Certified Algorithms for Implicit Surfaces

Simon Plantinga

Contents

1 Introduction **1**
- 1.1 Implicit surfaces . . . 1
  - 1.1.1 Modeling . . . 2
  - 1.1.2 Radial basis functions . . . 4
- 1.2 Contour generators . . . 4
- 1.3 Meshing . . . 6
  - 1.3.1 Continuation methods . . . 6
  - 1.3.2 Adaptive enumeration techniques . . . 6
- 1.4 Surface reconstruction . . . 8
- 1.5 Interval arithmetic . . . 8
  - 1.5.1 Interval analysis . . . 9
  - 1.5.2 Interval Newton Method . . . 9
  - 1.5.3 Affine arithmetic . . . 10
- 1.6 Implicit function theorem . . . 10
- 1.7 Morse theory . . . 11
- 1.8 Overview and main results . . . 12
  - 1.8.1 Publications . . . 13

2 Contour generators **15**
- 2.1 Introduction . . . 15
- 2.2 Contour generator and apparent contour . . . 17
  - 2.2.1 Contour generators of implicit surfaces . . . 17
  - 2.2.2 Singular points of contour generators . . . 19
  - 2.2.3 Generic projections: fold and cusp points . . . 20
- 2.3 Evolving contours . . . 22
- 2.4 Transformations and normal forms . . . 25
  - 2.4.1 Example: local model at a truncated cusp point . . . 26
- 2.5 Deriving local models . . . 27

2.5.1 Local model at a fold point 28
2.5.2 Local model at a cusp point 29
2.6 Time dependent contours 30
2.6.1 Lips and beak-to-beak bifurcations 31
2.6.2 The swallowtail bifurcation 34

3 Contour tracing **39**
3.1 Introduction 39
3.2 Finding initial points 40
3.3 Tracing step 40
3.3.1 Interval test 41
3.4 Evolving surfaces 44
3.5 Results 46
3.6 Conclusion and future research 47

4 Isotopic Approximation **51**
4.1 Introduction 51
4.2 Related work 52
4.3 Curves 54
4.4 Marching cubes revisited 59
4.5 Octrees 64
4.6 Results 69
4.7 Conclusion 69

5 Level sets of implicit functions **75**
5.1 Introduction 75
5.2 Isotopic meshing with tetrahedra 76
5.3 Level sets 79
5.3.1 Algorithm 79
5.4 Results 81
5.5 Conclusion and future work 82

A Singularities of functions on surfaces **89**
A.1 Non-degenerate singular points 89

B The Morse Lemma with parameters **93**

C Normal form of a Whitney fold **95**
C.1 Fold points of good mappings 95
C.2 Normal form of a good map near a fold point 98
C.3 Introducing parameters 101

Bibliography **103**

Chapter 1

Introduction

1.1 Implicit surfaces

Three-dimensional objects are extensively used in computer graphics applications, such as computer aided design, animation, games and numerous other areas. These objects are often represented by triangle meshes. This is ideal for objects with flat parts, such as buildings. Organic shapes however, have to be built up with a large number of tiny triangles to approximate the curved surface, like geodesic domes approximating a sphere. For curved surfaces, an implicit representation is a convenient alternative. In animation, ray-traced implicit surfaces are very useful, for example in modeling water or deformable object [58, 26, 27]. Also, in computer aided geometric design, implicit curves and surfaces have several advantages over the more widely used parametric representations [39].

Implicit surfaces are defined as the zero set of a function $\mathsf{F}\colon \mathbb{R}^3 \to \mathbb{R}$. See Figure 1.1. In this thesis we assume F is a smooth function defining a regular two-manifold, that is, the gradient does not disappear on the surface (see also Section 1.6).

For visualization of implicit surfaces raytracing can be used to create high-quality images. Several techniques have been suggested to improve raytracing computations for implicit functions [33, 34, 25]. However, for interactive visualization, raytracing is usually too slow. Approximating the implicit surface with a conventional mesh allows fast rendering, taking full advantage of the hardware of graphics cards. This meshing of implicit surfaces is one of the major topics of this thesis.

Accuracy is an important but often neglected issue in scientific visualization. Lopes [47] shows that inaccuracy in isosurfacing usually is due to incorrect topology. Several algorithms for implicit surfaces try to improve accuracy by taking certain topological issues into consideration, for example

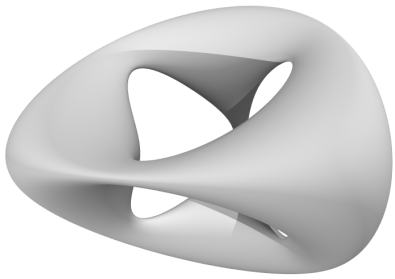


Figure 1.1: Implicit surface of the 'chair' function (Section 5.4).

by reconstructing the topology of a function 'close to' the implicit function itself. An example is to use the trilinear interpolation of the function values at a regular grid. Algorithms constructing meshes (regularly) isotopic to the implicit surface itself have only shown up recently. Regular isotopy means that the mesh can be continuously deformed into the surface without leaving the embedded space, and is therefore an even stronger result than correct topology. However, we should note that for general implicit surfaces, it is difficult if not impossible to come up with a proof of correct topology without guaranteeing isotopy at the same time.

Regarding visualization, we should also mention the use of particle systems. Small disks are distributed over the surface by using repelling forces between these disks, together with an attracting force towards the surface. The resulting collection of disks gives a good indication of the shape of the surface, without reconstructing its structure [71].

1.1.1 Modeling

One way to model implicit surfaces, originating from molecular modeling, is the use of blobs or metaballs (Figure 1.2). A single blob consists of a Gaussian density function around a point, the corresponding isosurface being a sphere. When using multiple blobs, the density functions interact with eachother, giving a smooth transition between the different balls. By varying the strenght and decay of the density functions, the size of the blobs and the degree of interaction can be manipulated. Such a blobby surface is implicitly given by summing the individual density functions:

$$F(x, y, z) = \sum_k b_k e^{-a_k r_k^2} - T.$$

Here, r_k represents the distance from (x, y, z) to the centre of blob k. The parameters a_k and b_k determine the details of the individual blobs. In particular, negative values of b_k can be used to produce dents instead of bumps. The parameter T is a fixed threshold determining the particular iso-value.

In computer graphics, the Gaussian blobs are usually replaced by metaballs or by soft objects, which are polynomial functions with compact support. These metaballs provide a kind of digital clay to sculpt models by smoothly adding or removing claylike pieces to the model.

For a metaball with radius d, the density function at distance r is given by:

$$f(r) = \begin{cases} b\left(1 - \frac{3r^2}{d^2}\right), & \text{if } 0 < r \leq \frac{d}{3} \\ \frac{3}{2}b\left(1 - \frac{r}{d}\right)^2, & \text{if } \frac{d}{3} < r \leq d \\ 0, & \text{if } r > d \end{cases}$$

For soft objects, this density function is:

$$f(r) = \begin{cases} 1 - \frac{22r^2}{9d^2} + \frac{17r^4}{9d^4} - \frac{4r^6}{9d^6}, & \text{if } 0 < r \leq d \\ 0, & \text{if } r > d \end{cases}$$

As with the blob model, the implicit function consists of a weighted sum of these density functions.

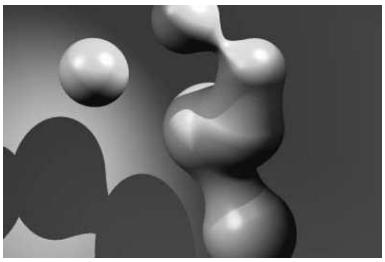


Figure 1.2: Metaball objects.

Apart from this clay-like approach, constructive solid geometry (CSG) can also be applied to implicit surfaces. By performing simple arithmetic operations on two functions, boolean operations on multiple implicit surfaces can be applied. For example, taking the maximum of two functions results in the union of two objects. However, these CSG objects are not smooth.

Implicit objects can also be blended together with a smooth transition, for example by using convolution surfaces [11].

1.1.2 Radial basis functions

Another approach to modeling with implicit surfaces is the use of radial basis functions. At certain *control points* the function value is specified. A radial basis function can be constructed interpolating this data. This type of function has been used extensively to interpolate higher dimensional data. For modeling purposes, points on the surface are defined having function value 0. Outside the surface, points with non-zero value are specified. Computing a radial basis function for this data set results in an implicit function, whose zero-set passes through the given surface points [74, 30, 68, 19, 45].

In the RBF approach, the implicit function consists of a weighted sum of functions depending only on the distance to the specified points, hence the name 'radial':

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(|\mathbf{x} - \mathbf{c}_i|) + P(\mathbf{x})$$

In this formula the $\mathbf{c_i}$ are the specified control points. For the radial function ϕ the thin-plate spline is a convenient choice. For three dimensionsal approximation, this function is given by $\phi(r) = r^3$, in 2D $\phi(r) = r^2 \log r$ is used. The thin-plate spline minimizes the total curvature, and is therefore expected to result in a smooth surface through the surface points. The term $P(\mathbf{x})$ is a low-degree polynomial giving some extra freedom in choosing F, without changing the minimal total curvature. Substituting the control points, together with their required values, results in a system of linear equations. From these, the weights w_i can be solved.

An approach similar to the use of radial basis functions, is Poisson surface reconstruction [43]. The *indicator function* is defined as 1 inside the surface, and 0 outside. The corresponding indicator gradient is zero almost everywhere, and corresponds to the surface normal at points on or near the surface. The indicator function can now be found by inverting the gradient operator. Transforming this into a standard Poisson problem results in a smooth implicit surface.

1.2 Contour generators

The contour generator is the curve on the surface where the normal is perpendicular to the view-direction. For implicit surfaces, this contour generator can easily be defined as an implicit curve (see Section 2.2). It seperates the front-facing region from the back-facing part of the surface (Figure 1.3). Simple drawings of three-dimensional objects usually consist of the visible contours of the object. These contours are important in recognizing

the 3D shape, and are often used in so-called non-photorealistic rendering [16, 50, 37], resulting in recognizable, drawing-like, artistic renderings of the object.

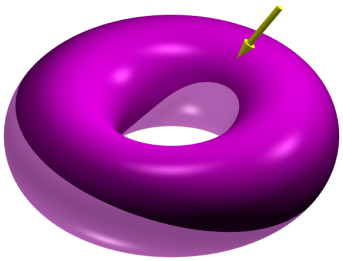


Figure 1.3: Torus with front-facing (dark) and back-facing (light) regions seperated by the contour generator. The arrow indicates the view-direction.

Another application of contour generators is in view-dependent meshing. When fast rendering is important, such as in computer games, it is important to use low-polygon models (i.e. having a low number of polygons). By using smooth lighting models, transitions between adjacent polygons appear to be smoothened (Figure 1.4. The low number of polygons is hardly visible, except along the contours, where large, straight line segments form a jaggy outline. The trick of view-dependent meshing, is to keep the polygon count low over the largest part of the surface, and to refine the mesh along the contour generator [3].

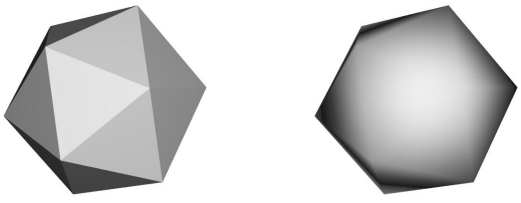


Figure 1.4: A smooth lighting model gives a low-polygon model of a sphere a smooth looking surface, except for the jaggy contour.

1.3 Meshing

As stated earlier, a fast and convenient visualization method for implicit surfaces is to generate a polyhedral mesh, usually a triangulation, approximating the surface. Several algorithms have been suggested for meshing implicit surfaces [55]. These algorithms can be roughly divided into two groups: continuation methods and adaptive enumeration techniques. The former group starts with a triangle close to the surface, and spreads out over the implicit surface by adding new triangles along the boundary of the already triangulated part. The latter group subdivides space into cells and computes local approximations for the intersection of the surface with a cell. By combining the local approximations a mesh is produced. The most famous example of this type of algorithm is Marching Cubes.

1.3.1 Continuation methods

Marching triangles [35] starts with a regular hexagon on the surface. The boundary is called the *front polygon*. From points of the front polygon new triangles of more or less uniform size are created to cover the surface. By checking the distance between points and front polygons, overlapping of triangles is prevented.

Adaptive versions of marching triangles use triangle sizes depending on local curvature. The mesh spreads out over the surface until it is close to existing triangles. This leaves untriangulated strips of the surface, that are filled in a separate phase [1, 42].

Another adaptive version of continuation methods is Edge Spinning [20, 51]. Here, triangles are constructed along the boundary of the already triangulated part, and 'spinned' around the connecting edge until the third vertex lies on the surface.

1.3.2 Adaptive enumeration techniques

Marching cubes Marching Cubes [49] subdivides space into a regular grid of cubes. For implicit surfaces, the function sign at the vertices of a cube are used to determine a local triangle mesh. Symmetry, rotation and flipping of function signs reduces the 256 possible cases to 15 different sign configurations (Figure 1.5). Due to the ambiguities of some of these cases, Marching Cubes is not guaranteed to produce a two-manifold. An improved version that prevents this problem has been presented by Chernyaev [22]. Another way to solve the problems of the original Marching Cubes has been suggested by Lopes and Brodlie [46].

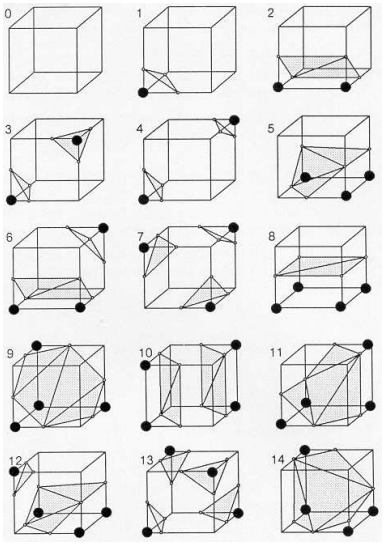


Figure 1.5: The 15 sign configurations of Marching Cubes.

These algorithms only require the function values at the vertices of a regular grid, and originate from meshing of voxel data, for example resulting from MRI scans. They can be easily adapted to implicit surfaces by only considering the function values at these vertices.

Yet another adaptation of Marching Cubes is Marching Tetrahedrons [67]. Using a tetrahedral subdivision not only prevents the ambiguities of cubical subdivision, but also circumvented the recently expired patent on Marching Cubes. Tetrahedra can also be used for adaptive polygonization [40].

Bloomenthal [10] presents a continuation version of Marching Cubes, by only meshing cubes along the boundary of the already meshed part of the surface.

Similar algorithms exist for meshing with octrees instead of regular grids [61, 6, 7, 73, 38]. This enables local refinement of the mesh, for example in regions with higher curvature.

Dual contouring A dual version of Marching Cubes is called Dual Contouring [41]. Here, triangular patches are not created for the cells of the subdivision, but for the edges of the regular grid intersecting the surface. This algorithm uses Hermite data, that is, exact intersection points and normals. For non-smooth level sets of voxel data, this enables Dual Contouring

to handle both sharp vertices and sharp edges.

1.4 Surface reconstruction

An important related topic in computer graphics is surface reconstruction. The problem is to construct a surface approximating a given point cloud, for example the data from a 3D scanner. Several algorithms exist to generate both meshes and implicit surfaces approximating these point clouds. Major problems in surface reconstruction are handling inaccurate data, sharp edges and non-closed manifolds.

If the point sample is dense enough, topological guarantees can be given for the reconstructed surface. Sample density is usually defined in terms of the local feature size, which is defined as the distance of a point on the surface to the medial axis. It is not difficult to imagine that in order to determine the topology of the surface, point samples need to be denser in regions of high curvature, or where two parts of the surface are close to eachother.

For implicit surfaces, topologically correct approximations can be computed by creating a sufficiently dense point sample for this surface. Surface reconstruction then gives a topologically correct mesh of the implicit surface [12]. The problem for generic implicit surfaces is to determine what sufficiently dense means, since computing the local feature size is no trivial task.

1.5 Interval arithmetic

The one-dimensional analogue of surface meshing is solving the equation $F(x) = 0$. To find all roots of a function $F\colon \mathbb{R} \to \mathbb{R}$ we can use interval arithmetic. If the function is strictly positive over a given interval, this interval clearly contains no roots. If the function values at the endpoints of an interval have opposite signs, and the function is strictly increasing, we can be sure that the interval contains exactly one root. It is therefore sufficient to check whether the derivative over that interval is strictly positive. By isolating the roots in disjoint intervals we have determined the 'topology' of the zero set. Also in higher dimensions interval arithmetic is an important tool to guarantee correct topology.

1.5.1 Interval analysis

One way to prevent rounding errors due to finite precision numbers is to use interval arithmetic [53]. Instead of numbers, intervals containing the exact solution are computed. An *inclusion function* $\Box f$ for a function $f\colon \mathbb{R}^m \to \mathbb{R}^n$ computes for each m-dimensional interval I (i.e. an m-box) an n-dimensional interval $\Box f(I)$ such that

$$x \in I \quad \Rightarrow \quad f(x) \in \Box f(I)$$

An inclusion function is *convergent* if

$$\text{width}(I) \to 0 \quad \Rightarrow \quad \text{width}(\Box f(I)) \to 0$$

where the width of an interval is the largest width of I.

For example if $f\colon \mathbb{R} \to \mathbb{R}$ is the square function $f(x) = x \cdot x$, then a convergent inclusion function is given by

$$\Box f([a,b]) = \begin{cases} [\min(a^2,b^2), \max(a^2,b^2)], & a \cdot b < 0 \\ [0, \max(a^2,b^2)], & a \cdot b \geq 0 \end{cases}$$

Inclusion functions exist for the basic operators and functions. To compute an inclusion function it is often sufficient to replace the standard number type (e.g. double) by an interval type. Therefore, in practice almost all functions have convergent inclusion functions, which can be easily computed by using the same code together with an interval library.

We assume there are convergent inclusion functions for our implicit function F and its derivatives, and will denote them by F (and similarly for the derivatives). From the context it will be clear when the inclusion function is meant.

Interval arithmetic can be implemented using demand-driven precision. For the interval bounds, ordinary doubles can be used for fast computation. In the rare case that the interval becomes too small for the precision of a double, a multi-precision number type can be used.

1.5.2 Interval Newton Method

For precision small intervals around the required value are used. Another use of interval arithmetic is to compute function values over larger intervals. If for an implicit surface $F = 0$ and a box I we have $0 \notin \Box F(I)$, we can be certain that I contains no part of the surface. This observation can be extended to the Interval Newton Method, that finds all roots of a function $f\colon \mathbb{R}^n \to \mathbb{R}^n$ in a box I.

The first part of the algorithm recursively subdivides the box, discarding parts of space containing no roots. If the boxes are 'small enough' a Newton method refines the solutions and guarantees that all roots are found. Determining when a box is small enough influences the speed of the algorithm, but has no effect on the guarantee to find all solutions. For the Newton step we solve

$$f(x) + J(I)(z - x) = 0,$$

where x is the centre of I, J is the Jacobian matrix of f and $J(I)$ is the interval matrix of J over the interval I, resulting in an interval Y containing all roots z of f. This interval can be used to refine I. Also, if $Y \subset I$ there is a unique root of f in I.

See [32] or [60] for the mathematical details. A more practical introduction can be found in [62] or [65].

1.5.3 Affine arithmetic

Intervals can be regarded as a value together with an error bound. A drawback of interval arithmetic is that it does not keep track of these error bounds. For example, if we compute $X \cdot X$ for $X = [-1, 1]$, it results in the interval $[-1, 1]$. Stolfi proposed to use affine artithmetic, keeping track of which errors result from specific parameters [24, 66]. If we multiply X by itself, it 'knows' that both terms have the same value. This results in $X \cdot X = [0, 1]$. Affine arithmetic therefore gives more accurate inclusion functions than interval arithmetic. However, due to the much more complicated computations, affine arithmetic is considerably slower. In general, it is not clear if the use of affine arithmetic gives better results than using interval arithmetic with for example an extra level of subdivision.

1.6 Implicit function theorem

An important tool in working with implicit functions is the implicit function theorem. It gives a sufficient condition to represent implicit surfaces locally as the graph of a function. Though the general implicit function theorem is stated for functions $F\colon \mathbb{R}^m \to \mathbb{R}^n$ with $m > n$, we shall only use a few simpler cases.

For an implicit curve $F\colon \mathbb{R}^2 \to \mathbb{R}$ the implicit function theorem states that if $F(x, y) = 0$, where F is a continous function, and $\frac{\partial F}{\partial y} \neq 0$ at a point p, then y can be written as a funtion of x in some neighbourhood of p. Stated otherwise, there exists a function g over that neighbourhood such that $y = g(x)$.

For a smooth function $F\colon \mathbb{R}^2 \to \mathbb{R}$ this implies that the zero-set is a regular curve, as long as the gradient does not disappear on $F = 0$. At a point p with $F(p) = 0$, the curve is perpendicular to $\nabla F(p)$.

A similar result holds for implicit surfaces, that is, for $F\colon \mathbb{R}^3 \to \mathbb{R}$ the zero-set is a regular surface if $\nabla F(p) \neq 0$ for $F(p) = 0$. Again, the gradient $\nabla F(p)$ is perpendicular to the surface at point p.

For contour generators, we will also look at functions $F\colon \mathbb{R}^3 \to \mathbb{R}^2$. In this case, the zero set consists of a one-dimensional curve, if the determinant of the Jacobian matrix $J_F(p)$ does not disappear on the zero-set of F. In fact, this is easy to see by regarding the two components of F as two implicit surfaces. The condition on the Jacobian tells us that the gradients of these two surfaces are not parallel at p. The surfaces therefore have a proper intersection, consisting of a one-dimensional curve.

1.7 Morse theory

Morse theory can be used to determine the topology of implicit surfaces from the critical points of a real function over the surface. This is best illustrated by considering implicit curves. By regarding the implicit function as a height function, we can regard the zero set as the coastline of this landscape. Imagine that the water level rises. New coastlines appear when the water level reaches the bottom of a valley, such that a new lake is created. Coastlines of islands disappear when the water level reaches the highest point of the island. Finally, coastlines can merge when two lakes or two islands grow together. It is not difficult to see that this happens at points of the landscape where the ground is perfectly horizontal. This is exactly where the gradient of the height field function disappears. Moreover, examining the function we can determine the nature of the change in topology of the coastline. The examples given above correspond to local minima, local maxima and saddle points. The topology of the coastlines can now be determined by flooding the entire landscape, slowly lowering the water level, and keeping track of topological changes as the water level reaches so-called critical points, where the gradient disappears.

An important result is the following Morse lemma:

Lemma 1. *Let p be a non-degenerate critical point of F (i.e., the Hessian matrix of F at p is non-singular). Then there exists a neighbourhood U of p and smooth coordinates $(x_1, \ldots, x_n)$ on U such that $x_i(p) = 0$ for all i, and $F|_U = F(p) - x_1^2 - \ldots - x_\lambda^2 + x_{\lambda+1}^2 + \ldots + x_n^2 +$* higher order terms*, where λ is the index of F at point p.*

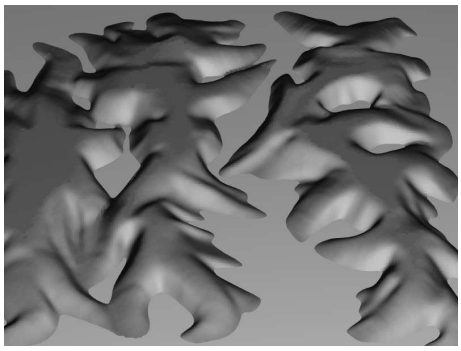


Figure 1.6: An implicit curve as the coastline of a mathematical landscape. Islands merge as the water level drops.

A corollary of the Morse lemma is that non-degenerate critical points are isolated.

To apply Morse theory to implicit surfaces, information about the critical points and their indices is required. In general this is not a trivial task, requiring interval arithmetic to construct unique enclosures of the critical points. See [13] for a sketch of how to do this. For specific classes of functions, such as algebraic surfaces and certain metaball-like skeleton based models [72], more convenient methods exist.

Another issue is finding a convenient Morse function on the surface, i.e. having no degenerate critical points with all critical levels being different. The particular choice of this function greatly influences the number of critical points and therefore the complexity of determining the topology [54].

1.8 Overview and main results

Chapter 2 provides a mathematical framework for the robust computation of contour generators and apparent contours of implicit surfaces. Morse theory is used to study the singularities of contour generators for evolving surfaces, or fixed surfaces with a moving viewpoint. We present a new derivation of the different types of singularities that can occur, together with local models that can be used to classify these singularities. These results can be used to update approximations of contour generators for dynamic surfaces with topological guarantees.

In Chapter 3 we present a new algorithm for tracing implicit curves in higher dimensional space, such that the resulting piecewise linear approximation is topological equivalent to the implicit curve itself. As far as we

know this is the first practical algorithm guaranteeing topological correctness. This algorithm is used to create a topologically correct approximation of a regular contour generator of a static surface. We also explain how a similar tracing process in a four-dimensional space can speed up the updating of contour generators for evolving surfaces.

Chapter 4 presents one of the first algorithms to mesh implicit surfaces, such that the resulting mesh is isotopic to the implicit surface itself. Compared to conventional (not necessarily isotopic) meshing algorithms it adds little overhead, and is therefore quite fast. In particular, to guarantee isotopy, we do not have to do a relatively slow computation of critical points. Instead, a simple test based on interval arithmetic tests whether the local deviation in gradient is small enough for a simple local approximation.

Finally, in Chapter 5 we examine how the same data structure used for isotopic approximation of implicit surfaces, can be used to move through the level sets of an implicit function. With a predetermined precision, the algorithm produces meshes for the different level sets, and keeps track of isotopy. If the level set moves close to singularities, isotopy cannot be guaranteed. However, the algorithm indicates when and where this is the case, by isolating the problematic areas in small 'red' boxes. This way, areas where the isotopy class is not yet determined, are clearly indicated. In this case, further subdivision or analysis of the implicit function could be used to examine the behaviour inside these red boxes.

1.8.1 Publications

Chapters 2 and 3 are to appear as *Computing Contour Generators of Evolving Implicit Surfaces* in ACM Transactions on Graphics.

Chapter 4 and the tetrahedral meshing algorithm of Chapter 5 are to appear as *Isotopic Meshing of Implicit Surface* in The Visual Computer.

Chapter 2

Contour generators

2.1 Introduction

An important visibility feature of a smooth object seen under parallel projection along a certain direction is its *contour generator*, also known as outline, or profile. The contour generator is the curve on the surface, separating front-facing regions from back-facing regions. This curve may have singularities if the direction of projection is non-generic. The *apparent contour* is the projection of the contour generator onto a plane perpendicular to the view direction. In many cases, drawing just the visible part of the apparent contour gives a good impression of the shape of the object. In this chapter, we will not distinguish between visible and invisible parts of the contour generator. Stated otherwise, we assume the surface is transparent. Generically, the apparent contour is a smooth curve, with some isolated singularities. See Figure 2.1.

The contour generator and the apparent contour play an important role in computer graphics and computer vision. Rendering a polyhedral model of a smooth surface yields a jaggy outline, unless the triangulation of the surface is finer in a neighbourhood of the contour generator. This observation has led to techniques for view-dependent meshing and view-dependent refinement techniques, cf. [3].

Also, to render a smooth surface it is sufficient to render only the part of the surface with front-facing normals, so the contour generator, being the boundary of the potentially visible part plays a crucial role here. In non-photorealistic rendering [50] one just visualizes the apparent contour, perhaps enhanced by strokes indicating the main curvature directions of the surface. This is also the underlying idea in silhouette rendering of implicit surfaces [16]. In computer vision, techniques have been developed for partial reconstruction of a surface from a sequence of apparent contours correspond-

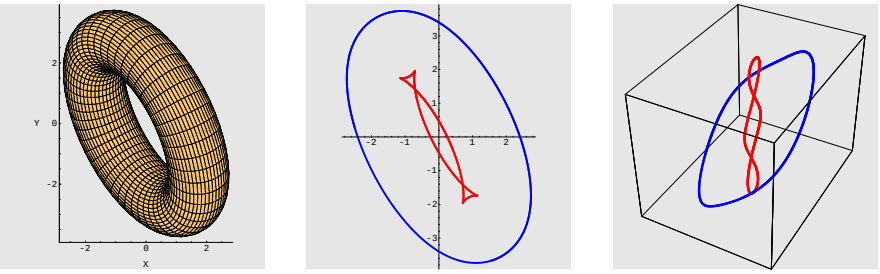


Figure 2.1: A smooth surface (left), its apparent contour under parallel projection along the z direction (middle), and its contour generator, seen from a different position (right). For generic surfaces (or, generic parallel projections) the contour generator is a smooth, possibly disconnected curve on the surface, whereas the apparent contour may have isolated cusp points.

ing to a discrete set of nearby projection directions. We refer to the book by Cipolla and Giblin [23] for an overview, and for a good introduction to the mathematics underlying this chapter. Other applications use silhouette interpolation [31] from a precomputed set of silhouettes to obtain the silhouette for an arbitrary projection. In computational geometry, rapid silhouette computation of polyhedral models under perspective projection with moving viewpoints has been achieved by applying suitable preprocessing techniques [8].

This chapter presents a mathematical framework for the robust computation of contour generators and apparent contours of implicit surfaces. For an introduction to the use of implicit surfaces for smooth deformable object modeling we refer to Chapter 1, and to [57] and [9].

We first consider generic static views, where both the surface and the direction of projection are static. Then we pass to time-dependent views, where the direction of projection changes with time. We derive conditions that locate the changes in the topology of the contour generator and the apparent contour. It turns out that generically there are three types of events, or bifurcations, leading to such a change in topology. These bifurcations have been studied from a much more advanced mathematical point of view, where they are known under the names lips, beak-to-beak, and swallowtail bifurcation. See also [18]. For a nice non-mathematical description we refer to the book by Koenderink [44]. Also see [5] and [17] for a sketch of some of the mathematical details related to singularity theory. Some of the results of this chapter are given in a complex analytic setting in [4]. Our approach is somewhere inbetween the level of Koenderink's book and the sophisticated mathematical approach. We use only elementary tools, like

the Inverse and Implicit Function Theorem, and finite order Taylor expansions. These techniques are used to design algorithms, in the same way as the Implicit Function Theorem gives rise to Newton's method.

The main result of this chapter is a general framework to examine and approximate contour generators of implicit surfaces. We present a new derivation of the different types of singularities, together with local models. For e.g. algebraic surfaces, these local models can be used to classify singularities. As a first step to implementation, we present a new curve tracing algorithm, with a correct topology of the output.

In Section 2.2 we present the framework, and discuss criteria for a point on the contour generator and apparent contour to be regular. Section 2.3 examines singularities under some time-dependent view, for example when the viewpoint moves or when the surface deforms. In section 2.4 we explain the transformation to local models. Section 2.5 shows in detail how to derive local models near fold and cusp points. In section 2.6 we examine contours for time-dependent views. An overview of the existing theory used in this chapter is given in the appendices.

2.2 Contour generator and apparent contour

2.2.1 Contour generators of implicit surfaces

To understand the nature of regular and singular points of the contour generator, and their projections on the apparent contour, we assume S is given as the zero-set of a smooth function $F\colon \mathbb{R}^3 \to \mathbb{R}$, so $S = F^{-1}(0)$. *Smooth* means that the required derivatives exist; for most of the results C^3 will be sufficient. In the section on evolving surfaces sometimes C^5 is required. Furthermore, we assume that 0 is a regular value of F, i.e., the gradient ∇F is non-zero at every point of the surface. The gradient vector $\nabla F(p)$ is the normal of the surface at p, i.e., it is normal to the tangent plane of S at p. This tangent plane is denoted by $T_p(S)$. If v is the direction of parallel projection, then the contour generator Γ is the set of points at which the normal to S is perpendicular to the direction of projection, i.e., $p \in \Gamma$ iff the following conditions hold:

$$\begin{aligned} F(p) &= 0 \\ \langle \nabla F(p), v \rangle &= 0. \end{aligned} \tag{2.1}$$

For convenience, we assume that $v = (0, 0, 1)$. Then the preceding equations reduce to

$$\begin{aligned} F(x, y, z) &= 0 \\ F_z(x, y, z) &= 0. \end{aligned} \tag{2.2}$$

For arbitrary view directions the formulae become more complicated. It is more convenient to rotate the surface instead, by applying a rotation transformation to the implicit function. Also, a linear skewing transformation can be used. For view direction $(\alpha, \beta, 1)$ we consider the function $F(x - \alpha z, y - \beta z, z)$ with view direction $(0, 0, 1)$, and use the inverse skewing to move the approximation of the contour generator back to the original surface.

We assume that S is a generic surface, i.e. there are no degenerate singular points on its contour generator. Some functions can yield degenerate contour generators. For example, a cylinder has a two-dimensional contour generator for the view direction along its axis. Using a small perturbation, we can remove these degeneracies. In the case of the cylinder, the two-dimensional contour generator collapses to a one-dimensional curve. See for example [65].

We now derive conditions for the contour generator Γ and the apparent contour γ to be regular at a given point. Recall that a curve is regular at a certain point if it has a non-zero tangent vector at that point. The next result gives conditions in terms of the function defining the surface.

Proposition 2.
1. A point $p \in \Gamma$ is a regular point of the contour generator if and only if

$$F_{zz}(p) \neq 0 \quad \textit{or} \quad \Delta(p) \neq 0, \tag{2.3}$$

where $\Delta(p)$ is a Jacobian determinant defined by

$$\Delta(p) = \left.\frac{\partial(F, F_z)}{\partial(x, y)}\right|_p = \begin{vmatrix} F_x(p) & F_y(p) \\ F_{xz}(p) & F_{yz}(p) \end{vmatrix}.$$

2. A point $p \in \gamma$ is a regular point of the apparent contour if and only if

$$F_{zz}(p) \neq 0. \tag{2.4}$$

Proof. 1. The condition for p to be regular is

$$\nabla F(p) \wedge \nabla F_z(p) \neq 0.$$

Since $F_z(p) = 0$, a straightforward calculation yields

$$\nabla F(p) \wedge \nabla F_z(p) = F_{zz} \cdot (F_y, -F_x, 0) + \frac{\partial(F, F_z)}{\partial(x, y)} \cdot (0, 0, 1). \tag{2.5}$$

Here all derivatives are evaluated at p. Since $\nabla F(p) = (F_x(p), F_y(p), 0) \neq 0$, we see that $(F_y, -F_x, 0)$ and $(0, 0, 1)$ are linearly independent vectors. Therefore, the linear combination of these vectors in the right-hand side of (2.5)

is zero iff the corresponding scalar coefficients are zero. The necessity and sufficiency of condition (2.3) is a straightforward consequence of this observation.

2. Since $\nabla F(p) \neq 0 \in \mathbb{R}^3$, and $F_z(p) = 0$, we see that $(F_x(p), F_y(p)) \neq (0,0)$. Assuming $F_y(p) \neq 0$, we get

$$\left.\frac{\partial(F, F_z)}{\partial(y, z)}\right|_p = \left.\begin{vmatrix} F_y & F_z \\ F_{yz} & F_{zz} \end{vmatrix}\right|_p = F_y(p)F_{zz}(p) \neq 0. \tag{2.6}$$

Let $p = (x_0, y_0, z_0)$. Then, the Implicit Function Theorem yields locally defined functions $\eta, \zeta : \mathbb{R} \to \mathbb{R}$, with $\eta(x_0) = y_0$ and $\zeta(x_0) = z_0$, such that (2.2) holds iff $y = \eta(x)$ and $z = \zeta(x)$. The contour generator is a regular curve parametrized as $x \mapsto (x, \eta(x), \zeta(x))$, locally near p, whereas the apparent contour is a regular curve in the plane parametrized as $x \mapsto (x, \eta(x))$, locally near (x_0, y_0). □

2.2.2 Singular points of contour generators

We apply the preceding result to detect non-degenerate singularities of contour generators of implicit surfaces. This result will be applied later in this chapter, when we consider contour generators of time-dependent surfaces.

Again, let the regular surface S be the zero set of a C^3-function $F\colon \mathbb{R}^3 \to \mathbb{R}$, for which 0 is a regular value. We consider the contour generator Γ of S under parallel projection along the vector $v = (0, 0, 1)$. The equations for Γ are

$$F = F_z = 0.$$

We consider the contour generator as the zero-set of the function F_z, restricted to S.

Corollary 3. *Point p is a non-degenerate singular point of Γ iff the following two conditions hold:*

$$F(p) = F_z(p) = F_{zz}(p) = \left.\frac{\partial(F, F_z)}{\partial(x, y)}\right|_p = 0, \tag{2.7}$$

and $\Sigma(p) \neq 0$, where, for $F_x(p) \neq 0$,

$$\begin{aligned} \Sigma(p) = &- {F_x}^2 {F_{xzz}}^2 {F_y}^2 + 2{F_x}^3 F_{xzz} F_y F_{yzz} - {F_x}^4 {F_{yzz}}^2 \\ &- 2{F_x}^3 F_{xyz} F_y F_{zzz} + 2{F_x}^2 F_{xy} F_{xz} F_y F_{zzz} \\ &+ {F_x}^2 F_{xxz} {F_y}^2 F_{zzz} - F_x F_{xx} F_{xz} {F_y}^2 F_{zzz} \\ &- {F_x}^3 F_{xz} F_{yy} F_{zzz} + {F_x}^4 F_{yyz} F_{zzz}, \end{aligned} \tag{2.8}$$

whereas, for $F_y(p) \neq 0$*, we have*

$$\begin{aligned}\Sigma(p) = &- {F_{xzz}}^2 {F_y}^4 + 2 F_x F_{xzz} {F_y}^3 F_{yzz} - {F_x}^2 {F_y}^2 {F_{yzz}}^2 \\ &- 2 F_x F_{xyz} {F_y}^3 F_{zzz} + F_{xxz} {F_y}^4 F_{zzz} \\ &+ {F_x}^2 {F_y}^2 F_{yyz} F_{zzz} + 2 F_x F_{xy} {F_y}^2 F_{yz} F_{zzz} \\ &- F_{xx} {F_y}^3 F_{yz} F_{zzz} - {F_x}^2 F_y F_{yy} F_{yz} F_{zzz}.\end{aligned} \tag{2.9}$$

Proof. Condition (2.7) reflects the fact that p is a singular point of $F_z|S$, cf. (2.3), whereas (2.8) expresses non-degeneracy of this singular point. Condition (2.8) is obtained by a straightforward expansion of (A.3) (see appendix A), with $G = F_z$, and $V = X$ as in (A.4), where $\lambda = \frac{F^0_{xz}}{F^0_x}$, $F^0_z = F^0_{zz} = 0$, and $F^0_{yz} = \frac{F^0_y}{F^0_x} F^0_{xz}$. Condition (2.9) is derived similarly. □

2.2.3 Generic projections: fold and cusp points

In view of Proposition 2, regular points of the apparent contour are projections of points $(x, y, z) \in \mathbb{R}^3$ satisfying

$$F(x, y, z) = F_z(x, y, z) = 0, \quad \text{and} \quad F_{zz}(x, y, z) \neq 0.$$

This being a system of two equations in three unknowns, we expect that the regular points of the apparent contour form a one-dimensional subset of the plane. Furthermore, the singular points of the apparent contour are projections of points satisfying an additional equation, viz. $F_{zz}(x, y, z) = 0$, and are therefore expected to be isolated. This is true for *generic* surfaces. To make this more precise, we consider the set of functions $F : \mathbb{R}^3 \to \mathbb{R}$, satisfying

$$(\, F(x, y, z), F_z(x, y, z), F_{zz}(x, y, z), \Delta(x, y, z)\,) \neq (0, 0, 0, 0), \tag{2.10}$$

and

$$(\, F(x, y, z), F_z(x, y, z), F_{zz}(x, y, z), F_{zzz}(x, y, z)\,) \neq (0, 0, 0, 0). \tag{2.11}$$

If F satisfies (2.10), then Proposition 2 tells us that the contour generator Γ of $S = F^{-1}(0)$ under parallel projection along ν is a regular curve. Moreover, for a point $(x, y, z) \in \Gamma$ there are two cases:

1. $F_{zz}(x, y, z) \neq 0$; in this case the point projects to a regular point (x, y) of the apparent contour γ. Such a point is called a *fold point* of the

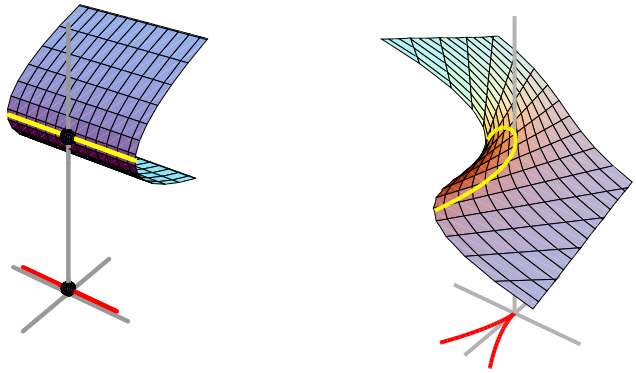


Figure 2.2: (a) A local model of the surface at a fold point is $x + z^2 = 0$. Both the contour generator Γ, and the visible contour γ, are regular at the fold point, and its projection onto the image plane, respectively. (b) A local model at a cusp point is $x+yz+z^3 = 0$. Here the contour generator is regular, but the apparent contour has a regular cusp.

contour generator. This terminology is justified by the *local model* of the surface near a fold point, viz.

$$x + z^2 = 0. \tag{2.12}$$

See also Figure 2.2a. Here the contour generator is the y-axis in three space, so the apparent contour is the y-axis in the image plane.

2. $F_{zz}(x, y, z) = 0$; in this case the point projects to a singular point (x, y) of γ. Such a point is called a *cusp point* of the contour generator if, in addition to (2.10), condition (2.11) is satisfied, i.e., if both $\Delta(x, y, z) \neq 0$ and $F_{zzz}(x, y, z) \neq 0$. In this case the surface has the following local model near the cusp point:

$$G(x, y, z) = x + yz + z^3 = 0. \tag{2.13}$$

See also Figure 2.2b. The local model G is sufficiently simple to allow for an explicit computation of its contour generator and apparent contour: the former is parametrized by $z \mapsto (2z^3, -3z^2, z)$, the latter is a *regular cusp* parametrized by $z \mapsto (2z^3, -3z^2)$.

Intuitively speaking, a local model of the surface near a point is a 'simple' expression of the defining equation in suitably chosen local coordinates.

Usually, as in the cases of fold and cusp points, a local model is a low degree polynomial, which can be easily analyzed in the sense that the contour generator and the apparent contour are easily determined.

So far we have only considered parallel projection. The standard perspective transformation [36], which moves the viewpoint to ∞, reduces perspective projections to parallel projections. By deforming the surface using this transformation, perspective projections can be computed by using parallel projection on the transformed implicit function. Note that this perspective transform does not influence the smoothness of the implicit surface function.

2.3 Evolving contours

As we have seen, generic surfaces satisfy conditions (2.10) and (2.11), since violation of one of these conditions would correspond to the existence of a solution of four equations in three unknowns. However, *evolving surfaces* depend on an additional variable, t say. Evolving surfaces occur for example when the viewpoint moves along a predetermined path, but also under deformation (or morphing) of the surface. Time dependency is expressed by considering implicitly defined surfaces

$$S_t = \{(x, y, z) \in \mathbb{R}^3 \mid F(x, y, z, t) = 0\},$$

where $F : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ is a smooth function of the space variables (x, y, z) and time t. Generically we expect that exactly one of the conditions (2.10) and (2.11) will be violated at isolated values of (x, y, z, t). For definiteness, we assume $(0, 0, 0, 0)$ is such a value.

Violation of (2.10) corresponds to a singularity of the contour generator. In this case the implicit surfaces, defined by $F(x, y, z, 0) = 0$ and $F_z(x, y, z, 0) = 0$, are tangent at $(x, y, z) = (0, 0, 0)$, but the tangency is non-degenerate. Stated otherwise, the function $G : \mathbb{R}^3 \to \mathbb{R}$, defined by $G(x, y, z) = F_z(x, y, z)$, *restricted* to the surface S_0, has a non-degenerate singularity at $(0, 0, 0)$.

Generically, there are two types of *bifurcations*, corresponding to different scenarios for changes in topology of the contour generator. The *beak-to-beak* bifurcation corresponds to the *merging* or *splitting* of connected components of the contour generator. Under some additional generic conditions (inequalities), a local model for this phenomenon is the surface, defined by

$$G(x, y, z, t) = x + (-y^2 + t)z + z^3, \tag{2.14}$$

Here the contour generator is defined by $x = 2z^3, -y^2 + 3z^2 = -t$. See also Figure 2.3.

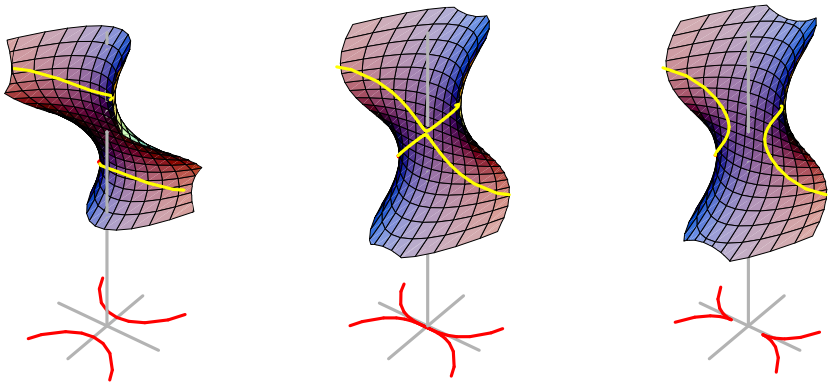


Figure 2.3: The beak-to-beak bifurcation. With respect to the local model (2.14) the bifurcation corresponds to $t < 0$ (left), $t = 0$ (middle), and $t > 0$ (right).

Putting $G^t(x, y, z) = G(x, y, z, t)$, we check that G^0 satisfies (2.7) at $p = (0, 0, 0)$, and that $|\Sigma(p)| = -4$. (In fact, $G^0_x = 1$, so all higher order derivatives of G^0_x vanish identically, so only the last term in the right hand side of (2.8) is not identically equal to zero.) Therefore, $G^0_z|S$ has a non-degenerate singular point of saddle type at p. According to the Morse lemma [28, 52], the level set of $G^0_z|S$ through p consists of two regular curves, intersecting transversally at p, which concurs with Figure 2.3 (middle).

A second scenario due to the violation of (2.11) is the *lips* bifurcation, corresponding to the *birth* or *death* of connected components of the contour generator. Again, under some additional generic conditions a local model for this phenomenon is the surface, defined by

$$G(x, y, z, t) = x + (y^2 + t)z + z^3, \tag{2.15}$$

Here the contour generator is defined by $x = 2z^3, y^2 + 3z^2 = -t$. In particular, for $t > 0$ the surface S_t has no connected component of the contour generator near $(0, 0, 0)$, for $t = 0$, the point $(0, 0, 0)$ is isolated on the contour generator, and for $t < 0$ there is a small connected component growing out of this isolated point as t decreases beyond 0. See also Figure 2.4.

As for the beak-to-beak bifurcation, we show that $G^0|S$ has a non-degenerate singular point at $(0, 0, 0)$, which in this case is an extremum.

Violation of (2.11) involves the occurrence of a higher order singularity of the apparent contour. Note, however, that in this situation the contour generator is still regular at the point (x, y, z), cf. Proposition 2. Imposing some additional generic conditions a local model for this type of bifurcation

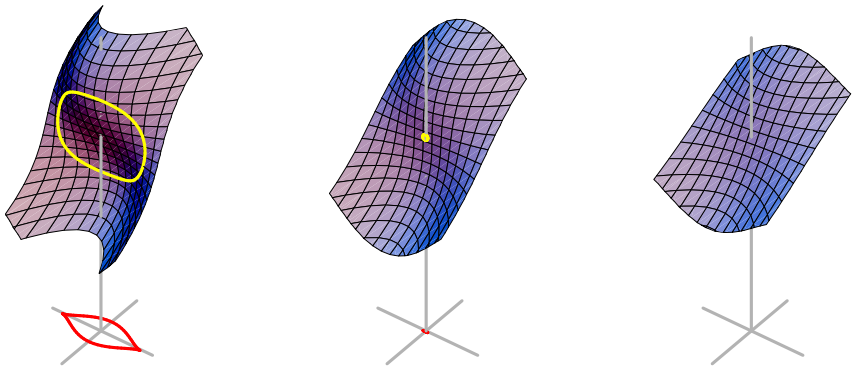


Figure 2.4: The lips bifurcation. Left: $t < 0$. Middle: $t = 0$. Right: $t > 0$.

is

$$G(x, y, z, t) = x + yz + tz^2 + z^4 = 0. \tag{2.16}$$

Here the apparent contour is parametrized as $z \mapsto (tz^2 + z^4, -2tz - 4z^3)$. See also Figure 2.5.

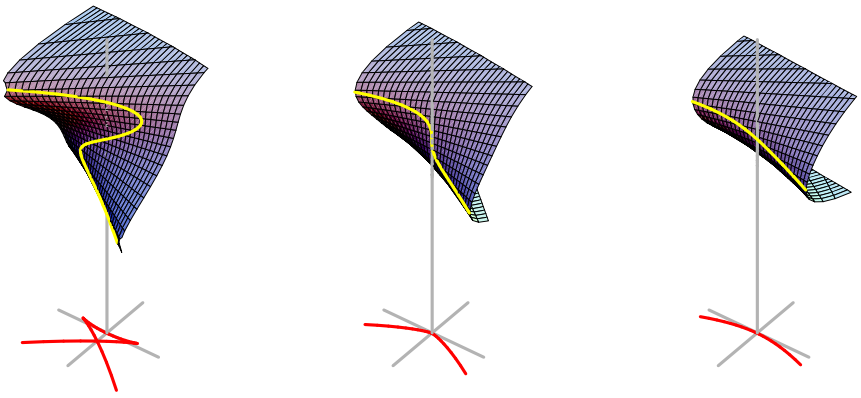


Figure 2.5: The swallowtail bifurcation. Left: $t < 0$. Middle: $t = 0$. Right: $t > 0$.

2.4 Transformations and normal forms

In Sections 2.2 and 2.3 we presented local models of various types of regular and singular points on contour generators and apparent contours, both for generic static surfaces, and for surfaces evolving generically in time. These local models are low degree polynomials, which are easy to analyze, and which yet capture the *qualitative behavior* of the contour generator and the apparent contour in a neighbourhood of the point of interest. In this section we explain more precisely what we mean by capturing local behavior.

Consider two regular implicit surfaces $S = F^{-1}(0)$ and $T = G^{-1}(0)$. An invertible smooth map $\Phi\colon \mathbb{R}^3 \to \mathbb{R}^3$ for which

$$F \circ \Phi = G \tag{2.17}$$

maps T to S. In fact, we consider Φ to be defined only locally near some point of T, but we will not express this in our notation. The map Φ need not map the contour generator of T onto that of S, however. To enforce this, we require that Φ maps vertical lines onto vertical lines, i.e., Φ should be of the form

$$\Phi(x, y, z) = (h(x, y), H(x, y, z)), \tag{2.18}$$

where $h\colon \mathbb{R}^2 \to \mathbb{R}^2$ and $H\colon \mathbb{R}^3 \to \mathbb{R}$ are smooth maps. The map h is even invertible, since Φ is invertible. To allow ourselves even more flexibility in the derivation of local models, we relax condition (2.17) by requiring the existence of a non-zero function $\varphi\colon \mathbb{R}^3 \to \mathbb{R}$ such that

$$F(\Phi(x, y, z)) = \varphi(x, y, z)G(x, y, z). \tag{2.19}$$

Definition 4. *Let $S = F^{-1}(0)$ and $T = G^{-1}(0)$ be regular surfaces, near $p = (0, 0, 0) \in \mathbb{R}^3$. An admissible local transformation from T to S, locally near p, is a pair (Φ, φ), where $\varphi\colon \mathbb{R}^3 \to \mathbb{R}$ is non-zero at p, and $\Phi\colon \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ is locally invertible near p, and of the form (2.18), such that (2.19) holds. We also say that Φ brings F in the normal form G.*

If the surfaces S and T depend smoothly on k parameters, i.e., they are defined by functions $F\colon \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$ and $G\colon \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}$, respectively, then we require that the parameters are not mixed with the (x, y, z)-coordinates, i.e., we require that (2.19) is replaced with

$$F(\Phi(x, y, z, \mu)) = \varphi(x, y, z, \mu)\, G(x, y, z, \mu),$$

where $\Phi\colon \mathbb{R}^3 \times \mathbb{R}^k \to \mathbb{R}^3 \times \mathbb{R}^k$ is of the form

$$\Phi(x, y, z, \mu) = (h(x, y, \mu), H(x, y, z, \mu), \psi(\mu)).$$

Then Φ^μ, defined by $\Phi^\mu(x, y, z) = \Phi(x, y, z, \mu)$, maps T^μ to $S^{\psi(\mu)}$, and preserves contour generators. Furthermore, the map $h^\mu\colon \mathbb{R}^2 \to \mathbb{R}^2$, defined by $h^\mu(x, y) = h(x, y, \mu)$, maps the apparent contour of T^μ onto that of $S^{\psi(\mu)}$.

Proposition 5. *If Φ is an admissible local transformation from T to S, locally near a point p on the contour generator of T, where Φ is of the form (2.18), then*

1. *Φ maps T to S, locally near $p \in S$;*
2. *Φ maps the contour generator of T to the contour generator of S, locally near p;*
3. *h maps the apparent contour of T onto the apparent contour of S, locally near the projection $\pi(p) \in \mathbb{R}^2$.*

Proof. From (2.18) and (2.19) it is easy to derive

$$\begin{aligned} F(\Phi(p)) &= \psi(p)\, G(p), \\ F_z(\Phi(p))\, H_z(p) &= \psi_z(p)\, G(p) + \psi(p)\, G_z(p). \end{aligned}$$

Since $\psi(p) \neq 0$, and $H_z(p) \neq 0$, we conclude that $G(p) = G_z(p) = 0$ iff $F(\Phi(p)) = F_z(\Phi(p)) = 0$. □

2.4.1 Example: local model at a truncated cusp point

We now illustrate the use of admissible transformations by deriving a local model for the class of implicit surfaces defined as the zero set of a function of the form:

$$F(x, y, z) = a(x, y) + b(x, y)z + c(x, y)z^2 + z^3, \tag{2.20}$$

with $a(0, 0) = b(0, 0) = c(0, 0) = 0$, and

$$\left.\frac{\partial(a, b)}{\partial(x, y)}\right|_0 \neq 0. \tag{2.21}$$

Note that the local model $x + y\,z + z^3 = 0$, derived in Section 2.2 for a cusp point, belongs to this class. Our goal is to show that the latter is indeed a local model for all surfaces of the type (2.20). Since $F(0) = F_z(0) = F_{zz}(0) = 0$, and

$$\left.\frac{\partial(F, F_z)}{\partial(x, y)}\right|_0 = \left.\frac{\partial(a, b)}{\partial(x, y)}\right|_0 \neq 0,$$

we see that $0 \in \mathbb{R}^3$ is a regular point of the contour generator of S, whereas $(0, 0)$ is a singular point of the apparent contour, cf. Proposition 2

As a first step towards a normal form of the implicit surface, we apply the *Tschirnhausen transformation* $z \mapsto z - \frac{1}{3}c(x,y)$ to transform the quadratic term (in z) in F away. More precisely,

$$\begin{aligned} F(x,y,z-\tfrac{1}{3}\,c(x,y)) &= a(x,y) - \tfrac{1}{3}\,b(x,y)\,c(x,y) + \tfrac{2}{27}\,c(x,y)^3 \\ &\quad + (b(x,y) - \tfrac{1}{3}\,c^2(x,y))\,z + z^3 \\ &= G(\overline{\varphi}(x,y),z), \end{aligned}$$

where $G(x,y,z) = x + yz + z^3$, and

$$\begin{aligned} \overline{\varphi}(x,y) = (\; & a(x,y) - \tfrac{1}{3}\,b(x,y)\,c(x,y) + \tfrac{2}{27}\,c(x,y)^3, \\ & b(x,y) - \tfrac{1}{3}\,c(x,y)^2\;). \end{aligned}$$

It is not hard to check that the Jacobian determinant of $\overline{\varphi}$ at $0 \in \mathbb{R}^2$ is equal to

$$\left.\frac{\partial(a,b)}{\partial(x,y)}\right|_0,$$

so $\overline{\varphi}$ is a local diffeomorphism near $0 \in \mathbb{R}^2$. Let φ be its inverse, then, putting

$$\Phi(x,y,z) = (\varphi(x,y), z - \tfrac{1}{3}\,c(\varphi(x,y)),$$

we get

$$F \circ \Phi(x,y,z) = G(x,y,z).$$

In other words, the *admissible* transformation Φ brings F into the normal form G. In particular, it maps the surface $T = G^{-1}(0)$ and its contour generator onto $S = F^{-1}(0)$, and φ maps the apparent contour of T onto the apparent contour of S.

2.5 Deriving local models

In this section, we derive local models of an implicit surface near fold and cusp points of contour generators. The local model for the fold point is as stated in Section 2.2, the one for the cusp point is slightly weaker in the sense that it includes terms of higher order. To derive these models, we use only elementary means, like the Implicit Function Theorem and Taylor expansion up to finite order. Although these tools are strong enough to derive the essential qualitative features of the contour generator and apparent contour at a fold or a cusp point, they are not strong enough to obtain the simple polynomial form for the cusp as stated earlier. Although the cubic model for the cusp (See Figure 2.2b) can be obtained by elementary means, the

derivation is quite involved, cf. [70]. Even worse, the polynomial models of the contour generator of evolving surfaces (See Figures 2.3, 2.4, and 2.5) can only be obtained using sophisticated methods from Singularity Theory. Therefore, our more modest goal will be to obtain a local model only for the *lower order part* of the implicit function, which is sufficient for our purposes. See Proposition 7 for a more precise statement.

2.5.1 Local model at a fold point

Proposition 6. *Let $p \in S$ be a regular point of the contour generator of S, and let its image under parallel projection along v be a regular point of the apparent contour of S. Then there is an admissible transformation, bringing S into the normal form $x \pm z^2 = 0$ (See also Figure 2.2a).*

Proof. Let $p = 0 \in \mathbb{R}^3$ be a regular point of the contour generator of $S = F^{-1}(0)$, and let its projection $(0,0)$ be a regular point of the apparent contour. Then $F(0) = F_z(0) = 0$, and $F_{zz}(0) \neq 0$. Consider F as a 2-parameter family of real-valued univariate functions depending on the parameters (x, y), i.e.,

$$F_{(x,y)}(z) = F(x, y, z).$$

Then $F_{(0,0)}$ has a non-degenerate singularity at $0 \in \mathbb{R}$, and hence the function $F_{(x,y)}$ has a non-degenerate singularity $z = \zeta(x, y)$, depending smoothly on the parameters (x, y), such that $\zeta(0,0) = 0$. According to the Morse Lemma with parameters, cf. Appendix B, there is a local diffeomorphism $(x, y, z) \mapsto (x, y, Z(x, y, z))$ on a neighborhood of $0 \in \mathbb{R}^3$, such that

$$F(x, y, Z(x, y, z)) = \alpha(x, y) \pm z^2.$$

where $\alpha(x, y) = F(x, y, \zeta(x, y))$. Since $(\alpha_x^0, \alpha_y^0) = (F_x^0, F_y^0) \neq (0,0)$, let us assume that $\alpha_x^0 \neq 0$. Then there is a local diffeomorphism $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ of the form $\varphi(x, y) = (\xi(x, y), y)$, such that

$$\alpha(\varphi(x, y)) = x.$$

In fact, $\xi(x, y)$ is the solution of the equation $\alpha(\xi, y) - x = 0$, with $\xi(0,0) = 0$. The existence and local uniqueness follows from the Implicit Function Theorem. Furthermore, $\xi_x^0 \alpha_x^0 = 1$, which proves that φ is a local diffeomorphism. Therefore,

$$F \circ \Phi(x, y, z)) = x \pm z^2,$$

where the local diffeomorphism $\Phi : \mathbb{R}^3 \to \mathbb{R}^3$ is defined by

$$\Phi(x, y, z) = (\xi(x, y), y, Z(\xi(x, y), y, z)).$$

Now Φ is an admissible transformation, so it maps the surface T, defined by $x \pm z^2 = 0$ and its contour generator, viz. the line $x = z = 0$, onto S and its contour generator. Furthermore, the diffeomorphism $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ maps the apparent contour $x = 0$ of T onto the apparent contour of S. □

2.5.2 Local model at a cusp point

Proposition 7. *Let $p \in S$ be a point of the contour generator of S, satisfying conditions (2.10) and (2.11). Then p projects onto a cusp point of the apparent contour of S under parallel projection along v. More precisely, there is an admissible transformation, defined on a neighborhood of p, transforming S into a local model of the form*

$$G(x, y, z) = x + yz + z^3 + z^4 R(x, y, z) = 0,$$

where R is a smooth function (See also Figure 2.2b).

Before giving the proof, we observe that this result involves the existence of a map $h : \mathbb{R}^2 \to \mathbb{R}^2$, mapping the apparent contour of $T = G^{-1}(0)$ onto the apparent contour of $S = F^{-1}(0)$. The apparent contour of T is easily determined by solving x and y from the following system of equations:

$$\begin{aligned} G(x, y, z) &= x + yz + z^3 + z^4 R(x, y, z), \\ G_z(x, y, z) &= y + 3z^2 + z^3 (4R(x, y, z) + zR_z(x, y, z)). \end{aligned} \tag{2.22}$$

Using the Implicit Function Theorem, it is easy to see that we can solve x and y from (2.22), yielding:

$$\begin{aligned} x(z) &= 2z^3 + O(z^4) \\ y(z) &= -3z^2 + O(z^3). \end{aligned} \tag{2.23}$$

Therefore, the apparent contour of S has a *cusp* at $(0, 0)$.

Proof. We follow the same strategy as in the example of the truncated cusp in Section 2.4 by transforming the quadratic term of F in z away. For general (not necessarily polynomial) functions F the Tschirnhausen transformation is obtained as follows. Since $F_{zzz}(p) \neq 0$, the Implicit Function Theorem guarantees that

$$F_{zz}(x, y, z) = 0 \tag{2.24}$$

has a solution $z = z(x, y)$, locally near $(0, 0)$, with $z(0, 0) = 0$. In this situation, Taylor expansion yields:

$$F(x, y, z + z(x, y)) = C(x, y)\big(U(x, y) + V(x, y)\, z + z^3 + z^4\, R_4(x, y, z)\big), \tag{2.25}$$

where R_4 is a smooth function, and

$$\begin{aligned} C(x,y) &= \tfrac{1}{6}F_{zzz}(x,y,z(x,y)). \\ U(x,y) &= F(x,y,z(x,y))/C(x,y) \\ V(x,y) &= F_z(x,y,z(x,y))/C(x,y) \end{aligned} \tag{2.26}$$

Note that $C(0,0) \neq 0$. Furthermore, $U(0,0) = V(0,0) = 0$. Note that there is no quadratic term in (2.25), because $F_{zz}(x,y,z(x,y)) = 0$.

The right hand side of (2.25) is in fact the form of F after the parameter-dependent change of coordinates $(x,y,z) \mapsto (x,y,z+z(x,y))$. We try to polish (2.25) further by applying additional admissible transformations of the parameters (x,y) and the variable z. To this end, observe that

$$\left.\frac{\partial(U,V)}{\partial(x,y)}\right|_{(0,0)} = \frac{6}{F^0_{zzz}} \left.\frac{\partial(F,F_z)}{\partial(x,y)}\right|_{(0,0)} \neq 0.$$

Therefore, there is an invertible smooth local change of x,y-coordinates $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$U \circ \varphi(x,y) = x, \quad \text{and} \quad V \circ \varphi(x,y) = y. \tag{2.27}$$

Therefore

$$F(\,\varphi(x,y), z + z(\,\varphi(x,y)\,)\,) = C(x,y)\,G(x,y,z),$$

where

$$G(x,y,z) = x + y\,z + z^3 + z^4\,R(x,y,z),$$

with $R(x,y,z) = R_4(\varphi(x,y),z)$. In other words: G is a local model of F. □

2.6 Time dependent contours

If we allow the direction of projection to change over time, or, equivalently, fix the direction of projection and allow the surface to depend on time, the implicit function, defining the surface, depends on four variables (x,y,z,t), where t corresponds to time. In this situation, we expect one of the conditions (2.10) and (2.11) to be violated at isolated values of (x,y,z,t). See also the description of evolving contours in Section 2.2. In this section, we derive approximate local models for these degenerate situations, cf. Figures 2.4, 2.3 and 2.5.

As said before, we consider a smooth function $F : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$, which defines a time-dependent surface $S_t = (F^t)^{-1}(0)$, where $F^t : \mathbb{R}^3 \to \mathbb{R}$ is defined by $F^t(x,y,z) = F(x,y,z,t)$.

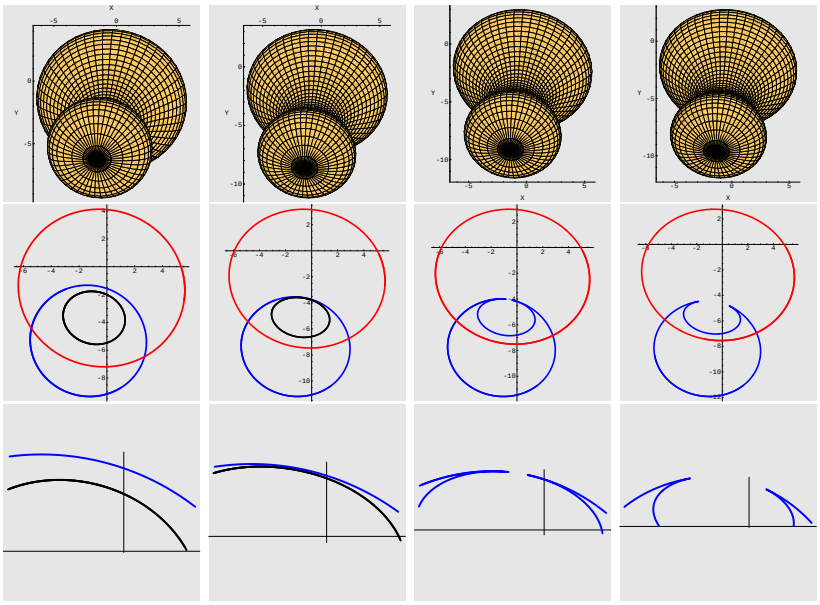


Figure 2.6: A beak-to-beak bifurcation. Top row: a sequence of views of a smooth surface. Middle row: the corresponding apparent contours. Bottom row: blow-up of the apparent contour near the bifurcation event.

2.6.1 Lips and beak-to-beak bifurcations

The occurrence of lips and beak-to-beak bifurcations are associated with a violation of condition (2.10).

Proposition 8. *Let* p *be a non-degenerate singular point of the contour generator for* $t = t_0$*, i.e.,*

$$F(p, t_0) = F_z(p, t_0) = F_{zz}(p, t_0) = \left.\frac{\partial(F, F_z)}{\partial(x, y)}\right|_{(p,t_0)} = 0, \text{ and } F_{zzz}(p, t_0) \neq 0, \tag{2.28}$$

and

$$\Sigma(p) \neq 0,$$

where $\Sigma(p)$ *is as in (2.8) or (2.9). Moreover, assume that the matrix*

$$\begin{pmatrix} F^0_x & F^0_y & F^0_t \\ F^0_{xz} & F^0_{yz} & F^0_{tz} \end{pmatrix} \quad \textit{has rank two.} \tag{2.29}$$

Then the surface has a local model at (p, t_0) *of the form*

$$G(x, y, z, t) = x + \big(\sigma y^2 + \alpha(x, t)\big)\, z + z^3 + z^4 R(x, y, z, t),$$

where $\sigma = \operatorname{sign}(\Sigma(p))$*, with* $\Sigma(p)$ *defined by (2.8) or (2.9),* R *is a smooth function for* (x, y, z, t) *near* $(0, 0, 0, 0)$*, and* $\frac{\partial \alpha}{\partial t}(0, 0) \neq 0$*.*

Remark 9. Before giving the proof, we observe that the contour generator is determined by the equations $G = G_z = 0$, i.e.,

$$\begin{aligned} x + (\sigma y^2 + \alpha(x, y))\, z + z^3 &= O(z^4), \\ (\sigma y^2 + \alpha(x, y))\, z + 3z^2 &= O(z^3). \end{aligned}$$

Solving this system for x and y yields

$$\begin{aligned} x &= 2\, z^3 + O(z^4) \\ \sigma y^2 + 3\, z^2 &= -\alpha(x, t) + O(z^3). \end{aligned}$$

Since $\alpha(0, 0) = 0$, it is easy to see that the singularity of the contour generator for $t = 0$ is of saddle type if $\sigma = -1$, and corresponds to an extremum if $\sigma = +1$. The former case corresponds to a beak-to-beak bifurcation, the latter to a lips bifurcation. We have a scenario as depicted in Figure 2.3 and 2.4, respectively, where it depends on the sign of $\frac{\partial \alpha}{\partial t}(0, 0)$ whether we have to read the Figure from left-to-right or from right-to-left, as t passes zero from negative to positive values.

Proof. For ease of notation we assume that $p = (0, 0, 0)$ and $t_0 = 0$. Our approach is as in the derivation of the local normal form near a cusp point, i.e., we again consider the equation

$$F_{zz}(x, y, z, t) = 0. \tag{2.30}$$

cf. (2.24). Since $F_{zzz}(0, 0, 0, 0) \neq 0$, there is a locally unique solution $z = \zeta(x, y, t)$ of (2.30), with $\zeta(0, 0, 0) = 0$. Also in this situation, Taylor expansion yields:

$$F(x, y, z+z(x, y), t) = C(x, y, t)\big(U(x, y, t)+V(x, y, t)\, z+z^3+z^4\, R_4(x, y, z, t)\big),$$

where R_4 is a smooth function, and

$$\begin{aligned} C(x, y, t) &= \tfrac{1}{6} F_{zzz}(x, y, z(x, y, t)). \\ U(x, y, t) &= F(x, y, z(x, y, t), t)/C(x, y, t) \\ V(x, y, t) &= F_z(x, y, z(x, y, t), t)/C(x, y, t) \end{aligned}$$

Also in this case $C(0,0,0) \neq 0$, and $U(0,0,0) = V(0,0,0) = 0$. To avoid superscripts, we introduce the functions $u, v : \mathbb{R}^2 \to \mathbb{R}$, defined by $u(x,y) = U(x,y,0)$, and $v(x,y) = V(x,y,0)$. In this case we have

$$\left.\frac{\partial(U,V)}{\partial(x,y)}\right|_{(0,0)} = \frac{6}{F^0_{zzz}} \left.\frac{\partial(F,F_z)}{\partial(x,y)}\right|_{(0,0)} = 0, \tag{2.31}$$

so, unlike the situation at a cusp point, we don't have the normal form (x,y) for the pair (u,v). Yet, we can obtain a normal form as stated in the Proposition. To this end introduce the map $\Psi : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$, defined by

$$\Psi(x,y,t) = (U(x,y,t), V(x,y,t)). \tag{2.32}$$

It follows from (2.32) that the Jacobian determinant $J = U_x V_y - U_y V_x$ of Ψ satisfies

$$J(0,0,0) = 0. \tag{2.33}$$

Since, moreover,

$$(u^0_x, u^0_y) = \frac{6}{F^0_{zzz}} (F^0_x, F^0_y) \neq (0,0), \tag{2.34}$$

the derivative of the map $\psi : \mathbb{R}^2 \to \mathbb{R}^2$, defined by $\psi(x,y) = (u(x,y), v(x,y))$ at $(0,0) \in \mathbb{R}^2$ has rank one.

This observation brings us to the context of Singularity Theory, in particular the theory of normal forms of generic maps from the plane to the plane. Whitney's paper [70] presents a local normal form for the two types of generic singularities of this type of maps, viz. the fold and the cusp. As said before, although, or perhaps, because, this paper uses elementary tools, the technical details are quite involved. We refer the reader to Appendix C for a rather self-contained derivation of the Whitney fold, fine-tuned to our current context.

To make the link with Whitney's result more precise, observe that the Jacobian determinant j of ψ satisfies the identity $j(x,y) = J(x,y,0)$. Since $\nabla u(0,0) \neq (0,0)$, cf. (2.34), Proposition 19, or rather its parametrized version presented in Section C.3, guarantees the existence of a change of parameters $\varphi(x,y,t)$ of the form $\varphi(x,y,t) = (\cdot,\cdot,t)$, such that

$$U \circ \varphi(x,y,t) = x, \quad \text{and} \quad V \circ \varphi(x,y,t) = \sigma y^2 + \alpha(x,t).$$

Here α is a smooth function with $\alpha(0,0) = 0$, and

$$\sigma = \operatorname{sign}(\langle \nabla u(0,0), \nabla j(0,0)^{\perp} \rangle) = \operatorname{sign}(-u^0_x j^0_y + u^0_y j^0_x).$$

A straigthforward, but tedious computation shows that

$$-u^0_x j^0_y + u^0_y j^0_x = \Sigma(p),$$

where $\Sigma(p)$ is defined by (2.8) or (2.9). Therefore,

$$F(\,\varphi(x,y,t), z + z(\,\varphi(x,y,t)\,)\,) = C(x,y,t)\,G(x,y,z,t),$$

where

$$G(x,y,z) = x + (\sigma y^2 + \alpha(x,t))\,z + z^3 + z^4\,R(x,y,z),$$

with $R(x,y,z) = R_4(\varphi(x,y,t),z)$. In other words: G is a local model of F. □

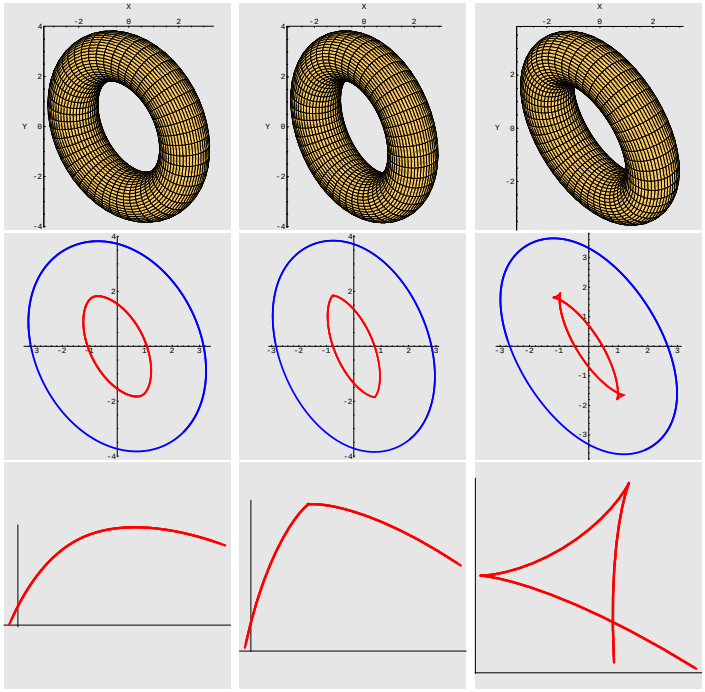


Figure 2.7: A swallowtail bifurcation. Top row: a sequence of views of a smooth surface. Middle row: the corresponding apparent contours. Bottom row: blow up of the apparent contour near the bifurcation event.

2.6.2 The swallowtail bifurcation

In this section we study the discontinuous change of the apparent contour associated with a violation of (2.11), whereas (2.10) still holds. In this case the contour generator is regular.

Proposition 10. *Let* $F : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ *be such that at* $(p, t_0) \in \mathbb{R}^3 \times \mathbb{R}$ *the following conditions are satisfied:*

$$F(p, t_0) = F_z(p, t_0) = F_{zz}(p, t_0) = F_{zzz}(p, t_0) = 0. \tag{2.35}$$

If at (p, t_0) *the generic conditions:*

$$F_{zzzz}(p, t_0) \neq 0, \left.\frac{\partial(F, F_z)}{\partial(x, y)}\right|_{(p,t_0)} \neq 0, \text{ and } \left.\frac{\partial(F, F_z, F_{zz})}{\partial(x, y, t)}\right|_{(p,t_0)} \neq 0. \tag{2.36}$$

are satisfied, then F *has a local model at* (p, t_0) *of the form*

$$G(x, y, z, t) = x + y\, z + w(x, y, t)\, z^2 + z^4 + z^5\, R(x, y, z, t), \tag{2.37}$$

with $w(0, 0, t) = \alpha\, t + O(t^2)$*, where*

$$\alpha = \frac{4!}{\Delta_0\, F^0_{zzzz}} \left.\frac{\partial(F, F_z, F_{zz})}{\partial(x, y, t)}\right|_{(0,0,0)} \neq 0. \tag{2.38}$$

In particular, there is a time-dependent change of coordinates $h^t : \mathbb{R}^2 \to \mathbb{R}^2$*, mapping the apparent contour of* S^t *onto the apparent contour of* $(G^t)^{-1}(0)$*, which has a parametrization of the form* $z \mapsto (\, x(z, t), y(z, t)\,)$*, where:*

$$\begin{aligned} x(z, t) &= w^0(t)\, z^2 - 2w^0(t)\, w^0_x(t)\, z^3 - (3 + w^0(t)\, W(t))\, z^4 + X(z, t)\, z^5 \\ y(z, t) &= -2w^0(t)z + 4w^0(t)\, w^0_y(t)\, z^2 - (2 + w^0(t)\, W(t))\, z^3 + Y(z, t)\, z^4, \end{aligned}$$

with

$$\begin{aligned} w^0(t) &= w(0, 0, t) = \alpha\, t + O(t^2) \\ w^0_x(0) &= -\frac{4!}{\Delta_0\, F^0_{zzzz}} \left.\frac{\partial(F_z, F_{zz})}{\partial(x, y)}\right|_0, \\ w^0_y(0) &= \frac{4!}{\Delta_0\, F^0_{zzzz}} \left.\frac{\partial(F, F_{zz})}{\partial(x, y)}\right|_0, \\ W(0) &= w^0_x + 4\,(w^0_y)^2. \end{aligned}$$

Remark 11. The parametrization of the apparent contour is of the form

$$\begin{aligned} x(z, t) &= \big(\alpha\, t + O(t^2)\big)\, z^2 + O(t)\, z^2 - \big(3 + O(t)\big)\, z^4 + O(z^5), \\ y(z, t) &= \big(-2\,\alpha\, t + O(t^2)\big)\, z + O(t)\, z^2 - \big(2 + O(t^2)\big)\, z^3 + O(z^4). \end{aligned}$$

For $\alpha > 0$, the evolution of the apparent contour is as the one depicted in Figure 2.5, whereas for $\alpha < 0$ the evolution corresponds to reading the latter figure from right to left.

Proof. As before, we assume that $p = (0,0,0)$ and $t_0 = 0$. As in the Tschirnhausen transformation, we try to transform the cubic term away by considering the equation

$$F_{zzz}(x,y,z,t) = 0. \tag{2.39}$$

Since $F_{zzzz}(0,0,0,0) \neq 0$, there is a locally unique solution $z = \zeta(x,y,t)$ of (2.39), with $\zeta(0,0,0) = 0$. Also in this situation, Taylor expansion yields:

$$F(x,y,z+\zeta(x,y,t),t) = C(U + Vz + Wz^2 + z^4 + z^5 R(x,y,z,t)),$$

where R is a smooth function, and

$$\begin{aligned}
C = C(x,y,t) &= \tfrac{1}{4!} F_{zzzz}(x,y,\zeta(x,y,t),t),\\
U = U(x,y,t) &= F(x,y,\zeta(x,y,t),t)/C(x,y,t),\\
V = V(x,y,t) &= F_z(x,y,\zeta(x,y,t),t)/C(x,y,t),\\
W = W(x,y,t) &= \tfrac{1}{2} F_{zz}(x,y,\zeta(x,y,t),t)/C(x,y,t).
\end{aligned}$$

In particular, $U(0,0,0) = 0$, $V(0,0,0) = 0$, $W(0,0,0) = 0$, and $C(0,0,0) \neq 0$. Since, according to (2.35):

$$\left.\frac{\partial(U,V)}{\partial(x,y)}\right|_{(0,0,0)} = \frac{4!}{F^0_{zzzz}} \left.\frac{\partial(F,F_z)}{\partial(x,y)}\right|_{(0,0,0,0)} = \frac{4!\,\Delta_0}{F^0_{zzzz}} \neq 0,$$

we see that the map $\psi : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2 \times \mathbb{R}$, defined by

$$\psi(x,y,t) = (U(x,y,t), V(x,y,t), t),$$

is a local diffeomorphism, which has an inverse of the form

$$\varphi(x,y,t) = (\xi(x,y,t), \eta(x,y,t), t),$$

i.e.,

$$U \circ \varphi(x,y,t) = x \quad \text{and} \quad V \circ \varphi(x,y,t) = y. \tag{2.40}$$

Putting

$$\Phi(x,y,z,t) = (\xi(x,y,t), \eta(x,y,t), z, t),$$

we see that Φ is an admissible transformation bringing F into the normal form $G = F \circ \Phi$, which is of the form

$$G(x,y,z,t) = x + y\,z + w(x,y,t)z^2 + z^4 + O(|z|^5),$$

where

$$w(x,y,t) = W(\xi(x,y,t),\eta(x,y,t),t) = F_{zz}(\xi(x,y,t),\eta(x,y,t),t).$$

Therefore $w(0,0,0) = 0$. Furthermore,

$$w_t(0,0,0) = \frac{4!}{F^0_{zzzz}}\,(F^0_{xzz}\,\xi^0_t + F^0_{yzz}\,\eta^0_t + F^0_{tzz})$$

From (2.40) we derive

$$\begin{aligned} U^0_x\,\xi^0_t + U^0_y\,\eta^0_t + U^0_t &= 0,\\ V^0_x\,\xi^0_t + V^0_y\,\eta^0_t + V^0_t &= 0. \end{aligned}$$

In other words:

$$\begin{pmatrix} F^0_x & F^0_y \\ F^0_{xz} & F^0_{yz} \end{pmatrix} \begin{pmatrix} \xi^0_t \\ \eta^0_t \end{pmatrix} = \begin{pmatrix} -F^0_t \\ -F^0_{zt} \end{pmatrix}$$

Using Cramer's rule, we obtain

$$\xi^0_t = -\frac{1}{\Delta_0}\begin{vmatrix} F^0_t & F^0_y \\ F^0_{zt} & F^0_{yz} \end{vmatrix}, \quad \text{and} \quad \eta^0_t = -\frac{1}{\Delta_0}\begin{vmatrix} F^0_x & F^0_t \\ F^0_{xz} & F^0_{zt} \end{vmatrix}.$$

Therefore,

$$\begin{aligned} w^0_t &= -\frac{4!}{\Delta_0\,F^0_{zzzz}}\Big(F^0_{xzz}\begin{vmatrix} F^0_t & F^0_y \\ F^0_{zt} & F^0_{yz} \end{vmatrix} + F^0_{yzz}\begin{vmatrix} F^0_x & F^0_t \\ F^0_{xz} & F^0_{zt} \end{vmatrix} - F^0_{tzz}\begin{vmatrix} F^0_x & F^0_y \\ F^0_{xz} & F^0_{yz} \end{vmatrix}\Big)\\ &= \frac{4!}{\Delta_0\,F^0_{zzzz}}\begin{vmatrix} F^0_x & F^0_y & F^0_t \\ F^0_{xz} & F^0_{yz} & F^0_{zt} \\ F^0_{xzz} & F^0_{yzz} & F^0_{zzt} \end{vmatrix}\\ &= \frac{4!}{\Delta_0\,F^0_{zzzz}}\;\frac{\partial(F,F_z,F_{zz})}{\partial(x,y,z)}\Big|_0. \end{aligned} \tag{2.41}$$

By a similar computation we obtain:

$$w^0_x = -\frac{4!}{\Delta_0\,F^0_{zzzz}}\;\frac{\partial(F_z,F_{zz})}{\partial(x,y,z)}\Big|_0. \tag{2.42}$$

and

$$w^0_y = \frac{4!}{\Delta_0\,F^0_{zzzz}}\;\frac{\partial(F,F_{zz})}{\partial(x,y,z)}\Big|_0. \tag{2.43}$$

Finally, we can solve x and y as functions of (z,t) from the set of equations $G(x,y,z,t) = G_z(x,y,z,t) = 0$, with G as in (2.37). A straightforward computation yields:

$$\begin{aligned}
x(0,t) &= 0\\
y(0,t) &= 0\\
x_z(0,t) &= 0\\
y_z(0,t) &= -2\,w(0,0,t)\\
x_{zz}(0,t) &= 2\,w(0,0,t)\\
y_{zz}(0,t) &= 8\,w(0,0,t)\,w_y(0,0,t)\\
x_{zzz}(0,t) &= -12\,w(0,0,t)\,w_y(0,0,t)\\
y_{zzz}(0,t) &= -12\,\big(2+4\,w(0,0,t)\,w_y(0,0,t)^2+2\,w(0,0,t)^2\,w_{yy}(0,0,t)+\\
&\quad w(0,0,t)\,w_x(0,0,t)\big)\\
x_{zzzz}(0,t) &= 24\,\big(3+4\,w(0,0,t)\,w_y(0,0,t)^2+2\,w(0,0,t)^2\,w_{yy}(0,0,t)+\\
&\quad w(0,0,t)\,w_x(0,0,t)\big)
\end{aligned}$$

The low order Taylor expansions of $x(z,t)$ and $y(z,t)$, as stated in Proposition 10 are obtained by combining these expressions with the identities for w_t^0, w_x^0, and w_y^0 in (2.41), (2.42), and (2.43), respectively. □

Chapter 3

Contour tracing

3.1 Introduction

In this chapter we will introduce a new curve tracing algorithm with topological guarantees. Our goal is to approximate the contour generator by a piecewise linear curve. This initial approximation can then be maintained under some time-dependent view. To this end the singularities of the contour generator for an evolving view or surface can be precomputed using interval analysis. Since the topology doesn't change between these singularities, the initial contour generator can be updated continuously, until we reach a time where a singularity arises. The local model at this singularity indicates how the topology has to be updated. See Section 2.4 in the previous chapter.

Note that for a singularity of the contour generator of a time-dependent surface, we have

$$\begin{pmatrix} F(x,y,z,t) \\ F_z(x,y,z,t) \\ F_{zz}(x,y,z,t) \\ \Delta(x,y,z,t) \end{pmatrix} = 0,$$

with Δ as in Proposition 2.

These singularities can therefore be considered as the zeroes of a function from $\mathbb{R}^4$ to $\mathbb{R}^4$. Using the Interval Newton Method, we can find all t for which a singularity occurs.

For the initial contour generator, generically there are no singularities. The construction consists of two steps.

Firstly, for each component of the contour generator we have to find an initial point to start the tracing process. Interval analysis will enable us to find points on all components of the contour generator. These (regular) points serve as starting points for the tracing process.

Secondly, we trace the component by stepping along the contour generator. For each starting point we trace the component, by moving from a point p^i to the next point p^{i+1}. We take a small step in the direction of the tangent to the contour generator at p^i. Then, we move the resulting point back to the contour generator, giving us p^{i+1}. An interval test guarantees that we stay on the same component, without skipping a part. If the test fails, we decrease the step size and try again, until the interval test succeeds. If we reach the initial point p^0, the component is fully traced.

3.2 Finding initial points

To be able to trace all components of the contour generator, we have to find at least one point on each component from where we can start the tracing process.

A point p on the contoru generator satisfies both $F(p) = 0$ and $F_z(p) = 0$. A tangent vector to the contour generator at p can be found by computing

$$w(p) = \nabla F(p) \wedge \nabla F_z(p) = \begin{pmatrix} F_y F_{zz} \\ -F_x F_{zz} \\ F_x F_{yz} - F_y F_{xz} \end{pmatrix}.$$

Since the components of the contour generator are bounded, closed curves, there are at least two points on each component where the x-component of $w(p)$ disappears, i.e. where $F_y F_{zz} = 0$.

Let $R, S \colon \mathbb{R}^3 \to \mathbb{R}^3$ be the functions

$$R(p) = \begin{pmatrix} F(p) \\ F_z(p) \\ F_y(p) \end{pmatrix} \text{ and } S(p) = \begin{pmatrix} F(p) \\ F_z(p) \\ F_{zz}(p) \end{pmatrix}$$

The Interval Newton Method finds all roots of R and S. These roots result in a list of (regular) initial points. During the tracing process we can identify which initial points lie on the same component.

3.3 Tracing step

Starting at an initial point we now trace the component by taking small steps along the contour generator in the direction of the tangent vector. To control the step size we will use the normalized tangent vectors. The step size is crucial in guaranteeing correct topology of the approximation.

Let $T(x)$ be the normalized vector field

$$T(x) = \frac{\nabla F(x) \wedge \nabla F_z(x)}{\|\nabla F(x) \wedge \nabla F_z(x)\|}.$$

For x on the contour generator, $T(x)$ is a tangent vector at x. From p^i we first move to $q^0 = p^i + \delta T(p^i)$, where δ determines the step size. To move back to the contour generator, we alternately move towards $F = 0$ and $F_z = 0$ by replacing q^i by

$$\begin{cases} q^{i+1} = q^i - \dfrac{F(q^i)\nabla F(q^i)}{\|\nabla F(q^i)\|^2} & \text{towards } F \\ q^{i+2} = q^{i+1} - \dfrac{F_z(q^{i+1})\nabla F_z(q^{i+1})}{\|\nabla F_z(q^{i+1})\|^2} & \text{towards } F_z \end{cases}$$

until $\|q^{i+2} - q^i\|$ is sufficiently small. The resulting point is the next point on the contour generator, p^{i+1}. For this new point we perform the interval test (explained below) to determine whether $p^i p^{i+1}$ is a good approximation of the contour generator. If not, we decrease δ (e.g. by setting it to $\delta/2$), and repeat the tracing step from p^i. To prevent δ from decreasing to 0 we multiply it by 1.2 after each successful step.

3.3.1 Interval test

The main problem of existing tracing methods is that for a fixed step size, there is always a possibility of accidentally jumping to another component of the contour generator, or of skipping a part (Figure 3.1). This prevents topologically correct approximation of curves. Using interval arithmetic we can ensure that the step size is small enough to guarantee correct topology.

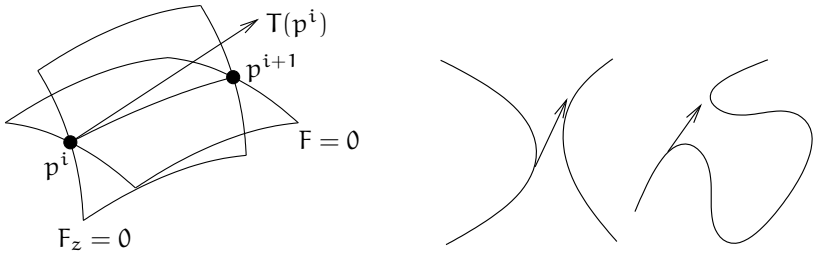


Figure 3.1: Left: $T(p^i)$ is a tangent to the intersection of $F = 0$ and $F_z = 0$. Right: a fixed step size can miss part of the contour generator.

To assure that $p^i p^{i+1}$ is a good approximation for the contour generator, first we construct a sphere S with centre p^i, that contains p^{i+1}. Then we

take the bounding box B of S:

$$B = [p_x^i - D, p_x^i + D] \times [p_y^i - D, p_y^i + D] \times [p_z^i - D, p_z^i + D],$$

where $D = \|p^{i+1} - p^i\|$. Over this box we compute the interval

$$I = \langle T(p^i), T(B) \rangle,$$

where $T(B)$ contains all normalized vectors $\dfrac{\nabla F(s) \wedge \nabla F_z(t)}{\|\nabla F(s) \wedge \nabla F_z(t)\|}$, with $s, t \in B$.

Lemma 12. *If $I > \frac{1}{2}\sqrt{2}$, (i.e. $\forall i \in I\colon i > \frac{1}{2}\sqrt{2}$), then the part of the contour generator within S consists of a connected part of a single component of the contour generator (Figure 3.2).*

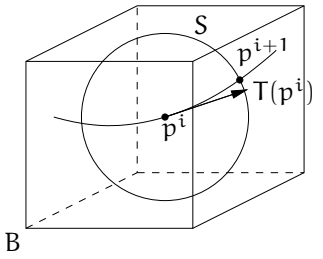


Figure 3.2: The sphere containing the line segment, and its bounding box.

Proof. We define $G(x) = \langle x - p^i, T(p^i) \rangle$. The level sets of G are planes perpendicular to $T(p^i)$. Let $f\colon \mathbb{R}^3 \to \mathbb{R}^3$ be the function

$$f(x) = \begin{pmatrix} F(x) \\ F_z(x) \\ G(x) \end{pmatrix}.$$

Suppose there are two points x and y of the contour generator in B, lying in a plane perpendicular to $T(p^i)$, i.e. $f(x) = f(y) = (0, 0, \theta)$ for some θ. According to the Mean Value Theorem, there are points s and t on xy where

$$\begin{pmatrix} \nabla F(s) \\ \nabla F_z(t) \\ T(p^i) \end{pmatrix} (y - x) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

For $x \neq y$, we find that $\langle \nabla F(s) \wedge \nabla F_z(t), T(p^i) \rangle = 0$ (all vectors in a plane perpendicular to $y - x$). Since $s, t \in B$ this contradicts the interval test.

Therefore, within box B each plane perpendicular to $T(p^i)$ contains at most one point of the contour generator.

The interval condition $I > \frac{1}{2}\sqrt{2}$ implies that the angle between $T(p^i)$ and $T(x)$ is at most $\frac{\pi}{4}$ for all $x \in B$ and therefore also for $x \in \Gamma \cap B$. The contour generator lies in a cone C around $T(x)$ (fig. 3.3) with top angle $\frac{\pi}{2}$, for if it leaves the cone at point a, then $\langle T(p^i), T(a)\rangle$ would be smaller than $\frac{1}{2}\sqrt{2}$. It can only leave the sphere in $S \cap C$. In this part of the sphere the contour generator cannot re-enter S, because that would require an entry point b where $\langle T(p^i), T(b)\rangle < \frac{1}{2}\sqrt{2}$. Therefore, there can only be a single connected component of the contour generator within sphere S. □

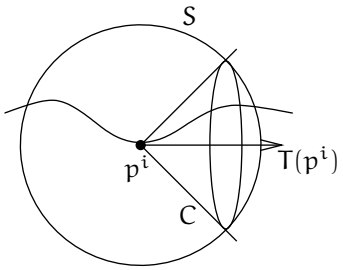


Figure 3.3: The contour generator lies within a cone.

If the interval test succeeds, we can use the same sphere S to remove redundant points from the initial point list. If there are points in the list that lie within S, they must be part of the component we are tracing, so they can be discarded.

To test whether the component is fully traced, we test if the current sphere S overlaps the sphere of the initial point p^0. If it does, testing

$$\langle p^{i+1} - p^i, p^0 - p^i\rangle > 0$$

tells us whether we are done with this component (otherwise we just started the trace and are still moving away from p^0).

The interval bound is valid if the centre of the cube is on the contour generator. Since p^i is in general close to, but not on the contour generator, in practice we take a slightly larger bound (e.g. by multiplying the radius by 1.1). Also, D should be slightly larger than $\|p^{i+1} - p^i\|$, to prevent p^{i+1} from being too close to other parts of the contour generator outside S. Again, multiplying by 1.1 should suffice to prevent rounding errors.

For algorithms on implicit surfaces it is difficult to derive a complexity bound, because timing results depend not only on the complexity of the

surface, but also on the behaviour of the function outside the surface, and on the accuracy of the convergent inclusion functions. For algebraic surfaces it might be possible to derive the complexity, similar to [2].

3.4 Evolving surfaces

Time-dependent surfaces occur for example if the viewpoint moves along a fixed path, or if the surface changes shape under a continuous deformation (morphing). We assume that the transformation is also a smooth function, e.g. rotation. For a time-dependent surface $F(x, y, z, t) = 0$ there is no need to compute the initial points on the contour generator for each t. Instead, we can trace the initial points as time evolves. Recall that the initial points are zeroes of the functions $R, S\colon \mathbb{R}^4 \to \mathbb{R}^3$:

$$R(x, y, z, t) = \begin{pmatrix} F(x, y, z, t) \\ F_z(x, y, z, t) \\ F_y(x, y, z, t) \end{pmatrix} \text{ and } S(x, y, z, t) = \begin{pmatrix} F(x, y, z, t) \\ F_z(x, y, z, t) \\ F_{zz}(x, y, z, t) \end{pmatrix}$$

For time-dependent surfaces, these are 1-dimensional curves in $\mathbb{R}^4$. These curves can be traced similarly to the contour generator itself, i.e. by stepping along the curve using a step size such that the interval inequality ($I > \frac{1}{2}\sqrt{2}$) is satisfied.

The tangent vector to the curve is perpendicular to the gradients of the three components of R and of S. As time evolves, generically initial points are created and destroyed in pairs. This happens when the tangent vector is perpendicular to the t-axis. Otherwise, according to the implicit function theorem there exists a unique solution.

For R the tangent vector can be found using

$$\operatorname{Ker} \begin{bmatrix} F_x & F_y & F_z & F_t \\ F_{xz} & F_{yz} & F_{zz} & F_{zt} \\ F_{xy} & F_{yy} & F_{yz} & F_{yt} \end{bmatrix}$$

A solution (t_1, t_2, t_3, t_4) in the kernel with $\sum t_i = 1$ can be found by adding a row $[1, 1, 1, 1]$ to the matrix, resulting in a 4×4-matrix M, provided $\det(M) \neq 0$. Since $R(x, y, z, t) = 0$ we have $F_z = F_y = 0$. Using Cramer's rule to solve

$$\begin{pmatrix} F_x & F_y & F_z & F_t \\ F_{xz} & F_{yz} & F_{zz} & F_{zt} \\ F_{xy} & F_{yy} & F_{yz} & F_{yt} \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

we find a tangent vector for the curve:

$$\left(\begin{vmatrix} 0 & 0 & F_t \\ F_{yz} & F_{zz} & F_{zt} \\ F_{yy} & F_{yz} & F_{yt} \end{vmatrix}, -\begin{vmatrix} F_x & 0 & F_t \\ F_{xz} & F_{zz} & F_{zt} \\ F_{xy} & F_{yz} & F_{yt} \end{vmatrix}, \begin{vmatrix} F_x & 0 & F_t \\ F_{xz} & F_{yz} & F_{zt} \\ F_{xy} & F_{yy} & F_{yt} \end{vmatrix}, -\begin{vmatrix} F_x & 0 & 0 \\ F_{xz} & F_{yz} & F_{zz} \\ F_{xy} & F_{yy} & F_{yz} \end{vmatrix}\right)$$

This tangent vector is perpendicular to the t-axis if

$$\begin{pmatrix} F \\ F_z \\ F_y \\ F_x \end{pmatrix} = 0 \text{ or } \begin{pmatrix} F \\ F_z \\ F_y \\ F_{yz}^2 - F_{yy}F_{zz} \end{pmatrix} = 0$$

The first equality has no solutions, since the surface does not contain singularities.

Similarly, for S we have:

$$\operatorname{Ker} \begin{bmatrix} F_x & F_y & F_z & F_t \\ F_{xz} & F_{yz} & F_{zz} & F_{tz} \\ F_{xzz} & F_{yzz} & F_{zzz} & F_{tzz} \end{bmatrix}$$

For the tangent vector of S we find:

$$\left(\begin{vmatrix} F_y & 0 & F_t \\ F_{yz} & 0 & F_{tz} \\ F_t & F_{tz} & F_{tzz} \end{vmatrix}, -\begin{vmatrix} F_x & 0 & F_t \\ F_{xz} & 0 & F_{tz} \\ F_{xzz} & F_{zzz} & F_{tzz} \end{vmatrix}, \begin{vmatrix} F_x & F_y & F_t \\ F_{xz} & F_{yz} & F_{tz} \\ F_{xzz} & F_{yzz} & F_{zzz} \end{vmatrix}, -\begin{vmatrix} F_x & F_y & 0 \\ F_{xz} & F_{yz} & 0 \\ F_{xzz} & F_{yzz} & F_{zzz} \end{vmatrix}\right)$$

This vector is perpendicular to the t-axis if

$$\begin{pmatrix} F \\ F_z \\ F_{zz} \\ F_{zzz} \end{pmatrix} = 0 \text{ or } \begin{pmatrix} F \\ F_z \\ F_{zz} \\ F_x F_{yz} - F_y F_{xz} \end{pmatrix} = 0$$

The first equality corresponds to swallowtail bifurcations, the second equality corresponds to a singularity of the contour generator.

For dynamic surfaces depending on a single parameter, computing the four-dimensional initial points shows us at which time steps the set of initial points for the contour generator changes. As long as the time parameter t does not pass such a time value, we can adjust the position of the current initial points to the new value of t, by tracing a small part of the four-dimensional curve. For small time steps this requires only a small number of 4D tracing steps. The time to compute the new contour generator is therefore only slightly larger than the time required for the tracing phase,

removing the time required to find the initial points. When t does pass one of the precomputed values, generically a pair of initial points of the contour generator is created or destroyed. In this case we can adjust the set of initial points by adding or removing a pair of points.

Doing more preprocessing, we could also precompute the approximation of the entire 4D curve, enabling us to find the initial points for any fixed t, by intersecting this 4D curve with the t-plane.

For a single parameter function, we can handle evolving surfaces such as level sets of an implicit function, or moving along a one-dimensional view path. Arbitrary view directions require two parameters. In this case we also have other types of bifurcations, such as the butterfly bifurcation. How to handle contour generators of surfaces with more than one parameter remains an open problem.

3.5 Results

We have implemented the computation of the static contour generator, using the interval classes and interval Newton solver from the `c-xsc` [56] library.

The table below shows the results on a Pentium 2.80GHz, running Linux. The implicit surface was transformed with a shearing to change the view direction from the default $(0, 0, 1)$. The resulting contour generators are shown in Figure 3.4, 3.5, and 3.6 for the tangle cube, and in Figure 3.7 and 3.8 for the non-algebraic surface. The table shows the number of contours, the total number of line segments, and timing results for the two phases: finding the initial points, and tracing the contour generator.

Surface	Contours	Segments	Initial pts	Tracing	Total
Tangle (0.095, 0.295, 1)	4	4148	3.6s	2.2s	5.8s
Tangle (0.09, 0.29, 1)	8	3940	3.4s	1.5s	4.9s
Tangle (0.08, 0.25, 1)	8	3184	3.0s	1.0s	4.0s
Non-alg. (0.2, 0.4, 1)	62	11869	22.1s	3.4s	25.5s
Non-alg. (0.8, 0.4, 1)	29	23812	75.3s	6.8s	82.1s

At the moment the (inclusion) functions and derivatives are hard-coded into the implementation, since we do not have an interval newton implementation working together with automatic differentiation. A solution would be to use a code generator that outputs inclusion functions of all derivatives. The lack of such a preprocessor makes experimenting with 4D curves tedious. However, as mentioned in the previous section, the time required for updating the contour generator for small time steps is only slightly larger than the results in the table under 'Tracing'.

The preprocessing requires computing the initial points of the 4D curve. We have computed these points for the tangle cube, for the view direction

moving along the one-dimensional path $(x - tz, y - (0.2 + t)z, z)$. This path includes the views shown in Figure 3.4 and Figure 3.5. We separately solved the three sets of equations from the previous section: $F = F_z = F_y = F_{yz}^2 - F_{yy}F_{zz} = 0$, $F = F_z = F_{zz} = F_{zzz} = 0$ and $F = F_z = F_{zz} = F_x F_{yz} - F_y F_{xz} = 0$. The following table shows the results for these three equations, as well as the total time for computing the initial points of the 4D curve. We solved over the intervals $[0, 1]$ and over the ten subintervals of width 0.1. The final row gives the sum over these ten smaller intervals.

t Interval	$F_{yz}^2 - F_{yy}F_{zz}$	F_{zzz}	$F_x F_{yz} - F_y F_{xz}$	Total
$[0, 1]$	977s	7894s	1972s	10843s
$[0.0, 0.1]$	0.2s	9.0s	13.5s	22.7s
$[0.1, 0.2]$	0.7s	23.9s	21.2s	45.8s
$[0.2, 0.3]$	1.7s	52.4s	27.2s	81.3s
$[0.3, 0.4]$	4.6s	90.8s	30.8s	126.2s
$[0.4, 0.5]$	7.7s	132.5s	34.1s	174.3s
$[0.5, 0.6]$	11.4s	181.7s	40.4s	233.5s
$[0.6, 0.7]$	17.4s	227.6s	51.6s	296.6s
$[0.7, 0.8]$	33.8s	298.2s	67.3s	399.3s
$[0.8, 0.9]$	78.4s	388.6s	93.1s	560.1s
$[0.9, 1.0]$	70.7s	476.2s	127.9s	674.8s
10 intervals	226.6s	1880.9s	507.1s	2614.6

The time for solving over the interval $[0, 1]$ is just over three hours, whereas solving over ten intervals of width 0.1 takes less than 44 minutes. This indicates that the interval Newton implementation in the `c-xsc` library is far from optimal. Results can be greatly improved by subdividing the search space first. For algebraic functions such as the tangle cube, of course other methods for solving systems of equations are available.

3.6 Conclusion and future research

In Chapters 2 and 3 we presented a framework for the analysis of an implicit surface near regular and singular points of its contour generator and apparent contour, and also derived conditions for detecting changes of topology of these visibility features in generic one-parameter families of implicit surfaces.

We developed an algorithm to compute a topologically correct approximation of curves, in particular of the initial contour generator. A dynamic step size, combined with an interval test, guarantees that no part of the contour generator is skipped.

For further experiments we require an interval Newton method working together with automatic differentiation, or a preprocessor generating inclusion functions for the partial derivatives of the implicit function. More

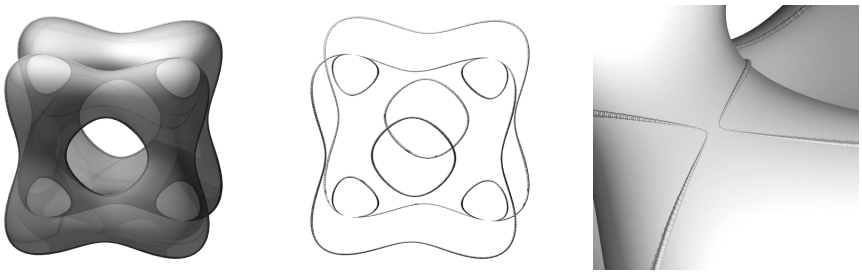


Figure 3.4: Surface with contour generator, contour generator alone, and detail of the contour generator of the tangle cube $x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 10 = 0$. The view direction $(0.095, 0.295, 1)$. The contour generator is plotted as the set of spheres, such that larger spheres indicate a larger step size. The actual contour generator connects the centres of these spheres. The detail shows the contour generator on the top left of the tangle cube, shown from above.

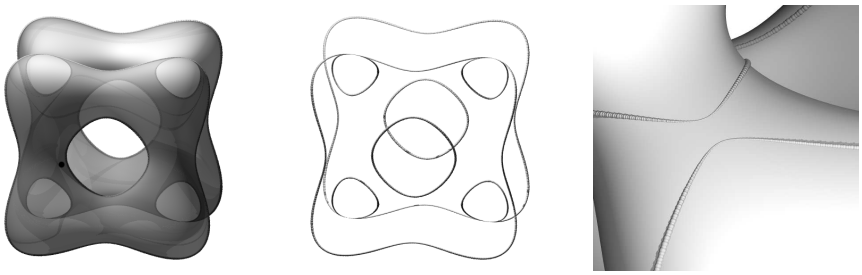


Figure 3.5: Contour generator of the tangle cube, with view direction $(0.09, 0.29, 1)$.

research needs to be done on solving systems of equations. The interval Newton method used is far from optimal. Faster solvers are required for a practical implementation of dynamic contours. In particular, for algebraic surfaces other methods should be considered. For this class of surfaces, it is also easier to examine the singularties, using the theory on local models. This information can be used to construct singular contours.

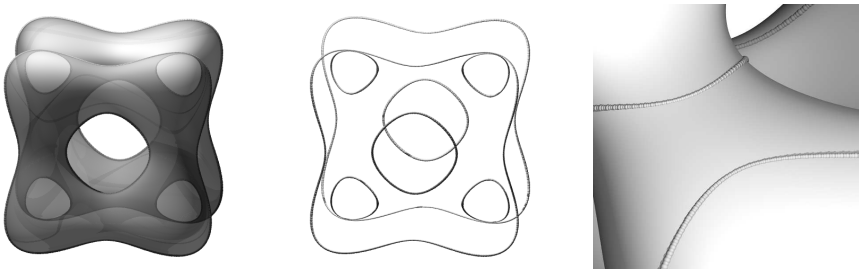


Figure 3.6: Contour generator of the tangle cube, with view direction $(0.08, 0.25, 1)$.

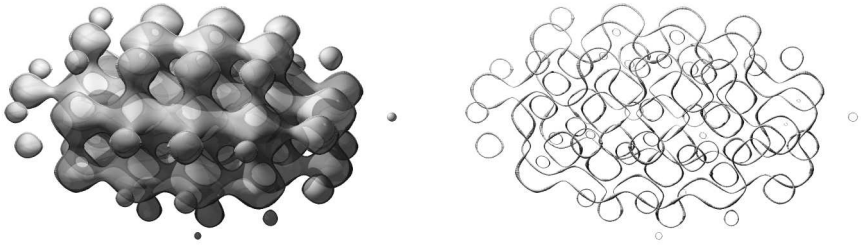


Figure 3.7: Contour generator of the non-algebraic surface $-0.4(\sin(5x) + \sin(5y) + \cos(5z)) + 0.1x^2 + 0.3y^2 + 0.2z^2 - 0.5$, view direction $(0.2, 0.4, 1)$.

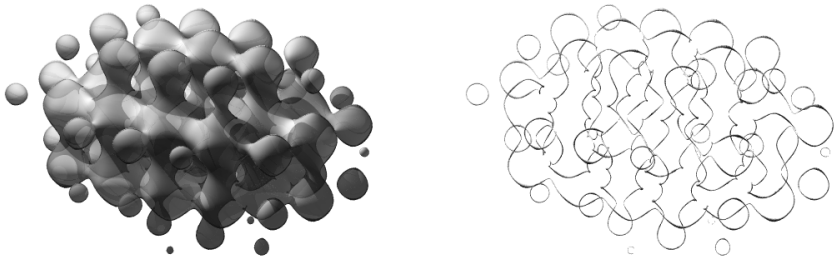


Figure 3.8: Contour generator of the non-algebraic surface, view direction $(0.8, 0.4, 1)$.

Chapter 4

Isotopic Approximation

4.1 Introduction

Implicit functions provide a convenient representation of smooth surfaces. However, piecewise linear approximations are often required, for example for visualization. Meshing algorithms can only compute function values at a finite number of points. Since grid based schemes can miss important details of the surface, correct topology usually cannot be guaranteed. To capture the global properties of implicit surfaces, we somehow need to extract information about the surroundings of these points. Lipschitz conditions give a bound on the gradient and can therefore sometimes discard the neighbourhood of a point. Another tool that can be used is interval arithmetic.

In this chapter we present an algorithm that creates a regularly isotopic approximation of implicit curves in $\mathbb{R}^2$ and of implicit surfaces in $\mathbb{R}^3$. Regularly isotopic means that the approximation is equivalent to the curve or surface under continuous deformation within the embedding space. In particular, the approximation has the same topology as the implicit manifold. Correct topology is important to determine connectedness, and for example in constructing an initial mesh for time-dependent surfaces (e.g. morphing). Our implicit surface meshing algorithm subdivides space using an octree. Interval arithmetic is used to decide which cells of the octree require further subdivision. For each leaf of the resulting octree a local approximation is constructed.

The algorithm given in this chapter falls in the category of enumeration methods. A novelty is that we use a fast and simple interval test to decide which cells need subdivision, in such a way that we can guarantee isotopy. Compared to other enumeration methods it adds little overhead, and is therefore quite fast. In particular, we do not need to compute the critical points of the implicit function.

The octree based subdivision is very flexible. For example we could define a minimal subdivision level to improve the Hausdorff distance between the surface and the approximation. A maximal subdivision level could be used to improve the speed of the meshing process and to handle surfaces with singularities. In this case colour coding of the mesh could be used to identify the (arbitrarily small) regions where the approximation might be topologically incorrect.

In section 4.2 we give an overview of the existing techniques for implicit curve and surface approximation. Section 4.3 explains our approximation algorithm for implicit curves. After giving the algorithm, we will prove that the resulting piecewise linear approximation is isotopic to the curve itself. Since both the algorithm and its proof will be generalized to the three-dimensional case, this section also provides an introduction to the meshing of implicit surfaces. In section 4.4 we provide a first step towards our meshing algorithm, by adapting the well-known marching cubes algorithm to construct isotopic approximations. Section 4.5 extends the previous section by using octrees instead of a regular grid, thereby greatly reducing the complexity of both the algorithm and the resulting mesh. Results of the algorithms are shown in section 4.6. Finally, in section 4.7 we discuss some possible improvements and possible changes to our approximation scheme.

Main contribution This chapter presents a new algorithm to approximate not necessarily algebraic regular implicit curves and surfaces. For surfaces, it is one of the first schemes guaranteeing that the resulting mesh is isotopic to the implicit surface. For curves, it is the first practical algorithm giving this guarantee. Since we do not need to compute the critical points of the implicit function, both algorithms are fast enough to be of practical use. Also, they can easily be adapted to improve the accuracy of the approximation and to deal with singular surfaces. We expect that the algorithm generalizes to isotopic approximation of all codimension one manifolds in Euclidean spaces.

4.2 Related work

Several algorithms exist for the approximation of implicit curves and surfaces, but very few can guarantee topological correctness. Since we can only compute function values at a finite number of points, there is a risk of missing important details of the curve or surface. Some approximation algorithms depend on a user specified value to give a trade-off between accuracy and speed. For these methods it is not clear what level of accuracy is required to capture the topology of the implicit manifold. Algorithms that do guarantee

a topologically correct result depend either on extra information about the underlying implicit function (e.g. algebraicity of surfaces or bounds on the Lipschitz conditions), or on interval arithmetic.

In this section we will give an overview of the different existing techniques for approximation of implicit curves and surfaces.

Curves The schemes for approximation of implicit curves can be roughly divided into continuation methods and adaptive enumeration techniques.

Continuation methods approximate the curve or surface by first finding seed points on all components, and then using these as starting points for tracing the components.

In [16] an implicit curve (in this case a contour in $\mathbb{R}^3$) is approximated using a predictor-corrector method. Initial points are found by shooting random rays to the surface and by moving from the first intersection point along the surface towards the contour. In [59] we adapted this method to guarantee topological correctness of an approximation of the contour generator, but to achieve this the relatively slow interval Newton method was required. Also, a small step size for the tracing process is required to guarantee that the resulting approximation is isotopic to the traced component.

Approximation schemes based on *adaptive enumeration* start with a bounding box and subdivide until the cells are 'small enough', for example by looking at the local curvature. Then, for each cell of the subdivided box a local approximation is constructed. In [48] interval arithmetic is used to get a good approximation of implicit curves. The size of the subdivided cells depends on the geometry of the curve, such that in areas with high curvature the level of detail is also higher. However, to terminate the subdivision process, the method requires user-specified bounds on cell size, and on the variation of the normalized gradient within a cell. Although the edge length varies with the curvature of the curve, it cannot guarantee correct topology.

As in this chapter, [63] uses parametrizability to subdivide a quadtree. This algorithm guarantees topological correctness, but because of a slightly weaker test for parametrizability, it requires a large number of (slow) interval Newton searches for intersection points.

Surfaces A continuation algorithm is given in [35]. Starting with a seed point on the surface, a triangulation is generated by expanding over the surface, along the boundary of the already triangulated part. As with curves, this algorithm depends on seed points, and on a user-specified step size. The triangles of the resulting mesh have more or less the same size, independent

of the geometry of the surface. An improvement of this algorithm using a dynamic triangle size is given in [1].

Enumeration methods use either a regular grid or an octree to subdivide space. Regular grids are used in the well-know marching cubes algorithm [49]. The main purpose of these methods is to mesh sampled voxel data, but they can easily be adapted to approximate implicit surfaces. In [10], cubes are constructed along the surface. Although this saves examining a large number of grid cells, there is a risk of missing components of the implicit surface. Another improvement of marching cubes is given in [22]. The linear variation along an edge used in marching cubes, is extended to trilinear variation over the cube, using the function values at the eight corners of the cube. The algorithm guarantees that the approximation has the same topology as the isosurface of this trilinear function. For implicit surfaces there is no such guarantee.

For dynamic surfaces, critical points and their indices indicate when, where and how the topology changes. This can be used to update the topology of the triangulation, as in [64]. Also, the shrinkwrap algorithm [69] can be adapted to use critical points for updating the topology [15].

An algorithm for isotopic meshing based on critical points is given in [13]. It is based on Morse theory to determine the topology, and therefore requires a priori knowledge of the critical points of the implicit surface, as well as of their indices.

Finally, sampling based algorithms construct a sufficiently dense sample of points on the surface, such that surface reconstruction on this point set results in a homeomorphic approximation. To construct such a sampling, in [14] bounds on the distance to the medial axis are needed, while in [21] the critical points of heightfunctions on the surface and on intersection curves are required. Also, both algorithms assume that the intersection points of a given line segment with the surface can be computed. These Delaunay based algorithms result in a better mesh quality, but for generic implicit surfaces it is very difficult to meet these assumptions.

4.3 Curves

In this section we will introduce an algorithm to construct a piecewise linear approximation of an implicit curve S, such that the approximation and S have the same (regular) isotopy. The construction is based on a quadtree. After subdivision this quadtree is *balanced*, i.e. the tree is further subdivided until two neighbouring squares differ at most a factor two in size. Balancing does not increase the complexity of the tree.

Possible improvements regarding accuracy and singularities will be discussed in Section 4.7 and in the next chapter.

Let $S = F^{-1}(0)$ be a bounded implicit curve, where $F\colon \mathbb{R}^2 \to \mathbb{R}$ is a smooth function and 0 is a regular value of F, i.e. the gradient ∇F is non-zero at every point of the curve. The following algorithm constructs an isotopic piecewise linear approximation of S. Details are explained right after the presentation of the algorithm.

Algorithm ApproximateCurve(F, B)
Input. An implicit function F, and a square bounding box B.
Output. A piecewise linear approximation of the curve $F = 0$.

1. Initialize a quadtree $\mathcal{T}$ to the bounding square B of $F = 0$.
2. Subdivide $\mathcal{T}$ until for all leaves C we have $0 \notin \Box F(C) \vee \langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0$.
3. BalanceQuadtree($\mathcal{T}$)
4. **for** each edge of the quadtree
5. **do if** the signs of F at its two endpoints are opposite
6. **then** construct a vertex at the midpoint of the edge
7. **for** each leaf C of quadtree $\mathcal{T}$
8. **do if** the leaf contains two vertices
9. **then** connect the vertices by a line segment
10. **if** the leaf contains four vertices
11. **then** find the two vertices on the same side and connect each of them to the other adjacent vertex

Note that a single side of a cell may consist of two quadtree edges, if the neighbouring cell along that side is subdivided. See Figures 4.18 and 4.19. Each cell of the quadtree is therefore surrounded by four to eight quadtree edges.

The construction of the approximating edges handles only a few different cases of vertex placement on the boundary of a cell: either two vertices, or four vertices with two of them along one side of the cell. In the correctness proof we will show that these are the only possible cases satisfying our subdivision condition.

Before proving that this algorithm constructs an isotopic approximation, we will show that the resulting quadtree cells are parametrizable and prove that the algorithm terminates by examining the interval condition:

$$0 \notin \Box F(C) \quad \vee \quad \langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0.$$

The first clause discards cells where $0 \notin \Box F(C)$. Note that this discards boxes of which we are certain that they do not contain part of the curve S.

The righthand clause implies that $\langle \nabla F(x), \nabla F(y) \rangle > 0$, for all $x, y \in C$. Hence, the direction of the gradient (and, therefore, of the curve) does not change more than $\pi/2$ over C.

We say the curve S is *parametrizable in direction* v, if every line parallel to v intersects S at most once. This definition extends easily to three dimensions.

If $\langle \Box\nabla F(C), \Box\nabla F(C) \rangle > 0$, at least one of the two terms $\Box F_x(C) \cdot \Box F_x(C)$ and $\Box F_y(C) \cdot \Box F_y(C)$ (where we write F_x for $\frac{\partial F}{\partial x}$) does not contain 0. This implies that F is strictly increasing or decreasing in the x or y direction, and therefore locally (i.e. within this cell) parametrizable in the direction of one of the axes.

When we continue the subdivision process, the squares shrink towards a point. Since S is a regular curve, at least one of the two clauses converges to a non-zero value. Therefore, the subdivision process terminates.

After these observations we can move on to the following theorem:

Theorem 13. *The approximation of* S *constructed by algorithm* ApproximateCurve *is (regularly) isotopic to* S.

Proof. To prove that the approximation is isotopic to the implicit curve C we will proceed as follows:

- Firstly, we construct an approximation using a regular grid, assuming that S satisfies certain constraints, and show that this approximation is isotopic to S.
- Secondly, we will remove the contraints on S.
- Finally, we will show that the approximation constructed by the regular grid is equivalent to the one created by ApproximateCurve. To this end, we will show that further subdivision of the quadtree does not change the isotopy of the corresponding approximation. By repeating the subdivision process until the quadtree is complete, we end up with a regular grid. This grid was already shown to be isotopic to the implicit curve.

Regular grid For the first part of the proof we will start with a grid-based approximation instead of a quadtree. Let G be a regular grid, such that for each cell C we have $0 \notin \Box F(C) \vee \langle \Box\nabla F(C), \Box\nabla F(C) \rangle > 0$. Furthermore we assume that S intersects each edge of G at most once, and that $F \neq 0$ at the nodes of G. These assumptions will be removed later. This implies that we have to construct at most one vertex at each edge of the grid. Due to the inner product constraint, S is parametrizable in the direction of one of

the axes. Therefore we cannot have alternating signs of F at the vertices of C, since F would have to increase along one edge, and decrease along the other parallel edge (Figure 4.1). We conclude that S intersects at most two edges of C. For the approximation we connect the two midpoints of these two edges (if any) by a straight line segment s.

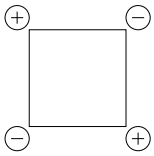


Figure 4.1: A sign configuration contradicting the inner product constraint. F decreases along the top edge and increases along the bottom edge. It is therefore not parametrizable in the x-direction. A similar argument holds for the y-direction.

The curve S is locally parametrizable, and therefore cannot contain closed loops inside a cell. Also, since S is a regular curve, there are no self-intersections. This implies that within the cell C, if there are two intersection points along the edges, the part of the curve inside C is isotopic to a line segment.

In fact we can easily construct a deformation to move the curve locally to the segment s of the approximation. Suppose for example that S is locally parametrizable in the y-direction, i.e. within the grid cell it is of the form $y = f(x)$. If S intersects the left and right edge, the approximation has the same x-domain as S. By linear interpolation in the y-direction we can continuously move this part of S to segment s. If S does not intersect the left and right edge, we may need to stretch/shrink the x-domain first. An example is given in Figure 4.2. By mapping $[0, t]$ linearly to $[0, \frac{1}{2}]$ and $[t, 1]$ to $[\frac{1}{2}, 1]$ the intersection point $(t, 0)$ moves to $(\frac{1}{2}, 0)$. Now, both curves have the same x-domain, so we can do a vertical interpolation again. Note that an edge of the grid maps onto itself. Therefore, we can stitch the local deformations within each square together, resulting in a global deformation that moves the whole implicit curve continuously to the approximation.

Removing constraints In the second part of the proof we remove the constraints on S. The first assumption was that S intersects each edge of the grid at most once. Now, suppose S intersects an edge of the grid more than once. In this case the curve must be parametrizable in the direction perpendicular to this edge, for both adjacent cells. For two adjacent intersection

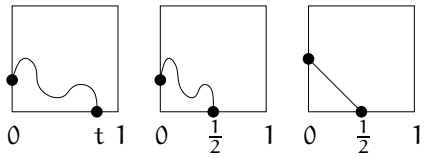


Figure 4.2: Deformation of the curve (left) to the approximation (right).

points α_1 and α_2 we look at the curve between these two points. Using the mean value theorem it is easy to see that the inner product constraint prevents S from bending 'too much' (i.e. more than $\pi/2$). Also, recall that the adjacent cells are squares. Therefore, S cannot leave the two cells between α_1 and α_2 (Figure 4.3). By a local interpolation in the parametrizable direction between S and the edge, we can continuously deform the part of S between α_1 and α_2 towards the edge, and 'push' this part of S through the edge (Figure 4.4), thereby removing the two intersection points. Since we can continue removing pairs of intersection points, S is isotopic to a curve that intersects each edge at most once, and hence it is isotopic to the piecewise linear approximation.

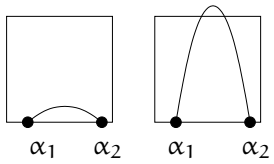


Figure 4.3: Multiple intersection along an edge. The curve on the right has too much curvature to satisfy the interval constraint.

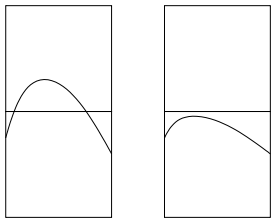


Figure 4.4: Removing a pair of intersection points by pushing the curve through the edge. In the union of the two squares the topology of the curve does not change.

The second assumption was that $\mathsf{F} \neq 0$ on nodes of the grid. If S passes through a node of the grid, we can again deform S, this time by moving it continuously to $\mathsf{F} + \epsilon = 0$. For small ϵ this again yields an isotopy, since S is regular. We can take ϵ arbitrarily small using a symbolic perturbation, by considering F to be strictly positive at a vertex whenever $\mathsf{F} \geq 0$.

Quadtrees Now that we have removed the constraints on F we complete the proof by showing that the quadtree approximation is isotopic to the grid approximation. To this end, we consider a leaf C of the quadtree $\mathcal{T}$, together with its approximation edges. When we split C, the inner product constraint still holds for its four children. Furthermore, since S is parametrizable within C, S is parametrizable in its children, and also in the same direction. The only topological changes can therefore occur if subdivision introduces two new vertices at a single edge of C, as shown in Figure 4.5.

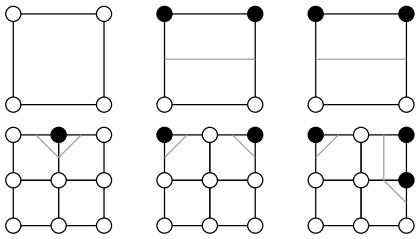


Figure 4.5: Topological changes after subdivision

Since two new vertices were created, the neighbouring cell along the edge containing these vertices is not subdivided. After subdivision of C, the neighbouring cell will detect the two new vertices and connect them either to each other or to two existing vertices. Both types of change correspond to 'pushing' part of the curve through the edge. In the union of the cell and its neighbour the approximation is isotopically unaltered (Figure 4.6).

By repeatedly subdividing the leaves of $\mathcal{T}$ we can turn it into a complete quadtree, with the same approximation as a regular grid. Since this does not change the isotopy of the approximation, the output of algorithm ApproximateCurve is isotopic to S. □

4.4 Marching cubes revisited

For the meshing of implicit surfaces we proceed similarly to the two-dimensional case. As a first step towards our meshing algorithm, we approximate the sur-

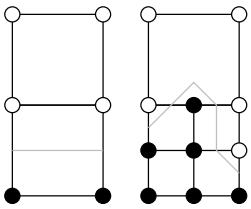


Figure 4.6: After subdivision the topology inside a square changes, but inside the union of both squares it stays the same.

face on a regular grid, assuming only single intersections on the edges. Later, we shall remove these constraints. In the next section we will present our meshing algorithm, using a balanced octree instead of a regular grid.

In this section we introduce a slightly modified version of the well-known marching cubes ($\mathcal{MC}$) algorithm. Recall that $\mathcal{MC}$ subdivides space into a uniform cubic grid, constructing a triangulation for each cell of the grid, depending only on the function sign at its eight vertices. Sometimes the function value at the vertices is used to position the vertex by linear interpolation. Since our main goal is topological correctness we will just put the vertex at the centre of the edge. The $\mathcal{MC}$ algorithm as introduced by Lorensen and Cline [49] cannot guarantee topological correctness. Due to ambiguities in the sign configuration of a cell face, the resulting surface could also contain holes. To fix these problems we will introduce a few assumptions, most of which can later be removed.

Let S be a regular, bounded, implicit surface, given as the zero set of a smooth function $F\colon \mathbb{R}^3 \to \mathbb{R}$, so $S = F^{-1}(0)$. Furthermore, we assume that 0 is a regular value of F, i.e., the gradient ∇F is non-zero at every point of the surface.

For now, we will assume that S does not contain vertices of the uniform grid, and that S intersects each edge of the grid at most once. Furthermore, the following interval condition should hold for each cell C:

$$0 \notin \square F(C) \quad \vee \quad 0 \notin \langle \square \nabla F(C), \square \nabla F(C) \rangle.$$

As in the 2-dimensional case, the righthand clause implies:

$$\forall x, y \in C\colon \langle \nabla F(x), \nabla F(y) \rangle > 0.$$

Again we have that for each cell where $0 \in \square F(C)$ the surface S is parametrizable in the direction of one of the axes.

For the signs at the vertices, there are 14 possibilities (up to rotation, mirroring and change of sign), shown in Figure 4.7.

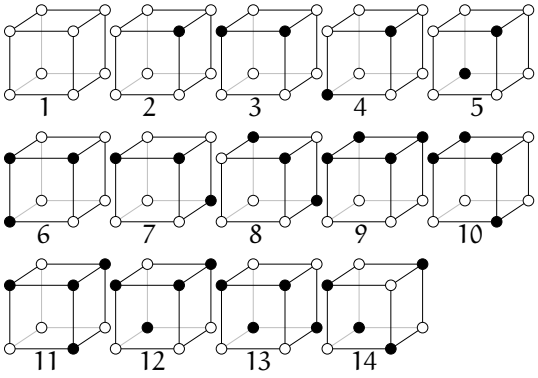


Figure 4.7: The 14 possible sign configurations.

Of these, configurations 5, 8, 12, 13 and 14 are impossible, due to the interval condition. For example in configuration 5 (see Figure 4.8), on edges $\overline{cd}$ and $\overline{ef}$ the function changes sign in opposite directions. Therefore, F is not parametrizable in this direction. For $\overline{ad}$, $\overline{fg}$ and for $\overline{bf}$, $\overline{dh}$ we find that F is not parametrizable in the direction of the other two axes, contradicting the inner product condition.

For configurations 8, 12 and 14 a similar argument holds. For configuration 13 we have to look at the diagonal pairs $(\overline{ac}, \overline{eg})$, $(\overline{bd}, \overline{fh})$ and $(\overline{ae}, \overline{cg})$, and note that the inner product condition does not depend on any particular orthogonal coordinate system.

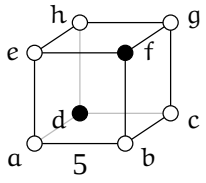


Figure 4.8: Configuration 5.

The remaining 9 configurations can now be triangulated. Two of these still contain an ambiguity (case 4 and 6 in Figure 4.9). To resolve this, note that the ambiguity is due to the sign configuration on a face of the cell. In both cases we have a single face where the function changes sign along all of its four edges, resulting in four vertices of the triangulation. There

are two ways to connect these vertices pairwise. We will choose to connect them, such that the resulting segments are parallel to one of the vectors $(1,1,0)$, $(1,0,1)$ and $(0,1,1)$, depending on the orientation of the face. This choice guarantees that the individual triangulations of two adjacent cells fit together. We have to show that this choice does not affect the isotopy.

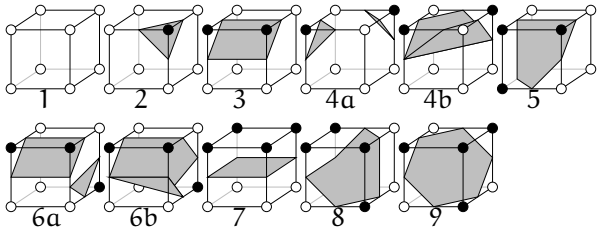


Figure 4.9: Triangulation of the subcubes.

Since cells have at most one ambiguous face, we regard the block of two cubes adjacent to such a face (see Figure 4.10). Note that the border of the union of these two cubes does not contain ambiguities. The sign changes along the edges of the shared face prevent a parametrization in the direction of one of these edges. Both cubes are therefore parametrizable in the vertical direction. Since the vertical component of the gradient does not disappear, the union of these two cells is also parametrizable in the vertical direction. This leads to only three possibilities to join two ambiguous cells, shown in Figure 4.10.

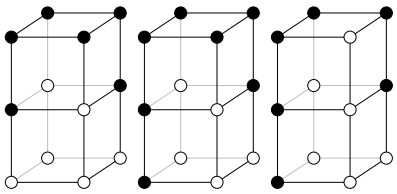


Figure 4.10: Two cubes sharing an ambiguous face: two type 4 cells, a type 4 and a type 6 cell, and two type 6 cells.

For the ambiguous face we have two ways to connect the vertices on the edges pairwise. Both possibilities are shown in Figure 4.11. Since both choices lead to the same isotopic approximation, we can choose either of them, as long as the triangulations for both cells fit together.

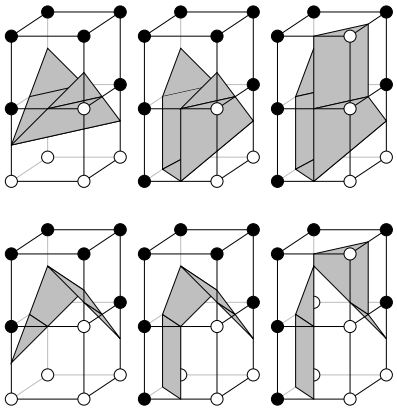


Figure 4.11: The two different triangulations of the three ambiguous cell combinations.

If the surface intersects the grid-faces in curved segments connecting the intersections of S with the edges of the grid, we are done. However, it is possible that S intersects a face of the grid in a closed curve, lying completely withing that face (Figure 4.12). In this case both cubes adjacent to this face must be parametrizable in the direction perpendicular to this face. Also, the 'bubble' does not leave the cube on the opposite face, since that would require the gradient to change more than $\pi/2$ (see also Figure 4.3). By linear interpolation in the direction of parametrizability, we can continuously flatten the bubble towards the face, and push it through this face, thereby removing the intersection loop.

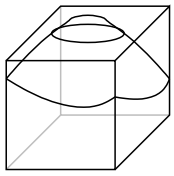


Figure 4.12: Surface S intersects a face of the regular grid in a closed loop.

As in the two-dimensional case we still have to remove the constraints on the function. The case where the surface passes through vertices of the grid can be handled as with implicit curves, i.e., by considering F to be strictly positive whenever $F \geq 0$.

Now assume that S intersects an edge e of the grid more than once, say at two consecutive points α and β (Figure 4.13). Since F changes sign at α and at β, there is a point p between α and β where $\nabla F(p)$ is perpendicular to e. This gradient lies inside the gradient cone of all four cubes adjacent to e. (Due to the interval constraint all gradients over a cell lie within a cone with top angle $\pi/2$.) Therefore, in the union of these four cubes, the surface is parametrizable in the direction of $\nabla F(p)$. Consider the plane through e in the direction of $\nabla F(p)$. Each of the two half-planes bounded by e intersects one of the cubes. Since $\nabla F(p)$ lies inside the cone of gradients of this cube, the projection of $\nabla F(x)$ does not change by more than $\pi/2$ when x ranges over the intersection of this half-plane with the cube. Therefore in one of these two half-planes, the intersection with S consists of a connected curve between α and β, that does not leave the cube. By linear interpolation in direction $\nabla F(p)$ between this curve and e we can locally deform the surface, and remove the pair α, β.

After removing all pairs of edge intersections we have continuously deformed the surface to one with only single edge intersections, for which the grid triangulation was shown to be correct.

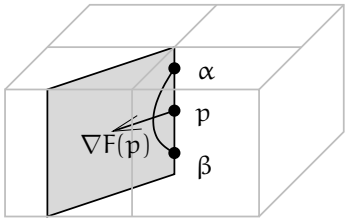


Figure 4.13: Surface S intersects an edge of the regular grid more than once.

We conclude that the triangulation of the regular grid is isotopic to the implicit surface.

4.5 Octrees

In the previous section we have shown that a regular grid satisfying an interval constraint can be used to create an isotopic approximation of an implicit surface. The regular grid approach usually results in a large number of mesh triangles. An octree based approximation has the advantage that we only produce a detailed mesh where a higher level of detail is required. In this section we will show how a balanced octree can be used to create an isotopic

mesh. Since we can subdivide the octree approximation to a complete octree, we have isotopy between the surface and the octree approximation, similarly to the two-dimensional case.

Overview The subdivision process is identical to the two-dimensional case. Starting with an octree initialized to a bounding box B, we subdivide the leaves until for each leaf C we have

$$0 \notin \square F(C) \quad \vee \quad \langle \square \nabla F(C), \square \nabla F(C) \rangle > 0.$$

After the subdivision process, we balance the octree.

Once we have constructed the octree, we can construct the approximation. First we put vertices at the center of each edge of the octree that has opposite function signs at its endpoints. Then, for each face of the octree we connect these vertices pairwise with segments. For each cell of the octree, these segments form closed chains on the boundary of the cell. If we have two nested loops, we connect these two loops with triangles, resulting in an annulus (this case is shown in Figure 4.14). Note that due to parametrizability we can consider the projection of the approximation. This way, it is clear that we cannot have more than two nested loops. In all other cases, we close each loop with triangles, constructing a topological disk for each closed chain of edges (Figure 4.15). Section 5.2 shows a more practical way to create the local triangle patches.

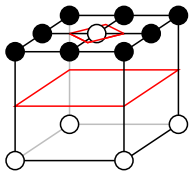


Figure 4.14: The top face indicates the only case where two nested loops occur, resulting in an annulus. All other configurations are triangulated as a set of disks.

To make sure that further subdivision does not change the isotopy, we have to be careful when connecting the vertices. The subdivision can 'push' the approximation up and down in the parametrizable direction. We shall assume a vertical parametrizable direction. For the top and bottom faces, it does not matter how we connect the vertices. Since the parametrizable direction is perpendicular to these faces, changing the connections corresponds to moving the surface slightly up or down through the surface. In the union of the adjacent cubes the isotopy does not change. This is similar to the

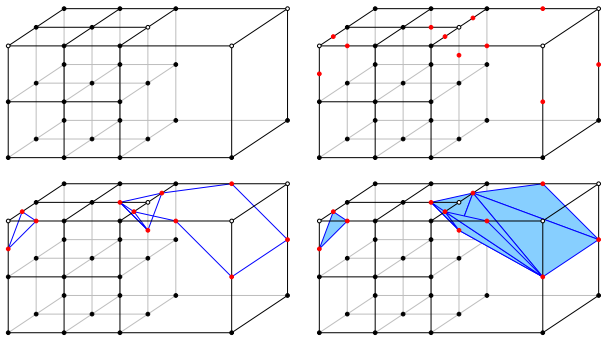


Figure 4.15: Meshing an octree by creating vertices, connecting these by line segments, and creating patches for segment loops.

regular grid case in Figure 4.11. The intersection curve in the common face changes, but the topology of the surface inside the union of the two adjacent cells stays the same.

For the faces of the octree along the side of a cell, the intersection of the surface with such a face is still parametrizable, but the $\pi/2$ bound on the variation of the gradient restricted to the face does not hold. The intersection curve can therefore pierce through two opposite edges (cf. Figure 4.3). This prevents us from uniquely determining the topology of the intersection curve. However, we will see later that a change in topology of the intersection curve corresponds to moving the surface slightly sideways through such a side face. Although the intersection curve changes, the topology of the surface itself is not affected.

Mesh construction To construct an isotopic approximation, we first construct mesh vertices on edges exhibiting a sign change. Then we examine these newly created mesh vertices on the boundary of an octree face. If the vertex configuration could correspond to a parametrizable intersection curve, we connect the vertices such that the piecewise linear approximation is parametrizable as well. For other vertex configurations we connect adjacent vertices pairwise, starting at an arbitrary vertex. Some examples are shown in Figure 4.16. For a non-ambiguous algorithm, see Section 5.2.

To show that this algorithm also constructs an isotopic mesh, we will subdivide the octree to a complete tree, similar to the two-dimensional case. The subdivision of an octree cell is split into several steps: first we add an octree vertex at the centre of an edge (if necessary), then at the centre of faces, and finally we add an octree vertex at the cell centre.

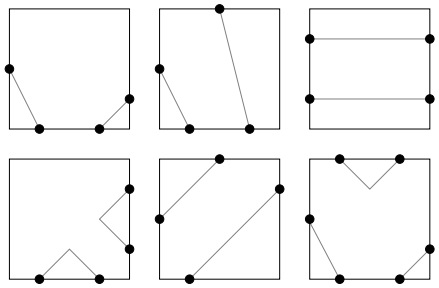


Figure 4.16: Examples of how to connect the vertices along the faces of the octree. The top row corresponds to potentially parametrizable intersections. Note that for example the middle figure could be parametrizable, since the vertices do not represent the exact intersection point. The bottom row is not parametrizable and can be connected arbitrarily.

Vertex on edge If an octree edge has opposite function signs at its endpoints, adding a centre octree vertex to that edge moves a mesh vertex from the centre of that edge to $\frac{1}{4}$th or $\frac{3}{4}$th of that edge. If the function signs are the same, adding the centre octree vertex can give rise to two new mesh vertices. However, the octree cells adjacent to this edge are at least as large as the edge itself, otherwise the octree vertex would have existed already. Therefore we can use the same argument with which we removed double edge intersections in section 4.4 (cf. Figure 4.12). That is, the isotopy of the mesh does not depend on the function sign at the centre of the edge.

Vertex on face Once all edges of a leaf are subdivided, we can add octree vertices at the centres of the faces.

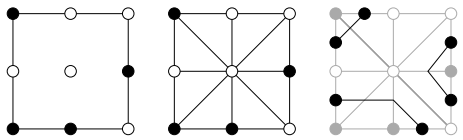


Figure 4.17: Meshing a face after adding a center vertex.

In Figure 4.17 we have subdivided the face and added diagonals in the four quadrants by connecting the center vertex to the four corners of the face. Meshing the resulting triangles connects adjacent mesh vertices. These intersection curves depend on our particular triangulation. However, choosing other diagonals corresponds to other ways to resolve ambiguous quadrants.

This is identical to the ambiguous faces in the regular grid approximation from section 4.4 (Figure 4.11). Although we could have constructed another intersection curve, the isotopy of the approximation surface does not change.

Both the regular grid method and the octree algorithm therefore connect adjacent mesh vertices of the face. The remaining problem is that the sign of the centre vertex determines which adjacent mesh vertices are connected.

For (possibly) parametrizable faces we constructed a parametrizable approximation. Along the line through the centre vertex we may shift the intersection curve in the parametrizable direction. By moving it over the center vertex we can change the function sign at this center vertex without changing the isotopy.

For non-parametrizable faces we can assume there are at least four mesh vertices along the boundary, otherwise there is only a single way to connect them. We have to show that both ways of connecting adjacent mesh vertices result in isotopic approximations.

We assume that the face contains at least four mesh vertices along its boundary, otherwise there is only a single way to connect them. Due to the gradient variation constraint, the part of the surface above or below the convex hull of these vertices stays close to the octree face, similar to the bubble in section 4.4 (Figure 4.12). It does not leave the adjacent cells, and therefore we may move the surface up and down slightly, resulting in different ways to connect the vertices inside the octree face. Again, different topology of intersection curves does not influence the isotopy of the surface inside the union of the adjacent octree leaves.

Vertex in cell Finally we have to add an octree vertex at the cell centre. Since the function is increasing in the parametrizable direction, this can shift a mesh vertex in that direction, but it cannot introduce new mesh vertices.

Summarizing, we now have shown that the octree together with the function signs at its vertices fully determines the isotopy of the surface. The signs at the newly created octree vertices have no influence on the isotopy.

Now we can use the regular grid approximation for the eight subcubes of the octree cell, which is isotopic to the surface. Since the approximation is parametrizable (except for vertical triangles), it is easy to see that connected patches usually form topological disks. The only way to create a hole is when the projection of that hole is a loop around the centre vertex, as in Figure 4.14. This is exactly the case for which we create an annulus. Therefore the regular grid approximation for the eight subcubes is isotopic to the grid created by the octree algorithm.

4.6 Results

Figure 4.18 and 4.19 show two examples of the curve approximation algorithm. The quadtree is also shown in the images. Both examples took 0.04 seconds to compute.

In Figures 4.20, 4.21, 4.22 and 4.23, four examples of implicit surface approximation are shown. The algorithm was implemented in `gcc`. The table below shows the number of leaves in the octree before and after subdivision, the number of triangles in the mesh, and the running time in seconds on a Pentium 667 MHz, running Linux.

Surface	Octree	Balanced	Triangles	Time
Tangle	24648	24816	8704	0.8
Chair	232072	233402	43014	6.2
Bear	497596	688794	366746	113.0
Non-alg	29275	40293	24612	2.0

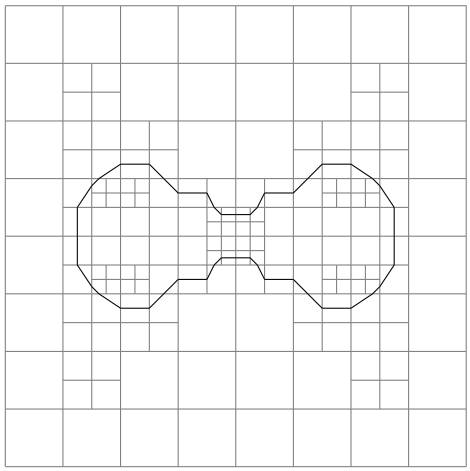


Figure 4.18: Implicit curve approximation of $f(x, y) = x^2(1 - x)(1 + x) - y^2 + 0.01$

4.7 Conclusion

We have presented a simple and fast algorithm for meshing implicit surfaces, that is among the first to guarantee isotopy between the mesh and the surface. The current implementation does not try to create a good approx-

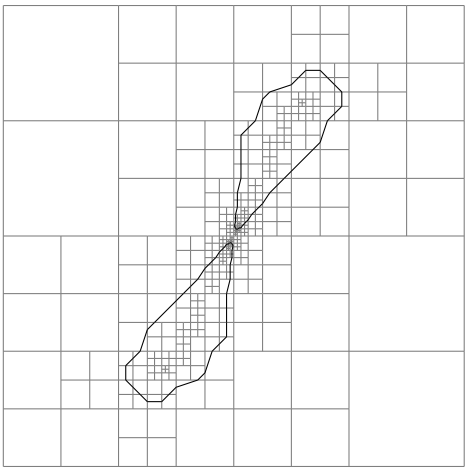


Figure 4.19: Implicit curve approximation of $f(x, y) = x^2 - xy + y^4 + 0.0001$

imation in terms of Hausdorff distance. An open problem remains how to improve the mesh quality without loosing isotopy. Other improvements to the algorithm are given below.

Improvements Both algorithms presented in this chapter use balanced trees. This balancing is not necessary for an isotopic result. Although the approximation for the leaves becomes more complicated, removal of the balancing would produce a slightly smaller approximation.

The octree based approach allows for local updates if S changes only locally, for example when using blobs or metaballs. For example, adding a single metaball does not change the implicit function outside the influence region of the metaball. Therefore, we only need to remesh the part of the octree intersecting the new metaball.

Although the resulting approximation is isotopic to S, it does not have to be close in terms of Hausdorff distance. To get a closer approximation the algorithm could be extended by subdividing cells where $0 \in \Box F(C)$ to a given minimal level, before starting the subdivision process (recall that the surface does not leave the neighbouring cell).

The algorithm can also be used for unbounded curves and surfaces. Recall that the undetected parts of the curves and surfaces cannot pierce through opposite edges or faces of a cell. By making the initial bounding box larger than the area of interest, and discarding the outer cells afterwards, we can construct an isotopic approximation for an unbounded manifold within a

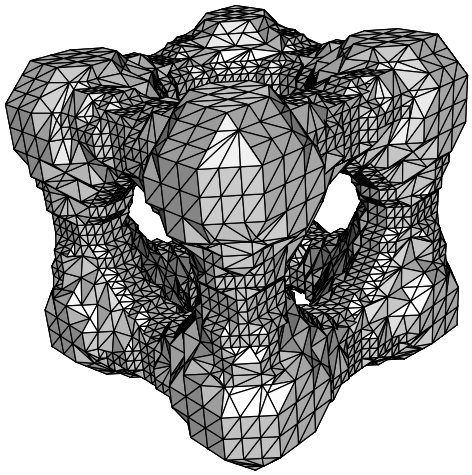


Figure 4.20: Tangle cube $f(x, y, z) = x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 10$

given box.

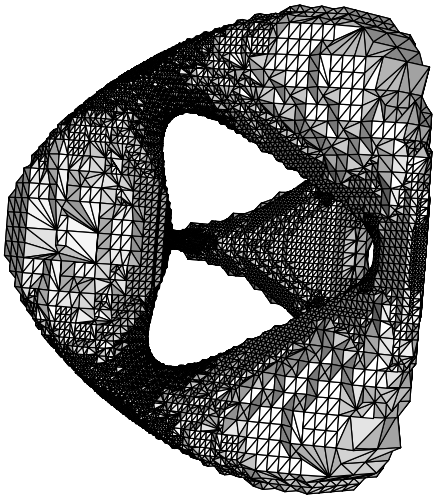


Figure 4.21: Chair $f(x,y,z) = \left(x^2+y^2+z^2-ak^2\right)^2 - b\left((z-k)^2-2x^2\right)\left((z+k)^2-2y^2\right)$, for $k=5$, $a=0.95$ and $b=0.8$

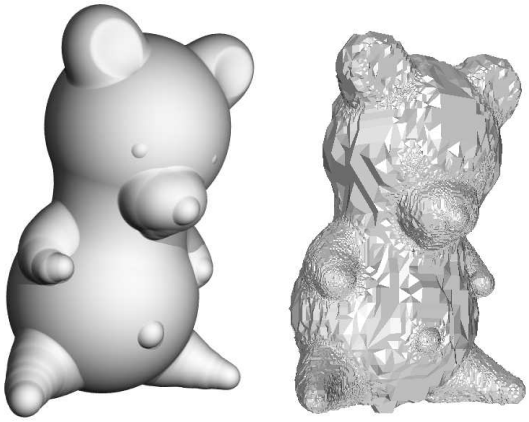


Figure 4.22: An implicit teddy bear together with its approximation. The bear consists of 48 metaballs.

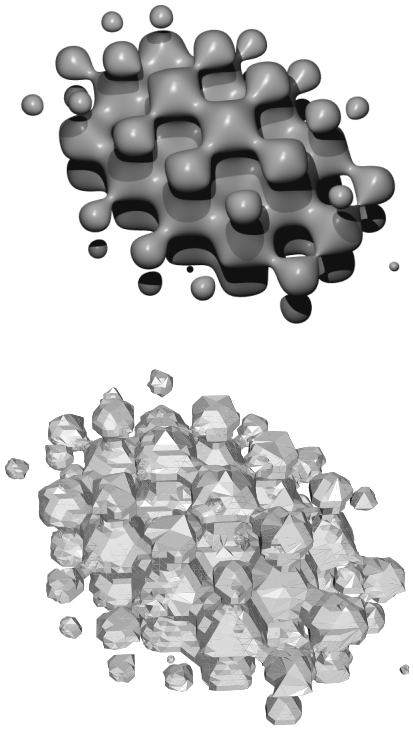


Figure 4.23: The non-algebraic surface $-0.4\,(\sin(5x)+\sin(5y)+\cos(5z))+0.1x^2+0.3y^2+0.2z^2-0.5$

Chapter 5

Level sets of implicit functions

5.1 Introduction

Level sets are isosurfaces of an implicit function $F\colon \mathbb{R}^3 \to \mathbb{R}$, that is the set of points satisfying $F(x, y, z) = 0$. In this chapter we introduce an algorithm to move at interactive speed through the different level sets. Furthermore the meshes of the level sets are isotopic to the isosurface itself, as long as the surface stays away from singularities. When the surface moves close to singularities, the algorithm indicates arbitrarily small boxes where the topology is not certified. In this case, the user can decide to decrease the size of the boxes by further refinement. For special classes of functions such as algebraic surfaces other methods could be used to determine the topology inside the singular boxes.

Recently a number of algorithms for regular isotopic meshing of implicit surfaces has appeared. Due to singularities for generic level sets, no such results exist for examining level sets of implicit functions. In this chapter we extend our algorithm for isotopic meshing of implicit surfaces to the meshing of level sets. As far as we know this is the first meshing algorithm for level sets giving topological guarantees for the resulting mesh. Furthermore, the meshes generated for different isovalues are disjoint. The algorithm has been implemented and is fast enough to examine level sets interactively. In this chapter we also improve this static algorithm, by using a tetrahedral subdivision of the octree, resulting in a less complex mesh generation.

Another meshing algorithm with topological correctness was presented in [13]. The algorithm is based on Morse theory and used information about the critical points of the function to determine the topology.

Morse theory is also used in [64] and [15], where the topologoy of a mesh is determined by starting an isovalue with trivial mesh topology, and updating the mesh as the isosurface passes through singularities.

It is not clear whether these algorithms can be extended to meshing of level sets. In particular, details about the singular points are required, and there is no fixed spatial subdivision that can be maintained for all level sets.

Sampling based algorithms for isotopic meshing of static implicit surfaces construct a sufficiently dense sample of points on the surface, such that surface reconstruction results in a homeomorphic approximation. To construct such a sample, in [14] bounds on the distance to the medial axis are needed, while [21] requires the critical points of heightfunctions on the surface and on intersection curves. Even if there requirements are met, the point sample cannot be reused for different isovalues.

In Section 5.2 we modify the isotopic meshing algorithm for static surfaces from the previous chapter to use tetrahedral subdivision. In Section 5.3 we show how the same data structure can be used for meshing level sets with topological guarantees. Results of this algorithm are shown in Section 5.4. Finally, in Section 5.5 we discuss some possible improvements to the current implementation.

5.2 Isotopic meshing with tetrahedra

To fascilitate mesh generation, in this section we will modify our isotopic meshing algorithm for implicit surfaces to use tetrahedral subdivision.

The isotopic meshing algorithm from Chapter 4 is based on an octree subdivision. Leaves of the octree are subdivided until we have determined that either it does not contain part of the surface ($0 \notin \Box F(C)$), or the variation in the gradient is small enough to determine the local topology. The latter condition is given by the interval inequality:

$$0 \notin \Box F(C) \vee \langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0.$$

It implies that for all $x, y \in C$ we have $\langle \nabla F(x), \nabla F(y) \rangle > 0$.

To fascilitate the meshing of the resulting octree, we balance it first, and then tetrahedrize it. Balancing means that we perform extra subdivision until neighbouring leaves differ at most one level in depth, and therefore at most a factor two in size. For the tetrahedrization we triangulate the faces of the octree in a non-ambiguous way. A vertex is added at the centre of each leaf, and tetrahedra are constructed by connecting this centre vertex with the triangles on the boundary of the leaf.

Algorithm ApproximateSurface(F, B)
Input. An implicit function F, and a cubic bounding box B.
Output. A piecewise linear approximation of the implicit surface $F = 0$.
1. Initialize an octree $\mathcal{T}$ to the bounding box B of $F = 0$.

2. Subdivide $\mathcal{T}$ until for all leaves C we have $0 \notin \Box F(C) \vee \langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0$.
3. BalanceOctree($\mathcal{T}$)
4. **for** each face of the octree $\mathcal{T}$
5. **do** triangulate the face according to the subdivision of its edges
6. **for** each leaf of the octree
7. **do** add a centre vertex and create tetrahedra by connecting it to the triangles on the boundary of the leaf
8. **for** each tetrahedron
9. **do** determine the function sign of its vertices and mesh the tetrahedron using linear interpolation

The sides of each face of the octree can consist of either one octree edge or of two octree edges in case of a subdivided side. We walk around an octree face in a fixed direction, for example counterclockwise as seen from the positive axis perpendicular to that face, and starting at the corner vertex with largest coordinates. By checking whether the four sides are subdivided, this results in a unique sign pattern. We then triangulate the face according to the table in Figure 5.1.

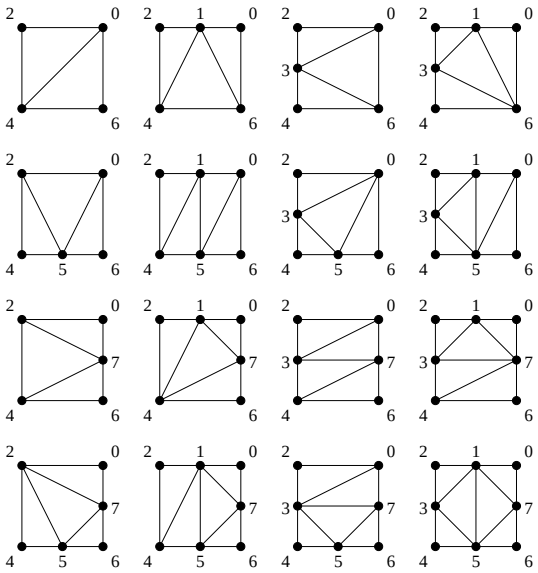


Figure 5.1: Triangulation of octree faces. The eight possible vertex positions are labeled 0 through 7 in counterclockwise direction around the positive axis.

Once all faces of an octree cell are triangulated, we can tetrahedrize the

cell by constructing a tetrahedron for each triangle, by connecting it to the centre of the cube. Since the triangulation of octree faces is determined uniquely, this results in a tetrahedrization of the entire octree (Figure 5.2).

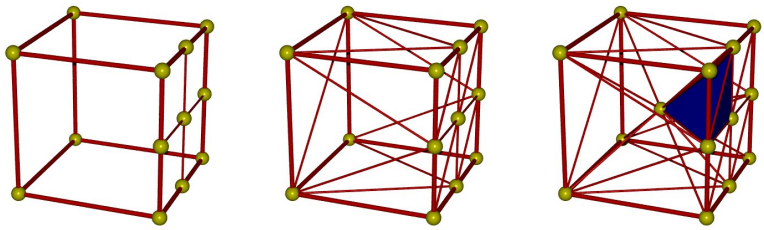


Figure 5.2: Tetrahedrization of a cell by triangulating its boundary. One of the 22 tetrahedra is shown in the bottom figure.

To generate the mesh, we construct at most two triangles for each tetrahedron. If one of the four vertices has an opposite sign we construct one triangle, if two vertices have the opposite sign of the other two, we construct two triangles forming a 4-gon (Figure 5.3).

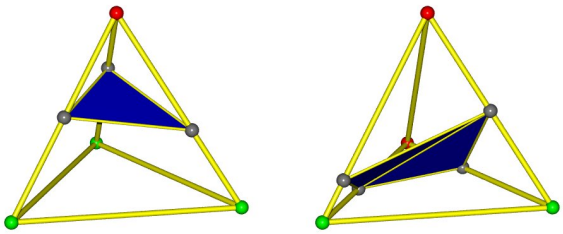


Figure 5.3: Meshing the tetrahedra.

For the proof of isotopic correctness of an octree based mesh we refer to Chapter 4. For isotopy of the tetrahedral based mesh, we use the fact that the octree leaves are parametrizable in the direction of one of the axes. By considering the projection in the parametrizable direction, it is easy to see that the tetrahedrization results in a mesh isotopic to the mesh we constructed in the previous chapter.

5.3 Level sets

In this section we shall extend the isotopic approximation algorithm to mesh level sets of an implicit function. Changing the isovalue means that the isosurface moves through the singularities of the implicit function. Our static algorithm only works for regular surfaces.

For the isotopic approximation algorithm, we required as little information about the implicit function as possible. We used a black box approach, where we can only compute function values and derivatives at points and over intervals. With this approach it is not possible to determine when and where exactly singularities occur. Note that the surface passing through a singularity of F only depends on the value of F at that point. Higher order derivatives do not give any more information. This implies that we would need to know the exact location and function value of singularities points. With the black box approach this is not possible. However, we can construct arbitrarily small boxes around the singularities, by terminating the subdivision at a predetermined maximum level. The leaves at the maximum level for which the gradient variation bound does not hold could contain singularities of the implicit function. We call these *singular* boxes, although they do not necessarily have to contain a singularity of F. Outside these singular boxes the topology is already guaranteed. For the singular boxes we can test whether $\theta \notin \Box F(C)$. If this is the case, then also inside the singular box we have correct topology. If not, there is not much more we can do. We could subdivide further until the isotopy class is determined, but since we do not know if the isosurface for the current value of θ is regular there is no guarantee that this process terminates. Instead, we choose to simply mark the box as 'red'. For visualization purposes, this means that we can easily identify for which isovalues we have small areas with uncertain topology. Depending on the requirements of the user, we could choose to do further octree refinement, which could remove the topological uncertainty. For specific classes of surfaces, such as algebraic surfaces, we could also examine the function itself to study the possible singularity inside a red box.

5.3.1 Algorithm

Instead of looking at the zero set $F = 0$ we now consider the level set $F = \theta$. For subdivision this means we have to examine the interval condition

$$\theta \notin \Box F(C) \quad \vee \quad \langle \Box \nabla F(C), \Box \nabla F(C) \rangle > 0.$$

The lefthand clause is dependent on the specific level set θ. We therefore subdivide the initial octree until the righthand clause holds or until a

predetermined maximum depth is reached. For a better Hausdorff accuracy we could also perform extra subdivision until all leaves have a fixed minimum depth. As in the static case we balance the tree for convenience. For the resulting octree we precompute $\square F(C)$ for all leaves for which the gradient variation bound does not hold. This way we can easily check whether $\theta \notin \square F(C)$, and therefore determine quickly if the topology inside a cell is guaranteed.

If we also store the function values at the octree vertices, the resulting data structure contains all the information we need to create the meshes for different level sets. Also, if the isovalue changes only by a small amount we can locally update the mesh.

Starting with an isovalue outside the range of function values at octree vertices, the initial mesh is empty. If we change the isovalue θ, the combinatory of the mesh only changes in cells for which θ passes the function value of one of the vertices on the boundary of that cell. Otherwise we only have to move existing mesh vertices to maintain linear interpolation. For the cells where the combinatory does change we simply remesh these octree leaves. Meshing of a single cell is done by tetrahedrizing the cell, and subsequently meshing the tetrahedra.

After (or during) updating the mesh, we examine all singular boxes. Recall that these are the octree leaves of maximal depth for which the gradient variation bound does not hold. If for such a leave $\theta \in \square F(C)$, we mark the leave as 'red'. In our implementation we display small red boxes for these leaves, indication that there is an uncertainty in topology. For nonempty leaves having vertices with function values both larger and smaller than θ, we could also construct a local triangle mesh, displaying these triangles in another colour to indicate the topological uncertainty.

In practice we can now move interactively through the level sets of an implicit function. For regular surfaces the mesh will be isotopic to the implicit surface, except for small red boxes appearing if the surface moves close to or through a singular box of the octree. If the maximum subdivision level is large enough, these singular boxes will only appear in the neighbourhood of singularities of the implicit function.

After further user input we could do a temporary refinement of the octree inside these red boxes, or perform some kind of analysis of the implicit function. For almost all θ the corresponding isosurface is regular, and a few extra levels of subdivision should remove most of the red boxes.

5.4 Results

For the implemention we computed the tetrahedrization immediately after balancing of the octree. We stored the information about the implicit function (such as function range, gradient variation bound and vertex values) in the tetrahedron structure. By storing this information of the octree leaves in the tetrahedron structure, the octree itself is not needed anymore. Unfortunately, storing the entire tetrahedrization of the octree takes up a lot of memory space, resulting in fairly low maximum subdivision levels.

We tested the implentation on several implicit functions. In this section we show some results for the Tangle Cube, the Chair surface and for a non-algebraic surface (Figure 5.4).

In Figure 5.5 two isosurfaces of the tangle cube are shown. This surface is defined by:

$$F(x, y, z) = x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2$$

Figure 5.6 shows a series of close-ups as the isosurface passes a singularity. With a maximum subdivision level of 9, the red boxes are small, and only appear when the surface is fairly close to the singular boxes. They could easily be removed by further subdivision, but this would require intervention of the user.

Figure 5.7 shows the 'chair' function:

$$F(x, y, z) = (x^2 + y^2 + z^2 - ak^2)^2 - b((z - k)^2 - 2x^2)((z + k)^2 - 2y^2),$$

with $k = 5$, $a = 0.95$ and $b = 0.8$. Due to the higher number of computations, the interval arithmetic converges slower, and the bound on the variation of the gradient is only satisfied at a higher level of subdivision. The maximum subdivision level used for Figure 5.7 is 6, resulting in large areas of red boxes. Although they could be removed with extra subdivision, it seems that in this case a higher initial subdivision level would perform better. However, this requires an implementation that does not store the entire tetrahedrization.

The non-algebraic surface is given by the function:

$$F(x, y, z) = -4\,(\sin(5x) + \sin(5y) + \cos(5z)) + x^2 + 3y^2 + 2z^2$$

This surface consists of many components and passes through a lot of singularities. Approximation the entire surface requires a large initial subdivision level. We approximated the part of the surface defined by the bounding box $[2, 2.5] \times [2, 2.5] \times [2, 2.5]$ with a maximum subdivision level of 8. Results are shown in Figure 5.8, giving a good indication of the stages where red boxes

appear and disappear. Notice in particular how red boxes appear just before a new component appears and again before it merges with the main surface component.

The timing results for the preprocessing are given in the following table. The time required for updating the mesh when the isovalue θ changes is dependent on the function and the values of the previous and new values for θ. In our experiments the updating typically takes a few tenths of a second, fast enough for interactive examination of the level sets. The table shows the time for computing the octree and the tetrahedrization, and the size of the octree (number of leaves), the balanced octree and the number of tetrahedra. The numbers in parentheses indicate the maximum subdivision level used.

	tangle (9)	chair (6)	non-alg. (8)
Octree subdiv	0.432s	0.484s	0.468s
Tetrahedrization	3.036s	5.468s	6.984s
Octree size	45032	66592	34987
Balanced size	46544	66760	49253
Tetrahedra	696432	862800	792638

Our experiments show that a maximal subdivision level of 8 or 9 results in a small number of red boxes. Since the tetrahedrization uses much more memory than the octree, an implementation that computes the tetrahedrization locally on demand should perform better. For small changes in the isovalue θ the number of octree leaves that require tetrahedrization is small. Comparing the time required for complete tetrahedrization with the size of the octree suggests that the time for updating the mesh would increase only slightly.

5.5 Conclusion and future work

Depending on the specific requirements there are lots of options in implementing the algorithm. For example, to move through an entire range of level sets, we could sort the vertices of the octree on function value. This list can be used to determine quickly where the next combinatorial changes in the mesh will occur.

From our results it is clear that we have to make a trade-off between speed and memory. Storing only the octree requires much less memory space, allowing a greater maximum depth of the initial tree. However, updating the mesh will take slightly more time due to the tetrahedrization and recomputation of function values. Another option is to precompute the tetrahedrization, and store it in external memory.

It is not yet clear how to deal with the red boxes. The static algorithm for

implicit functions terminates fairly quickly for regular surfaces. We expect that a few extra levels of subdivision would remove most of the red boxes. Again there is a trade-off between a lower initial subdivision combined with extra subdivision of red boxes, or starting with a higher initial subdivision level.

Another open problem is whether changing from interval arithmetic to affine arithmetic (AA) [24] will speed up the algorithm. AA computations are slower, but the results converge faster. Another option that we used in our implementation is to use standard interval arithmetic with extra levels of subdivision.

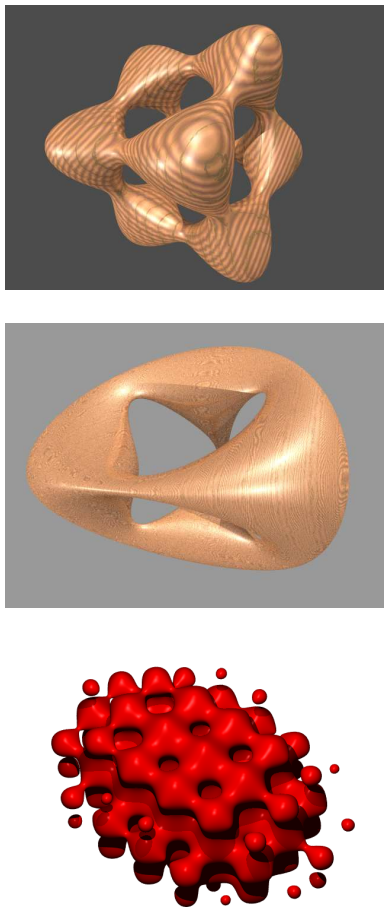


Figure 5.4: The tangle cube, the chair, and the non-algebraic surface.

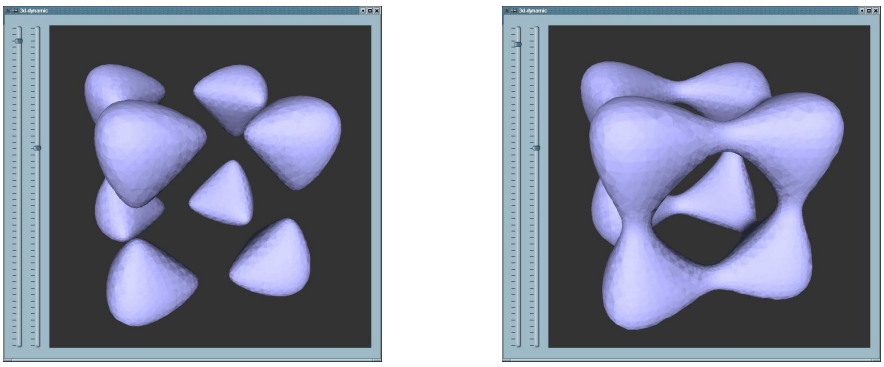


Figure 5.5: Level sets of the tangle cube.

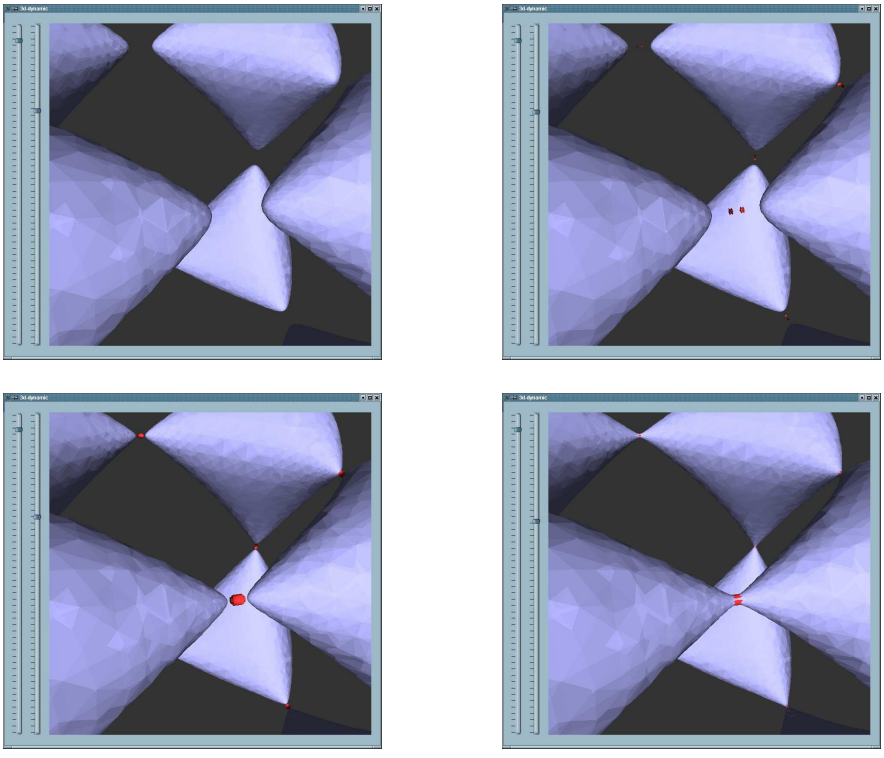


Figure 5.6: Close up of the 'red' boxes for level sets of the tangle cube. The maximum subdivision level is 9.

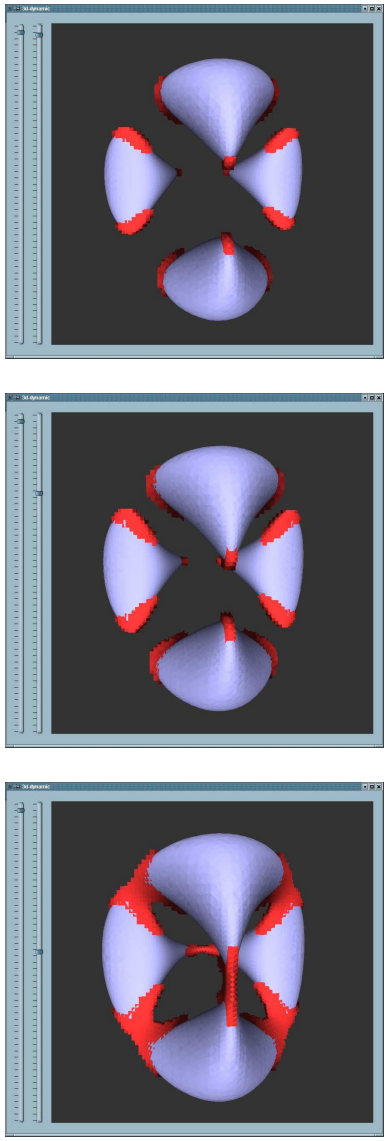


Figure 5.7: Level sets of the chair, with a maximum subdivision level of 6.

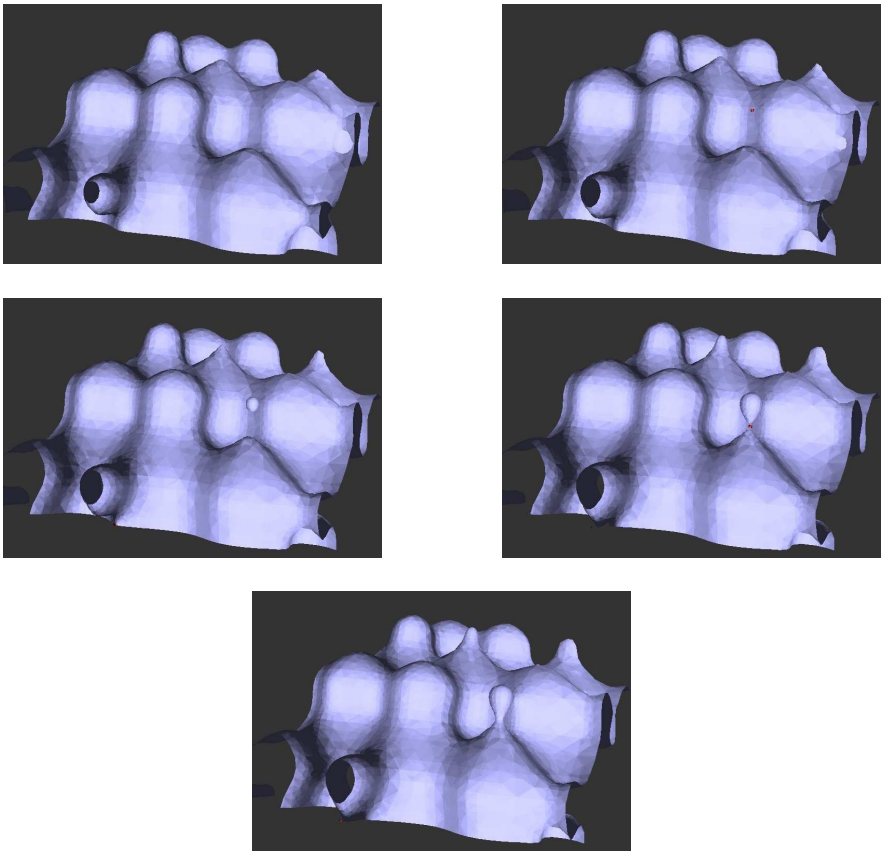


Figure 5.8: Level sets of part of the non-algebraic surface, with a maximum subdivision level of 8.

Appendix A

Singularities of functions on surfaces

A.1 Non-degenerate singular points

Consider an implicit surface $S = F^{-1}(0)$, where $F\colon \mathbb{R}^3 \to \mathbb{R}$ is a C^2 function. We assume that 0 is a regular value of F, so according to the Implicit Function Theorem S is a regular C^2-surface.

Our goal is to determine conditions which guarantee that the restriction of a C^2 function $G\colon \mathbb{R}^3 \to \mathbb{R}$ to the surface S has a non-degenerate singular point.

As for notation, the gradient of a function $F\colon \mathbb{R}^3 \to \mathbb{R}$ at $p \in \mathbb{R}$ will be denoted by $\nabla F(p)$. Furthermore, the *Hessian quadratic form* of a function $F\colon \mathbb{R}^3 \to \mathbb{R}$ at $p \in \mathbb{R}$ will be denoted by $H_F(p)$. Usually, we suppress the dependence on p from our notation, and denote this quadratic form by H_F. With respect to the standard euclidean inner product its matrix is the usual symmetric matrix whose entries are the second order partial derivatives of F. We denote partial derivatives using subscripts, e.g., F_x denotes $\frac{\partial F}{\partial x}$, F_{xy} denotes $\frac{\partial^2 F}{\partial x \partial y}$, etc.

Theorem 14. *Let $F, G\colon \mathbb{R}^3 \to \mathbb{R}$ be C^2 functions, and let 0 be a regular value of F. Let p be a point on the surface $S = F^{-1}(0)$.*

1. *p is a singular point of $G|S$ iff there is a real number λ such that*

$$\nabla G(p) = \lambda \nabla F(p). \tag{A.1}$$

2. *Furthermore, the singular point p is non-degenerate iff*

$$(H_G - \lambda H_F) \mid T_p S \tag{A.2}$$

is a non-degenerate quadratic form, where λ is as in (A.1).

Remark The scalar λ in (A.1) is traditionally called a *Lagrange multiplier.*

Corollary 15. *The singularity* p *of* $G|S$ *is non-degenerate iff the* 2×2*-matrix* Δ*, defined by (A.3), is non-singular:*

$$\Delta = V^T \cdot \left(\begin{pmatrix} G_{xx} & G_{xy} & G_{xz} \\ G_{xy} & G_{yy} & G_{yz} \\ G_{xz} & G_{yz} & G_{zz} \end{pmatrix} - \lambda \begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix} \right) \cdot V, \tag{A.3}$$

where λ *is the Lagrange multiplier defined by (A.1), and* V *is a* 3×2*-matrix whose columns span the tangent space* T_pS*. Here all first and second order derivatives are evaluated at* p*. Furthermore, for* $G|S$*, the singular point* p *is a maximum or minimum if* $\det(\Delta) > 0$*, and a saddle point if* $\det(\Delta) < 0$*.*

In particular, we may take $V = X$*,* $V = Y$*, or* $V = Z$ *if* $F_x(p) \neq 0$*,* $F_y(p) \neq 0$*, or,* $F_z(p) \neq 0$*, respectively, where*

$$X = \begin{pmatrix} F_y & F_z \\ -F_x & 0 \\ 0 & -F_x \end{pmatrix}, Y = \begin{pmatrix} -F_y & 0 \\ F_x & F_z \\ 0 & -F_y \end{pmatrix}, Z = \begin{pmatrix} -F_z & 0 \\ 0 & -F_z \\ F_x & F_y \end{pmatrix}. \tag{A.4}$$

Proof of Theorem 14 1. Saying that p is a singular point of $G|S$ is equivalent to $dG_p(v) = 0$, for all $v \in T_pS$. Since $T_pS = \ker dF_p$, we see that this is equivalent to the existence of a scalar λ such that $dG_p = \lambda\, dF_p$.

2. Since 0 is a regular value of F, we have $\nabla F(p) \neq 0$. We assume that $F_x(p) \neq 0$, and argue similarly in case $F_y(p) \neq 0$ or $F_z(p) \neq 0$. Furthermore, assume that $p = (0,0,0)$. According to the Implicit Function Theorem, there is a unique local solution $x = f(y,z)$, with $f(0,0) = 0$, of the equation $F(x,y,z) = 0$. Implicit differentiation yields

$$F_x f_y + F_y = 0, \tag{A.5}$$

$$F_x f_z + F_z = 0, \tag{A.6}$$

where f_y and f_z are evaluated at (y,z), and F_x, F_y and F_z are evaluated at $(f(y,z),y,z)$. Similarly,

$$F_{xx} f_y^2 + 2F_{xy} f_y + F_{yy} + F_x f_{yy} = 0. \tag{A.7}$$

Similar identities are obtained by differentiating (A.5) with respect to y, and (A.6) with respect to z.

Using y and z as local coordinates on S, we obtain the following expression of $G|S$ with respect to these local coordinates:

$$g(y,z) = G(f(y,z),y,z).$$

Differentiating this identity twice with respect to y we obtain

$$g_{yy} = G_{xx}\, f_y^2 + 2G_{xy} f_y + G_{yy} + G_x\, f_{yy}. \tag{A.8}$$

Since $F_x(p) \neq 0$, we solve f_{yy} from (A.7), and plug the resulting expression into (A.8), to get

$$g_{yy} = (G_{xx} - \lambda F_{xx})\, f_y^2 + 2(G_{xy} - \lambda F_{xy})\, f_y + (G_{yy} - \lambda F_{yy}), \tag{A.9}$$

where

$$\lambda = \frac{G_x}{F_x}$$

is the Lagrange multiplier, cf. (A.1). We rewrite (A.9) as

$$\begin{aligned} g_{yy} &= \begin{pmatrix} f_y & 1 & 0 \end{pmatrix} (H_G - \lambda H_F) \begin{pmatrix} f_y \\ 1 \\ 0 \end{pmatrix} \\ &= \frac{1}{F_x^2} \begin{pmatrix} F_y & -F_x & 0 \end{pmatrix} (H_G - \lambda H_F) \begin{pmatrix} F_y \\ -F_x \\ 0 \end{pmatrix} \end{aligned}$$

We similarly derive

$$g_{yz} = \frac{1}{F_x^2} \begin{pmatrix} F_y & -F_x & 0 \end{pmatrix} (H_G - \lambda H_F) \begin{pmatrix} F_z \\ 0 \\ -F_x \end{pmatrix},$$

and

$$g_{zz} = \frac{1}{F_x^2} \begin{pmatrix} F_z & 0 & -F_x \end{pmatrix} (H_G - \lambda H_F) \begin{pmatrix} F_z \\ 0 \\ -F_x \end{pmatrix}.$$

Since the vectors $(F_y, -F_x, 0)$ and $(F_z, 0, -F_x)$ span the tangent space $T_p S$, we see that

$$\begin{pmatrix} g_{yy} & g_{yz} \\ g_{yz} & g_{zz} \end{pmatrix} = \frac{1}{F_x^2} \Delta.$$

Appendix B

The Morse Lemma with parameters

The Morse Lemma gives a normal form for a smooth function in a neighborhood of a non-degenerate critical point. We discuss an extension of this result to functions depending on parameters. We refer to [4] for a proof. Let $F : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ be a smooth map, and let the function $f : \mathbb{R}^n \to \mathbb{R}$, defined by $f(x) = F(x, 0)$, have a non-degenerate singularity at $p \in \mathbb{R}^n$. Then the function $F_\mu : \mathbb{R}^n \to \mathbb{R}$ has a non-degenerate singularity p_μ, which depends smoothly on $\mu \in \mathbb{R}^k$, and coincides with p for $\mu = 0$. In fact, the singular point $x = p_\mu$ is a solution of the system of equations

$$\frac{\partial F}{\partial x_1}(x, \mu) = \cdots = \frac{\partial F}{\partial x_n}(x, \mu) = 0.$$

The Jacobian determinant of this system at $(x, \mu) = (p, 0)$ is equal to the determinant of the Hessian matrix of f at 0, which is non-zero. Therefore, the Implicit Function Theorem guarantees the existence and uniqueness of the local solution p_μ.

Proposition 16. *There is a local diffeomorphism $\Phi : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^n \times \mathbb{R}^k$, defined on a neighborhood of $(p, 0)$, of the form*

$$\Phi(x, \mu) = (\varphi(x, \mu), \mu),$$

such that

$$F \circ \Phi(x, \mu) = \sigma_1 x_1^2 + \cdots + \sigma_n x_n^2 + F(p_\mu, \mu),$$

where $\sigma_i = \pm 1$, $i = 1, \ldots, n$, defines the signature of the quadratic part of f at p.

The proof of the Morse Lemma with parameters is completely similar to the proof of the 'regular' Morse Lemma.

Appendix C

Normal form of a Whitney fold

C.1 Fold points of good mappings

A celebrated theorem of Whitney [70] states that generic mappings of the plane into the plane have only two types of singular points, namely *folds* or *cusps*. In this section we discuss a normal form for such a mapping in the neighborhood of a fold point, and sketch a method for actually computing such a normal form.

Let $f : \mathbb{R}^2 \to \mathbb{R}^2$ be a mapping, and let $J : \mathbb{R}^2 \to \mathbb{R}$ be its Jacobian determinant. More precisely, if $f(x, y) = (u(x, y), v(x, y))$, then

$$J = u_x v_y - u_y v_x.$$

The mapping f is called *good* if $J(p) \neq 0$ or $\nabla J(p) \neq 0$, at every point $p \in \mathbb{R}^2$. In other words, the system of equations

$$J(x, y) = J_x(x, y) = J_y(x, y) = 0 \tag{C.1}$$

has no solutions in $\mathbb{R}^2$. It can be proven that this condition is independent of the choice of coordinates.

Whitney proves that generically a map from the plane into the plane is good. Intuitively, this may seem to follow from the observation that generic systems of three equations in two unknowns do not have a solution. However, it is not obvious that (C.1) is a generic system. A proof can be given using Thom's transversality theorem.

Consequently, the singular set $S_1(f) = J^{-1}(0)$ of a good map consists of a number of disjoint regular curves in the plane. This is a straightforward consequence of the Implicit Function Theorem, since the gradient of J is non-zero at every point of $S_1(f)$.

Furthermore, at a singular point p at least one of the partial derivatives of the component functions of f is non-zero, i.e.

$$(u_x(p), u_y(p), v_x(p), v_y(p)) \neq (0,0,0,0). \tag{C.2}$$

In particular, the rank of the derivative df_p at a singular point is one, or, equivalently, the kernel of df_p is one-dimensional. A singular point p of a good map f is called a *fold point* [29] if

$$T_p S_1(f) \oplus \mathrm{Ker} df_p = \mathbb{R}^2. \tag{C.3}$$

Note that the tangent space $T_p S_1(f)$ is well-defined, since $S_1(f)$ is a regular curve. Whitney's version [70] of (C.3) is slightly different: if w is a non-zero tangent vector at p to $S_1(f)$, then the directional derivative of f in the direction w is non-zero, i.e., $\nabla_w f(p) \neq 0$. Obviously, the two versions are equivalent, and, more importantly, neither version depends on a particular choice of coordinates. The latter observation follows from Lemma 17.

Lemma 17. *Let f be a good map, $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ an invertible non-singular map, and $g = f \circ \varphi$. Then*

(i) $S_1(g) = \varphi^{-1}(S_1(f))$;

(ii) $g(S_1(g)) = f(S_1(f))$;

(iii) *g is a good map;*

(iv) *p is a fold point of g iff $\varphi(p)$ is a fold point of f.*

Proof. Let J and $\tilde{J}$ be the Jacobian determinants of f and g, respectively. Since

$$dg_p = df_{\varphi(p)} \circ d\varphi_p, \tag{C.4}$$

we see that

$$\tilde{J}(p) = J(\varphi(p)) \cdot \det d\varphi_p. \tag{C.5}$$

Therefore, $\varphi(\tilde{J}^{-1}(0)) = J^{-1}(0)$, which is equivalent to first claim. The second claim is an immediate consequence of the first one. To prove the third claim, let $p \in S_1(g)$, i.e., $\tilde{J}(p) = 0$. Then (C.5) implies $J(\varphi(p)) = 0$. Therefore, applying the product rule to (C.5) yields

$$d\tilde{J}_p = \det d\varphi_p \circ dJ_{\varphi(p)} \circ d\varphi_p. \tag{C.6}$$

Since $d\varphi_p$ is invertible, we see that the scalar $\det d\varphi_p$ is non-zero, and hence that

$$d\tilde{J}_p \neq 0 \ \text{ iff } \ dJ_{\varphi(p)} \neq 0.$$

This proves the third claim. Finally, observe that (C.4) implies

$$d\varphi_p(\mathrm{Ker} dg_p) = \mathrm{Ker} df_{\varphi(p)}, \tag{C.7}$$

and that (C.6) implies

$$d\varphi_p(\mathrm{Ker} d\tilde{J}_p) = \mathrm{Ker} dJ_{\varphi(p)}. \tag{C.8}$$

Since $\mathrm{Ker} d\tilde{J}_p = T_p S_1(g)$, and $\mathrm{Ker} dJ_{\varphi(p)} = T_{\varphi(p)} S_1(f)$, we conclude from (C.7) and (C.8) that

$$T_p S_1(g) \oplus \mathrm{Ker} dg_p = \mathbb{R}^2 \text{ iff } T_{\varphi(p)} S_1(f) \oplus \mathrm{Ker} df_{\varphi(p)} = \mathbb{R}^2,$$

in other words: p is a fold point of g iff $\varphi(p)$ is a fold point of f. This concludes the proof of the fourth claim. □

For later use, we isolate the following result.

Lemma 18. *Let $f : \mathbb{R}^2 \to \mathbb{R}^2$ be a good map, with component functions $u, v : \mathbb{R}^2 \to \mathbb{R}$, and Jacobian determinant $J : \mathbb{R}^2 \to \mathbb{R}$. The following statements are equivalent:*

(i) *p is a fold point of f;*

(ii) *$W_p \neq (0,0)$, where*

$$W_p = (\langle \nabla u(p), \nabla J(p)^\perp \rangle, \langle \nabla v(p), \nabla J(p)^\perp \rangle).$$

In either case, W_p is a tangent vector of the regular curve $f(S_1(f))$ at p.

Proof. Since f is good, $J^{-1}(0) = S_1(f)$ is a regular curve, containing p. Its tangent vector at p is $\nabla J(p)^\perp$.

Since df_p has rank one, we have $\nabla u(p) \neq (0,0)$ or $\nabla v(p) \neq (0,0)$. Assume the former inequality holds. Then there is a real constant c such that

$$\nabla v(p) = c \nabla u(p).$$

Now

$$df_p(W) = (\langle \nabla u(p), W \rangle, \langle \nabla v(p), W \rangle), \tag{C.9}$$

for $W \in R^2$, so $df_p(\nabla u(p)^\perp) = (0,0)$. Hence $\mathrm{Ker} df_p = \mathbb{R} \nabla u(p)^\perp$.

It now follows that (C.3) is holds iff $\nabla u(p)^\perp$ is not parallel to $\nabla J(p)^\perp$, or, iff $\langle \nabla u(p), \nabla J(p)^\perp \rangle \neq 0$. Similarly, if $\nabla v(p) \neq 0$, then (C.3) is equivalent to $\langle \nabla v(p), \nabla J(p)^\perp \rangle \neq 0$. In either case, $W_p \neq 0$. This proves the equivalence of (i) and (ii).

To prove that W_p is tangent to $f(S_1(f))$ at p, let $s \mapsto (x(s), y(s))$ be a regular parametrization of $J^{-1}(0)$, with $(x(0), y(0)) = p$, and $(x'(0), y'(0)) = \nabla J(p)^{\perp}$. Then $\alpha(s) = f(x(s), y(s))$ is a parametrization of $f(J^{-1}(0))$, and, in view of (C.9 we have

$$\alpha'(0) = df_p(\nabla J(p)^{\perp}) = W_p. \tag{C.10}$$

□

C.2 Normal form of a good map near a fold point

Our goal is to obtain a simple form of a good map near a fold point. Such a *normal form* can be obtained by applying a change of coordinates on the domain of the map, on its range, or both. We start by applying only changes of coordinates on the *domain* of the map.

Proposition 19. *Let $f : \mathbb{R}^2 \to \mathbb{R}^2$ be a good map with component functions $u, v : \mathbb{R}^2 \to \mathbb{R}$, and Jacobian determinant $J : \mathbb{R}^2 \to \mathbb{R}$. Let $p = (0, 0)$ be a fold point of f, with $f(p) = (0, 0)$. The following properties are equivalent:*

(i) *$f(J^{-1}(0))$ is transversal to the y-axis at $f(p)$;*

(ii) *$\nabla u(p) \neq (0, 0)$;*

(iii) *There is a local change of coordinates $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$, with $\varphi(0, 0) = p$, such that*

$$f \circ \varphi(x, y) = (x, \sigma y^2 + \alpha(x)), \tag{C.11}$$

where $\sigma = \operatorname{sign}(\langle \nabla u(p), \nabla J(p)^{\perp} \rangle) = \pm 1$, and $\alpha : \mathbb{R} \to \mathbb{R}$ is a smooth function with $\alpha(0) = 0$.

Remark 20. If $g = f \circ \varphi$, we shall say that φ *brings* f *into the form* g. The right hand side of (C.11) will be considered as a *normal form*. The transformations bringing the mapping into normal form should be *computationally feasible*. Therefore, we shall be very precise about each and every step we take.

Proof. of Proposition 19
The equivalence of conditions (i) and (ii) follows from Lemma 18. To see that (iii) implies (i), let $g = f \circ \varphi$, then $f(S_1(f)) = g(S_1(g))$, according to Lemma 17.(ii). Since $g(S_1(g)) = \{(x, \alpha(x)) \mid x \in \mathbb{R}^2\}$, the first claim follows.

So it remains to prove that the (ii) implies (iii). The proof goes in two steps. We assume that $p = (0,0)$.

Step 1. *There is a local change of coordinates bringing* f *into one of the forms* $g(x,y) = (x, V(x,y))$.

To see this, we first consider the case

$$u_x^0 \neq 0, \tag{C.12}$$

where the notation is as in Section C.1. Here the superscript 0 indicates that the corresponding function has to be evaluated at $p = (0,0) \in \mathbb{R}^2$. In view of (C.12), the change of coordinates $\psi : \mathbb{R}^2 \to \mathbb{R}^2$, defined by

$$\psi(x,y) = (u(x,y), y),$$

is non-singular. Let φ be the inverse of ψ, then $\varphi(x,y) = (\xi,\eta)$ iff $\eta = y$ and

$$u(\xi,y) - x = 0. \tag{C.13}$$

Let $\xi = \xi(x,y)$ be the locally unique solution of (C.13) with $\xi(0,0) = 0$, then

$$\varphi(x,y) = (\xi(x,y), y),$$

and

$$f \circ \varphi(x,y) = (x, V(x,y)), \quad \text{with} \quad V(x,y) = v(\xi(x,y), y).$$

If $u_x^0 = 0$, then condition (ii) implies that $u_y^0 \neq 0$. This case is reduced to (C.12) by first applying the transformation $(x,y) \mapsto (y,x)$.

Step 2. *If* $f(x,y) = (x, V(x,y))$ *is a good map with a fold point at* $(0,0)$*, then there is a local change of coordinates bringing* g *into the form* $g(x,y) = (x, \pm y^2 + \alpha(x))$*, where* $\alpha : \mathbb{R} \to \mathbb{R}$ *is a smooth function with* $\alpha(0) = 0$.

Since

$$df_{(x,y)} = \begin{pmatrix} 1 & 0 \\ V_x & V_y \end{pmatrix},$$

we see that $(0,0)$ is a singular point of f iff

$$V_y^0 = 0. \tag{C.14}$$

In that case, $\mathrm{Ker} df_{(0,0)}$ is the span of the vector $(0,1)$. Furthermore, $J = V_y$, so the fact that f is a good mapping implies in particular that $\nabla J(0,0) = (V_{xy}^0, V_{yy}^0) \neq (0,0)$. A non-zero tangent vector to $S_1(f) = J^{-1}(0)$ at $(0,0)$ is $\nabla J(0,0)^\perp = (-V_{yy}^0, V_{xy}^0)$. It is now easy to see that condition (C.3) is equivalent to

$$V_{yy}^0 \neq 0. \tag{C.15}$$

Since also $V(0,0)=0$, we apply the Morse Lemma with parameters to bring the function V into normal form. More precisely, there is a local change of parameters $\varphi:\mathbb{R}^2\to\mathbb{R}^2$ of the form $\varphi(x,y)=(x,\Phi(x,y))$, such that

$$V\circ\varphi(x,y)=\pm y^2+\alpha(x),$$

where α is a smooth function with $\alpha(0)=0$. It is now obvious that $f\circ\varphi$ is the desired normal form.

Concatenating steps 1 and 2 concludes the proof. □

Corollary 21. *If $f:\mathbb{R}^2\to\mathbb{R}^2$ is a good mapping with a fold point at p, then there is a local change of coordinates $\varphi:\mathbb{R}^2\to\mathbb{R}^2$, near $(0,0)$, with $\varphi(0,0)=p$, and a local change of coordinates $\psi:\mathbb{R}^2\to\mathbb{R}^2$, with $\psi(f(p))=(0,0)$, such that*

$$\psi\circ f\circ\varphi(x,y)=(x,y^2).$$

Proof. Applying Proposition 19, we may assume that $f(x,y)=(x,\pm y^2+\alpha(x))$ or $f(x,y)=(\pm x^2+\alpha(y),y)$. In the first case, take $\psi(x,y)=(x,\pm(y-\alpha(x)))$. Then $\psi\circ f(x,y)=(x,y^2)$. The second case is treated similarly. □

Remark 22. In Step 1 of the proof of Proposition 19, in case $u_x\neq 0$, near $p=(0,0)$, we define the function V by $V(x,y)=v(\xi(x,y),y)$, where $\xi=\xi(x,y)$ is the solution of $u(\xi,y)-x=0$. The derivatives V_y and V_{yy} can be computed from this definition in a straightforward way:

$$V_y=v_x\xi_y+v_y=v_x(-\frac{u_y}{u_x})+v_y=\frac{J}{u_x},$$

where all functions in the right hand sides are evaluated at $(\xi(x,y),y)$. Since $J(0,0)=0$, we obtain

$$V_{yy}(0,0)=\frac{(J^0_x\xi^0_y+J^0_y)u^0_x}{(u^0_x)^2}=\frac{-u^0_yJ^0_x+u^0_xJ^0_y}{(u^0_x)^2}.$$

Therefore, $V_{yy}(0,0)\neq 0$ iff $\langle\nabla u(0,0),\nabla J(0,0)^\perp\rangle\neq 0$ iff the vectors $\nabla u(0,0)^\perp$ and $\nabla J(0,0)^\perp$ are not parallel.

We claim that $\nabla u(0,0)^\perp=(-u^0_y,u^0_x)$ is a non-zero vector in $\mathrm{Ker}\,df_0$. Indeed, since df_0 has rank 1, we see that $\nabla v(0,0)=c\nabla u(0,0)$, for some real constant c. Therefore

$$\begin{pmatrix}u^0_x & u^0_y\\ v^0_x & v^0_y\end{pmatrix}\begin{pmatrix}-u^0_y\\ u^0_x\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix},$$

which proves our claim. Since $\nabla J(0,0)^{\perp}$ is tangent to $J^{-1}(0) = S_1(f)$, we see that the latter condition is equivalent to (C.3). In other words: $V_{yy}(0,0) \neq 0$ iff condition (C.3) holds. This is what we also observed in Step 2 of the proof of Proposition 19.

C.3 Introducing parameters

If f depends on an additional parameter t, we can still obtain a normal form like (C.11), provided we let both the transformation φ and the function α depend on this new parameter. More precisely, consider a family of maps $f_t : \mathbb{R}^2 \to \mathbb{R}^2$, depending smoothly on the parameter $t \in \mathbb{R}$, such that f_0 satisfies any of the conditions (i), (ii) and (iii) of Proposition 19. With this family we associate the map $F : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$, defined by $F(x,t) = f_t(x)$. Then there is a one-parameter family $\Phi : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2 \times \mathbb{R}$ of local coordinate transformations of the form

$$\Phi(x,y,t) = (\varphi_t(x,y),t),$$

such that

$$F \circ \Phi(x,y,t) = (x, \sigma y^2 + \alpha(x,t)). \tag{C.16}$$

The latter identity may be rephrased as

$$f_t \circ \varphi_t(x,y) = (x, \sigma y^2 + \alpha(x,t)).$$

The image of the fold curve of f_t is $f_t(S_1(f_t)) = \{(x, \alpha(x,t)) \mid x \in \mathbb{R}\}$. The point of intersection of this curve with the y-axis, viz $(0, \alpha(0,t))$, passes through the origin with speed α_t^0 (Here we consider t as time) . To express this velocity in terms of f_t, let $u(x,y,t)$ and $v(x,y,t)$ be the component functions of $F(x,y,t)$.

Lemma 23. *If* $u_x(p,0) \neq 0$*, then*

$$\alpha_t^0 = \frac{1}{u_x(p,0)} \frac{\partial(u,v)}{\partial(x,t)}(p,0),$$

and if $u_y(p,0) \neq 0$*, then*

$$\alpha_t^0 = \frac{1}{u_y(p,0)} \frac{\partial(u,v)}{\partial(y,t)}(p,0).$$

Proof. Let $\Phi(x,y,t) = (\xi(x,y,t), \eta(x,y,t))$. Then (C.16) boils down to

$$\begin{aligned} u(\xi(x,y,t), \eta(x,y,t), t) &= x, \\ v(\xi(x,y,t), \eta(x,y,t), t) &= \sigma y^2 + \alpha(x,t). \end{aligned}$$

Differentiating the latter identity with respect to t and evaluating at $(p, 0)$ yields

$$\begin{aligned} u_x^0 \xi_t^0 + u_y^0 \eta_t^0 + u_t^0 &= 0, \\ v_x^0 \xi_t^0 + v_y^0 \eta_t^0 + v_t^0 &= \alpha_t^0. \end{aligned} \tag{C.17}$$

Since the second condition of Proposition 19 holds, i.e., $(u_x^0, u_y^0) \neq (0, 0)$, there is a scalar constant c such that $(u_x^0, u_y^0) = c(v_x^0, v_y^0)$. Therefore, (C.17) implies:

$$\alpha_t^0 = v_t^0 - c u_t^0 = \frac{1}{u_x^0}(u_x^0 v_t^0 - v_x^0 u_t^0).$$

If $u_y^0 \neq 0$, the second identity is derived similarly. □

Remark 24. Lemma 23 implies that $\alpha_t^0 \neq 0$ iff the matrix

$$\begin{pmatrix} u_x^0 & u_y^0 & u_t^0 \\ v_x^0 & v_y^0 & v_t^0 \end{pmatrix}$$

has rank two. Furthermore, since $u_x^0 v_y^0 - u_y^0 v_x^0 = 0$, it follows that

$$u_y^0 \begin{vmatrix} u_x^0 & u_t^0 \\ v_x^0 & v_t^0 \end{vmatrix} = u_x^0 \begin{vmatrix} u_y^0 & u_t^0 \\ v_y^0 & v_t^0 \end{vmatrix}.$$

In view of Lemma 23 both sides of this equality are equal to $u_x^0 u_y^0 \alpha_t^0$.

Bibliography

[1] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. In D. Duke and R. Scopigno, editors, *Computer Graphics Forum*, volume 20(2), pages 67–80. Blackwell Publishing, 2001. [6, 54]

[2] L. Alberti, G. Compte, and B. Mourrain. Meshing implicit algebraic surfaces: the smooth case. Technical Report TR. 5405, INRIA Sophia-Antipolis, December 2004. [44]

[3] P. Alliez, N. Laurent, H. Sanson, and et al. Efficient view-dependent refinement of 3D meshes using sqrt3-subdivision. In *The Visual Computer*, volume 19(4), pages 205–221. Springer, July 2003. [5, 15]

[4] V.I. Arnol'd. Wave front evolution and equivariant morse lemma. *Comm. Pure Appl. Math.*, 29:557–582, 1976. [16, 93]

[5] V.I. Arnol'd. *Catastrophe Theory*. Springer-Verlag, 1986. [16]

[6] Koji Ashida. *Adaptive construction of manifold meshes*. PhD thesis, University of Pennsylvania, 2003. [7]

[7] Koji Ashida and Norman I. Badler. Feature preserving manifold mesh from an octree. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 292–297, New York, NY, USA, 2003. ACM Press. [7]

[8] G. Barequet, C.A. Duncan, M.T. Goodrich, S. Kumar, and M. Pop. Efficient perspective-accurate silhouette computation. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 417–418, 1999. [16]

[9] J. Bloomenthal. *Introduction to Implicit Surfaces*. Morgan-Kaufmann, 1973. [16]

[10] Jules Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988. [7, 54]

[11] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 251–256, New York, NY, USA, 1991. ACM Press. [3]

[12] J. Boissonnat and S. Oudot. Provably good surface sampling and approximation, 2003. [8]

[13] J.D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Isotopic implicit surface meshing. In *Proceedings Thirty-Sixth Annual ACM Symposium on Theory of Computing*, Chicago, 2004. [12, 54, 75]

[14] J.D. Boissonnat and S. Oudot. An effective condition for sampling surfaces with guarantees. In *Proceedings ACM Symposium on Solid Modeling*, 2004. [54, 76]

[15] A. Bottino, W. Nuij, and K. van Overveld. How to shrinkwrap through a critical point: an algorithm for the adaptive triangulation of iso-surfaces with arbitrary topology. In *Proc. Implicit Surfaces*, pages 55–72, 1996. [54, 75]

[16] David Bremer and John F. Hughes. Rapid approximate silhouette rendering of implicit surfaces. In *Proc. Implicit Surfaces*, 1998. [5, 15, 53]

[17] J.W. Bruce. Seeing – the mathematical viewpoint. *The Mathematical Intelligencer*, 6:18–25, 1984. [16]

[18] J.W. Bruce and P.J. Giblin. Outlines and their duals. *Proc. London Math. Soc. (3)*, 50:552–570, 1985. [16]

[19] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM Press. [4]

[20] Martin Cermak and Vaclav Skala. Adaptive edge spinning algorithm for poligonization of implicit surfaces. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 36–43, Washington, DC, USA, 2004. IEEE Computer Society. [6]

[21] S.W. Cheng, T.K. Dey, E.A. Ramos, and T.Ray. Sampling and meshing a surface with guaranteed topology and geometry. In *Proceedings Symposium on Computational Geometry*, 2004. [54, 76]

[22] Evgeni V. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical Report CERN CN 95–17, 1995. [6, 54]

[23] R. Cipolla and P.J. Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, 2000. [16]

[24] J. L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. In *Proc. VI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'93)*, pages 9–18, 1993. [10, 83]

[25] Erwin de Groot and Brian Wyvill. Rayskip: faster ray tracing of implicit surface animations. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 31–36, New York, NY, USA, 2005. ACM Press. [1]

[26] Mathieu Desbrun and Marie-Paule Cani. Active implicit surface for animation. In *Graphics Interface*, pages 143–150, Jun 1998. Published under the name Marie-Paule Cani-Gascuel. [1]

[27] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 287–290, New York, NY, USA, 1995. ACM Press. [1]

[28] A.T. Fomenko and T.L. Kunii. *Topological Modeling for Visualization*. Springer-Verlag, 1997. [23]

[29] M. Golubitsky and V. Guillemin. *Stable Mappings and Their Singularities*, volume 14 of *Graduate Texts in Mathematics*. Springer-Verlag, 1973. [96]

[30] T. Greg and J. O'Brien. Variational implicit surfaces, 1999. [4]

[31] X. Gu, S.J. Gortler, H. Hoppe, L. McMillan, B.J. Brown, and A.D. Stone. Silhouette mapping. Technical Report TR-1-99, Harvard University, March 1999. [16]

[32] E.R. Hansen and R.I. Greenberg. An interval newton method. *Applied Mathematics and Computation*, 12:89–98, 1983. [10]

[33] John C. Hart. Ray tracing implicit surfaces. In *SIGGRAPH 93 Modeling, Visualizing, and Animating Implicit Surfaces course notes*, pages 13–1 to 13–15. 1993. [1]

[34] John C. Hart. Sphere tracing: Simple robust antialiased rendering of distance-based implicit surfaces. In *SIGGRAPH 93 Modeling, Visualizing, and Animating Implicit Surfaces course notes*, pages 14–1 to 14–11. 1993. [1]

[35] Erich Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998. [6, 53]

[36] D. Hearn and M.P. Baker. *Computer Graphics, 2nd edition.* Prentice-Hall, 1994. [22]

[37] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 517–526. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000. [5]

[38] Chien-Chang Ho, Fu-Che Wu, Bing-Yu Chen, Yung-Yu Chuang, and Ming Ouhyoung. Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum*, 24(3):537–545, August 2005. Special issue: Proceedings of EUROGRAPHICS 2005. [7]

[39] Christoph M. Hoffman. Implicit curves and surfaces in cagd. *IEEE Comput. Graph. Appl.*, 13(1):79–88, 1993. [1]

[40] K.C. Hui and Z.H. Jiang. Tetrahedra based adaptive polygonisation of implicit surface patches. *Computer Graphics Forum*, 18(1):57–68, 1999. [7]

[41] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, New York, NY, USA, 2002. ACM Press. [7]

[42] Tasso Karkanis and A. James Stewart. High quality, curvature dependent triangulation of implicit surfaces. [6]

[43] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics symposium on geometry processing*, 2006. [4]

[44] J.J. Koenderink. *Solid Shape*. MIT-Press, 1990. [16]

[45] Nikita Kojekine, Vladimir Savchenko, and Ichiro Hagiwara. Surface reconstruction based on compactly supported radial basis functions. pages 218–231, 2004. [4]

[46] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing, 2002. [6]

[47] Adriano Martins Lopes. *Accuracy in Scientific Visualization*. PhD thesis, University of Leeds, 1999. [1]

[48] H. Lopes, J. B. Oliveira, and L. H. de Figueiredo. Robust adaptive polygonal approximation of implicit curves. *Computers and Graphics*, 26(6):841–852, 2002. [53]

[49] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, 1987. [6, 54, 60]

[50] L. Markosian, M.A. Kowalski, S.J. Trychin, L.D. Bourdev, D. Goldstein, and J.F. Hughes. Real-time nonphotorealistic rendering. In *SIGGRAPH Proceedings*, pages 415–420, 1997. [5, 15]

[51] Čermák Martin and Skala Václav. Polygonization by the edge spinning. In *Algoritmy 2002*, pages 245–252, Slovakia, 2002. Univ.of Technology. [6]

[52] J. Milnor. *Morse Theory*. Princeton University Press, 1963. [23]

[53] Ramon E. Moore. *Interval analysis*. Prentice Hall, 1966. [9]

[54] Xinlai Ni, Michael Garland, and John C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Trans. Graph.*, 23(3):613–622, 2004. [12]

[55] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993. [6]

[56] University of Wuppertal. C-xsc class library. http://www.math.uni-wuppertal.de/~xsc/. [46]

[57] A. Opalach and S. Maddock. An overview of implicit surfaces, 1995. [16]

[58] A. Opalach and S. C. Maddock. Disney effects using implicit surfaces. In *Proc. 5th Eurographics Workshop on Animation and Simulation*, 1994. [1]

[59] Simon Plantinga and Gert Vegter. Contour generators of evolving implicit surfaces. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 23–32. ACM Press, 2003. [53]

[60] H. Ratschek and J. Rokne. *Geometric computations with interval and new robust methods*. Horwood Publishing, 2003. [10]

[61] William J. Schroeder, Berk Geveci, and Mathieu Malaterre. Compatible triangulations of spatial decompositions. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 211–218, Washington, DC, USA, 2004. IEEE Computer Society. [7]

[62] J.M. Snyder. Interval analysis for computer graphics. In *SIGGRAPH Proceedings*, pages 121–130, 1992. [10]

[63] John M. Snyder. Interval analysis for computer graphics. *Computer Graphics*, 26(2):121–130, 1992. [53]

[64] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 279–286. ACM Press/Addison-Wesley Publishing Co., 1997. [54, 75]

[65] B.T. Stander and J.C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *SIGGRAPH Proceedings*, pages 279–286, 1997. [10, 18]

[66] J. Stolfi and L. de Figueiredo. *Self-Validated Numerical Methods and Applications*. Institute for Pure and Applied Mathematics (IMPA), Rio de Janeiro, 1997. Monograph for the 21st Brazilian Mathematics Colloquium (CBM'97), IMPA. [10]

[67] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4):583–598, 1999. [7]

[68] Greg Turk and James F. O'brien. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.*, 21(4):855–873, 2002. [4]

[69] C.W.A.M. van Overveld and B. Wywill. Shrinkwrap: An adaptive algorithm for polygonizing an implicit surface. Technical Report 93/514/19, University of Calgary, March 1993. [54]

[70] H. Whitney. On singularities of mappings of euclidean spaces I, mappings of the plane into the plane. *Ann. of Math.*, 62:374–410, 1955. [28, 33, 95, 96]

[71] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, 28(Annual Conference Series):269–277, 1994. [2]

[72] S. Wu, M. de, and G. Malheiros. On improving the search for critical points of implicit functions, 1999. [12]

[73] Dan X. Yang. *Mesh generation and information model for device simulation.* PhD thesis, Stanford University, 1994. [7]

[74] G. Yngve and G. Turk. Robust creation of implicit surfaces from polygonal meshes, 2002. [4]