distance matrix
0.2 0.1
0.1 0.2
0.0 0.2
0.0 0.0

```

FIGURE 26. A sample runtime input file which asks for a distance matrix for the points given.

and a list of points defined by their (u, v) coordinates. If the surface is open (such as a two-sheeted hyperboloid, as in the case in Figure 27), then the variable `copies` should be set to zero.

```

x=u;
y=v;
z=sqrt(1.0+u*u+v*v);

```

FIGURE 27. Contents of the file `metric.cc` for a two-sheeted hyperboloid.

3.1 Syntax for `metric.cc`

The file `metric.cc` is expected to contain either a set of definitions for the metric (1–2) as in Figure 25, or a set of definitions for a parametrization of an embedding of the surface in \mathbb{R}^3 followed by an optional list of “bumps” to distort this surface (see Figure 24 for an example).

Loki must be compiled anew for each change to `metric.cc`. The point is that a compiler will hopefully be able to make use of an explicit expression to be able to optimize generation of an exponential map. The effectiveness of this strategy will of course depend upon the compiler used, and, in particular, the optimization options selected. It is therefore difficult to say whether parsing of an input file would be a better solution, but this is an option that has been considered. Parsing of an input file will only be efficient if the time taken repeatedly interpreting the internal representation of the contents of that file is insignificant compared with the actual computations involved. It is however felt that the trend is that compilers do improve over time, and that the choice made is justified. In any case, the file `metric.cc` serves to define the surface to be worked on (up to a possible free parameter, which must be named `parameter`).

Since `metric.cc` is actually compiled as C++ source code, its syntax must conform to the C++ language definition (see [Stroustrup, 97] for an introduction). This is the reason why the lines assigning values to `x`, `y` and `z` must be terminated with a semicolon. Only the four elementary arithmetic operations (addition, subtraction, multiplication and division, using the notation `+`, `-`, `*` and `/`; and also the unary version of `-`) and the functions `cos`, `sin`, `cosh`, `sinh` and `sqrt` (i.e. $\sqrt{}$) have been implemented for automatic differentiation thus far. That is, only these operations have been overloaded to handle arguments with the types `u` and `v` are assigned in the different parts of the program where `metric.cc` is included. This means that no other function may be applied to expressions containing `u` or `v`. For example, the input line

```
x=log(u^2);
```

will simply produce compiler error messages, due to the appearance of a C++ mathematical function which has not been overloaded for Loki (`log`) and also an operator which has not been overloaded (`^` is the logical bitwise symmetric difference operator in C++).

One is however free to use any C++ standard mathematical function on constants appearing in `metric.cc`. For example,

```
x=u*log(1.2);
```

This also applies to the use of the special variable named `parameter`. This variable acts like a constant. Its value is set at runtime in the runtime input file. This allows one to compile only once for a family of related surfaces. For example:

```

x=(parameter+cos(u))*cos(v);
y=(parameter+cos(u))*sin(v);
z=sin(u);

```

```
BUMP(1.57, log(parameter+1.0), 0.2, 0.2, -1.0);
```

C++ syntax has one disadvantage in the context of Loki which cannot be avoided: There is a difference between integer and “real” (`double`) constants which influences the arithmetic used. For example, `1/2` is understood to be an integer division because both operands are integers. The result of an integer division is itself an integer, and `1/2` is rounded by the C++ compiler to zero before Loki is executed. For example, the file

```

x=(1.5+1/2+cos(u))*cos(v);
y=(1.5+1/2+cos(u))*sin(v);
z=sin(u);

```

does not define the standard torus given by Equation (1–1).

If however “one half” is entered using “real numbers”, as in

```
x=(1.5+1.0/2.0*cos(u))*cos(v);
y=(1.5+1.0/2.0*cos(u))*sin(v);
z=sin(u);
```

then the standard torus given by Equation (1–1) is indeed defined, because 1.0 and 2.0 are “real numbers” according to C++ syntax, so a “real” division is performed.

Since Loki does not necessarily have access to its own source code, there is no sensible way of detecting such problems, but the arguments of the `BUMP` macro are checked carefully. A fatal compilation error results if any of them is an integer rather than a “real number” (e.g. 0 instead of 0.0).

One may add several “bumps” to a surface, but these will all make use of a surface normal to the original surface (i.e. the one defined by the `x=...` statements).

Loki does output what it has understood its input to be, so it is always possible to double-check, simply by comparing Loki’s output with one’s own input. Loki also outputs a `Maple` [Heck, 96] version of the surface, and this also provides one with a possibility to check.

3.2 Syntax for the Runtime Input File

A sample runtime input file is to be found in Figure 23.

The runtime input file is read by Loki, and therefore does not follow C++ syntax. Loki parses it in a rather primitive way. As was the case with `metric.cc`, one is able to check that one’s input has been understood by looking at output files.

The rules used by Loki to read from standard input are:

1. If a line contains the substring `atrix`, assume that the user wants to compute a distance matrix.
2. If a line begins (ignoring spaces or tabs) with a “-”, “+”, “.” or a digit, assume that the line contains (u, v) coordinates of a point defining the distance matrix.
3. If a line contains the substring `tol` and an equals sign (=), assume that a single real number follows the equals sign, and that this number defines the input tolerance.
4. If a line contains the substring `max` and an equals sign, assume that a single real number follows the

equals sign, and that this number defines the maximum distance to be used in approximating the exponential map.

5. If a line contains the substring `par` and an equals sign, assume that a single real number follows the equals sign, and that this number defines the value of the variable in `metric.cc` named `parameter`.
6. If a line contains the substring `cop` and an equals sign, assume that a single integer follows the equals sign, and that this number defines the variable `copies` (which was discussed in Section 2.2).
7. If a line contains the substring `che` and an equals sign, assume that a single integer follows the equals sign, and that this number defines the variable `checks` (which was discussed in Section 2.3).
8. If a line contains one of the characters `u` or `v` and an equals sign, assume that a single real number follows the equals sign, and that this number defines either u_P or v_P , respectively.

These rules are applied in the order given, the first match determining how any line is understood. Note that these rules do not allow for any actual computation while parsing. For example, the line

```
tolerance = 1e-3 + 25.6
```

is understood as if it read

```
tolerance = 1e-3
```

(setting the input tolerance to be 10^{-3}).

All of the variables and constants mentioned above have default values which are used if they are not defined in the runtime input file.

3.3 Output (to Standard Error `cerr`)

If Loki is run using the UNIX command `batch`, then this output will usually appear in an E-mail informing the user that the job is finished. The idea is to give a short summary of what was computed and how long it took. This information may be of use in keeping track of exactly which job produced which output file. Also, if Loki should abort, information concerning the reason for this will be output here.

Here is the output corresponding to `metric.cc` as defined in Figure 24 and the runtime input file of Figure 23:

```
#####
##          LOKI          ##
#####
tolerance=0.001
parameter=0
u_0=-0.5
v_0=1
t_max=7.5
copies=1
checks=4
computing from (x,y,z)
size=5458
error=1.29506829232
processor time used on
    exponential map=256.423076 seconds
total processor time used=7198.078732 seconds
#####
```

3.4 Output (to Standard Output cout)

This file contains the actual output data defining the cut locus. The file contains enough information to allow a complete reconstruction of the runtime input file and the file `metric.cc`. This is to prevent one from inadvertently losing track of which output file corresponds to which surface and starting point etc. One can literally copy the sections of this output file beginning with `input file` and `metric.cc` into the runtime input file and `metric.cc` respectively and then (after removing the leading #s) recompute.

This file can be read by Maple. It defines a function called `trans`, which maps from (u, v) coordinates to \mathbb{R}^3 (if the surface has been defined in terms of (x, y, z)), the error estimate and time to compute the approximation to the exponential map, a list of angles to be explained below, and then the actual list named `l` which defines the cut locus. `l` consists of sextuples of numerical values. The first two entries are the (s, t) coordinates of a point of the cut locus. The next two are the (u, v) coordinates of the same point. The final two entries are usually the (s, t) coordinates of another geodesic of same minimal length, which can help understand the structure of the cut locus. For numerical reasons, these final two entries may sometimes simply be a copy of the first two. This is often the case near a conjugate point, where the algorithm can have difficulty identifying which two geodesics are intersecting at the cut point.

The list of angles output while Loki is computing indicates how the computation is progressing. Since a single cut locus computation can take several days, it is useful to have this practical information.

Here is the output corresponding to `metric.cc` as defined in Figure 24 and the runtime input file of Figure 23.

Lines have been truncated here. The actual output file of course contains full lines. Also, vertical dots indicate where lines have been deleted to save paper.

```
#####
##          LOKI          ##
#####
##### input file #####
# tolerance=0.001
# parameter=0.0
# u_0=-0.5
# v_0=1.0
# t_max=7.5
# copies=1
# checks=4
##### metric.cc #####
# x=(cos(u)+2.0)*(cos(v));
# y=(cos(u)+2.0)*(sin(v));
# z=sin(u);
# BUMP(1.5700000000000001,0.0,0.2000000000000001,
#####
trans:=proc(u,v)
[(cos(u)+2.0)*(cos(v))+((-0.0004000000000000013)/
(cos(u)+2.0)*(sin(v))+((-0.0004000000000000013)/
sin(u))+((-0.0004000000000000013)/((1.02-(cos(u)-(1.
end:
#####
# size=5458
# error=1.29506829232
# processor time used on exponential map=256.423076
##### computing #####
# -3.09159265359
# -3.04159265359
# -2.99159265359
.
:
#####
l:=[]
[-3.14159265359,5.60718139896,0.519178903231,-0.866
[-3.09159265359,6.12407684326,0.503320818319,-1.045
[-3.04159265359,6.11148834229,0.513546541913,-1.041
.
:
[3.14159265359,5.59918803018,0.519406330002,-0.8632
#####
# total processor time used=7198.078732 seconds
#####
```

3.5 Performance

Table 1 gives some idea of the performance of Loki for various input tolerances. In every case, the surface has been defined by `metric.cc` as in Figure 24 and the runtime input file of Figure 23 (with the various different tolerances).

Roughly speaking, for a reduction in input tolerance by a factor of ten, one can expect a doubling of memory

usage and a quadrupling of computing time. At the same time, the estimated error of the output drops by approximately a factor of ten (but with large fluctuations in this factor).

Some readers may be alarmed at the long computing times recorded in Table 1. First, it must be said that a “reasonable” time will always depend upon the context in which the program is used. Most mathematics departments have idle computing power. Most mathematicians have enough to keep them busy that they can wait a few days for a new cut locus. Loki has been written with robustness as the highest priority, and speed as a secondary consideration. If one were to find that a research project actually needs a large number of cut loci to be computed in a short time, then the algorithm would have to be improved. The first author does not believe that programs should be optimized unless there is a demonstrated need for it. We must remember that the purpose of this software package is primarily mathematical, and it was never intended as a contribution to high speed computing.

An obvious improvement to the inversion algorithm would be the use of a quadtree (a data structure which naturally represents objects distributed in space, allowing rapid access to the neighbours of any given object — see [Warren and Salmon, 92] for an excellent description of how this idea is used in astrophysical simulations) to group neighbouring (in terms of (u, v) coordinates) patches together. One can expect that this would significantly improve the efficiency of the inversion process, since one would be able to locate relevant patches without searching the entire data structure. The fact that the gaps/overlaps between patches are not stored, but recomputed each time the exponential map is inverted, would however make this a nontrivial modification.

4. KOKKENDORFF’S CONJECTURE

The point of this section is to illustrate how Loki has already contributed in a small way to research in global differential geometry, by providing inspiration for a new hypothesis.

Simon Kokkendorff, a Ph.D. student of Professor Markvorsen at the Department of Mathematics of the Technical University of Denmark, has used Loki to study the cut loci of surfaces similar to the one shown in Figure 1. This experimentation led to the following, very preliminary conjecture:

Given a smooth (or analytic) surface, the graph of the distance to cut points of a point as a

function of initial angle from the starting point is smooth at local minima, but not differentiable at local maxima.

Clearly this was only the first step in formulating a precise hypothesis.

The motivation for the conjecture was plots such as Figure 5, although one might be unsure as to what is happening for $s \approx 1.8$, where it is not quite clear whether the local minimum is smooth. Figure 28 is a close-up of the relevant portion, and shows that the graph is indeed smooth at the local minimum, although quite some computation was required to show it (the crosses correspond to 9 days’ computing time — see Table 1). Note that the curve joining the two vertices of Figure 9 could also have been examined more closely.

The conjecture was then studied from a purely theoretical point of view by the second author, leading to a proof of the first part (concerning minima) and a counterexample to the second (concerning maxima). These are presented in the next Section. Note that the proof of the first part given here is for the more general situation of the cut locus of a closed submanifold in a surface.

It is hoped that Loki will lead to more such conjectures. This is the very essence of experimental mathematics.

4.1 A Counterexample and a Proof to Kokkendorff’s Conjecture

Let \mathcal{C} be a closed submanifold of a complete 2-dimensional Riemannian manifold M (cf. [Chavel, 93] or [Sakai, 92] for Riemannian geometry). A unit speed geodesic segment $\gamma : [0, a] \rightarrow M$ emanating from \mathcal{C} is called a \mathcal{C} -segment if $t = d(\mathcal{C}, \gamma(t))$ on $[0, a]$. Here $d(\cdot, \cdot)$ denotes the Riemannian distance function of M . Let $\pi : N_1\mathcal{C} \rightarrow \mathcal{C}$ be the unit normal bundle over \mathcal{C} . We define a function $\rho(v)$, $v \in N_1\mathcal{C}$, which is called the *distance function to the cut locus* of \mathcal{C} , by

$$\rho(v) := \sup\{ t > 0 \mid \gamma|_{[0,t]} \text{ is a } \mathcal{C}\text{-segment} \}.$$

Each point $\exp(\rho(v)v)$ with $\rho(v) < \infty$ is called a *cut point* of \mathcal{C} and the set of all cut points is called the *cut locus* of \mathcal{C} . Here \exp denotes the exponential map on the tangent bundle TM of M . For each unit speed curve $v(\tau)$, $\tau \in (-\delta, \delta)$, on $N_1\mathcal{C}$

$$Y_{v(0)}(t) := \left. \frac{\partial}{\partial \tau} \right|_{\tau=0} \exp(tv(\tau))$$

is called a \mathcal{C} -Jacobi field along $\gamma_{v(0)}(t) := \exp(tv(0))$. We define a function $\lambda_1(v)$ on $N_1\mathcal{C}$ by

$$\lambda_1(v) := \inf\{ t > 0 \mid Y(t) = 0 \},$$

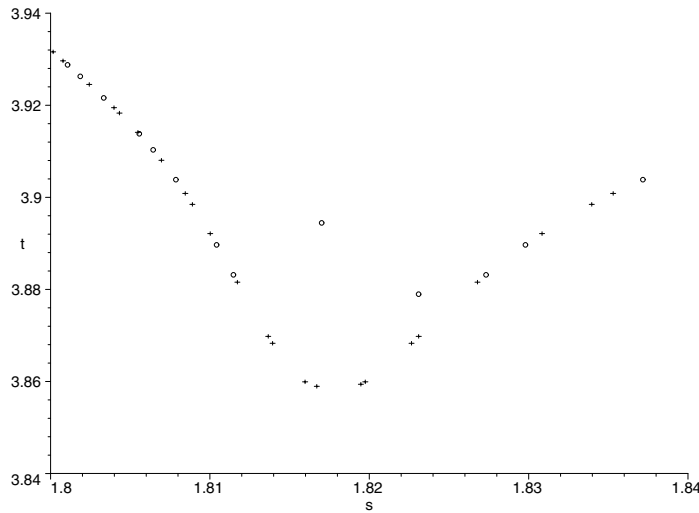


FIGURE 28. A close-up of Figure 5. The crosses correspond to the finer input tolerance. This illustration supports the conjecture that local minima of the cut locus in (s, t) coordinates are smooth.

where $Y(t)$ denotes a non-zero \mathcal{C} -Jacobi field along $\gamma_v(t)$. It is well known that λ_1 is smooth on $\{v \in N_1\mathcal{C} \mid \lambda_1(v) < \infty\}$ and $\lambda_1 \geq \rho$ on $N_1\mathcal{C}$ (see [Hartman, 64]).

Theorem 4.1. *Let \mathcal{C} be a closed submanifold of a complete 2-dimensional Riemannian manifold M . Then the distance function ρ to the cut locus of \mathcal{C} is differentiable where ρ attains a local minimum.*

Proof: Suppose ρ attains a local minimum at v_0 . Choose a positive number r which is less than the injectivity radius at $p := \exp(\rho(v_0)v_0)$. Suppose that $\lambda_1(v_0) > \rho(v_0)$. Then $\gamma_{v_0}|_{[0, \rho(v_0)]}$ is an isolated \mathcal{C} -segment. Let Σ^+, Σ^- denote the two distinct connected components of $B(p, r) \setminus \{\gamma[\rho(v_0) - r, \rho(v_0)] \mid \gamma \text{ is an } \mathcal{C}\text{-segment}\}$ whose boundaries contain $\{\exp(tv_0) \mid \rho(v_0) - r \leq t \leq \rho(v_0)\}$. Here, $B(p, r)$ denotes the open ball centered at p with radius r . Let $v : (-\delta, \delta) \rightarrow N_1\mathcal{C}$ denote a unit speed geodesic with $v(0) = v_0$. Since either $v(0, \delta_0) \subset \Sigma^+$ or $v(0, \delta_0) \subset \Sigma^-$ for a sufficiently small positive number δ_0 , we may assume $v(0, \delta_0) \subset \Sigma^+$. Hence $v(-\delta_0, 0) \subset \Sigma^-$. Let $Y(t)$ denote the Jacobi field along γ_{v_0} with $Y(0) = 0$, $Y'(0) = v'(0)$, where $Y'(t)$ denotes the covariant derivative of $Y(t)$ along $\gamma_{v_0}(t)$. It follows from Proposition 2.2 in [Itoh and Tanaka, 01] that

$$\lim_{t \rightarrow +0} \frac{\rho(v(t)) - \rho(v(0))}{t} = -\|Y(\rho(v(0)))\| \cot \theta^+ \quad (4-1)$$

and

$$\lim_{t \rightarrow -0} \frac{\rho(v(t)) - \rho(v(0))}{t} = \|Y(\rho(v(0)))\| \cot \theta^-, \quad (4-2)$$

where θ^+ and θ^- denote the inner angle at p of Σ^+, Σ^- respectively. Since ρ attains a local minimum at v_0 ,

$$\lim_{t \rightarrow +0} \frac{\rho(v(t)) - \rho(v(0))}{t} \geq 0$$

and

$$\lim_{t \rightarrow -0} \frac{\rho(v(t)) - \rho(v(0))}{t} \leq 0.$$

Therefore, by (4-1) and (4-2), we get $2\theta^+ \geq \pi$ and $2\theta^- \geq \pi$. Since $2\theta^+ + 2\theta^- \leq 2\pi$, we have $\theta^+ = \theta^- = \pi$, i.e.

$$\lim_{t \rightarrow +0} \frac{\rho(v(t)) - \rho(v(0))}{t} = \lim_{t \rightarrow -0} \frac{\rho(v(t)) - \rho(v(0))}{t} = 0.$$

This implies that ρ is differentiable at v_0 if $\lambda_1(v_0) > \rho(v_0)$. If $\lambda_1(v_0) = \rho(v_0)$, then $d\lambda_1$ is zero at v_0 (see Lemma 1 in [Itoh and Tanaka, 98] or Proposition 5.4 in [Hartman, 64]). Since $\lambda_1 \geq \rho$ on $N_1\mathcal{C}$ and ρ attains a local minimum at v_0 ,

$$\begin{aligned} 0 &\geq \lim_{t \rightarrow -0} \frac{\rho(v(t)) - \rho(v(0))}{t} \geq \lim_{t \rightarrow -0} \frac{\lambda_1(v(t)) - \lambda_1(v(0))}{t} \\ &= \lim_{t \rightarrow +0} \frac{\lambda_1(v(t)) - \lambda_1(v(0))}{t} \geq \lim_{t \rightarrow +0} \frac{\rho(v(t)) - \rho(v(0))}{t} \geq 0. \end{aligned}$$

Hence ρ is differentiable at v_0 , if $\rho(v_0) = \lambda_1(v_0)$. \square

Remark 4.2. Let \mathcal{C}_ϵ denote a circle centred at a point p in the manifold M with radius ϵ and ρ_ϵ the distance function to the cut locus of \mathcal{C}_ϵ . Then for any sufficiently small ϵ , the cut locus of \mathcal{C}_ϵ equals the one of the point p and

$\rho_\epsilon \circ (d \exp_p)_{ev}(v) = \rho_p(v) - \epsilon$ holds for any tangent vector v at p , where \exp_p denotes the exponential map on the tangent space of M and ρ_p denotes the distance function to the cut locus of p . Therefore ρ_p is differentiable where ρ_p attains a local minimum.

Let S be an ellipsoid in the 3-dimensional Euclidean space defined by the equation

$$x^2 + y^2 + \frac{z^2}{c^2} = 1,$$

where c is a constant greater than 1. The Gaussian curvature $K(x, y, z)$ of S at a point $(x, y, z) \in S$ is given by

$$K(x, y, z) = \frac{c^6}{(c^4 + z^2(1 - c^2))^2}.$$

In particular, $K(\gamma(t)) = \frac{1}{c^2}$, where $\gamma(t) := (\cos t, \sin t, 0)$. The parallel γ is a geodesic on S . Hence the first conjugate point of $p := \gamma(0)$ along γ is $\gamma(\pi c)$. Therefore the geodesic segment $\gamma|_{[0, \pi]}$ has no conjugate point of p along γ .

Theorem 4.3. *Let S be an ellipsoid in the 3-dimensional Euclidean space defined by the equation*

$$x^2 + y^2 + \frac{z^2}{c^2} = 1,$$

where $c > 1$. The cut locus C_p of p is $\tau[-b, b]$ for some $b \in (0, d(p, N))$, where $N := (0, 0, c)$ and τ is the unit speed geodesic emanating from $(-1, 0, 0)$ with $\tau'(0) = (0, 0, 1)$. Hence the distance function ρ to the cut locus C_p is differentiable everywhere.

Proof: It is known that the cut locus C_p is a tree (cf. [Hebda, 94] or [Shiohama and Tanaka, 96]). Suppose that there exists a cut point (x, y, z) of p with $y \neq 0$. This implies that the cut locus has an endpoint (x_0, y_0, z_0) with $y_0 \neq 0$. Hence there exists a cut point (x_1, y_1, z_1) of p with $y_1 \neq 0$ and two minimizing geodesics α, β joining p to (x_1, y_1, z_1) such that these geodesics bound a disk domain D containing (x_0, y_0, z_0) . Since S is symmetric with respect to the xy -plane, we may assume that the two geodesics lie in $\{(x, y, z) \in S \mid z \leq 0\}$. Since the Gaussian curvature $K(x, y, z)$ is a monotone decreasing function in z on D , we get a contradiction by imitating the proof of Theorem 1.4 in [Tanaka, 92]. Hence the cut locus is a subset of $\tau[-d(p, N), d(p, N)]$. Since the point N is not a cut point of p , the cut locus of p is $\tau[-b, b]$ for some $b \in (0, d(p, N))$. Since $K(x, y, z)$ is

a strictly monotone decreasing function in $z \in [-c, 0]$, $\tau|_{[-2d(p, N), -d(p, N)]}$ is the unique minimizing geodesic joining p to N (see Proposition 1.5 in [Tanaka, 92]). This implies that ρ attains a local minimum at the tangent vectors $(\pm 1, 0, 0)$ at p . Hence the latter claim is clear from Theorem 1 and Proposition 2.2 in [Itoh and Tanaka, 01]. Note that this proposition is applicable for conjugate cut points, since the cut locus has no branch points. \square

5. CONCLUSION AND FURTHER WORK

The software package Loki is able to compute the cut loci of 2 dimensional torus-like surfaces, which may be defined via a parametrization of an embedding in \mathbb{R}^3 , or via an explicit metric. An input tolerance influences the error of the output data. In general, very high accuracy is achievable.

Loki has been designed with robustness and ease of use in mind. Once one has discovered which command one must use to compile for a new surface, it is in fact quite a simple software package to use. Usually, one can expect to have to spend some time transforming the output data into an illustration of some sort using Maple. This is also typically quite simple if one has some experience in using Maple.

The code is freely available. Interested parties should contact R. Sinclair.

It is clear that Loki should be able to be extended to also compute conjugate loci, and this is being looked into.

Loki could be extended to treat surfaces of genus greater than one by replacing the regular square lattice used to identify points on a torus (see Figure 13) with other regular lattices, and then identifying points in various manners. This would require a set of basis functions with the same symmetry, and that is not entirely trivial, but it would appear that such changes could be made with minimal alterations to the code.

One would like to study surfaces which are topologically equivalent to the surface of a sphere embedded in \mathbb{R}^3 . It is not immediately clear how to do this, since a change of chart may be necessary in all but the simplest cases. This would require a significant amount of new code allowing the algorithm to change from one chart to another.

We have been able to present a proof of the first part of what we call Kokkendorff's conjecture. It would be interesting to see if the second part of the conjecture can be saved by reformulation, taking into account the existence of the counterexamples presented here. In any case, it is

hoped that Loki will stimulate further analytical work on cut loci.

ACKNOWLEDGEMENTS

S. Kokkendorff was of great assistance in the final stages of testing of the software package, by volunteering to use it and thereby discovering errors which had not already been found. R. Sinclair was supported in the initial software development stage of this work by a DANVIS grant from the Danish Research Academy, and for his stay in Japan by a grant from Otto Mønstedts Fond.

REFERENCES

- [Aurenhammer, 1991] F. Aurenhammer. “Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure.” *ACM Computing Surveys*. **23**:3 (1991), 345–405.
- [Barth and Sethian, 98] T.J. Barth and J.A. Sethian. “Numerical Schemes for the Hamilton-Jacobi and Level Set Equations on Triangulated Domains.” *J. Comp. Physics*. **145** (1998), 1–40.
- [Berger and Gostiaux, 88] M. Berger and B. Gostiaux. *Differential Geometry: Manifolds, Curves, and Surfaces*, Springer-Verlag, Berlin, 1988.
- [Berger, 00] M. Berger. *Riemannian geometry during the second half of the twentieth century*, University Lecture Series, Vol. 17, AMS, Providence, Rhode Island, 2000.
- [Bishop, 77] R.L. Bishop. “Decomposition of cut loci.” *Proc. of the AMS*. **65** (1977), 133–136.
- [Bleecker, 81] D.D. Bleecker. “Cut loci of closed surfaces without conjugate points.” *Colloquium Mathematicum*. **44** (1981), 263–276.
- [Buchner, 78] M.A. Buchner. “The structure of the cut locus in dimension less than or equal to six.” *Compositio Mathematica*. **37** (1978), 103–119.
- [Chavel, 93] I. Chavel. *Riemannian Geometry : A Modern Introduction*, Cambridge University Press, Cambridge, UK, 1993.
- [do Carmo, 76] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*, Prentice-Hall International, London, 1976.
- [do Carmo, 92] M.P. do Carmo. *Riemannian Geometry*, Birkhäuser, Boston, 1992.
- [Degen, 97] W.L.F. Degen. “The Cut Locus of an Ellipsoid.” *Geometriae Dedicata*. **67**, (1997) 197–198.
- [Gluck and Singer, 78] H. Gluck and D.A. Singer. “Scattering of geodesic fields. I.” *Ann. of Math., II. Ser.* **108** (1978), 347–372.
- [Gluck and Singer, 79] H. Gluck and D.A. Singer. “Scattering of geodesic fields. II.” *Ann. of Math., II. Ser.* **110** (1979), 205–225.
- [Hartman, 64] P. Hartman. “Geodesic parallel coordinates in the large.” *Amer. J. Math.* **86** (1964), 705–727.
- [Hebda, 94] J.J. Hebda. “Metric structure of cut loci in surfaces and Ambrose’s problem.” *J. Diff. Geometry*. **40** (1994), 621–642.
- [Heck, 96] A. Heck. *Introduction to Maple*, Second Edition, Springer-Verlag, New York, 1996.
- [IEEE, 85] Institute of Electrical and Electronics Engineers Inc. *IEEE Standard for Binary Floating Point Arithmetic*, IEEE, New York, 1985.
- [Itoh and Tanaka, 98] J. Itoh and M. Tanaka. “The Hausdorff dimension of a cut locus on a smooth Riemannian manifold.” *Tôhoku Math. J.* **50** (1998), 571–575.
- [Itoh and Tanaka, 01] J. Itoh and M. Tanaka. “The Lipschitz continuity of the distance function to the cut locus.”, *Trans. Amer. Math. Soc.* **353**:1 (2001), 21–40.
- [Kimmel and Sethian, 98] R. Kimmel and J.A. Sethian. “Computing geodesic paths on manifolds.” *Proc. of the Natl. Acad. Sciences USA*. **95** (1998), 8431–8435.
- [Kimmel and Sethian, 99] R. Kimmel and J.A. Sethian. “Fast Voronoi Diagrams and Offsets on Triangulated Surfaces.” In *Proceedings of AFA Conference on Curves and Surfaces*, Saint-Malo, France, July, 1999.
- [Klingenberg, 82] W. Klingenberg. *Riemannian Geometry*, de Gruyter Studies in Mathematics 1, Walter de Gruyter, Berlin, 1982.
- [Kunze, et al. 97] R. Kunze, F.-E. Wolter and T. Rausch. “Geodesic Voronoi Diagrams on Parametric Surfaces.” In *Proceedings of CGI '97*, IEEE, Computer Society Press Conference Proceedings, pp. 230–237, IEEE, 1997.
- [Lawall, 98] J. Lawall. “Faster Fourier Transforms via Automatic Program Specialization.” In *Partial Evaluation: Practice and Theory*, J. Hatcliff, T. Æ. Mogensen and P. Thiemann (eds.), Lecture Notes in Computer Science 1706, pp. 338–355, Springer-Verlag, Berlin, 1998.
- [Maekawa, 96] T. Maekawa. “Computation of Shortest Paths on Free-Form Parametric Surfaces.” *J. of Mech. Design*. **118** (1996). 499–508.
- [Myers, 35] S.B. Myers. “Connections between Differential Geometry and Topology: I. Simply Connected Surfaces.” *Duke Math. J.* **1** (1935), 376–391.
- [Myers, 36] S.B. Myers. “Connections between Differential Geometry and Topology: II. Closed Surfaces.” *Duke Math. J.* **2** (1936), 95–102.
- [Ozols, 74] V. Ozols. “Cut loci in Riemannian manifolds.” *Tohoku Mathematical Journal, II. Ser.* **26** (1974), 219–227.
- [Rall, 81] L.B. Rall, *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Computer Science 120, Springer-Verlag, Berlin, 1981.

- [Rausch et al. 97] T. Rausch, F.-E. Wolter and O. Sniehotta. *Computation of Medial Curves on Surfaces*, in Proceedings of Conference on the Mathematics of Surfaces VII, IMA Conference Series, pp. 43–68, 1997.
- [Rebel, 95] J. Rebel. *Tower of Babylon*, Master of Science Thesis, Mathematical Institute, Technical University of Denmark, 1995.
- [Sakai, 92] T. Sakai. *Riemannian Geometry, Translation of Mathematical Monographs*, Vol. 149, Amer. Math. Soc., 1992.
- [Sethian, 99] J.A. Sethian. *Level Set Methods and Fast Marching Methods*, Cambridge Monograph on Applied and Computational Mathematics, Cambridge University Press, 1999.
- [Shiohama and Tanaka, 96] K. Shiohama and M. Tanaka. *Cut loci and distance spheres on Alexandrov surfaces*, Séminaires & Congrès, In *Collection SMF No.1, Actes de la table ronde de Géométrie différentielle en l'honneur Marcel Berger*, pp. 531–560, 1996.
- [Stroustrup, 97] B. Stroustrup. *The C++ Programming Language*, 3rd. edition, Addison-Wesley, Reading, Massachusetts, 1997.
- [Tanaka, 92] M. Tanaka. “On the cut loci of a von Mangoldt’s surface of revolution.” *J. Math. Soc. Japan.* **44** (1992), 631–641.
- [Taubin, 94] G. Taubin. “Rasterizing Algebraic Curves and Surfaces.” *IEEE Computer Graphics and Applications.* **14** (1994), 14–23.
- [Tsuji, 97] Y. Tsuji. “On a cut locus of a complete Riemannian manifold homeomorphic to cylinder.” *Proceedings of the School of Science of Tokai University.* **32** (1997), 23–34.
- [Warren and Salmon, 92] M.S. Warren and J.K. Salmon. “Astrophysical N-Body Simulations Using Hierarchical Tree Data Structures.” in *Supercomputing '92*, pp. 570–576, IEEE Computer Society, Los Alamitos, 1992.

Robert Sinclair, Department of Mathematics & Statistics, The University of Melbourne, Parkville, Victoria 3010, Australia
(R.Sinclair@ms.unimelb.edu.au)

Minoru Tanaka, Department of Mathematics, Tokai University, Hiratsuka City, Kanagawa Pref. 259-1292, Japan
(m-tanaka@sm.u-tokai.ac.jp)

Received December 17, 2000; accepted in revised form May 30, 2001.