Topologically correct Intersection Curves of Tori and Natural Quadrics

Dissertation
zur Erlangung des Grades
"Doktor der Naturwissenschaften"

am Fachbereich Physik, Mathematik und Informatik
der Johannes Gutenberg-Universität in Mainz,

vorgelegt von
Tobias Reithmann,
geboren in Wiesbaden

Mainz, den 14. Juli 2009

Abstract

This thesis provides efficient and robust algorithms for the computation of the intersection curve between a torus and a simple surface (e.g. a plane, a natural quadric or another torus), based on algebraic and numeric methods.

The algebraic part includes the classification of the topological type of the intersection curve and the detection of degenerate situations like embedded conic sections and singularities. Moreover, reference points for each connected intersection curve component are determined. The required computations are realised *efficiently* by solving quartic polynomials at most and *exactly* by using exact arithmetic.

The numeric part includes algorithms for the tracing of each intersection curve component, starting from the previously computed reference points. Using interval arithmetic, accidental incorrectness like jumping between branches or the skipping of parts are prevented. Furthermore, the environments of singularities are correctly treated.

Our algorithms are *complete* in the sense that any kind of input can be handled including degenerate and singular configurations. They are *verified*, since the results are topologically correct and approximate the real intersection curve up to any arbitrary given error bound. The algorithms are *robust*, since no human intervention is required and they are *efficient* in the way that the treatment of algebraic equations of high degree is avoided.

Kurzzusammenfassung

Diese Arbeit liefert effiziente und robuste Algorithmen zur Berechnung der Schnittkurve zwischen einem Torus und einer einfachen Fläche (d.h. einer Ebene, einer natürlichen Quadrik oder eines weiteren Torus), basierend auf algebraischen und numerischen Methoden.

Der algebraische Teil umfasst die Klassifizierung des topologischen Typus der Schnittkurve und das Auffinden von degenerierten Fällen, wie z.B. eingebettete Kegelschnitte und Singularitäten. Darüberhinaus werden Zeugenpunkte für jede zusammenhängende Schnittkurvenkomponente bestimmt. Durch das Lösen von Polynomen mit maximalem Grad von vier und durch Verwendung exakter Arithmetik werden die erforderlichen Rechnungen *effizient* und *exakt* durchgeführt.

Der numerische Teil umfasst Algorithmen für die Verfolgung der Schnittkurvenkomponenten, ausgehend von den zuvor berechneten Zeugenpunkten. Durch die Verwendung von Intervall-Arithmetik werden Fehler, wie z.B. das Springen zwischen Komponenten oder das Auslassen von Kurventeilen, unterbunden. Desweiteren werden die Umgebungen von Singularitäten korrekt behandelt.

Unsere Algorithmen sind *komplett* in dem Sinne, dass jegliche Eingabe, inklusive degenerierter und singulärer Konfigurationen, bearbeitet werden kann. Sie sind *verifiziert*, da die Ergebnisse topologisch korrekt sind und die eigentliche Schnittkurve bis zu jeder beliebigen Genauigkeit approximieren. Die Algorithmen sind *robust*, da keine Benutzerintervention erforderlich ist, und sie sind *effizient*, da das Bearbeiten von höhergradigen algebraischen Gleichungen vermieden wird.

Contents

Chapter 1

Introduction

Computer Aided Design (CAD) and *Computer Aided Manufacturing* (CAM) applications work with curved objects for nearly fifty years already. One of the two most common approaches, handling curved objects, uses free-form surfaces like *B-splines* and *NURBS* [62, 63]. A well-known application in this context is Rhino[1]. Free-form surfaces are easy to design and to render, but it is very difficult to examine, for instance, the intersection of two surfaces. The second approach uses *Constructive Solid Geometry* (CSG), which combines simple objects like planes, spheres, cylinders, cones and tori to more complex objects using Boolean operators like difference, union and intersection. The advantage of working with CSG is the reduction of complex to simple objects. However, the common implementations of both approaches suffer from approximation and rounding errors while using fast but inexact floating-point arithmetic. Thus degenerate configurations, which are frequent in the design of geometric objects, are either treated with much effort or not even detected. The consequences are incorrect results or even application failure.

To counteract this defectiveness, new approaches in the range of CSG were developed, based on *Algebraic Geometry*. In these approaches, objects are represented and treated by algebraic equations. On the one hand, this is realisable exactly. On the other hand, these approaches are very expensive due to the occurrence of large equation systems. Thus the implementations result in unacceptable running times.

In this thesis we present a hybrid approach, which combines the efficiency of floating-point based approaches with the exactness of algebraic approaches. Our goal was the ability to work on any kind of input including degenerate configurations and to present verified results with exact topology. Furthermore, our algorithms should be that efficient to be competitive to other applications already in use.

We concentrate on a small group of objects consisting of plane, sphere, circular cylinder, circular cone and torus, which we denote by *simple surfaces*. The set of sphere, circular cylinder and circular cone is usually denoted by *natural quadrics* as well. Some papers also include the plane as degenerate quadric. According to Requicha and Voelcker [65], these simple surfaces constitute the major component of objects handled in CAD/CAM applications. Moreover simple surfaces have nice properties we will get on to in Section 2.3. We compute the intersection curve of two of these objects, where we consider only intersections involving the torus, since there already exists a multitude of algorithms on intersecting natural quadrics, see next section.

Therewith this thesis provides a fundamental tool for CSG operations and the conversion into *Boundary Representation* (B-rep), while working on the most established group of objects and following the paradigm of exact computation.

1.1 Previous Work

The solid modeling community more and more favours exact and verified results, moving away from approximations by polygonal meshes to exactly representable curved objects. This implicates

[1] see http://www.rhino3d.com/

algorithms which provide exact results as well. There exist several projects in this context.

The software package CGAL[2] (Computational Geometry Algorithms Library) provides data structures and algorithms for numerous geometric problems like triangulations, Voronoi diagrams, arrangements, convex hull computations and many more, see [31].

A similar package is LEDA[3], which is specialised in graph algorithms, see also [14, 56]. Furthermore LEDA provides data types for an exact arithmetic like `leda::integer` and `leda::rational`, which represent arbitrary large integer and rational numbers, respectively.

Another library, providing exact arithmetic, is CORE[4], supporting the *Exact Geometric Computation* (EGC) approach for numerically robust algorithms, see also [42]. This library relies on the big number package GMP[5] (GNU Multiple Precision Arithmetic Library). Data types for arbitrary large numbers are `CORE::BigInt` and `CORE::BigRat`. Since the implementation of our algorithms is generic, we are able to interchange the number types, providing more flexibility in the low level arithmetic.

The project, being decisive for the prospected problem of this thesis, is EXACUS[6] (Efficient and Exact Algorithms for Curves and Surfaces). There exists a lot of literature in this context. Berberich et al. [7, 6, 8] considered the arrangement and the computation of planar maps of conics and conic arcs. Eigenwillig et al. [25, 28, 29] extended this framework to cubic curves and generalised it further to arbitrary algebraic plane curves [27, 26, 44]. Furthermore Hemmer et al. [34, 39, 72, 66] considered the arrangement of quadrics. Implementations are available through the EXACUS libraries CONIX, CUBIX , ALCIX and QUADRIX, respectively.

Quite recently some work has been published by Berberich et al. [10], considering the topology of algebraic surfaces of arbitrary degree. Their approach is similar to Collins' *cylindrical algebraic decomposition* (cad) [18], which consists of two phases: In the first phase, the projection phase, a given algebraic manifold is projected onto subvarieties until a total decomposition into distinct cells is possible. In the second phase, the lifting phase, these cells serve as a base for gaining higher dimensional cells. The complete algebraic decomposition consists then of all cells being determined during the lifting phase. The main problems of a cad are first the handling of sophisticated algebraic equations, mainly consisting of polynomials whose coefficients are roots of algebraic equations as well. The second problem is the loss of topological information during the projection phase, which has to be regained during the lifting phase. Therefore most of the related approaches presume the prospected algebraic manifold to lie in a generic position, i.e. there are no occurring degeneracies during the projection phase such as vertical lines with respect to the projection direction. Even though, Berberich et al. [10] presented an applicable implementation which computes a complete decomposition for any arbitrary real algebraic surface, providing its exact topology. The ongoing work is about the consideration of multiple surfaces and, relating to this, the examination of real algebraic curves. However, the question is open, if this approach is then still applicable due to the remaining problem of handling sophisticated algebraic equations of high degree.

A related approach is the computation of the arrangement induced by arbitrary algebraic surfaces on a parametrised ring Dupin cyclide by Berberich and Kerber [9]. An arrangement is the subdivision of a manifold into distinct cells of less dimensionality. A Dupin cyclide, introduced by Dupin [23], can be seen as the generalisation of a torus. The approach of Berberich and Kerber is mainly based on Eigenwillig and Kerber [26], which is used to compute the 2D arrangement of the induced intersection curves in the parameter space of the referenced Dupin cyclide. Like in [10], Berberich and Kerber have to cope with algebraic equations of high degree as well. Another point is the lack of an explicit representation of the intersection curves in real affine space.

Concerning this point, Lazard et al. [49] provide the exact parametrisation of intersection curves of quadrics. This seems to be a restriction compared with [9], but quadrics make the biggest part of primitives in a CSG framework and thus are worth to be considered in particular. Hemmer [39]

[2]The project CGAL was originally funded by European Union's information technologies programme Esprit and followed by similar projects GALIA, ECG and ACS, see http://www.cgal.org/

[3]LEDA is distributed by *Algorithmic Solutions Software GmbH*, see http://www.algorithmic-solutions.com/

[4]see http://cs.nyu.edu/exact/

[5]see http://gmplib.org/

[6]EXACUS is developed by the Max Planck-Institute for Computer Science in Saarbrücken, see http://www.mpi-inf.mpg.de/projects/EXACUS/

extended this work to the full adjacency graph of an arrangement of quadrics, where an adjacency graph specifies the connectivity between the lower dimensional cells in an arrangement. This is a major step towards the full and exact 3D arrangement of quadrics.

The last project to be mentioned here is ESOLID[7], presented by Keyser et al. [45]. They compute already the exact boundary of Boolean operators on low-degree curved objects, in particular quadrics. The approach is based on the examination of intersection curves in the parameter space of each object and the determination of the correspondence between common intersection curve components in different parameter space domains. ESOLID provides the highest level of functionality, compared with all other presented frameworks, and thus can be directly used by CAD/CAM applications for CSG to B-rep conversion. However, it is not complete in the sense that it may fail or even crash on degenerate input, involving intersection curves with singularities or common intersection components. To countervail this, Keyser and Ouchi [61] presented an approach for the detection of degeneracies in advance by the examination of common parts in the algebraic equation systems. In a degenerate case they perturb numerically the input data, assuming that the objects are endued with a given tolerance, to generate a generic position. However, not all types of degeneracies can be detected automatically, namely when two surfaces intersect in an exclusively tangential curve.

Concerning the representation of implicitly given algebraic curves, there exist already several algorithms. For a good survey see Krishnan and Manocha [47]. Some of the approaches even claim to be able to detect degeneracies like singular points. In [47] this detection is based on the vanishing of some energy function. In [3] singularities are indicated by the appearance of near-singular matrices. However, they still use floating-point arithmetic for this detection. Hence they need a user given error bound which operates as threshold to decide, whether a singularity is present or two curve components just lie very close. Another problem of most of the numerically based approaches is the use of Newton-like methods as in [4, 5]. These methods sometimes do not converge, resulting in an inaccurate behaviour of the tracing up to component jumping or the miss of some parts.

Our approach uses a marching method to trace the intersection curve. Therefor we compute algebraically initial starting points on each intersection curve component, including singularities. Outgoing from these points we trace each curve component using a numerical predictor-corrector step method. For the corrector step we use interval arithmetic to verify the convergence of the involved Newton method. Furthermore we make use of a test by Plantinga and Vegter [64], based on interval arithmetic, which prevents component jumping. Due to this and the fact that we compute singularities algebraically in advance, we guarantee topologically correct results.

Recapitulating there are approaches, considering a very general problem but coping with algebraic equations of high degree and are thus only marginally applicable. There are specialised approaches, considering just a small group of objects, namely quadrics. And there are approaches, which are practical but not complete, or they are complete but assume a given tolerance attached to the input data. Thus our motivation was firstly to extend the group of concerned objects by including the torus as the next simple primitive occurring in most CAD/CAM applications aside from quadrics. Secondly the degree of any occurring algebraic equation should be as small as possible to speed up computation. Lastly our approach should provide exact results, based on an exactly given input. The results should be extendable immediately to a full 3D arrangement and thus serve for exact boundary computation. Regarding the first two directives, this thesis is mainly based on the dissertation of Kim [46]. He considered the intersection between tori and simple surfaces based on a configuration space approach, see Section 2.3. Since he used floating-point arithmetic, we were compelled to develop new techniques based on exact arithmetic, complying with the last directive.

[7] see http://research.cs.tamu.edu/keyser/geom/esolid/

1.2 Outline

We compute the topologically exact intersection curve between a torus T and a simple surface S. The intersection curve is given by the set of distinct intersection curve components, where each component consists either of a regular closed loop, an isolated touching point or some regular curve branches, which are connected by singular points. Each curve segment is given exactly by some square root expressions in case of singular points and embedded conic sections. This ensures an exact comparison of common intersection parts between pairs of objects, which is essential for the computation of the arrangement of more then two objects. Otherwise the curve segment is approximated by a linear spline, where it is guaranteed that this approximation never exceeds a given error bound.

The overall algorithm proceeds as follows:

1. Detect any common conic section of T and S.
2. Transform T and S into a trajectory C and an obstacle O in configuration space, see Section 2.3.
3. Calculate the maximal connected components of C inside O and the touching points between C and O.
4. Transform C and O back to the original surfaces T and S. Thereby for each connected component of C compute a reference point on the corresponding intersection curve component between T and S. Touching points between C and O become singular points on the intersection curve.
5. For each singular point calculate the directions of all outgoing intersection curve branches.
6. Trace each intersection curve branch starting from its singular point into the calculated direction.
7. Trace each remaining intersection curve component starting from its reference point.

Apart from the latter two points, any calculation can be done exactly by using exact arithmetic. In the cases of conic sections and singular points we operate with a symbolic representation of square root expressions. Calculating reference points on each intersection curve component requires the computation of roots of algebraic equations. Our approach is efficient since these algebraic equations are of degree four at most. Moreover there is no need of further operating on these roots in contrast to algebraic approaches based on a cad. It suffices to approximate the roots and thus the initial starting points for the tracing step due to the approximative nature of the whole tracing step itself.

Hence this thesis splits into two parts: An algebraic one, where singularities and conic sections are determined. Furthermore initial starting points for each intersection curve component are computed. For some cases even a parametric representation of the intersection curve can be provided. In this part all computations are based on an exact arithmetic and therewith the results are given exactly as well.

In Chapter 2 we explain the notations used in this thesis and overview the mathematical background of our problem. Moreover some useful data type concepts of our implementation are discussed. In Chapter 3 we give the necessary and sufficient conditions for any occurring conic section in a torus-simple surface intersection. Chapter 4 provides algorithms for the intersection between two objects, handling the remaining tasks of computing singular points, initial starting points and parametric curves.

The second part is of numerical nature. From the set of singular points and initial starting points from the Algebraic Part before, the intersection curve is traced using a typical predictor-corrector approach. Additionally we guarantee a correct result, i.e. we are able to avoid branch-hopping and the miss of curve parts. By using floating-point numbers with arbitrarily large bitsize the computations are fast as well as correct in arbitrary environments, e.g. a nearly degenerate situation with very small perturbation. Chapter 5 provides algorithms for tracing a regular intersection

curve component without any singularities. Furthermore the environments of singular points are examined to compute the exact directions of outgoing intersection curve branches.

We conclude with a summary of our work and give an outlook on possible future work in Chapter 6.

Chapter 2

Preliminaries

In this chapter we introduce some basic notations and mathematical backgrounds which will be used in this thesis. First we introduce general notations and notations for geometric primitives we use. We then present some useful definitions and basic data types we used in our prototypical implementation. In this context we describe the arithmetical concepts which our algorithms are based on. Since this thesis is closely linked to Ku-Jin Kim's dissertation [46] we try to adopt his notations as far as possible.

2.1 Basic Notations

Let $\mathbb{K}$ be a field. We presume that $\mathbb{K}$ is embeddable into $\mathbb{R}$, e.g. the field of rational numbers $\mathbb{Q}$ or some field extension of $\mathbb{Q}$. Then $\mathbb{K}$ is particularly comparable and we can define $\mathbb{K}^+ = \{a \in \mathbb{K} \,|\, a > 0\}$.

We denote by $\mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1, \ldots, x_n]$ the ring of polynomials in the variables $x_1, \ldots, x_n$. If $n = 3$ we prefer to name the variables x, y, z. In the case of $n = 2$ we prefer u, v and in the case $n = 1$ we prefer t. For $n = 1, 2, 3$ we denote polynomials to be *univariate*, *bivariate* or *trivariate*, respectively. By using multi-indices $\alpha = (\alpha_1, \ldots, \alpha_n)$, a polynomial $f \in \mathbb{K}[\mathbf{x}]$ can be represented by its polynomial function $f(\mathbf{x}) = \sum_\alpha c_\alpha \mathbf{x}^\alpha$ with $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and $c_\alpha \in \mathbb{K}$. The *total degree* $\deg f$ of f is the highest occurring sum of exponents, i.e. $\deg f = \max_\alpha |\alpha| = \max_\alpha \sum_{k=1}^n \alpha_k$ with $c_\alpha \neq 0$. In case of univariate polynomials we simply say *degree*. Depending on the degree we call the first four types of univariate polynomials *linear*, *quadratic*, *cubic* or *quartic*. In our algorithms we use the predicate $\texttt{coeff} : \mathbb{K}[t] \times \mathbb{N}_0 \to \mathbb{K}$, where $\texttt{coeff}(f, k)$ returns the coefficient c_k of an univariate polynomial f.

A root of an univariate polynomial $f \in \mathbb{K}[x]$, i.e. a number ξ with $f(\xi) = 0$, is called *algebraic*. The field $\mathbb{K}$ is called *algebraically closed*, if any root ξ of any polynomial $f \in \mathbb{K}[x]$ is again an element of $\mathbb{K}$. For each field $\mathbb{K}$ it exists an algebraically closed field containing $\mathbb{K}$, called the *algebraic closure* of $\mathbb{K}$ which we denote by $\overline{\mathbb{K}}$. If f is irreducible, i.e. f is non-constant and f cannot be written as product of two or more non-constant polynomials, we call f the *minimal polynomial* of ξ. A square root $\sqrt{c}$ with $c \in \mathbb{K}^+$ is algebraic, since $x^2 - c$ is the corresponding minimal polynomial. For a fixed $c \in \mathbb{K}^+$, the set $\mathbb{K}(\sqrt{c}) = \left\{a + b\sqrt{c} \middle| a, b \in \mathbb{K}\right\}$ is called an *algebraic extension* of $\mathbb{K}$. It is clear that $\mathbb{K}(\sqrt{c})$ is a field again. We denote elements of such field extensions by *one-root numbers*.

We call an univariate polynomial $f \in \mathbb{K}[x]$ *square-free*, if f has no root with a higher multiplicity then 1. In this case we always have a sign change at a root ξ, i.e. $f(\xi - \epsilon) \cdot f(\xi + \epsilon) < 0$ for an arbitrary small ϵ. Each univariate polynomial f can be made square-free by dividing it by the greatest common divisor of f and the first derivative $\frac{d}{dx} f$, see [6].

Let $\mathbb{V}^n$ denote the Euclidean vector space over $\mathbb{K}$. To explicitly specify the underlying field we sometimes write $\mathbb{V}(\mathbb{K})^3$. We represent elements of $\mathbb{V}^n$, namely points and vectors, by lowercase letters in boldface and scalars by plain lowercase letters or Greek letters. By plain uppercase letters we represent matrices, i.e. elements of $\mathbb{V}^{n \times m}$. The only exception is the major radius of the

$\mathbf{p}, \mathbf{q}, \mathbf{a}, \mathbf{b}$	points in $\mathbb{V}^n$
A, B, M	matrices in $\mathbb{V}^{n\times m}$
$\mathfrak{p}, \mathfrak{q}, \mathfrak{a}, \mathfrak{b}$	homogeneous coordinates in $\mathbb{P}^n$
$\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$	standard unit vectors, i.e. $\mathbf{e}_1 = (1,0,0), \mathbf{e}_2 = (0,1,0), \mathbf{e}_3 = (0,0,1)$
$\mathbf{n}, \mathbf{n_q}$	normal unit vector and normal unit vector at specific point
$L(\mathbf{p}, \mathbf{d})$	line at point $\mathbf{p}$ and direction $\mathbf{d}$, i.e. $L(\mathbf{p}, \mathbf{d}) = \{\mathbf{p} + t\mathbf{d} \mid t \in \mathbb{K}\}$
$P(\mathbf{p}, \mathbf{n})$	plane containing $\mathbf{p}$ and normal to $\mathbf{n}$
$B_\delta(\mathbf{p})$	ball with centre $\mathbf{p}$ and radius δ
$S_\delta(\mathbf{p})$	sphere with centre $\mathbf{p}$ and radius δ, i.e. $S_\delta(\mathbf{p}) = \partial B_\delta(\mathbf{p})$
$C_\delta(\mathbf{p}, \mathbf{n})$	circle with centre $\mathbf{p}$ and radius δ, contained in the plane $P(\mathbf{p}, \mathbf{n})$
$E(\mathbf{p}, M)$	conic with coefficient matrix E, contained in a plane specified by $\mathbf{p}$ and local coordinate system M
$Y_\delta(\mathbf{p}, \mathbf{n})$	cylinder with radius δ and axis $L(\mathbf{p}, \mathbf{n})$
$K_\delta(\mathbf{p}, \mathbf{n})$	cone with apex $\mathbf{p}$, axis direction $\mathbf{n}$ and half angle θ, where $\delta = \tan\theta$
$T_{r,R}(\mathbf{p}, \mathbf{n})$	torus with minor radius r, major radius R, centre $\mathbf{p}$ and main circle $C_R(\mathbf{p}, \mathbf{n})$

Table 2.1: Notations for geometric primitives

torus which we denote by R due to consistency to the minor radius r. In Section 2.4.2 we make a short excurse to interval arithmetic, where we also use plain capitals for intervals. Even though we make sure that no confusion occurs. For the inner product of two vectors $\mathbf{p}, \mathbf{q} \in \mathbb{V}^n$ we write $\langle \mathbf{p}, \mathbf{q} \rangle$. Since we generally do not differ between row and column vectors, we also sometimes use the notation $\mathbf{p}^T\mathbf{q}$. In case of $n = 3$ we write the vector product $\mathbf{p} \times \mathbf{q}$.

Let $\mathbb{P}^n$ denote the projective space over $\mathbb{K}$. We use a Gothic font type for homogeneous coordinates and corresponding homogeneous functions. In the case of $n = 3$ we write $\mathfrak{p} = (x, y, z, w) \in \mathbb{P}^3$ with $x, y, z, w \in \mathbb{K}$. The point $\mathfrak{p}$ can be dehomogenised again by dividing x, y, z by the last coordinate w. In our algorithms we describe this step by the predicate $\texttt{dehom} : \mathbb{P}^n \to \mathbb{V}^n$. In case of $n = 2$ we prefer to name the components u, v, w, in case of $n = 1$ we name the components t, w.

Table 2.1 summarises the most usual notations. We reserve the letters r and R for the radii of the torus, since the torus is the main object of interest of this thesis. Thus we mostly use δ for the radii of other objects. To simplify the computations in Chapter 3 we presume, that all direction vectors and normals of geometrically given algebraic surfaces are unit, i.e. their length is equal to 1. We use the letter $\mathbf{n}$ for unit vectors, otherwise we prefer $\mathbf{d}$ for direction vectors which do not necessarily have unit length.

2.2 Geometric Primitives

Let $\mathbb{K}$ be a field. An *algebraic hypersurface* H in $\mathbb{V}^d$ is the zero set of a polynomial equation $f(\mathbf{x}) = 0$ with $f : \mathbb{V}^d \to \mathbb{K}$, i.e.

$$H = \{\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{V}^d \mid f(x_1, \ldots, x_d) = 0\}.$$

Algebraic hypersurfaces partition the underlying vector space into three parts

$$\mathbb{V}^d = H \cup H_+ \cup H_-,$$

where $H_+ = \{\mathbf{x} \in \mathbb{V}^d \mid f(\mathbf{x}) > 0\}$ and $H_- = \{\mathbf{x} \in \mathbb{V}^d \mid f(\mathbf{x}) < 0\}$, named by *positive halfspace* and *negative halfspace*, respectively. In the case of $d = 3$ we use the term *algebraic surface*, and in case

of $d = 2$ we say *algebraic curve*. The *degree* of an algebraic hypersurface is determined by the total degree of the polynomial f, which is also called *algebraic function*.

2.2.1 Quadrics

In the next two sections we will sketch the most relevant information on quadrics and conics. For a more detailed discussion on this context we refer to [50].

A *quadratic algebraic surface* (or *quadric* for short) is an algebraic surface of degree 2. A quadric Q is specified by a symmetric 4×4 matrix

$$Q = \begin{pmatrix} a & f & g & l \\ f & b & h & m \\ g & h & c & n \\ l & m & n & d \end{pmatrix}.$$

The corresponding algebraic function with affine coordinates is determined by

$$f_Q(x, y, z) = \mathbf{p}Q\mathbf{p}^T \quad \text{with} \quad \mathbf{p} = (x, y, z, 1),$$

resulting in the affine polynomial representation of the quadric

$$f_Q(x, y, z) = ax^2 + by^2 + cz^2 + 2fxy + 2gxz + 2hyz + 2lx + 2my + 2nz + d.$$

Analogue, homogeneous coordinates yield

$$f_Q(x, y, z, w) = \mathfrak{p}Q\mathfrak{p}^T \quad \text{with} \quad \mathfrak{p} = (x, y, z, w)$$

with the homogeneous polynomial

$$f_Q(x, y, z, w) = ax^2 + by^2 + cz^2 + 2fxy + 2gxz + 2hyz + 2lxw + 2myw + 2nzw + dw^2.$$

A special class of quadrics, consisting of sphere, circular cylinder and circular cone, is called *natural quadrics*, see Figure 2.1. We extend this concept by adding the plane which can be considered as a degenerate quadric where l, m, n, d are the only non-zero coefficients. We will give a short overview of the relation between the geometric and the algebraic representation of natural quadrics.

The algebraic function of a plane $P(\mathbf{p}, \mathbf{n})$, containing the point $\mathbf{p}$ and normal to $\mathbf{n}$, is

$$f_P(\mathbf{x}) = \langle \mathbf{x} - \mathbf{p}, \mathbf{n} \rangle.$$

A sphere $S_\delta(\mathbf{p})$ with centre $\mathbf{p}$ and radius δ can be represented by

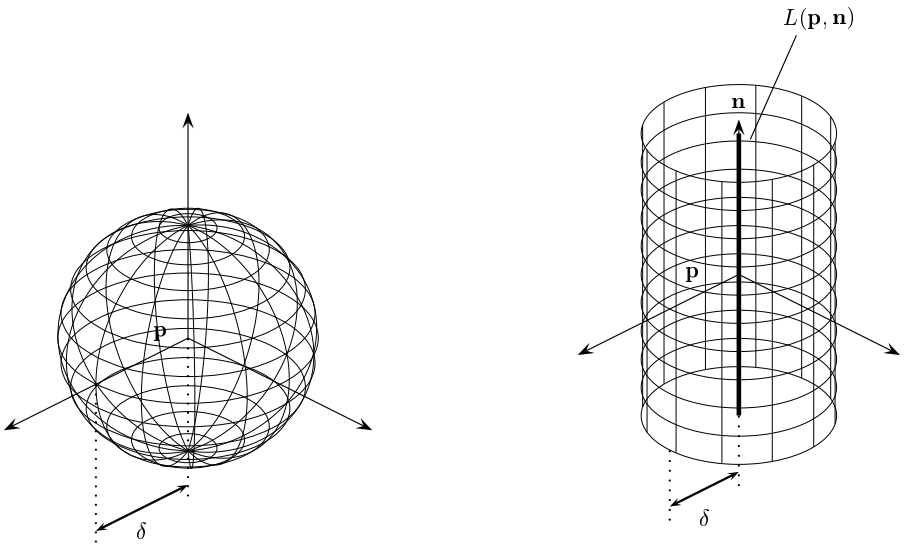
$$f_S(\mathbf{x}) = \|\mathbf{x} - \mathbf{p}\|^2 - \delta^2.$$

For a cylinder $Y_\delta(\mathbf{p}, \mathbf{n})$ with radius δ and axis $L(\mathbf{p}, \mathbf{n})$ we have

$$f_Y(\mathbf{x}) = \|(\mathbf{x} - \mathbf{p}) \times \mathbf{n}\|^2 - \delta^2.$$

In Table 2.1 we specified a cone $K_\delta(\mathbf{p}, \mathbf{n})$ with apex $\mathbf{p}$ and axis direction $\mathbf{n}$ by the tangent $\delta = \tan\theta$ instead of its half angle θ. This is because our algorithms presented in Chapters 3 and 4 use exact arithmetic and so they demand an exact input as well. If we allowed angles for parameters, we would have to evaluate trigonometric functions during computation, but these functions are not necessarily defined over the underlying field $\mathbb{K}$. Thus the algebraic function of a cone $K_\delta(\mathbf{p}, \mathbf{n})$ becomes

$$f_K(\mathbf{x}) = \|(\mathbf{x} - \mathbf{p}) \times \mathbf{n}\|^2 - \delta^2 \langle (\mathbf{x} - \mathbf{p}), \mathbf{n} \rangle^2.$$



(a) Sphere $S_\delta(\mathbf{p})$

(b) Cylinder $Y_\delta(\mathbf{p}, \mathbf{n})$

(c) Cone $K_\delta(\mathbf{p}, \mathbf{n})$

Figure 2.1: Natural quadrics

2.2.2 Conics

A *conic* is an algebraic curve of degree 2. Analogue to quadrics, conics are specified by a symmetric 3×3 matrix

$$E = \begin{pmatrix} a & f & l \\ f & b & m \\ l & m & d \end{pmatrix}$$

with the algebraic function

$$f_E(u, v) = \mathbf{p} E \mathbf{p}^T \quad \text{with} \quad \mathbf{p} = (u, v, 1)$$

and the resulting polynomial representation

$$f_E(u, v) = au^2 + bv^2 + 2fuv + 2lu + 2mv + d.$$

Similar to quadrics we also have the homogeneous counterpart

$$f_E(u, v, w) = \mathfrak{p} E \mathfrak{p}^T \quad \text{with} \quad \mathfrak{p} = (u, v, w)$$

and the homogeneous polynomial

$$f_E(u, v, w) = au^2 + bv^2 + 2fuv + 2luw + 2mvw + dw^2.$$

We call $\Delta = \det E$ the *determinant* of the conic and

$$\delta = \det \begin{pmatrix} a & f \\ f & b \end{pmatrix} = ab - f^2$$

its *discriminant*. If $\Delta = 0$, the conic degenerates into intersecting lines, parallel lines, a double line or even a single point. Otherwise, depending on the sign of δ, we can classify the conic as an *ellipse* ($\delta > 0$), a *parabola* ($\delta = 0$) or a *hyperbola* ($\delta < 0$). A *circle* is a specialised ellipse where $a = b$. If $\delta < 0$ and $a = -b$ we call the conic *rectangular hyperbola*.

Conics also emerge from the intersection between a quadric and a plane. Since we are working with three-dimensional objects, we identify a conic by its matrix representation and the plane it is embedded in. Therefore we have to specify a local coordinate system of the plane. So let $P = P(\mathbf{p}, \mathbf{n})$ be the plane, and let $\mathbf{a}, \mathbf{b} \in \mathbb{V}^3$ be two unit vectors with $\mathbf{a} \times \mathbf{b} = \mathbf{n}$. Then the mapping

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = M \begin{pmatrix} u \\ v \\ 0 \end{pmatrix} + \mathbf{p} \quad \text{with} \quad M = \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{n} \end{pmatrix}$$

transforms a point (u, v) from the local coordinate system of the plane to a point $\mathbf{x} = (x, y, z)$ in real affine space. The first two columns of the matrix M, namely the unit vectors $\mathbf{a}$ and $\mathbf{b}$, represent here the orthogonal basis, spanning the plane P. The third column $\mathbf{n}$ guarantees, that the matrix M is non singular and furthermore, that $\det M = 1$ which makes it easier to compute the inverse M^{-1}. We reverse the mapping by

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = M^{-1}(\mathbf{x} - \mathbf{p}).$$

If $w = 0$ we know that $\mathbf{x}$ lies on the plane P. In this case (u, v) is the local representation of the point $\mathbf{x}$ on P.

Thus we represent a conic by $E(\mathbf{p}, M)$ with M given from above, where E itself specifies the symmetric matrix of the conic. The point $\mathbf{p}$ lies on the plane, the conic is embedded in, and it represents the origin of the local coordinate system, given by M. In case of a circle it suffices to

specify the plane normal instead of a matrix M, since we always specify a circle with the centre at the origin of the local coordinate system, and circles are invariant under rotation. Hence we write $C_\delta(\mathbf{p}, \mathbf{n})$, where δ denotes the radius, $\mathbf{p}$ denotes the origin of the local coordinate system and $\mathbf{n}$ denotes the plane normal. From the normal $\mathbf{n}$ we can always specify a local coordinate system by e.g.

$$M_{\mathbf{n}} = \begin{cases} \mathbb{1} & \text{, if} \quad n_x^2 + n_y^2 = 0 \\ \begin{pmatrix} -\frac{n_x n_y (1+n_z)}{n_x^2+n_y^2} & \frac{n_y^2 - n_x^2 n_z}{n_x^2+n_y^2} & n_x \\ \frac{n_x^2 - n_y^2 n_z}{n_x^2+n_y^2} & -\frac{n_x n_y (1+n_z)}{n_x^2+n_y^2} & n_y \\ n_y & n_x & n_z \end{pmatrix} & \text{, otherwise.} \end{cases} \tag{2.1}$$

Though all conics have a parametrisation, we concentrate on circles since parametric representations of other conic types do not occur in our algorithms. A canonical parametrisation for a circle C with radius δ in the plane is

$$C = \left\{ \delta \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} \in \mathbb{V}(\mathbb{R})^2 \middle| \phi \in [0, 2\pi) \right\}.$$

Since we want to avoid transcendental functions, we prefer to use the equivalent rational substitutions $\cos_r, \sin_r : \mathbb{K} \to \mathbb{K}$, defined by

$$\cos_r(t) = \frac{1-t^2}{1+t^2} \quad \text{and} \quad \sin_r(t) = \frac{2t}{1+t^2}. \tag{2.2}$$

It should be clear from context, when transcendental functions or their rational respective homogeneous substitutions are used, thus we may omit the subscript r. Therewith we have a rational parametric representation of the circle and we can solve emerging equations even exactly. However, we cannot represent the angle $\phi = \pi$ which would require evaluations at infinity. By using homogeneous coordinates we bypass this problem. Let be $\mathfrak{t} = (t, w) \in \mathbb{P}$. The trigonometric substitutions become then

$$\cos_r(\mathfrak{t}) = \frac{w^2 - t^2}{w^2 + t^2} \quad \text{and} \quad \sin_r(\mathfrak{t}) = \frac{2wt}{w^2 + t^2}. \tag{2.3}$$

Hence a point at infinity, represented by $(t, 0)$, can be safely evaluated.

With homogeneous coordinates we specify the much simpler parameter form of a circle

$$\mathfrak{C} = \left\{ \begin{pmatrix} \delta(w^2 - t^2) \\ 2\delta wt \\ w^2 + t^2 \end{pmatrix} \in \mathbb{P}^2 \middle| \mathfrak{t} = (t, w) \in \mathbb{P} \right\}. \tag{2.4}$$

2.2.3 Torus

The *torus* belongs to the set of algebraic surfaces of degree 4. Given a plane $P = P(\mathbf{p}, \mathbf{n})$, the torus can be constructed by revolving a circle with radius r around the axis $L(\mathbf{p}, \mathbf{n})$, where the circle is coplanar to the axis and its centre lies on the plane P in the distance R to $\mathbf{p}$. We call P the *main plane* of the torus, $\mathbf{p}$ its *centre* and r and R the *minor radius* and *major radius*, respectively. Furthermore we denote the circle $C_R(\mathbf{p}, \mathbf{n})$ by *main circle*, which is the trajectory of the revolving circle's centre, see Figure 2.2. Depending on the radii there are three types of a torus, see Figure 2.3, namely *ring torus* ($r < R$), *horn torus* ($r = R$) and *spindle torus* ($r > R$). In this thesis we considered only ring tori as valid input. However, in our algorithms the other types my occur as well by constructing configuration space obstacles, see Section 2.3.

The algebraic function of a torus $T_{r,R}(\mathbf{p}, \mathbf{n})$ in general position and orientation is

$$f_T(\mathbf{x}) = \left(\|\mathbf{x} - \mathbf{p}\|^2 + R^2 - r^2 \right)^2 - 4R^2 \|(\mathbf{x} - \mathbf{p}) \times \mathbf{n}\|^2. \tag{2.5}$$

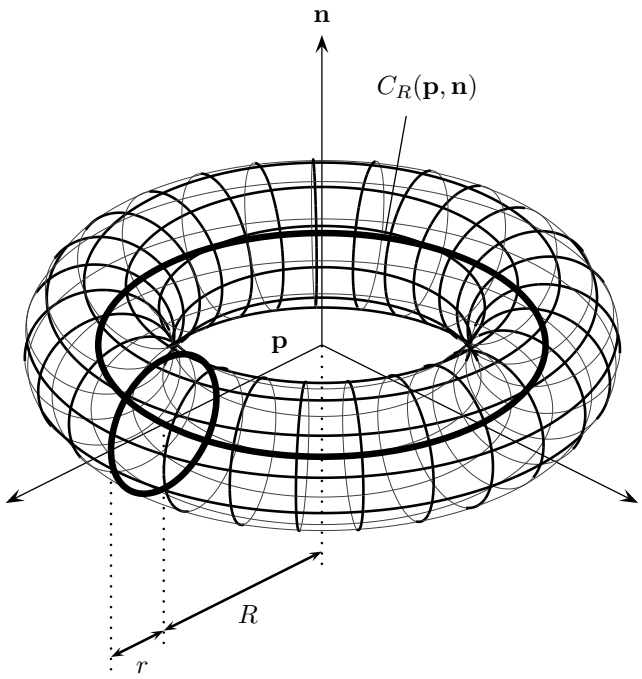


Figure 2.2: Torus $T_{r,R}(\mathbf{p}, \mathbf{n})$ with revolving circle

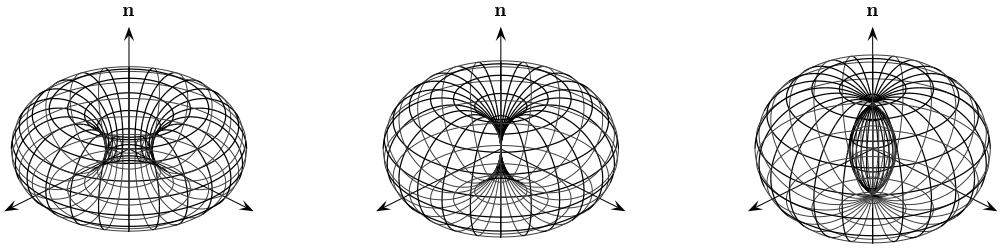


Figure 2.3: Ring torus (left), horn torus (middle) and spindle torus (right)

By *canonical position* we denote the position at the origin with orientation collinear to the z-axis. In this case, the algebraic representation for a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ simplifies to

$$T = \left\{(x, y, z) \in \mathbb{V}^3 \middle| (x^2 + y^2 + z^2 + R^2 - r^2)^2 - 4R^2(x^2 + y^2) = 0\right\}.$$

Since the torus is a surface of revolution, it can be easily parametrised. For simplicity and due to the fact that we solely need this case, we parametrise the torus T in canonical position by

$$T = \left\{ \begin{pmatrix} (R + r\cos\phi)\cos\theta \\ (R + r\cos\phi)\sin\theta \\ r\sin\phi \end{pmatrix} \in \mathbb{V}(\mathbb{R})^3 \middle| \phi, \theta \in [0, 2\pi) \right\}.$$

Analogue to the previous section we substitute the trigonometric functions by their rational counterparts and gain the rational homogeneous parametrisation

$$\mathfrak{T} = \left\{ \begin{pmatrix} [R(1 + s^2) + r(1 - s^2)](1 - t^2) \\ [R(1 + s^2) + r(1 - s^2)]2t \\ 2rs(1 + t^2) \\ (1 + s^2)(1 + t^2) \end{pmatrix} \in \mathbb{P}^3 \middle| s, t \in \mathbb{K} \right\}. \tag{2.6}$$

For a constant parameter s we obtain the parametric representation of a circle which is a subset of the torus. We denote these circles by *profile circles*. Circles with a constant parameter t are denoted by *cross-sectional circles*. In parameter space profile and cross-sectional circles are represented by straight lines collinear to the s- respective t-axis. Furthermore there is a third type of circles embedded in a torus called *Villarceau circles* after the french mathematician Antoine Joseph François Yvon-Villarceau [75]. For more details see Section 3.1.3.

For a practical use this approach has the disadvantage, that we can neither represent the profile circle $C_{R-r}(\mathbf{o}, \mathbf{e}_3)$ nor the cross-sectional circle $C_r((-R, 0, 0), \mathbf{e}_2)$ parametrically, which would require evaluation at infinity. We could again bypass this problem by using homogeneous coordinates for the parameters s and t, which would result in a four-dimensional parameter space. So we rather use the substitutions $\hat{s} = 1/s$ and $\hat{t} = 1/t$ instead, resulting in some sign changes in the parametric representation (2.6). By the restriction of the parameter domain to the interval $[-1, 1]$, see [12], we represent the torus by the union of four separate parametrisations.

2.3 Configuration Space

The *Configuration Space* (C-space) approach has been originally developed in robotics [51, 52]. Motivated by collision-free path finding, the C-space approach tests the trajectory of the moving object against the blown-up area (so-called *forbidden region*) of the obstructive object, instead of considering the relative positions of both objects. Thus it reduces the collision detection problem of two objects to a simpler collision problem with less dimensionality.

In our case we treat one object as the envelope surface of a moving ball along a corresponding trajectory (a sphere in C-space becomes a point, a cylinder becomes a line and a torus becomes a circle). The other object is considered as an obstacle in C-space by adding an offset about the moving ball's radius. The computations reduce from pure 3D intersections to 3D-2D intersections. The big advantage of working with tori and natural quadrics is, that an offset surface belongs to the same object class as the original surface, i.e. the offset surface of a sphere becomes a sphere again, etc., see [2].

For illustration we consider the case of intersecting a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ with a cylinder $Y = Y_\delta(\mathbf{p}, \mathbf{n})$, see Figure 2.4. We regard the torus as envelope surface of a moving ball with radius r along the circle $C = C_R(\mathbf{o}, \mathbf{e}_3)$. We gain the C-space obstacle by offsetting the cylinder about the moving ball's radius, which results in two cylinders $Y^O = Y_{\delta+r}(\mathbf{p}, \mathbf{n})$ and $Y^I = Y_{\delta-r}(\mathbf{p}, \mathbf{n})$. Instead of intersecting the torus T with the cylinder Y, we consider intersections of the circle C with the offset surfaces Y^O and Y^I. These intersections reduce furthermore to cylinder-plane intersections and circle-conic intersections.

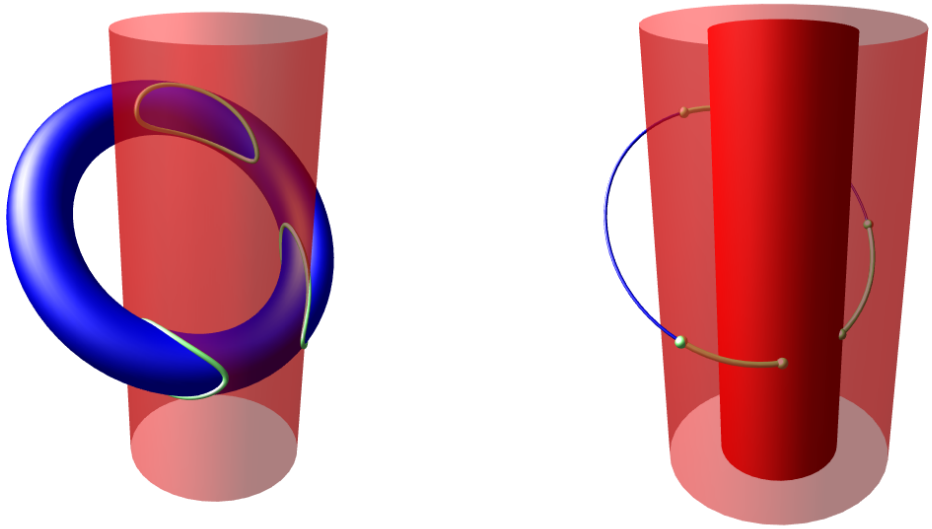


Figure 2.4: Torus-cylinder intersection in real affine space (left) and in C-space (right)

In case of $\delta < r$, we consider the cylinder Y as a moving ball with radius δ along the line $L(\mathbf{p}, \mathbf{n})$ and build the obstacle by offsetting the torus T, resulting in two tori $T_{r+\delta,R}(\mathbf{o}, \mathbf{e}_3)$ and $T_{r-\delta,R}(\mathbf{o}, \mathbf{e}_3)$. In this case we apply torus-line intersections.

Table 2.2 summarises all intersection types and their respective C-space operations.

affine space	C-space
torus-plane	plane-plane, circle-line
torus-sphere	(i) sphere-plane, circle-circle
	(ii) torus-point
torus-cylinder	(i) cylinder-plane, circle-conic
	(ii) torus-line
torus-cone	cone-plane, circle-conic
torus-torus	torus-circle

Table 2.2: Real affine space intersections and C-space intersections

Kim [46] showed that connected components in C-space correspond to connected components in real affine space. In our illustration we have three separate circle arcs without tangential intersections in C-space, hence we get three corresponding regular surface patches in real affine space. He showed further that tangential intersections in C-space correspond to intersection curve components with singularities in real affine space. Thus we can determine the topology of the intersection curve already in C-space by analysing the relative positions of simple objects like points, lines and circles.

2.4 Arithmetic and Data Type Concepts

In this section we present the basic concepts for the arithmetic and the corresponding data types we used in our implementation. Furthermore we clarify the terminology which is essentially used in our algorithms.

Since we claim to provide exact results, we make no use of machine-intern floating-point arithmetic due to rounding errors. Instead we use variable sized rational numbers for the computational basis. This has the advantage that standard operators like $+, -, *, /$ provide exact results, which is essential for testing the sign of an expression, for instance. Note that, using floating-point arithmetic, the question *"Is $f(x)$ equal to zero?"* with some function f may return three different answers for the same value x, depending on how rounding errors propagate through the expression. For a detailed discussion and some examples on this subject see [73, 56]. The disadvantages in

using exact arithmetic are i.a. a higher computation time since the bitsize of numbers increases normally after operations. Especially rational numbers should be cancelled after a few operations to prevent exploding bitsizes in numerator and denominator. Another disadvantage is the lack of transcendental functions like trigonometric functions, square-root and logarithm. In particular, this makes the normalisation of a vector difficult. The major work of this thesis is made up of bypassing these problems and finding alternative solutions.

For our implementation we used `leda::rational`[1] and `CORE::BigRat`[2] as data type for variable sized rational numbers. Both types provide comparable functionality and by using generic programming techniques we are able to interchange them without any additional effort. In some cases it is necessary to handle objects with parameters lying in some algebraic extension of $\mathbb{Q}$. We use `NiX::Sqrt_extension` to handle these cases, see next section.

2.4.1 Algebraic Numbers

As we saw in Section 2.1, a root ξ of a polynomial $f \in \mathbb{K}[x]$ is called *algebraic*. In case of $\deg f = 1$ we write

$$f(x) = ax + b \quad \text{with} \quad a, b \in \mathbb{K} \quad \text{and} \quad a \neq 0.$$

Then the root ξ can be computed exactly by

$$\xi = -\frac{b}{a}$$

and therewith $\xi \in \mathbb{K}$. In the case of $\deg f = 2$ we apply the well-known quadratic formula to determine both roots $\xi_{1,2}$, which lie in the algebraic closure $\overline{\mathbb{K}}$. So let be

$$f(x) = ax^2 + bx + c \quad \text{with} \quad a, b, c \in \mathbb{K} \quad \text{and} \quad a \neq 0.$$

Then the quadratic formula yields

$$\xi_{1,2} = -\frac{b}{2a} \pm \frac{1}{2a}\sqrt{b^2 - 4ac}.$$

We call the term under the square root *discriminant* of f and set $d := b^2 - 4ac$. Depending on the sign of d we classify the roots:

- If $d < 0$, both roots $\xi_{1,2}$ lie in the algebraic closure $\overline{\mathbb{K}}$.
- If $d = 0$, the roots coincide and we gain the double root $\xi = -\frac{b}{2a}$ which is an element of $\mathbb{K}$.
- If $d > 0$, we call the roots *simple* and both roots lie in the algebraic extension field $\mathbb{K}(\sqrt{d})$.

Interesting for us are the second and third case only, since we assume $\mathbb{K}$ to be embeddable into $\mathbb{R}$ and so we omit possibly occurring complex roots.

To represent simple roots, we recall that elements of an algebraic extension $\mathbb{K}(\sqrt{d})$ are one-root numbers and thus have the form $a_0 + a_1\sqrt{d}$ with some $a_0, a_1 \in \mathbb{K}$. A data type for representing one-root numbers is `NiX::Sqrt_extension`[3], which allows the interchangeability of the underlying field $\mathbb{K}$ by the use of generic programming techniques. Therewith we can even represent nested algebraic extensions by using `NiX::Sqrt_extension` recursively. We generally denote such numbers by *square-root expressions* and especially elements of a double nested extension $\mathbb{K}(\sqrt{d})(\sqrt{e})$ with $e \in \mathbb{K}(\sqrt{d})$ by *two-root numbers*. We note that only elements of the same algebraic extension are interoperable, i.e. operating on two one-root numbers $a \in \mathbb{K}(\sqrt{d})$ and $b \in \mathbb{K}(\sqrt{e})$ with $d \neq e$ does

[1] LEDA is a `C++` class library for efficient data types and algorithms, distributed by *Algorithmic Solutions Software GmbH* (see http://www.algorithmic-solutions.com/)

[2] CORE is a `C++` class library, supporting the *Exact Geometric Computation (EGC)* approach for numerically robust algorithms (see http://cs.nyu.edu/exact/)

[3] NiX is a part of the EXACUS library, developed by the Max Planck-Institute for Computer Science in Saarbrücken (see http://www.mpi-inf.mpg.de/projects/EXACUS/)

not normally yield an one-root number again. So we have to take care, that, once we defined an algebraic extension, we have to stay inside of it.

However, we sometimes cannot prevent comparing elements of different algebraic extensions. Therefor we define the predicates `sign`, `less` and `equal` for one-root numbers, given in Algorithms 1, 2 and 3, respectively. Thereby we presume the availability of standard operators and the `sign` predicate on elements of $\mathbb{K}$. Although we can evolve all standard comparison operators $<, \leq, >, \geq$ including $=, \neq$ by use of the `less` predicate, we implemented an `equal` predicate due to running time aspects. Using `C++` as programming language, which provides template-based operator overloading, we can even compare elements of nested algebraic extensions, e.g. if $\mathbb{K}$ itself is an algebraic extension like $\mathbb{Q}(\sqrt{c})$.

Roots of polynomials of degree three and four can be determined explicitly by the formulae of Cardano and Ferrari [16], but we do not dwell on these due to the utilisation of transcendental functions like cubic roots and cosines. In the following we consider roots of a polynomial f with $\deg f > 2$. If f is irreducible under $\mathbb{K}$, roots of f are not simple anymore, so we cannot represent them exactly by using `NiX::Sqrt_extension`. Instead we first determine a separate domain for each root to isolate them. There are various methods to achieve this like Uspensky's algorithm [19], Sturm sequences and isolation by differentiation. For a survey see [20]. A generic algebraic number α with $f(\alpha) = 0$ can then be represented by its defining polynomial f plus the corresponding isolating interval $[a, b]$ with $a, b \in \mathbb{K}$, i.e.

$$\alpha \in [a, b] \quad \text{and} \quad \beta \notin [a, b] \quad \text{for all} \quad \beta \neq \alpha \quad \text{with} \quad f(\beta) = 0.$$

For $a \neq b$ we furthermore presume $f(a) \neq 0$, $f(b) \neq 0$ and f to be square-free to guarantee a sign change at the bounds of the interval, i.e. $f(a) \cdot f(b) < 0$. Otherwise we have $\alpha = a = b \in \mathbb{K}$ and so we even found an exact representation.

The isolating intervals can be arbitrarily refined by the bisection method. Since we often make use of this and its quite fundamental, we shortly present the method. For an enhancement of this method see [1].

Let $[a, b]$ be an isolating interval. We know that $[a, b]$ contains exactly one root α. Furthermore it holds $f(a) \cdot f(b) < 0$. Setting $c := \frac{a+b}{2} \in \mathbb{K}$, the rational midpoint of the interval, there are three cases for the sign of $f(c)$:

- $f(c) = 0$: We found the rational root $\alpha = c$ and we can stop the algorithm.
- $f(c) \cdot f(a) < 0$: There is a sign change in $[a, c]$ and so α must lie in this interval. We set $[a, c]$ to be the new isolating interval and continue.
- $f(c) \cdot f(a) > 0$: There is no sign change in $[a, c]$ and so α must lie in the interval $[c, b]$. We set $[c, b]$ to be the new isolating interval and continue.

We repeat bisecting the interval until $b - a \leq \epsilon$ for a given upper bound ϵ. A data type providing such functionality is `NiX::Algebraic_real`. From now on we call instances of this data type *algebraic reals*. In our algorithms we use the term `AlgebraicReal`$(f, [a, b])$ which creates an algebraic real with defining polynomial f and isolating interval $[a, b]$. From a given algebraic real α we extract the lower and upper bound of the isolating interval by the predicates `lower`(α) and `upper`(α), respectively. To perform one interval refining iteration step, we use the predicate `refine`(α). To refine to a given precision ϵ, such that `upper`$(\alpha) -$ `lower`$(\alpha) \leq \epsilon$, we write `refine`(α, ϵ).

Even though arbitrary algebraic numbers, represented by isolating intervals, can be compared exactly, see [30, 72], we just need to compare algebraic numbers with common defining polynomial, which can easily be done by comparing the corresponding isolating intervals. Therefor the intervals must not overlap. Let α and β be two algebraic reals with common defining polynomial f and isolating intervals $[a, b]$ and $[c, d]$. We refine both intervals such that either $b < c$ or $d < a$. In the first case we have $\alpha < \beta$, in the latter case we have $\beta < \alpha$. To stay consistent to other number types, we use the predicate `less` for comparison. Note that we change the status of the algebraic reals inside the predicate. So we may assume, that two algebraic numbers are represented by distinct intervals after comparison.

Algorithm 1 $\texttt{sign}(a_0 + a_1\sqrt{d})$

Requires: $a_0, a_1 \in \mathbb{K}\,,\ d \in \mathbb{K}^+$

Returns: $\begin{cases} -1, & a_0 + a_1\sqrt{d} < 0 \\ 0, & a_0 + a_1\sqrt{d} = 0 \\ 1, & a_0 + a_1\sqrt{d} > 0 \end{cases}$

1: **if** $\texttt{sign}(a_0) = \texttt{sign}(a_1)$ **then**
2: **return** $\texttt{sign}(a_0)$
3: **else**
4: **return** $\texttt{sign}(a_1)\cdot\texttt{sign}(a_1^2 d - a_0^2)$
5: **end if**

Algorithm 2 $\texttt{less}(a_0 + a_1\sqrt{d}, b_0 + b_1\sqrt{e})$

Requires: $a_0, a_1, b_0, b_1 \in \mathbb{K}\,,\ d, e \in \mathbb{K}^+$

Returns: $\begin{cases} \textbf{true}, & a_0 + a_1\sqrt{d} < b_0 + b_1\sqrt{e} \\ \textbf{false}, & \text{otherwise} \end{cases}$

1: $s \leftarrow \texttt{sign}(b_1)$
2: **if** $s = 0$ **then**
3: **return** $(\texttt{sign}(b_0 - a_0 - a_1\sqrt{d}) = 1)$
4: **end if**
5: **if** $s = \texttt{sign}(a_1)$ **then**
6: $s \leftarrow s\cdot\texttt{sign}(b_1^2 e - a_1^2 d)$
7: **end if**
8: $t \leftarrow \texttt{sign}(a_0 - b_0)$
9: **if** $t = 0$ **then**
10: **return** $(s = 1)$
11: **else**
12: **if** $s \neq t$ **then**
13: **return** $(t = -1)$
14: **else**
15: **return** $(t\cdot\texttt{sign}(a_1^2 d + b_1^2 e - (a_0 - b_0)^2 - 2a_1 b_1\sqrt{de}) = 1)$
16: **end if**
17: **end if**

Algorithm 3 $\texttt{equal}(a_0 + a_1\sqrt{d}, b_0 + b_1\sqrt{e})$

Requires: $a_0, a_1, b_0, b_1 \in \mathbb{K}\,,\ d, e \in \mathbb{K}^+$

Returns: $\begin{cases} \textbf{true}, & a_0 + a_1\sqrt{d} = b_0 + b_1\sqrt{e} \\ \textbf{false}, & \text{otherwise} \end{cases}$

1: **if** $a_0 = b_0$ **then**
2: **return** $(\texttt{sign}(a_1) = \texttt{sign}(b_1) \wedge a_1^2 d = b_1^2 e)$
3: **else**
4: $s \leftarrow \texttt{sign}(b_1)$
5: **if** $s = 0$ **then**
6: **return** $(\texttt{sign}(b_0 - a_0 - a_1\sqrt{d}) = 0)$
7: **end if**
8: **if** $s = \texttt{sign}(a_1)$ **then**
9: $s \leftarrow s\cdot\texttt{sign}(b_1^2 e - a_1^2 d)$
10: **end if**
11: **return** $(s = \texttt{sign}(a_0 - b_0)$ **and** $\texttt{sign}(a_1^2 d + b_1^2 e - (a_0 - b_0)^2 - 2a_1 b_1\sqrt{de}) = 0)$
12: **end if**

2.4.2 Interval Arithmetic

Interval arithmetic is applicable in a broad field of scientific engineering, for a good survey see [43]. We focus on the estimation of function ranges, but nonetheless we shortly recall the basic ideas in the following. For a detailed discussion see [70].

In this work we delimit intervals by brackets, $\mathbb{IR}$ denotes the set of intervals and capital letters denote elements of $\mathbb{IR}$. Interval vectors are therewith written in boldface capital letters and called *boxes*. Further on underscores denote lower bounds of intervals and overscores denote upper bounds. Basic operators on intervals with exact boundary number types are defined by:

$$\begin{aligned} A + B &= [\underline{A} + \underline{B}, \overline{A} + \overline{B}] \\ A - B &= [\underline{A} - \overline{B}, \overline{A} - \underline{B}] \\ A * B &= [\min\{\underline{A}\underline{B}, \underline{A}\overline{B}, \overline{A}\underline{B}, \overline{A}\overline{B}\}, \max\{\underline{A}\underline{B}, \underline{A}\overline{B}, \overline{A}\underline{B}, \overline{A}\overline{B}\}] \\ 1/A &= [1/\overline{A}, 1/\underline{A}] \quad \text{if} \quad \underline{A} > 0 \quad \text{or} \quad \overline{A} < 0 \\ A/B &= A * (1/B) \end{aligned}$$

The fundamental property of interval arithmetic is the *inclusion property*, i.e. for a function f that can be extended to a function $\square f$, defined on intervals, we always have $f(x) \in \square f(X)$ for all $x \in X$. We call such a function $\square f$ *inclusion function*. The basic operators from above are apparently inclusion functions. This property is very useful to gain some information about large areas in one step. An example: If $0 \notin \square f(X)$, i.e. $\underline{\square f(X)} > 0$ or $\overline{\square f(X)} < 0$, we know that there is no root of f in the whole interval X. Note that the negation is normally not correct, i.e. from $0 \in \square f(X)$ we cannot conclude that there is a root $x \in X$ with $f(x) = 0$, since inclusion functions usually return an overestimation of the real function range. A task on working with interval arithmetic is to minimise this overestimation.

The data type we used for our implementation is `boost::numeric::interval`[4]. The use of generic programming allows again the interchangeability of the number type for the interval bounds. For built-in types like `float` or `double`, a correct rounding is provided to guarantee the inclusion property. To be able to work with arbitrarily small intervals, we used `leda::bigfloat` respectively `CORE::BigFloat` which have the same structure as floating-point numbers, but with arbitrarily large mantissa and exponent.

2.4.3 Homogeneous Parameter Forms

Beside conic sections, which are in fact circles and exactly representable, the Algebraic Part returns a set of initial starting points for the later tracing step. This set includes singularities as well as regular starting points for each intersection curve component. We will see that singularities can also be represented exactly by two-root numbers. Regular starting points on the other hand are in general evolved from simple roots of quartic polynomials and thus are represented by algebraic reals, which save the polynomials and the root isolating intervals, see Section 2.4.1. More precisely:

Let $P(t)$ be the parameter form of object A and $f_B(\mathbf{x})$ the implicit algebraic function of object B. The intersection points $\mathbf{q}$ between A and B can be computed by substituting the parameter form into the implicit algebraic function, solving for roots t_i and evaluating the parameter form at these roots, i.e.

$$A \cap B = \big\{\mathbf{q}_i = P(t_i) \big| t_i \text{ with } f_B(P(t_i)) = 0\big\}.$$

For exact roots t_s we can compute $P(t_s)$ exactly as well. An algebraic real α on the other hand cannot be plugged into the parameter form P in practice. However, the regular curve tracing operates with approximations of curve points, hence regular starting points as well, fulfilling a given error bound condition in advance, see Section 5.1. Thus it suffices to refine the isolating interval $I = [a, b]$ of α and to evaluate $\mathbf{Q} = P(I)$, using interval arithmetic, until $\mathbf{Q}$ satisfies the error bound condition.

[4] BOOST is a collection of software libraries, extending the functionality of `C++` (see http://www.boost.org/)

Let us now concentrate on the functionality to separate the Algebraic from the Numeric Part. Since then we are able to change the error bound for the curve tracing without recomputing the starting points.

We keep simple roots as algebraic reals, and we keep also the parameter form P for an evaluation at a later point. Due to generalisation, we prefer homogeneous forms. A *homogeneous parameter form* is therewith a map

$$\mathfrak{P} : \mathbb{K}^n \to \mathbb{P}^3.$$

We have already seen such a form in (2.6).

The advantage to work with homogeneous coordinates is, we can apply linear transformations, i.e.

$$M(\mathfrak{P}(t)) = (M\mathfrak{P})(t) \quad \text{with} \quad M \in \mathbb{P}^{3\times 3}.$$

The homogeneous parameter form of a circle $C = C_\beta(\mathbf{p}, \mathbf{n})$ with $M_\mathbf{n} = \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{n} \end{pmatrix}$ becomes then

$$\mathfrak{P}_C(t) = \begin{pmatrix} M_\mathbf{n} & \mathbf{p} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \beta(1-t^2) \\ 2\beta t \\ 0 \\ 1+t^2 \end{pmatrix} = \begin{pmatrix} [p_x - \beta a_x]t^2 + [2\beta b_x]t + [p_x + \beta a_x] \\ [p_y - \beta a_y]t^2 + [2\beta b_y]t + [p_y + \beta a_y] \\ [p_z - \beta a_z]t^2 + [2\beta b_z]t + [p_z + \beta a_z] \\ t^2 + 1 \end{pmatrix}. \tag{2.7}$$

The homogeneous parameter form of a line $L = L(\mathbf{p}, \mathbf{d})$ is

$$\mathfrak{P}_L(t) = \begin{pmatrix} d_x t + p_x \\ d_y t + p_y \\ d_z t + p_x \\ 1 \end{pmatrix}. \tag{2.8}$$

The data type $\texttt{IntersectionPoint}(t, \mathfrak{P})$ saves an algebraic real t as parameter and a homogeneous parameter form $\mathfrak{P}$. In the Numeric Part t can be refined arbitrarily and $\mathfrak{P}$ can be evaluated at an interval representation of t, i.e. the enclosing interval of the real root. The set of all starting points consists then of the union of exactly represented singularities and regular starting points in the form of `IntersectionPoint` data types. We usually write $\{\mathbf{q}_s\}$ for the set of singularities and $\{\mathbf{q}_r\}$ for the set of regular starting points. To stay variable, we may even represent tangential intersection points by the `IntersectionPoint` data type, since we sometimes need to compare intersection points depending on their parameter t.

Additionally to initial starting points, the Algebraic Part returns even intersection curves which can be parametrised. At any time we can parametrise a curve we prefer this representation since then we can spare expensive computations during the Numeric Part which includes some Newton-like iteration schemes in each tracing step. With a given parametrisation of the whole curve we just have to evaluate some algebraic functions at different parameters. We give a more detailed description in Section 5.2. Fortunately we can express all occurring parametrisations of intersection curves by rational polynomial functions. So even here we use homogeneous parameter forms which evaluate to a proper value after the dehomogenising step.

2.4.4 Useful Data Types

For a simpler reading of our algorithms we introduce some useful data types, which save the results of both the Algebraic as well as the Numeric Part.

The Numeric Part is responsible for the tracing of the intersection curve. Since we use a typical predictor-corrector approach to trace a regular curve, the results consists of a set of linear splines, one linear spline for each intersection curve component. We save a linear spline in a list of vertices $[\mathbf{v}]$. We use brackets in order to distinguish a list from an unordered set. Although we use a different approach to trace parametrised curves, the output here are linear splines as well.

In summary our algorithms for computing the intersection curve between a torus and another simple surface return a data structure, representing the intersection curve by

- a set of circles $\{C\}$ (containing conic sections),
- a set of singular points $\{\mathbf{q}_s\}$,
- a set of regular starting points $\{\mathbf{q}_r\}$,
- a set of homogeneous parameter forms $\{\mathfrak{P}\}$ (containing parametrisable curve components),
- a set of linear splines $\{[\mathbf{v}]\}$ (containing traced curve components).

A vertex $\mathbf{v}$ of a linear spline is represented by variable-sized floating-point numbers. A regular starting point $\mathbf{q}_r$ is either an exact point from $\mathbb{V}^3$ or an `IntersectionPoint` data type. We will see that a circle C can be represented by one-root numbers, a singularity $\mathbf{q}_s$ by two-root numbers and that a homogeneous parameter form $\mathfrak{P}$ consists of polynomials of degree four at most.

Algebraic Part

Chapter 3

Conic Sections

The detection of conic sections in intersection algorithms is important since conics can be represented exactly and evaluated efficiently. Moreover they frequently occur in engineering applications. In the case of intersecting quadrics, there are several approaches to detect conic sections [35, 58, 68, 69], some even provide a rational parametrisation [32]. In combination with the torus, the conic sections become circles, which is a result of classical geometry. A proof can be found in [46]. Since our algorithms for computing the intersections between a torus and a simple surface are based on a geometric approach, we represent these circles geometrically, i.e. by the centre $\mathbf{q}$, the radius β and the normal $\mathbf{n}$ of the plane the circle is embedded in. As input we have a torus $T_{r,R}(\mathbf{o}, \mathbf{e}_3)$, lying in canonical position, and a simple surface like a plane $P(\mathbf{p}, \mathbf{n})$, a natural quadric $S_\delta(\mathbf{p})$, $Y_\delta(\mathbf{p}, \mathbf{n})$ or $K_\delta(\mathbf{p}, \mathbf{n})$, or another torus $T_{\delta,\Delta}(\mathbf{p}, \mathbf{n})$. Depending on the parameters $r, R, \delta, \Delta, \mathbf{p}, \mathbf{n}$ we formulate all necessary and sufficient conditions for any occurring conic section and we present its exact representation.

This chapter is organised like Chapter 3 in Kim's thesis [46]. First we determine the different sets of possible circles embedded in each type of simple surface. Then we determine the necessary and sufficient conditions by comparing any pair of circle sets in torus-plane intersections (TPI), torus-sphere intersections (TSI), torus-cylinder intersections (TYI), torus-cone intersections (TKI) and torus-torus intersections (TTI) (Sections 3.2-3.6). At last we present an alternative approach for detecting conic sections by analysing the intersection curve in parameter space of the torus (Section 3.7).

We note that we compute an exact representation of conic sections. Furthermore we try to avoid transcendental functions like sine, cosine and square roots as far as possible. If we cannot get around square roots, we use a minimal number of algebraic field extensions. This requires new approaches of computing conic sections in most cases (e.g. in torus-sphere intersections). For the few other cases, where basic operations suffice, we adopt Kim's computations.

Since all coloured figures in this thesis are screenshots from our implementation, they may contain additional intersection curve components which are no circles. Therefore we highlight circles in intersection curves by using a yellow colour, whereas other intersection curve components are coloured in green. Furthermore we mark singularities with small orange spheres.

Since this chapter consists almost completely of demanding computations, the reader may skip this chapter without taking the risk of missing important parts for the understanding of the superior algorithm to compute the whole intersection curve.

3.1 Circle Sets of Simple Surfaces

For simplicity we adopt Kim's notations. So let Γ_T and Γ_Q denote the sets of all circles embedded in a torus T and a quadric Q, respectively, where Q can be specialised into a sphere S, a cylinder Y and a cone K. For simpler computations during the comparison step later on, we demand each set to be a one-parameter family of circles. Therefore we partition the circle set of the torus Γ_T into four disjoint subsets Γ_T^i with $i \in \{p, c, v+, v-\}$, where p denotes profile circles, c denotes

cross-sectional circles and v denotes Villarceau circles. Furthermore we consider the torus lying in canonical position first. Then we can easily evolve arbitrary positions from that by applying an appropriate transformation on the results, since a torus $T = T_{r,R}(\mathbf{p}, \mathbf{n})$ can be represented by

$$T = \left\{ \mathbf{p} + M_{\mathbf{n}}\mathbf{x} \middle| \mathbf{x} \in T_{r,R}(\mathbf{o}, \mathbf{e}_3) \right\}$$

with matrix $M_{\mathbf{n}}$ from (2.1).

3.1.1 Profile Circles of the Torus

Given a torus $T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ in canonical position with the rational parametric function (2.6) from Section 2.2.3. Profile circles are obtained by retaining the parameter s to a constant s_0. The parametric function $T(s_0, t)$ yields then a rotation of a point $\mathbf{q} = (R + r\cos(s_0), 0, r\sin(s_0))$ around the z-axis, which describes a circle with radius q_x, centre $(0, 0, q_z)$ and normal $\mathbf{e}_3$. The set of profile circles consists therewith of

$$\Gamma_T^p = \left\{ C_{R+r\cos(s_0)}(r\sin(s_0)\mathbf{e}_3, \mathbf{e}_3) \middle| s_0 \in \mathbb{K} \right\}. \tag{3.1}$$

3.1.2 Cross-sectional Circles of the Torus

Analogue to Section 3.1.1 we can represent cross-sectional circles of a torus in canonical position with parametric function (2.6) by retaining the parameter t to a constant t_0. For the value $t_0 = 0$ we get a rotation of the point $\mathbf{q} = (R + r, 0, 0)$ around the line $L((R, 0, 0), \mathbf{e}_2)$, which describes a circle $C_r((R, 0, 0), \mathbf{e}_2)$. Other values for t_0 just rotate this circle around the z-axis. Thus the set of cross-sectional circles consists of

$$\Gamma_T^c = \left\{ C_r((R\cos(t_0), R\sin(t_0), 0), (-\sin(t_0), \cos(t_0), 0)) \middle| t_0 \in \mathbb{K} \right\}. \tag{3.2}$$

3.1.3 Villarceau Circles of the Torus

Villarceau [75] states, that each point on a torus lies on four different circles. Two of them are obviously a profile and a cross-sectional circle. The other two circles lie in planes, each tangential to the torus at two points. Consider the point $\mathbf{q} = (0, -(R + r), 0)$ on the y-axis. The profile and cross-sectional circles, going through $\mathbf{q}$, are $C_{R+r}(\mathbf{o}, \mathbf{e}_3)$ and $C_r((0, -R, 0), \mathbf{e}_1)$, respectively. The tangential planes are

$$P^{\pm} = P(\mathbf{o}, \frac{1}{R}(\pm r, 0, \sqrt{R^2 - r^2})).$$

Due to symmetry we may concentrate on one plane. The tangential intersections between P^+ and the torus are the points

$$\pm\frac{1}{R}(R^2 - r^2, 0, -r\sqrt{R^2 - r^2}),$$

see Figure 3.1(b). If we rotate the two objects around the y-axis about the angle $\phi = \arcsin(\frac{r}{R})$, plane P^+ becomes equal to the xy-plane. Then the implicit function (2.5) of the rotated torus $T_{r,R}(\mathbf{o}, \frac{1}{R}(-r, 0, \sqrt{R^2 - r^2}))$, constrained to the xy-plane, decomposes into

$$f_T(\mathbf{x})\big|_{z=0} = [x^2 + (y + r)^2 - R^2][x^2 + (y - r)^2 - R^2]$$

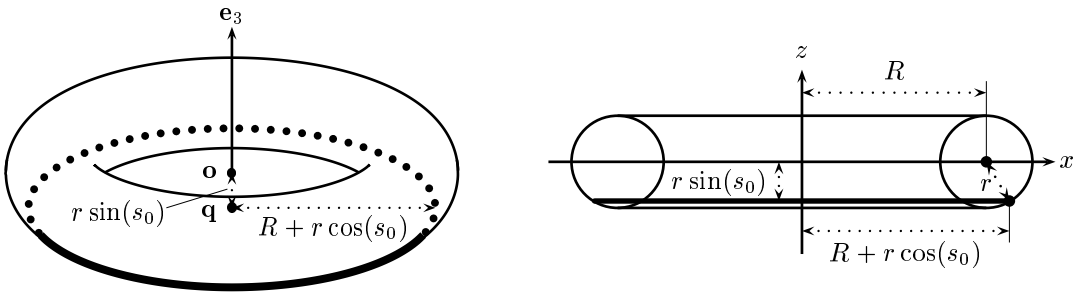
which represent two circles with radius R and centres $(0, \pm r, 0)$. Thus the Villarceau circles embedded in P^+ are

$$C_R((0, \pm r, 0), \frac{1}{R}(r, 0, \sqrt{R^2 - r^2})).$$

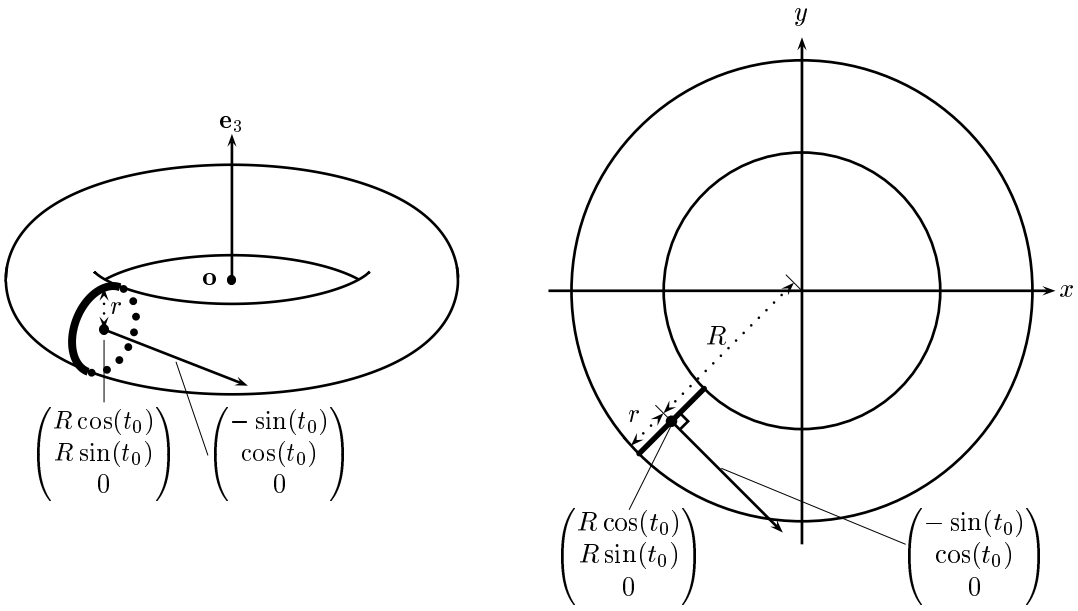
Rotating around the z-axis yields the two distinct sets of Villarceau circles

$$\Gamma_T^{v+} = \left\{ C_R((-r\sin(u), r\cos(u), 0), \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2})) \middle| u \in \mathbb{K} \right\} \tag{3.3}$$

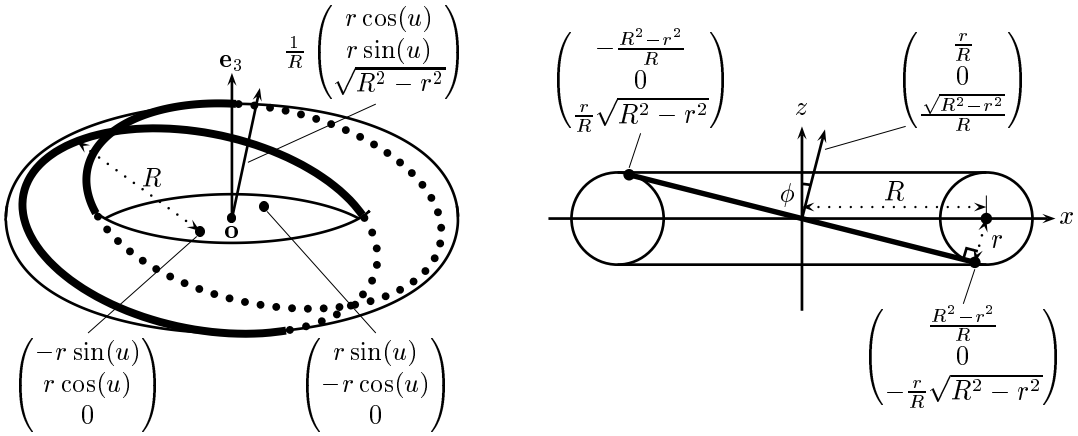
$$\Gamma_T^{v-} = \left\{ C_R((r\sin(u), -r\cos(u), 0), \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2})) \middle| u \in \mathbb{K} \right\}. \tag{3.4}$$



(a) Profile circle



(b) Cross-sectional circle



(c) Villarceau circles

Figure 3.1: Circles of a Torus

3.1.4 Circles of the Sphere

Each point $\mathbf{q}$ inside the ball $B_\delta(\mathbf{p})$ can be centre of a circle lying on the sphere $S_\delta(\mathbf{p}) = \partial B_\delta(\mathbf{p})$. Let $\mathbf{q}$ be represented by $\mathbf{q} = \mathbf{p} + c\mathbf{n}$ with some unit normal vector $\mathbf{n} \in S_1(\mathbf{o})$ and $0 \leq c < \delta$. In case of $c = \delta$, $\mathbf{q}$ would lie on the sphere and the corresponding circle would degenerate into a single point. We can easily disregard this case since these points (called touching points or isolated singularities) will be detected by another modality anyway.

The resulting circle is then embedded in a plane $P(\mathbf{q}, \mathbf{n})$. The radius d of the circle can be computed as follows, see Figure 3.2(a):

$$d = \sqrt{\delta^2 - c^2}.$$

In summary the set of embedded circles in a sphere $S_\delta(\mathbf{p})$ is represented by

$$\Gamma_S = \left\{ C_{\sqrt{\delta^2-c^2}}(\mathbf{p} + c\mathbf{n}, \mathbf{n}) \middle| 0 \leq c < \delta,\ \mathbf{n} \in S_1(\mathbf{o}) \right\}. \tag{3.5}$$

3.1.5 Circles of the Cylinder

A circle, embedded in a cylinder $Y_\delta(\mathbf{p}, \mathbf{n})$, must lie in a plane perpendicular to the cylinder's axis $L(\mathbf{p}, \mathbf{n})$. The centre $\mathbf{q}$ of the circle can therewith be expressed by $\mathbf{q} = \mathbf{p} + c\mathbf{n}$ with some parameter $c \in \mathbb{K}$. The circle's radius equals the radius δ of the cylinder, see Figure 3.2(b).

The set of embedded circles in a cylinder $Y_\delta(\mathbf{p}, \mathbf{n})$ consists therewith of

$$\Gamma_Y = \left\{ C_\delta(\mathbf{p} + c\mathbf{n}, \mathbf{n}) \middle| c \in \mathbb{K} \right\}. \tag{3.6}$$

3.1.6 Circles of the Cone

Likewise to the cylinder case, a circle, embedded in a cone $K_\delta(\mathbf{p}, \mathbf{n})$, must lie in a plane perpendicular to the cone's axis $L(\mathbf{p}, \mathbf{n})$. The circle's centre $\mathbf{q}$ can be expressed again by $\mathbf{q} = \mathbf{p} + c\mathbf{n}$ with some parameter $c \in \mathbb{K}$. But here the radius d computes to $d = \delta\|\mathbf{q} - \mathbf{p}\| = \delta|c|$, see Figure 3.2(c).

The set of embedded circles in a cone $K_\delta(\mathbf{p}, \mathbf{n})$ consists therewith of

$$\Gamma_K = \left\{ C_{\delta|c|}(\mathbf{p} + c\mathbf{n}, \mathbf{n}) \middle| c \in \mathbb{K} \right\}. \tag{3.7}$$

3.2 Circle Detection in TPI

To detect circles in a torus-plane intersection (Figure 3.3) it suffices to consider the four circle sets of the torus Γ_T^p, Γ_T^c and $\Gamma_T^{v\pm}$. Let $P = P(\mathbf{p}, \mathbf{n})$ be the given plane and $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ the torus in canonical position. Then a circle $C = C_\delta(\mathbf{q}, \mathbf{m})$ is embedded in P, if and only if

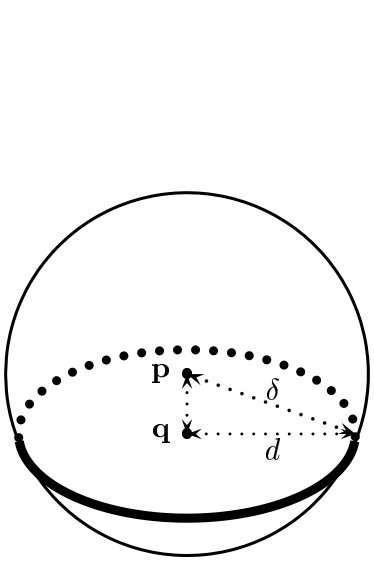
- the normals $\mathbf{n}$ and $\mathbf{m}$ are collinear, i.e.

$$\mathbf{n} \times \mathbf{m} = \mathbf{o}, \tag{3.8}$$

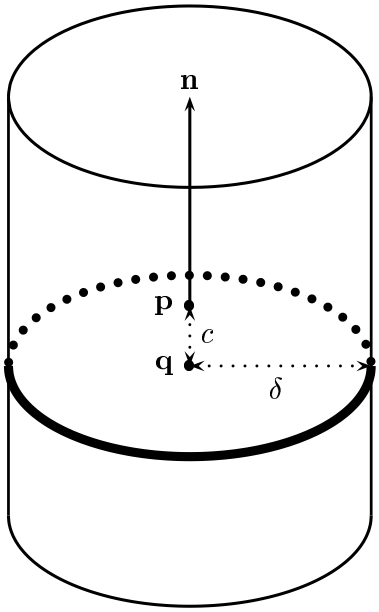
- $\mathbf{q}$ lies in the plane P, i.e.

$$\langle \mathbf{q} - \mathbf{p}, \mathbf{n} \rangle = 0. \tag{3.9}$$

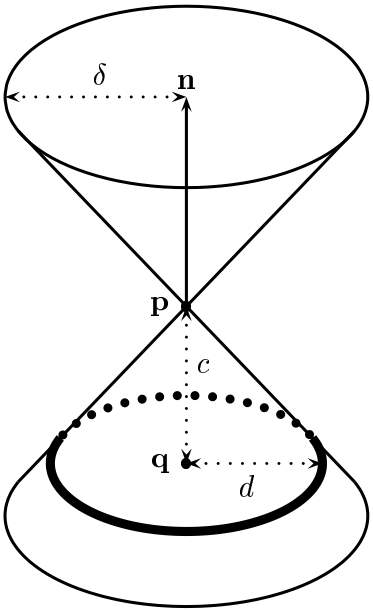
In the following we specialise these conditions for any type of circles (profile, cross-sectional and Villarceau circles). We will see that in each case of occurring circles exactly two circles are involved. Since the algebraic degree of the intersection curve is equal to four and the algebraic degree of two circles sum to four as well, there is no additional part of the intersection curve other than these two circles. So we can even stop the algorithm for computing the whole intersection curve already at this point.



(a) Sphere $S_\delta(\mathbf{p})$



(b) Cylinder $Y_\delta(\mathbf{p}, \mathbf{n})$



(c) Cone $K_\delta(\mathbf{p}, \mathbf{n})$

Figure 3.2: Circles of Natural Quadrics

3.2.1 Profile Circles in TPI

Adopting condition (3.8) on the set

$$\Gamma_T^p = \left\{ C_{R+r\cos(s_0)}(r\sin(s_0)\mathbf{e}_3, \mathbf{e}_3) \middle| s_0 \in \mathbb{K} \right\}$$

yields

$$n_x = n_y = 0. \tag{3.10}$$

With condition (3.9) we get

$$r\sin(s_0) - p_z = 0.$$

From this we can demand

$$|p_z| \leq r. \tag{3.11}$$

If the conditions (3.10) and (3.11) are fulfilled, the intersection curve consists of two circles of the form

$$C_{R\pm\sqrt{r^2-p_z^2}}((0,0,p_z), \mathbf{e}_3).$$

If $|p_z| = r$, the two circles coincide to one singular circle $C_R((0,0,p_z), \mathbf{e}_3)$, where each point on this circle is a touching point.

3.2.2 Cross-sectional Circles in TPI

Considering the set

$$\Gamma_T^c = \left\{ C_r((R\cos(t_0), R\sin(t_0), 0), (-\sin(t_0), \cos(t_0), 0)) \middle| t_0 \in \mathbb{K} \right\},$$

condition (3.8) yields

$$\mathbf{n} = \pm(-\sin(t_0), \cos(t_0), 0)$$

with an appropriate $t_0 \in \mathbb{K}$, in particular

$$n_z = 0. \tag{3.12}$$

Together with the second condition (3.9) we get

$$\langle \mathbf{p}, \mathbf{n} \rangle = 0. \tag{3.13}$$

Since we can derive the midpoint of the resulting cross-sectional circles in terms of $\mathbf{n}$, there is no need to explicitly compute t_0. So if conditions (3.12) and (3.13) are fulfilled, we get two cross-sectional circles of the form

$$C_r(\pm R(\mathbf{n} \times \mathbf{e}_3), \mathbf{n}).$$

3.2.3 Villarceau Circles in TPI

We can express the union of the circle sets Γ_T^{p+} and Γ_T^{p-} by one set

$$\Gamma_T^{p\pm} = \left\{ C_R(\pm R(\mathbf{n}_u \times \mathbf{e}_3), \mathbf{n}_u) \middle| \mathbf{n}_u = \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2}), u \in \mathbb{K} \right\}.$$

Condition (3.8) demands $\mathbf{n} \times \mathbf{n}_u = \mathbf{o}$, in particular

$$n_x^2 + n_y^2 = \frac{r^2}{R^2}, \tag{3.14}$$

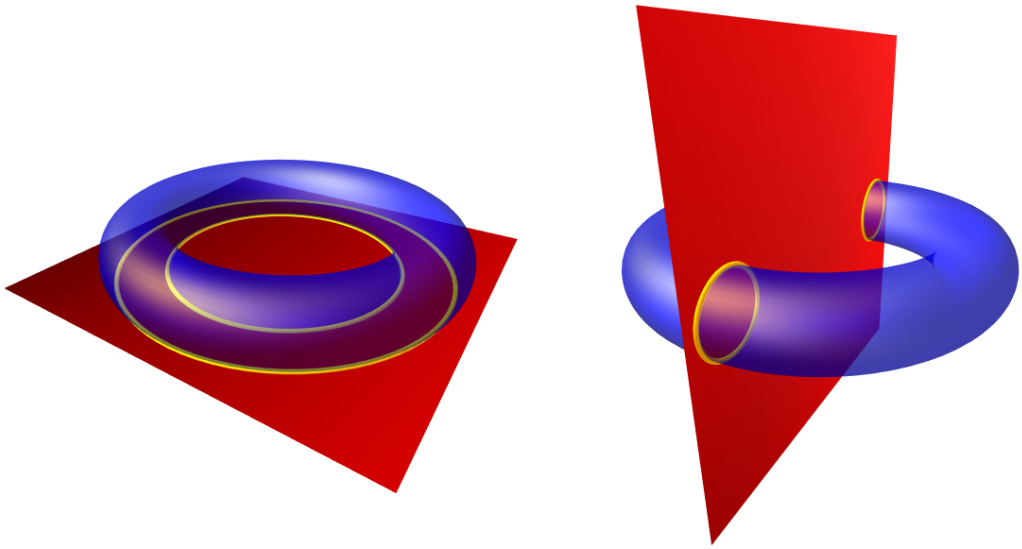
avoiding some square root expressions. Condition (3.9) yields again

$$\langle \mathbf{p}, \mathbf{n} \rangle = 0. \tag{3.15}$$

Similar to Section 3.2.2 the intersection curve consists of two circles of the form

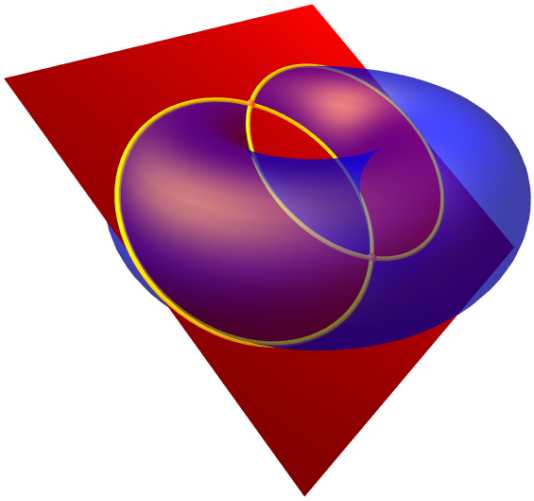
$$C_R(\pm R(\mathbf{n} \times \mathbf{e}_3), \mathbf{n}),$$

if conditions (3.14) and (3.15) are fulfilled.



(a) Profile circles

(b) Cross-sectional circles



(c) Villarceau circles

Figure 3.3: Circles in torus-plane intersection

3.3 Circle Detection in TSI

Let $S = S_\delta(\mathbf{p})$ be a sphere and $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ be a torus in canonical position. A circle embedded in the intersection curve of S and T must lie in one of the intersections of the circle set of the sphere Γ_S and the circle sets of the torus $\Gamma_T^i,\ i \in \{p, c, v\pm\}$. In the following we formulate all necessary and sufficient conditions to find non empty intersections $\Gamma_S \cap \Gamma_T^i$ and we specify the representations of their elements. We will see that in each case a non empty intersection consists of either one or two circles. In case of one circle, each point of this circle is a touching point between S and T. We call such circles singular. Since the algebraic degree of an intersection curve between a sphere and a torus is equal to four, see [46], there are no additional intersection curve components.

We consider the set of all circles embedded in the sphere S

$$\Gamma_S = \Big\{ C_{\sqrt{\delta^2 - c^2}}(\mathbf{p} + c\mathbf{n}, \mathbf{n}) \big| 0 \le c < \delta,\ \mathbf{n} \in S_1(\mathbf{o}) \Big\}.$$

3.3.1 Profile Circles in TSI

We compare the set Γ_S with the set of profile circles of the torus

$$\Gamma_T^p = \Big\{ C_{R + r\cos(s_0)}(r\sin(s_0)\mathbf{e}_3, \mathbf{e}_3) \big| s_0 \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_S \cap \Gamma_T^p$ the following conditions must hold:

$$\begin{aligned} \mathbf{n} &= \pm\mathbf{e}_3, \\ p_x = p_y &= 0, \\ p_z \pm c &= r\sin(s_0), \\ \delta^2 - c^2 &= (R + r\cos(s_0))^2. \end{aligned} \tag{3.16}$$

Solving the last two equations for s_0 after eliminating c we get

$$\begin{aligned} s_0 &= \frac{1}{\delta^2 - p_z^2 - (R-r)^2}\big[-2rp_z \pm \sqrt{d}\big], \\ d &= -[(\delta - r)^2 - R^2 - p_z^2][(\delta + r)^2 - R^2 - p_z^2]. \end{aligned} \tag{3.17}$$

To gain real solutions, the discriminant d must be non negative. This yields the condition

$$(\delta - r)^2 \le p_z^2 + R^2 \le (\delta + r)^2. \tag{3.18}$$

In case of equality both solutions coincide and the intersection curve consists of one singular circle, where each point of this circle is a touching point between S and T.

Otherwise, if conditions (3.16) and (3.18) are fulfilled, we can compute two circles by substituting (3.17) into the profile circle representation of set Γ_T^p, which yields

$$C_\beta((0, 0, \gamma), \mathbf{e}_3)$$

with

$$\begin{aligned} \beta &= \frac{1}{2(R^2 + p_z^2)}\big[R(R^2 + p_z^2 - r^2 + \delta^2) \pm p_z\sqrt{d}\big], \\ \gamma &= \frac{1}{2(R^2 + p_z^2)}\big[p_z(R^2 + p_z^2 + r^2 - \delta^2) \pm R\sqrt{d}\big], \\ d &= -[(\delta - r)^2 - R^2 - p_z^2][(\delta + r)^2 - R^2 - p_z^2]. \end{aligned}$$

3.3.2 Cross-sectional Circles in TSI

We compare the set Γ_S with the set of cross-sectional circles of the torus

$$\Gamma_T^c = \Big\{ C_r((R\cos(t_0), R\sin(t_0), 0), (-\sin(t_0), \cos(t_0), 0)) \big| t_0 \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_S \cap \Gamma_T^c$ the following conditions must hold:

$$\begin{aligned} n_z &= 0, \\ p_x &= R\cos(t_0) \pm c\sin(t_0), && (3.19) \\ p_y &= R\sin(t_0) \mp c\cos(t_0), && (3.20) \\ p_z &= 0, && (3.21) \\ \delta^2 &= r^2 + c^2, \end{aligned}$$

where the last equation yields immediately

$$\delta \geq r. \tag{3.22}$$

Together with equations (3.19) and (3.20) we follow that

$$p_x^2 + p_y^2 = R^2 + \delta^2 - r^2. \tag{3.23}$$

If conditions (3.21)-(3.23) are fulfilled, we can find up to two cross-sectional circles embedded in the intersection curve. In case of $\delta = r$ both circles coincide to one singular circle. We note that $\mathbf{p}$ as well as the centres of the circles $\mathbf{q}$ lie in the main plane of the torus. So we can express $\mathbf{q}$ by

$$\mathbf{q} = \beta\mathbf{p} + \gamma(\mathbf{p} \times \mathbf{e}_3)$$

with some appropriate constants $\beta, \gamma \in \mathbb{K}$. From Figure 3.4(b) we derive the following relations:

$$\begin{aligned} \|\mathbf{p}\|^2 &= c^2 + R^2, \\ R^2 &= \beta^2 + \gamma^2, \\ c^2 &= (\|\mathbf{p}\| - \beta)^2 + \gamma^2. \end{aligned}$$

This yields

$$\begin{aligned} \beta &= \frac{R^2}{\|\mathbf{p}\|}, \\ \gamma &= \pm\frac{R}{\|\mathbf{p}\|}\sqrt{\delta^2 - r^2}. \end{aligned}$$

The cross-sectional circles can then be expressed by

$$C_r(\mathbf{q}, \mathbf{n})$$

with

$$\mathbf{q} = \frac{R}{\|\mathbf{p}\|^2}\begin{pmatrix} Rp_x \pm \sqrt{\delta^2 - r^2}p_y \\ Rp_y \mp \sqrt{\delta^2 - r^2}p_x \\ 0 \end{pmatrix} \quad \text{and} \quad \mathbf{n} = \frac{1}{\|\mathbf{p}\|^2}\begin{pmatrix} -Rp_y \pm \sqrt{\delta^2 - r^2}p_x \\ Rp_x \pm \sqrt{\delta^2 - r^2}p_y \\ 0 \end{pmatrix}.$$

3.3.3 Villarceau Circles in TSI

We compare the set Γ_S with the set of Villarceau circles of the torus

$$\Gamma_T^{v\pm} = \Big\{ C_R((\mp r\sin(u), \pm r\cos(u), 0), \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2})) \big| u \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_S \cap \Gamma_T^{v\pm}$ the following conditions must hold:

$$\begin{aligned} \mathbf{o} &= \mathbf{n} \times (r\cos(u), r\sin(u), \sqrt{R^2 - r^2}), \\ \mathbf{p} + c\mathbf{n} &= (\mp r\sin(u), \pm r\cos(u), 0), \qquad (3.24) \\ \delta^2 &= R^2 + c^2. \qquad (3.25) \end{aligned}$$

To avoid considering too many sign variations, we redefine the set Γ_S as follows:

$$\Gamma_S = \Big\{ C_{\sqrt{\delta^2 - c^2}}(\mathbf{p} + c\mathbf{n}, \mathbf{n}) \Big| -\delta < c < \delta,\ \mathbf{n} \in S_1(\mathbf{o}),\ n_z \geq 0 \Big\}.$$

Then we can write directly

$$\mathbf{n} = \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2}).$$

From equation (3.24) and (3.25) we gain the two conditions

$$p_x^2 + p_y^2 = \frac{\delta^2 r^2}{R^2}, \qquad (3.26)$$

$$p_z^2 = \frac{1}{R^2}(\delta^2 - R^2)(R^2 - r^2), \qquad (3.27)$$

where the latter one already implies

$$\delta \geq R.$$

If conditions (3.26) and (3.27) are fulfilled, the intersection curve of the sphere S and the torus T consists of two Villarceau circles

$$C_R(\mathbf{q}, \mathbf{n}).$$

To determine $\mathbf{q}$ and $\mathbf{n}$ we consider again the set of Villarceau circles of the torus and substitute

$$\mathbf{q} = (\mp r\sin(u), \pm r\cos(u), 0).$$

Then we can express the normal $\mathbf{n}$ by

$$\mathbf{n} = \frac{1}{R}\big[\pm (\mathbf{q} \times \mathbf{e}_3) + \sqrt{R^2 - r^2}\mathbf{e}_3 \big]. \qquad (3.28)$$

With equation (3.24) we get

$$R\mathbf{q} \mp c(\mathbf{q} \times \mathbf{e}_3) = R\mathbf{p} + c\sqrt{R^2 - r^2}\mathbf{e}_3. \qquad (3.29)$$

Building the cross-product of this equation with $\mathbf{e}_3$ yields

$$R(\mathbf{q} \times \mathbf{e}_3) \pm c\mathbf{q} = R(\mathbf{p} \times \mathbf{e}_3). \qquad (3.30)$$

With (3.29) and (3.30) we can eliminate the term $\mathbf{q} \times \mathbf{e}_3$ and gain the explicit formula for $\mathbf{q}$

$$\mathbf{q} = \frac{R}{R^2 + c^2}\big[R\mathbf{p} + c\sqrt{R^2 - r^2}\mathbf{e}_3 \pm c(\mathbf{p} \times \mathbf{e}_3) \big].$$

To avoid working with different radicals and loosing the right sign, we do not resolve c by (3.25) but by the third component of (3.24), what leads to

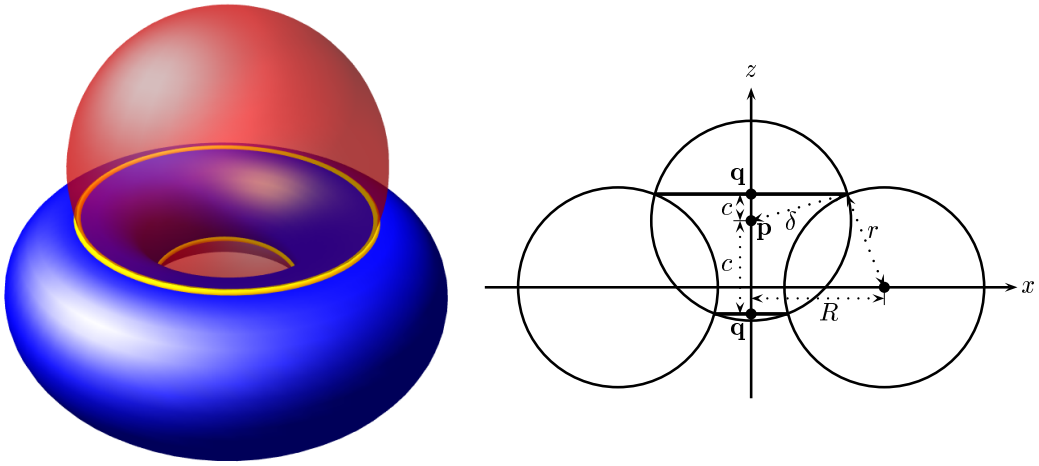
$$c = -\frac{p_z R}{\sqrt{R^2 - r^2}}.$$

With (3.25) the explicit representation of $\mathbf{q}$ becomes

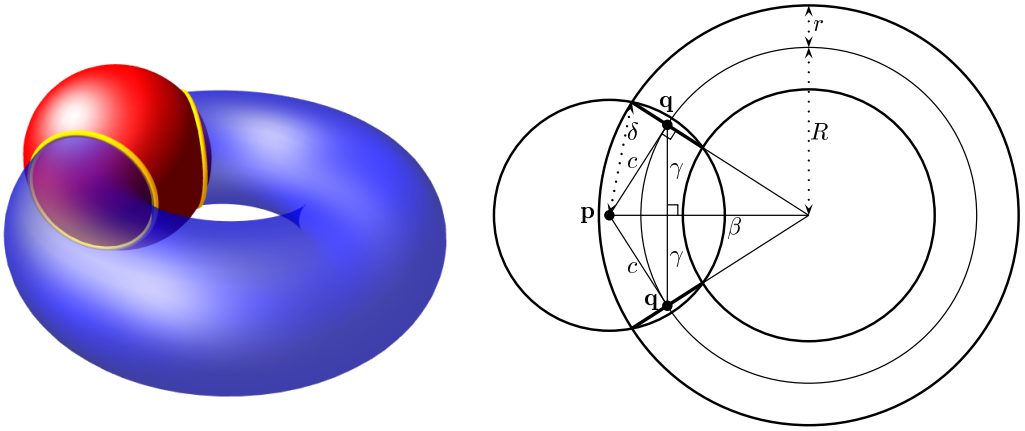
$$\mathbf{q} = \frac{R^2}{\delta^2}\begin{pmatrix} p_x \mp \frac{p_z}{R^2 - r^2}\sqrt{R^2 - r^2}p_y \\ p_y \pm \frac{p_z}{R^2 - r^2}\sqrt{R^2 - r^2}p_x \\ 0 \end{pmatrix}.$$

From (3.28) we get

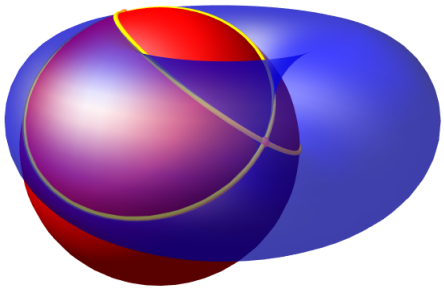
$$\mathbf{n} = \frac{R}{\delta^2}\begin{pmatrix} \pm p_y + \frac{p_z}{R^2 - r^2}\sqrt{R^2 - r^2}p_x \\ \mp p_x + \frac{p_z}{R^2 - r^2}\sqrt{R^2 - r^2}p_y \\ \frac{\delta^2}{R^2}\sqrt{R^2 - r^2} \end{pmatrix}.$$



(a) Profile circles



(b) Cross-sectional circles



(c) Villarceau circles

Figure 3.4: Circles in torus-sphere intersection

3.4 Circle Detection in TYI

Let $Y = Y_\delta(\mathbf{p}, \mathbf{n})$ be a cylinder and $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ be a torus in canonical position. A circle embedded in the intersection curve of Y and T must lie in one of the intersections of the circle set of the cylinder Γ_Y and the circle sets of the torus $\Gamma_T^i,\ i \in \{p, c, v\pm\}$. In the following we formulate all necessary and sufficient conditions to find non empty intersections $\Gamma_Y \cap \Gamma_T^i$ and we specify the representations of their elements.

We consider the set of all circles embedded in the cylinder Y

$$\Gamma_Y = \Big\{ C_\delta(\mathbf{p} + c\mathbf{n}, \mathbf{n}) \big| c \in \mathbb{K} \Big\}.$$

3.4.1 Profile Circles in TYI

We compare the set Γ_Y with the set of profile circles of the torus

$$\Gamma_T^p = \Big\{ C_{R+r\cos(s_0)}(r\sin(s_0)\mathbf{e}_3, \mathbf{e}_3) \big| s_0 \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_Y \cap \Gamma_T^p$ the following conditions must hold:

$$\mathbf{n} = \pm\mathbf{e}_3, \tag{3.31}$$
$$p_x = p_y = 0, \tag{3.32}$$
$$p_z \pm c = r\sin(s_0), \tag{3.33}$$
$$\delta = R + r\cos(s_0). \tag{3.34}$$

From (3.34) we follow directly

$$|\delta - R| \leq r. \tag{3.35}$$

Furthermore (3.34) leads to

$$s_0 = \pm\sqrt{\frac{r - \delta + R}{r + \delta - R}}.$$

Equation (3.33) contains no further information since p_z does not occur in other equations and c is free of choice. So if conditions (3.31),(3.32) and (3.35) are fulfilled, the intersection curve consists of two profile circles

$$C_\delta((0, 0, \gamma), \mathbf{e}_3) \quad \text{with} \quad \gamma = \pm\sqrt{r^2 - (\delta - R)^2}.$$

In case of equality in (3.35) both circles coincide to one singular circle

$$C_\delta(\mathbf{o}, \mathbf{e}_3).$$

3.4.2 Cross-sectional Circles in TYI

We compare the set Γ_Y with the set of cross-sectional circles of the torus

$$\Gamma_T^c = \Big\{ C_r((R\cos(t_0), R\sin(t_0), 0), (-\sin(t_0), \cos(t_0), 0)) \big| t_0 \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_Y \cap \Gamma_T^c$ the following conditions must hold:

$$\mathbf{n} = \pm(-\sin(t_0), \cos(t_0), 0), \tag{3.36}$$
$$\mathbf{p} + c\mathbf{n} = (R\cos(t_0), R\sin(t_0), 0), \tag{3.37}$$
$$\delta = r. \tag{3.38}$$

From (3.36)-(3.37) we get directly

$$n_z = 0, \tag{3.39}$$
$$p_z = 0. \tag{3.40}$$

From (3.37) we gain furthermore

$$\|\mathbf{p}\|^2 = R^2 + c^2,$$

and by building the dot-product with $\mathbf{n}$ we get

$$\langle \mathbf{p}, \mathbf{n} \rangle + c = 0.$$

Together we have

$$\|\mathbf{p}\|^2 = R^2 + \langle \mathbf{p}, \mathbf{n} \rangle^2. \tag{3.41}$$

If conditions (3.38)-(3.41) are fulfilled, one and only one cross-sectional circle is embedded in the intersection curve with the following representation:

$$C_r(\mathbf{p} - \langle \mathbf{p}, \mathbf{n} \rangle \mathbf{n}, \mathbf{n}).$$

3.4.3 Villarceau Circles in TYI

We compare the set Γ_Y with the set of Villarceau circles of the torus

$$\Gamma_T^{v\pm} = \Big\{ C_R((\mp r\sin(u), \pm r\cos(u), 0), \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2})) \big| u \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_Y \cap \Gamma_T^{v\pm}$ the following conditions must hold:

$$\mathbf{o} = \mathbf{n} \times (r\cos(u), r\sin(u), \sqrt{R^2 - r^2}), \tag{3.42}$$

$$\mathbf{p} + c\mathbf{n} = (\mp r\sin(u), \pm r\cos(u), 0), \tag{3.43}$$

$$\delta = R. \tag{3.44}$$

From (3.42) we can follow directly

$$n_z^2 = \frac{R^2 - r^2}{R^2}. \tag{3.45}$$

The dot-product of (3.43) with $\mathbf{n}$ yields

$$\langle \mathbf{p}, \mathbf{n} \rangle + c = 0. \tag{3.46}$$

If we do the same with the cross-product we get

$$\|\mathbf{p} \times \mathbf{n}\|^2 = r^2. \tag{3.47}$$

At last we get from the third component of (3.43) and with (3.46)

$$\langle \mathbf{p}, \mathbf{n} \rangle^2 = \frac{R^2 p_z^2}{R^2 - r^2}. \tag{3.48}$$

If conditions (3.44)-(3.47) and (3.48) are fulfilled, a Villarceau circle with the representation

$$C_R(\mathbf{p} - \langle \mathbf{p}, \mathbf{n} \rangle \mathbf{n}, \mathbf{n})$$

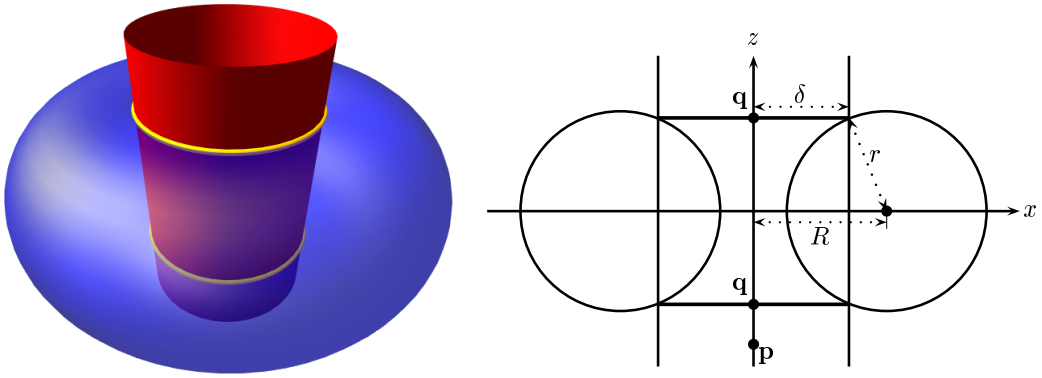
is embedded in the intersection curve.

3.5 Circle Detection in TKI

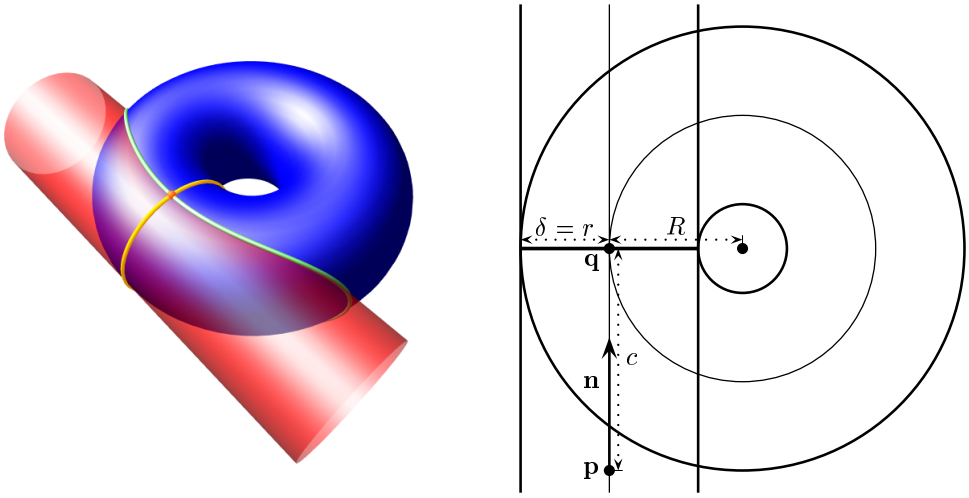
Let $K = K_\delta(\mathbf{p}, \mathbf{n})$ be a cone and $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ be a torus in canonical position. A circle embedded in the intersection curve of K and T must lie in one of the intersections of the circle set of the cone Γ_K and the circle sets of the torus $\Gamma_T^i,\ i \in \{p, c, v\pm\}$. In the following we formulate all necessary and sufficient conditions to find non empty intersections $\Gamma_K \cap \Gamma_T^i$ and we specify the representations of their elements.

We consider the set of all circles embedded in the cone K

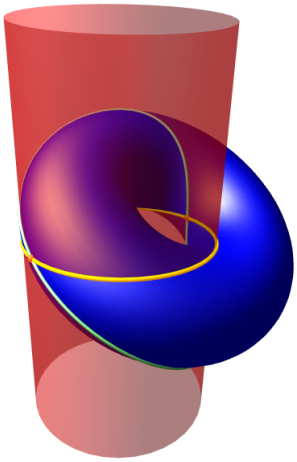
$$\Gamma_K = \Big\{ C_{\delta|c|}(\mathbf{p} + c\mathbf{n}, \mathbf{n}) \big| c \in \mathbb{K} \Big\}.$$



(a) Profile circles



(b) Cross-sectional circles



(c) Villarceau circles

Figure 3.5: Circles in torus-cylinder intersection

3.5.1 Profile Circles in TKI

We compare the set Γ_K with the set of profile circles of the torus

$$\Gamma_T^p = \Big\{ C_{R+r\cos(s_0)}(r\sin(s_0)\mathbf{e}_3, \mathbf{e}_3) \big| s_0 \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_K \cap \Gamma_T^p$ the following conditions must hold:

$$n_x = n_y = 0, \tag{3.49}$$
$$p_x = p_y = 0, \tag{3.50}$$
$$p_z + cn_z = r\sin(s_0), \tag{3.51}$$
$$\delta|c| = R + r\cos(s_0). \tag{3.52}$$

Note that (3.49) already implies

$$|n_z| = 1.$$

From (3.51) and (3.52) we derive then

$$R + r\cos(s_0) = \epsilon_1\delta(r\sin(s_0) - p_z) \quad \text{with} \quad \epsilon_1 = \pm 1.$$

This leads to

$$s_0 = \frac{\delta r + \epsilon_2\sqrt{d}}{\delta p_z + \epsilon_1(R-r)} \quad \text{with} \quad \epsilon_2 = \pm 1,\ d = r^2(1+\delta^2) - (R + \epsilon_1\delta p_z)^2.$$

Since ϵ_1 and ϵ_2 are independent, we may find up to four different solutions, provided that the discriminant $d \geq 0$. This is the case, if

$$-\frac{r\sqrt{1+\delta^2} + \epsilon_1 R}{\delta} \leq p_z \leq \frac{r\sqrt{1+\delta^2} - \epsilon_1 R}{\delta}. \tag{3.53}$$

Equality on one side in (3.53) leads to the case that two circles coincide to one singular circle, see Table 3.1.

If conditions (3.49) and (3.50) are fulfilled and (3.53) holds for an $\epsilon_1 \in \{1, -1\}$, we can find up to two profile circles in each case:

$$C_\beta((0,0,\gamma), \mathbf{e}_3)$$

with

$$\beta = \frac{\delta^2 R - \epsilon_1\delta(p_z \pm \sqrt{d})}{1+\delta^2},$$
$$\gamma = \frac{\delta^2 p_z + \epsilon_1\delta R \mp \sqrt{d}}{1+\delta^2},$$
$$d = r^2(1+\delta^2) - (R + \epsilon_1\delta p_z)^2.$$

3.5.2 Cross-sectional Circles in TKI

We compare the set Γ_K with the set of cross-sectional circles of the torus

$$\Gamma_T^c = \Big\{ C_r((R\cos(t_0), R\sin(t_0), 0), (-\sin(t_0), \cos(t_0), 0)) \big| t_0 \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_K \cap \Gamma_T^c$ the following conditions must hold:

$$\mathbf{n} = \pm(-\sin(t_0), \cos(t_0), 0), \tag{3.54}$$
$$\mathbf{p} + c\mathbf{n} = (R\cos(t_0), R\sin(t_0), 0), \tag{3.55}$$
$$\delta|c| = r. \tag{3.56}$$

circles	conditions
0	$p_z < c_1 \vee c_2 < p_z < c_3 \vee c_4 < p_z$
1 singular	$p_z \in \{c_1, c_4\} \vee p_z \in \{c_2, c_3\} \wedge c_2 < c_3$
2 regular	$c_1 < p_z < \min\{c_2, c_3\} \vee \max\{c_2, c_3\} < p_z < c_4$
2 singular	$p_z = c_2 = c_3 = 0$
2 regular + 1 singular	$p_z \in \{c_2, c_3\} \wedge c_2 > c_3$
4 regular	$c_3 < p_z < c_2$

$$c_1 = \frac{-r\sqrt{1+\delta^2} - R}{\delta}$$
$$c_2 = \frac{r\sqrt{1+\delta^2} - R}{\delta}$$
$$c_3 = \frac{-r\sqrt{1+\delta^2} + R}{\delta}$$
$$c_4 = \frac{r\sqrt{1+\delta^2} + R}{\delta}$$

Table 3.1: Cases of profile circles in TKI

From (3.54)-(3.55) we get directly

$$n_z = 0, \tag{3.57}$$
$$p_z = 0. \tag{3.58}$$

From (3.55) we gain furthermore

$$\|\mathbf{p}\|^2 = R^2 + c^2.$$

By building the dot-product with $\mathbf{n}$ we get

$$\langle \mathbf{p}, \mathbf{n} \rangle + c = 0.$$

Together we have

$$\|\mathbf{p}\|^2 = R^2 + \langle \mathbf{p}, \mathbf{n} \rangle^2 \tag{3.59}$$

and (3.56) yields

$$\delta |\langle \mathbf{p}, \mathbf{n} \rangle| = r. \tag{3.60}$$

If conditions (3.57)-(3.60) are fulfilled, one and only one cross-sectional circle is embedded in the intersection curve with the following representation:

$$C_r(\mathbf{p} - \langle \mathbf{p}, \mathbf{n} \rangle \mathbf{n}, \mathbf{n}).$$

3.5.3 Villarceau Circles in TKI

We compare the set Γ_K with the set of Villarceau circles of the torus

$$\Gamma_T^{v\pm} = \Big\{ C_R((\mp r\sin(u), \pm r\cos(u), 0), \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2})) \Big| u \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma_K \cap \Gamma_T^{v\pm}$ the following conditions must hold:

$$\mathbf{o} = \mathbf{n} \times (r\cos(u), r\sin(u), \sqrt{R^2 - r^2}), \tag{3.61}$$
$$\mathbf{p} + c\mathbf{n} = (\mp r\sin(u), \pm r\cos(u), 0), \tag{3.62}$$
$$\delta |c| = R. \tag{3.63}$$

From (3.61) we can follow directly

$$n_z^2 = \frac{R^2 - r^2}{R^2}. \tag{3.64}$$

The dot-product of (3.62) with $\mathbf{n}$ yields

$$\langle \mathbf{p}, \mathbf{n} \rangle + c = 0. \tag{3.65}$$

If we do the same with the cross-product we get

$$\|\mathbf{p} \times \mathbf{n}\|^2 = r^2. \tag{3.66}$$

At last we get from the third component of (3.62) and with (3.65)

$$\langle \mathbf{p}, \mathbf{n} \rangle^2 = \frac{R^2 p_z^2}{R^2 - r^2}. \tag{3.67}$$

Equation (3.63) yields

$$\delta |\langle \mathbf{p}, \mathbf{n} \rangle| = R. \tag{3.68}$$

If the four conditions (3.64) and (3.66)-(3.68) are fulfilled, a Villarceau circle with the representation

$$C_R(\mathbf{p} - \langle \mathbf{p}, \mathbf{n} \rangle \mathbf{n}, \mathbf{n})$$

is embedded in the intersection curve.

3.6 Circle Detection in TTI

Let $T_1 = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and $T_2 = T_{\delta,\Delta}(\mathbf{p}, \mathbf{n})$ be two tori. A circle embedded in the intersection curve of T_1 and T_2 must lie in one of the intersections $\Gamma^i_{T_1} \cap \Gamma^j_{T_2},\ i, j \in \{p, c, v\pm\}$. We do not need to discuss all possible combinations of circle sets, since the case $\Gamma^c_{T_1} \cap \Gamma^p_{T_2}$ is essentially the same as the case $\Gamma^p_{T_1} \cap \Gamma^c_{T_2}$ after an appropriate transformation and swapping the roles of T_1 and T_2. In practice, however, all cases have to be considered. So if we had a routine

$$\texttt{CheckProfileAndCrossSectionalCircles}\big(T_{r,R}(\mathbf{o}, \mathbf{e}_3),\ T_{\delta,\Delta}(\mathbf{p}, \mathbf{n})\big),$$

we would call the routine again with

$$\texttt{CheckProfileAndCrossSectionalCircles}\big(T_{\delta,\Delta}(\mathbf{o}, \mathbf{e}_3),\ T_{r,R}(-M_{\mathbf{n}}^{-1}\mathbf{p}, M_{\mathbf{n}}^{-1}\mathbf{e}_3)\big),$$

where $M_{\mathbf{n}}$ is given as in (2.1). Due to the same reason we omit the cases $\Gamma^{v\pm}_{T_1} \cap \Gamma^p_{T_2}$ and $\Gamma^{v\pm}_{T_1} \cap \Gamma^c_{T_2}$.

Note that we can represent circle sets of an arbitrary oriented torus $T = T_{\delta,\Delta}(\mathbf{p}, \mathbf{n})$ by applying an appropriate transformation on the circle set of a similar torus $T^\star = T_{\delta,\Delta}(\mathbf{o}, \mathbf{e}_3)$ in canonical position, i.e.

$$\Gamma^i_T = \big\{ C_\beta(\mathbf{p} + M_{\mathbf{n}}\mathbf{q}, M_{\mathbf{n}}\mathbf{n_q}) \big| C_\beta(\mathbf{q}, \mathbf{n_q}) \in \Gamma^i_{T^\star} \big\},\ i \in \{p, c, v\pm\}. \tag{3.69}$$

3.6.1 Profile and Profile Circles in TTI

We compare the set of profile circles of the torus T_1

$$\Gamma^p_{T_1} = \Big\{ C_{R + r\cos(s_1)}(r\sin(s_1)\mathbf{e}_3, \mathbf{e}_3) \big| s_1 \in \mathbb{K} \Big\}$$

with the set of profile circles of T_2

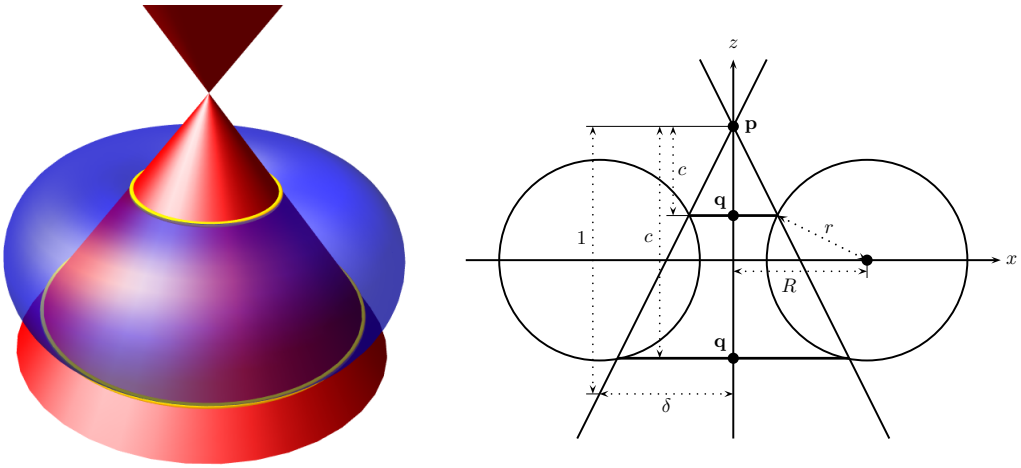
$$\Gamma^p_{T_2} = \Big\{ C_{\Delta + \delta\cos(s_2)}(\mathbf{p} + \delta\sin(s_2)\mathbf{n}, \mathbf{n}) \big| s_2 \in \mathbb{K} \Big\}.$$

For a circle $C \in \Gamma^p_{T_1} \cap \Gamma^p_{T_2}$ the following conditions must hold:

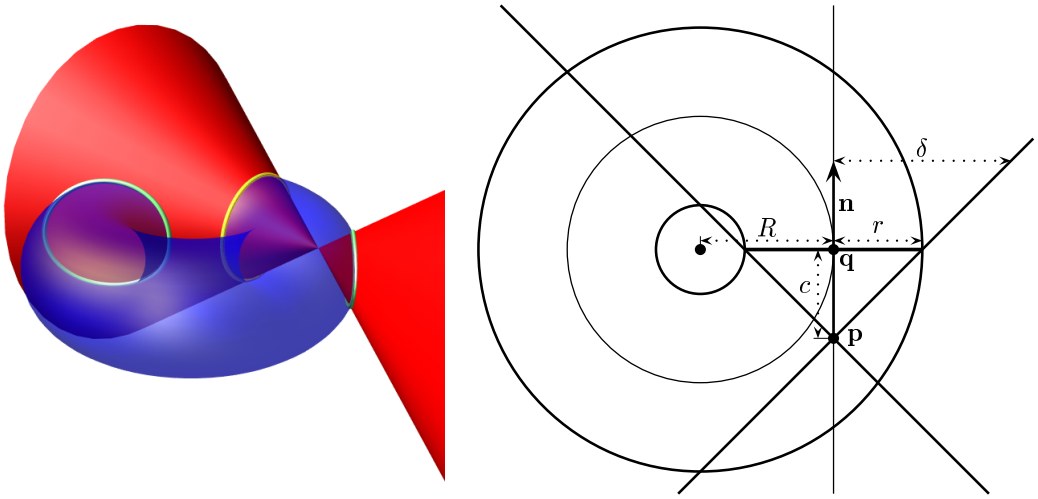
$$n_x = n_y = 0, \tag{3.70}$$
$$p_x = p_y = 0, \tag{3.71}$$
$$p_z + \delta\sin(s_2) = r\sin(s_1), \tag{3.72}$$
$$\Delta + \delta\cos(s_2) = R + r\cos(s_1). \tag{3.73}$$

From (3.72)-(3.73) we get after eliminating s_2

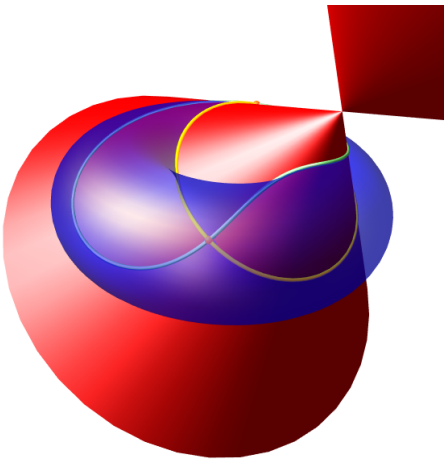
$$[p_z^2 - \delta^2 + (r - R + \Delta)^2]s_1^2 + [-4p_z r]s_1 + [p_z^2 - \delta^2 + (r + R - \Delta)^2] = 0.$$



(a) Profile circles



(b) Cross-sectional circles



(c) Villarceau circles

Figure 3.6: Circles in torus-cone intersection

This can be solved for s_1 and leads to

$$s_1 = \frac{2p_z r \pm \sqrt{d}}{p_z^2 - \delta^2 + (r - R + \Delta)^2}$$

with

$$d = -[(r-\delta)^2 - (R-\Delta)^2 - p_z^2][(r+\delta)^2 - (R-\Delta)^2 - p_z^2].$$

The two solutions coincide and the intersection curve consists of one singular circle, if and only if the discriminant d vanishes, i.e.

$$|r \pm \delta| = \sqrt{(R-\Delta)^2 + p_z^2},$$

see Figure 3.7(a).

If conditions (3.70)-(3.71) are fulfilled and the discriminant $d > 0$, we have two profile circles of the form

$$C_\beta((0,0,\gamma), \mathbf{e}_3)$$

with

$$\begin{aligned}
\beta &= \frac{R+\Delta}{2} - \frac{(R-\Delta)(r^2-\delta^2) \pm p_z\sqrt{d}}{2(p_z^2 + (R-\Delta)^2)}, \\
\gamma &= \frac{p_z}{2} + \frac{p_z(r^2-\delta^2) \mp (R-\Delta)\sqrt{d}}{2(p_z^2 + (R-\Delta)^2)}, \\
d &= -[(r-\delta)^2 - (R-\Delta)^2 - p_z^2][(r+\delta)^2 - (R-\Delta)^2 - p_z^2].
\end{aligned}$$

3.6.2 Profile and Cross-sectional Circles in TTI

We compare the set of profile circles of the torus T_1

$$\Gamma^p_{T_1} = \left\{ C_{R+r\cos(s_1)}(r\sin(s_1)\mathbf{e}_3, \mathbf{e}_3) \middle| s_1 \in \mathbb{K} \right\}$$

with the set of cross-sectional circles of T_2 (see 3.69)

$$\Gamma^c_{T_2} = \left\{ C_\delta(\mathbf{p} + \Delta M_\mathbf{n}\mathbf{m}, M_\mathbf{n}(\mathbf{e}_3 \times \mathbf{m})) \middle| \mathbf{m} = (\cos(t_0), \sin(t_0), 0), t_0 \in \mathbb{K} \right\}.$$

For a circle $C \in \Gamma^p_{T_1} \cap \Gamma^c_{T_2}$ the following conditions must hold:

$$\delta = R + r\cos(s_0), \tag{3.74}$$

$$M_\mathbf{n}(\mathbf{e}_3 \times \mathbf{m}) = \pm\mathbf{e}_3, \tag{3.75}$$

$$\mathbf{p} + \Delta M_\mathbf{n}\mathbf{m} = r\sin(s_0)\mathbf{e}_3. \tag{3.76}$$

Since $\det M_\mathbf{n} = 1$, see (2.1), we can write the left side of (3.75)

$$M_\mathbf{n}(\mathbf{e}_3 \times \mathbf{m}) = (M_\mathbf{n}\mathbf{e}_3) \times (M_\mathbf{n}\mathbf{m}) = \mathbf{n} \times (M_\mathbf{n}\mathbf{m}).$$

Building the dot product with $\mathbf{n}$ yields

$$\langle \mathbf{n}, \mathbf{e}_3 \rangle = 0. \tag{3.77}$$

Furthermore the cross product of $\mathbf{e}_3$ with (3.75) yields

$$\begin{aligned}
0 = \mathbf{e}_3 \times (\pm\mathbf{e}_3) &= \mathbf{e}_3 \times (\mathbf{n} \times (M_\mathbf{n}\mathbf{m})) \\
&= \langle \mathbf{e}_3, M_\mathbf{n}\mathbf{m} \rangle \mathbf{n} - \langle \mathbf{e}_3, \mathbf{n} \rangle M_\mathbf{n}\mathbf{m},
\end{aligned}$$

from which we can follow

$$\langle \mathbf{e}_3, M_\mathbf{n}\mathbf{m} \rangle = 0. \tag{3.78}$$

The cross product of $\mathbf{e}_3$ with (3.76) leads then to

$$p_z = r\sin(s_0).$$

With (3.74) we gain

$$(\delta - R)^2 + p_z^2 = r^2. \tag{3.79}$$

Equations (3.76) and (3.78) yield

$$\begin{aligned}\|\mathbf{p}\|^2 &= \|r\sin(s_0)\mathbf{e}_3 - \Delta M_{\mathbf{n}}\mathbf{m}\|^2 \\ &= r^2\sin^2(s_0) + \Delta^2.\end{aligned}$$

After subtraction of p_z^2 we get

$$p_x^2 + p_y^2 = \Delta^2. \tag{3.80}$$

Finally we build the dot product of $\mathbf{n}$ with (3.76) and gain

$$\begin{aligned}0 = \langle \mathbf{n}, r\sin(s_0)\mathbf{e}_3\rangle &= \langle \mathbf{n}, \mathbf{p} + \Delta M_{\mathbf{n}}\mathbf{m}\rangle \\ &= \langle \mathbf{n}, \mathbf{p}\rangle + \Delta\langle M_{\mathbf{n}}^{-1}\mathbf{n}, \mathbf{m}\rangle \\ &= \langle \mathbf{n}, \mathbf{p}\rangle + \Delta\langle \mathbf{e}_3, \mathbf{m}\rangle.\end{aligned}$$

Since $\langle \mathbf{e}_3, \mathbf{m}\rangle = 0$ we get

$$\langle \mathbf{n}, \mathbf{p}\rangle = 0. \tag{3.81}$$

If conditions (3.77) and (3.79)-(3.81) hold, the intersection curve contains a profile circle of T_1 which is a cross-sectional circle of T_2, represented by

$$C_\delta((0,0,p_z),\mathbf{e}_3).$$

3.6.3 Profile and Villarceau Circles in TTI

We compare the set of profile circles of the torus T_1

$$\Gamma_{T_1}^p = \Big\{ C_{R+r\cos(s_1)}(r\sin(s_1)\mathbf{e}_3, \mathbf{e}_3) \big| s_1 \in \mathbb{K} \Big\}$$

with the set of Villarceau circles of T_2

$$\Gamma_{T_2}^{v\pm} = \Big\{ C_\Delta(\mathbf{p} \pm M_{\mathbf{n}}(-\delta\sin(u), \delta\cos(u), 0), \frac{1}{\Delta} M_{\mathbf{n}}(\delta\cos(u), \delta\sin(u), \sqrt{\Delta^2 - \delta^2})) \big| u \in \mathbb{K} \Big\}.$$

Set $\mathbf{m}_u = (-\sin(u), \cos(u), 0)$ and $\mathbf{q} = \mathbf{p} \pm \delta M_{\mathbf{n}}\mathbf{m}_u$, then a circle in $\Gamma_{T_2}^{v\pm}$ can be expressed by

$$C_\Delta(\mathbf{q}, \mathbf{n}_\mathbf{q})$$

with

$$\begin{aligned}\mathbf{n}_\mathbf{q} &= \frac{1}{\Delta} M_{\mathbf{n}}\big[\delta(\mathbf{m}_u \times \mathbf{e}_3) + \sqrt{\Delta^2 - \delta^2}\mathbf{e}_3\big] \\ &= \frac{1}{\Delta}\big[\delta(M_{\mathbf{n}}\mathbf{m}_u) \times \mathbf{n} + \sqrt{\Delta^2 - \delta^2}\mathbf{n}\big] \\ &= \frac{1}{\Delta}\big[\pm\delta(\mathbf{q} - \mathbf{p}) \times \mathbf{n} + \sqrt{\Delta^2 - \delta^2}\mathbf{n}\big].\end{aligned}$$

We follow directly

$$\pm\langle \mathbf{q} - \mathbf{p}, \mathbf{n}\rangle = \delta\langle M_{\mathbf{n}}\mathbf{m}_u, \mathbf{n}\rangle = \delta\langle \mathbf{m}_u, \mathbf{e}_3\rangle = 0 \tag{3.82}$$

and

$$\|\mathbf{q} - \mathbf{p}\|^2 = \delta^2\|M_{\mathbf{n}}\mathbf{m}_u\|^2 = \delta^2. \tag{3.83}$$

For a circle $C \in \Gamma^p_{T_1} \cap \Gamma^{v\pm}_{T_2}$ the following conditions must hold:

$$\Delta = R + r\cos(s_0), \tag{3.84}$$
$$\mathbf{o} = \mathbf{e}_3 \times \mathbf{n_q}, \tag{3.85}$$
$$\mathbf{q} = r\sin(s_0)\mathbf{e}_3. \tag{3.86}$$

Building the dot product of $\mathbf{n}$ with (3.85) yields

$$\begin{aligned} 0 &= \langle \mathbf{n}, \pm\big[\langle \mathbf{e}_3, \mathbf{n}\rangle(\mathbf{q}-\mathbf{p}) - \langle \mathbf{e}_3, \mathbf{q}-\mathbf{p}\rangle\mathbf{n}\big] + \sqrt{\Delta^2-\delta^2}(\mathbf{e}_3 \times \mathbf{n})\rangle \\ &= \mp(q_z - p_z). \end{aligned}$$

With (3.86) we get

$$\mathbf{q} = (0, 0, p_z)$$

and considering additionally (3.84) yields

$$p_z^2 = r^2 - (\delta - R)^2. \tag{3.87}$$

From (3.82) and (3.83) we can derive two other conditions

$$p_x n_x + p_y n_y = 0 \tag{3.88}$$

and

$$p_x^2 + p_y^2 = \delta^2. \tag{3.89}$$

Consider (3.85) in the form

$$\pm n_z(p_x, p_y, 0) = \sqrt{\Delta^2 - \delta^2}(\mathbf{e}_3 \times \mathbf{n}).$$

With (3.89) we get

$$n_z^2 = \frac{\Delta^2 - \delta^2}{\delta^2}. \tag{3.90}$$

If conditions (3.87)-(3.90) hold, the intersection curve contains a profile circle of T_1 which is a Villarceau circle of T_2, represented by

$$C_\Delta((0, 0, p_z), \mathbf{e}_3).$$

3.6.4 Cross-sectional and Cross-sectional Circles in TTI

We compare the set of cross-sectional circles of the torus T_1

$$\Gamma^c_{T_1} = \Big\{ C_r((R\cos(t_1), R\sin(t_1), 0), (-\sin(t_1), \cos(t_1), 0)) \big| t_1 \in \mathbb{K} \Big\}$$

with the set of cross-sectional circles of T_2 (see (3.69))

$$\Gamma^c_{T_2} = \Big\{ C_\delta(\mathbf{p} + M_\mathbf{n}(\Delta\cos(t_2), \Delta\sin(t_2), 0), M_\mathbf{n}(-\sin(t_2), \cos(t_2), 0)) \big| t_2 \in \mathbb{K} \Big\}.$$

Set $\mathbf{m}_{t_2} = (\cos(t_2), \sin(t_2), 0)$ and $\mathbf{q} = \mathbf{p} + \Delta M_\mathbf{n}\mathbf{m}_{t_2}$, then a circle in $\Gamma^c_{T_2}$ can be expressed by

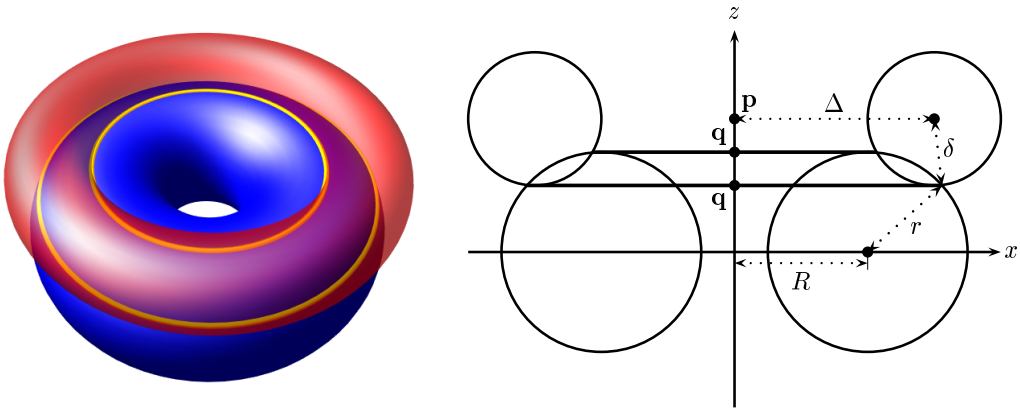
$$C_\delta(\mathbf{q}, \mathbf{n_q})$$

with

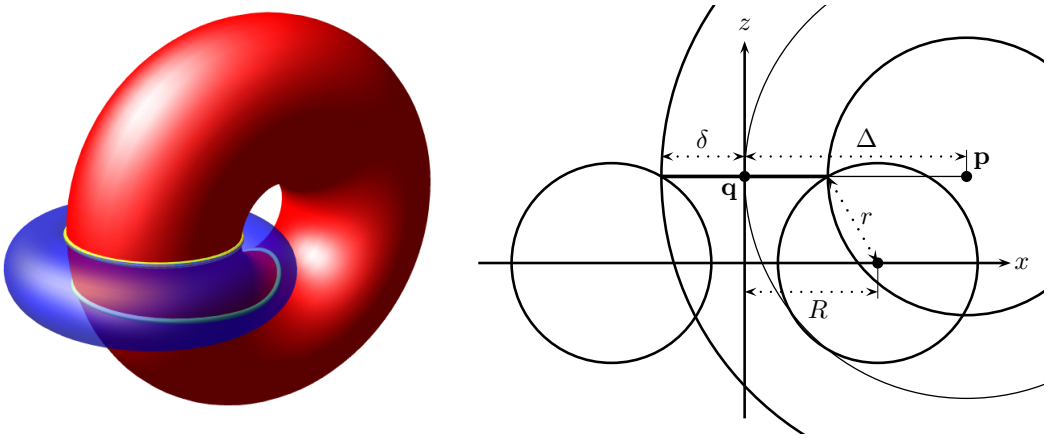
$$\mathbf{n_q} = \frac{1}{\Delta}\big[\mathbf{n} \times (\mathbf{q} - \mathbf{p})\big].$$

It is obvious that

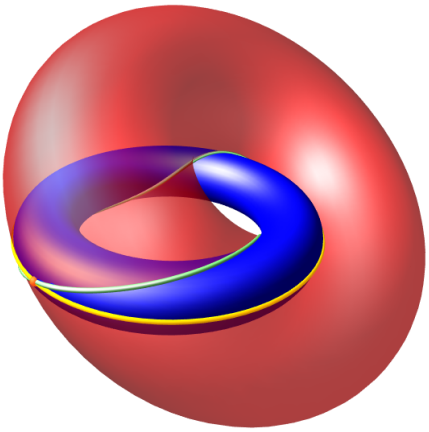
$$\langle \mathbf{q} - \mathbf{p}, \mathbf{n}\rangle = 0, \tag{3.91}$$
$$\|\mathbf{q} - \mathbf{p}\|^2 = \Delta^2. \tag{3.92}$$



(a) Profile circles



(b) Profile and cross-sectional circles



(c) Profile and Villarceau circles

Figure 3.7: Profile circles in torus-torus intersection

For a circle $C \in \Gamma^c_{T_1} \cap \Gamma^c_{T_2}$ the following conditions must hold:

$$\delta = r, \tag{3.93}$$
$$\mathbf{q} = R(\cos(t_1), \sin(t_1), 0), \tag{3.94}$$
$$\mathbf{n_q} = \epsilon \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q}) \tag{3.95}$$

with either $\epsilon = 1$ or $\epsilon = -1$. Building the cross product of $\mathbf{n}$ with (3.95) yields

$$\mathbf{p} = \frac{1}{R}\big[(R - \epsilon\Delta n_z)\mathbf{q} + \epsilon\Delta\langle \mathbf{n}, \mathbf{p}\rangle \mathbf{e}_3\big], \tag{3.96}$$

from which we can follow

$$p_x^2 + p_y^2 = \alpha^2, \tag{3.97}$$
$$\alpha p_z = \epsilon\Delta(n_x p_x + n_y p_y) \tag{3.98}$$

with $\alpha = R - \epsilon\Delta n_z$. Furthermore the cross product of $\mathbf{e}_3$ with (3.95) yields

$$\mathbf{q} = \epsilon \frac{R}{\Delta}\big[n_z(\mathbf{q} - \mathbf{p}) + p_z \mathbf{n}\big].$$

Squaring both sides yields

$$\|\mathbf{q}\|^2 = \frac{R^2}{\Delta^2}\big[n_z^2\|\mathbf{q} - \mathbf{p}\|^2 + 2n_z p_z\langle \mathbf{q} - \mathbf{p}, \mathbf{n}\rangle + p_z^2\big]$$

and with (3.91), (3.92) and (3.94) we get

$$p_z^2 = \Delta^2\|\mathbf{e}_3 \times \mathbf{n}\|^2. \tag{3.99}$$

At last we build the dot product of $\mathbf{e}_3$ with (3.95) and get

$$0 = \langle \mathbf{n} \times (\mathbf{q} - \mathbf{p}), \mathbf{e}_3\rangle = \langle \mathbf{e}_3 \times \mathbf{n}, \mathbf{q}\rangle - \langle \mathbf{e}_3 \times \mathbf{n}, \mathbf{p}\rangle.$$

Since $\langle \mathbf{e}_3 \times \mathbf{n}, \mathbf{q}\rangle = 0$, which follows from the dot product of $\mathbf{n}$ with equation (3.95), we gain the condition

$$n_x p_y - n_y p_x = 0. \tag{3.100}$$

Note that with (3.97) and (3.99) it holds

$$\big[\|\mathbf{p}\|^2 + R^2 - \Delta^2\big]^2 - 4R^2(p_x^2 + p_y^2) = 0, \tag{3.101}$$

i.e. $\mathbf{p}$ is embedded in a torus $T_{\Delta,R}(\mathbf{o}, \mathbf{e}_3)$. To avoid the consideration of singularities of spindle tori, we assume $\Delta \leq R$. Otherwise we swap the tori T_1 and T_2. Furthermore we can exclude the case $\mathbf{p} = \mathbf{o}$, since then both tori would coincide. So $\mathbf{p}$ is embedded on a ring torus, and since $p_x^2 + p_y^2 > 0$ this encloses our assumption to $\Delta < R$. Now we can derive a simple representation of the cross-sectional circle

$$C_r(\mathbf{q}, \mathbf{n_q}),$$

embedded in the intersection curve of T_1 and T_2. Therefore the conditions (3.93) and (3.97)-(3.100) have to be fulfilled. From (3.96) we derive

$$\mathbf{q} = \frac{R}{\alpha}(p_x, p_y, 0),$$

where α is determined by (3.97). From (3.95) we finally gain the normal

$$\mathbf{n_q} = \frac{1}{\alpha}(-p_y, p_x, 0).$$

3.6.5 Cross-sectional and Villarceau Circles in TTI

We compare the set of cross-sectional circles of the torus T_1

$$\Gamma^c_{T_1} = \Big\{ C_r((R\cos(t_0), R\sin(t_0), 0), (-\sin(t_0), \cos(t_0), 0)) \big| t_0 \in \mathbb{K} \Big\}$$

with the set of Villarceau circles of T_2

$$\Gamma^{v\pm}_{T_2} = \Big\{ C_\Delta(\mathbf{p} \pm M_\mathbf{n}(-\delta\sin(u), \delta\cos(u), 0), \frac{1}{\Delta} M_\mathbf{n}(\delta\cos(u), \delta\sin(u), \sqrt{\Delta^2 - \delta^2})) \big| u \in \mathbb{K} \Big\}.$$

Set $\mathbf{m}_u = (-\sin(u), \cos(u), 0)$ and $\mathbf{q} = \mathbf{p} + \epsilon_1 \delta M_\mathbf{n} \mathbf{m}_u$ with $\epsilon_1 = 1$ for circles in $\Gamma^{v+}_{T_2}$ and $\epsilon_1 = -1$ for circles in $\Gamma^{v-}_{T_2}$. A circle in $\Gamma^{v\pm}_{T_2}$ can then be expressed by

$$C_\Delta(\mathbf{q}, \mathbf{n}_\mathbf{q})$$

with

$$\mathbf{n}_\mathbf{q} = \frac{1}{\Delta}\big[\epsilon_1(\mathbf{q} - \mathbf{p}) \times \mathbf{n} + \sqrt{\Delta^2 - \delta^2}\mathbf{n}\big],$$

see Section 3.6.3. It is obvious that

$$\langle \mathbf{q} - \mathbf{p}, \mathbf{n} \rangle = 0, \tag{3.102}$$

$$\|\mathbf{q} - \mathbf{p}\|^2 = \delta^2. \tag{3.103}$$

For a circle $C \in \Gamma^c_{T_1} \cap \Gamma^{v\pm}_{T_2}$ the following conditions must hold:

$$\Delta = r, \tag{3.104}$$

$$\mathbf{q} = R(\cos(t_0), \sin(t_0), 0), \tag{3.105}$$

$$\mathbf{n}_\mathbf{q} = \epsilon_2 \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q}) \tag{3.106}$$

with either $\epsilon_2 = 1$ or $\epsilon_2 = -1$. Building the cross product of $\mathbf{n}$ with (3.106) yields

$$\mathbf{p} = \frac{1}{R}\big[(R + \epsilon_1\epsilon_2\Delta n_z)\mathbf{q} - \epsilon_1\epsilon_2\Delta\langle \mathbf{n}, \mathbf{p}\rangle \mathbf{e}_3\big], \tag{3.107}$$

from which we can follow

$$p_x^2 + p_y^2 = \alpha^2, \tag{3.108}$$

$$n_x p_x + n_y p_y = \epsilon_1\epsilon_2 \frac{\alpha}{\Delta} p_z \tag{3.109}$$

with $\alpha = R + \epsilon_1\epsilon_2\Delta n_z$. Since we just consider ring tori and the major radius Δ of T_2 is equal to the minor radius r of T_1, we note that $\alpha > 0$. Keeping this in mind we gain a representation for $\mathbf{q}$

$$\mathbf{q} = \frac{R}{\alpha}(p_x, p_y, 0),$$

and with (3.103), (3.105) and (3.108) we get

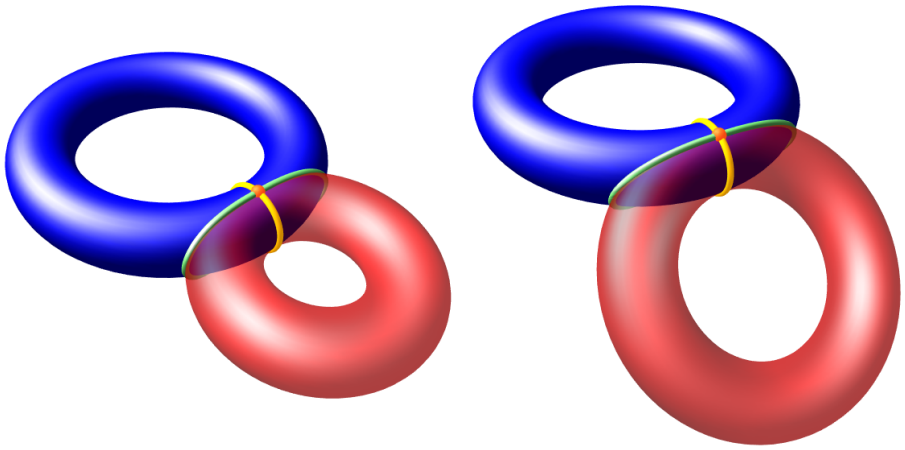
$$p_z^2 = \delta^2 - \Delta^2 n_z^2. \tag{3.110}$$

At last we build the dot product of $\mathbf{e}_3$ with (3.106) and get

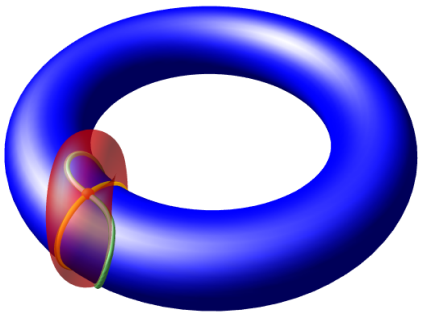
$$\begin{aligned} 0 &= \langle \epsilon_1(\mathbf{q} - \mathbf{p}) \times \mathbf{n} + \sqrt{\Delta^2 - \delta^2}\mathbf{n}, \mathbf{e}_3 \rangle \\ &= \epsilon_1\langle \mathbf{e}_3 \times \mathbf{q}, \mathbf{n}\rangle + \epsilon_1\langle \mathbf{p} \times \mathbf{e}_3, \mathbf{n}\rangle + \sqrt{\Delta^2 - \delta^2} n_z. \end{aligned}$$

With (3.106) we gain the condition

$$n_x p_y - n_y p_x = -\epsilon_2 \frac{\alpha}{\Delta}\sqrt{\Delta^2 - \delta^2}. \tag{3.111}$$



(a) Cross-sectional circles



(b) Cross-sectional and Villarceau circles

Figure 3.8: Cross-sectional circles in torus-torus intersection

Note that with (3.108) and (3.110) it holds

$$\left[\|\mathbf{p}\|^2 + R^2 - \delta^2\right]^2 - 4R^2(p_x^2 + p_y^2) = 0,$$

i.e. $\mathbf{p}$ is embedded on a torus $T_{\delta,R}(\mathbf{o}, \mathbf{e}_3)$.

If conditions (3.104) and (3.108)-(3.111) hold for some suitable $\epsilon_1, \epsilon_2 \in \{-1, +1\}$, a cross-sectional circle of T_1, which is a Villarceau circle of T_2, is embedded in the intersection curve. We represent this circle by

$$C_r(\mathbf{q}, \mathbf{n_q})$$

with

$$\begin{aligned}\mathbf{q} &= \frac{R}{\alpha}(p_x, p_y, 0),\\ \mathbf{n_q} &= \frac{1}{\alpha}(-p_y, p_x, 0),\end{aligned}$$

where α is determined by (3.108).

3.6.6 Villarceau and Villarceau Circles in TTI

We compare the set of Villarceau circles of the torus T_1

$$\Gamma_{T_1}^{v\pm} = \Big\{ C_R(\pm(-r\sin(u), r\cos(u), 0), \frac{1}{R}(r\cos(u), r\sin(u), \sqrt{R^2 - r^2})) \big| u \in \mathbb{K} \Big\}$$

with the set of Villarceau circles of T_2

$$\Gamma_{T_2}^{v\pm} = \Big\{ C_\Delta(\mathbf{p} \pm M_\mathbf{n}(-\delta\sin(v), \delta\cos(v), 0), \frac{1}{\Delta} M_\mathbf{n}(\delta\cos(v), \delta\sin(v), \sqrt{\Delta^2 - \delta^2})) \big| v \in \mathbb{K} \Big\}.$$

Like in the previous sections we substitute $\mathbf{m}_u = (-\sin(u), \cos(u), 0)$, $\mathbf{m}_v = (-\sin(v), \cos(v), 0)$, $\mathbf{q}_1 = \epsilon_1 r \mathbf{m}_u$ and $\mathbf{q}_2 = \mathbf{p} + \epsilon_2 \delta M_\mathbf{n} \mathbf{m}_v$ with $\epsilon_1, \epsilon_2 \in \{-1, +1\}$. Note that the following equations hold:

$$\|\mathbf{q}_1\| = r, \tag{3.112}$$
$$\|\mathbf{q}_2 - \mathbf{p}\| = \delta, \tag{3.113}$$
$$\langle \mathbf{q}_1, \mathbf{e}_3 \rangle = 0, \tag{3.114}$$
$$\langle \mathbf{q}_2 - \mathbf{p}, \mathbf{n} \rangle = 0. \tag{3.115}$$

For a circle $C \in \Gamma_{T_1}^{v\pm} \cap \Gamma_{T_2}^{v\pm}$ the following conditions must hold:

$$\Delta = R, \tag{3.116}$$
$$\mathbf{q}_1 = \mathbf{q}_2 =: \mathbf{q}, \tag{3.117}$$
$$\epsilon_1 \mathbf{q} \times \mathbf{e}_3 + \sqrt{R^2 - r^2}\mathbf{e}_3 = \epsilon_3 \big[\epsilon_2 (\mathbf{q} - \mathbf{p}) \times \mathbf{n} + \sqrt{R^2 - \delta^2}\mathbf{n} \big], \tag{3.118}$$

with $\epsilon_3 \in \{-1, +1\}$.

We build the dot product of $\mathbf{n}$ with (3.118) and get

$$\langle \mathbf{q} \times \mathbf{e}_3, \mathbf{n} \rangle = \epsilon_1 \big[\epsilon_3 \sqrt{R^2 - \delta^2} - n_z \sqrt{R^2 - r^2} \big]. \tag{3.119}$$

Applying the cross product to $\mathbf{n}$ and (3.118) yields

$$\mathbf{p} = \epsilon_1 \epsilon_2 \epsilon_3 \big[\alpha \mathbf{q} + \langle \mathbf{p}, \mathbf{n} \rangle \mathbf{e}_3 + \epsilon_1 \sqrt{R^2 - r^2} (\mathbf{e}_3 \times \mathbf{n}) \big] \tag{3.120}$$

with $\alpha = \epsilon_1 \epsilon_2 \epsilon_3 - n_z$. From this we follow

$$p_x^2 + p_y^2 = R^2 \alpha^2 + 2\epsilon_3 \alpha \sqrt{(R^2 - r^2)(R^2 - \delta^2)}, \tag{3.121}$$
$$\langle \mathbf{p}, \mathbf{n} \rangle = \epsilon_1 \epsilon_2 \epsilon_3 p_z. \tag{3.122}$$

Building the dot product of $\mathbf{e}_3$ with (3.118) yields

$$\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle = \alpha \big[\epsilon_1 \sqrt{R^2 - r^2} + \epsilon_2 \sqrt{R^2 - \delta^2} \big]. \tag{3.123}$$

At last we build the cross product of $\mathbf{e}_3$ with (3.118) and get

$$n_z \mathbf{p} = -\alpha \mathbf{q} + p_z \mathbf{n} + \epsilon_2 \sqrt{R^2 - \delta^2} (\mathbf{e}_3 \times \mathbf{n}).$$

Squaring yields

$$n_z^2 (p_x^2 + p_y^2) = \alpha^2 (r^2 + \delta^2 - R^2 - p_z^2) + 2\epsilon_1 \epsilon_2 n_z \alpha \sqrt{(R^2 - r^2)(R^2 - \delta^2)},$$

and with (3.121) and for $\alpha \neq 0$ (see below) we gain

$$p_z^2 = r^2 + \delta^2 - R^2(1 + n_z^2) + 2\epsilon_3 n_z \sqrt{(R^2 - r^2)(R^2 - \delta^2)}. \tag{3.124}$$

Assume $\alpha = 0$, i.e. $n_z = \epsilon_1\epsilon_2\epsilon_3$. From (3.119) we follow that $n_z = \epsilon_3$, $\delta = r$ and additionally $\epsilon_1 = \epsilon_2$. From (3.115) we follow $p_z = 0$ and with (3.121) we gain $\mathbf{p} = \mathbf{o}$, what results in identical tori. Thus this case can be neglected.

If conditions (3.116) and (3.121)-(3.124) hold, we find up to four Villarceau circles in the intersection curve. Since $\alpha \neq 0$ we can provide a representation for $\mathbf{q}$ and $\mathbf{n_q}$, resulting from (3.120):

$$\mathbf{q} = \frac{\epsilon_1}{\alpha}\begin{pmatrix}\epsilon_2\epsilon_3 p_x + n_y\sqrt{R^2-r^2}\\ \epsilon_2\epsilon_3 p_y - n_x\sqrt{R^2-r^2}\\ 0\end{pmatrix}, \quad \mathbf{n_q} = \frac{1}{\alpha R}\begin{pmatrix}\epsilon_2\epsilon_3 p_y - n_x\sqrt{R^2-r^2}\\ -\epsilon_2\epsilon_3 p_x - n_y\sqrt{R^2-r^2}\\ \alpha\sqrt{R^2-r^2}\end{pmatrix}. \tag{3.125}$$

It is left to determine ϵ_i for $i = 1\ldots 3$. From (3.122) we either get

$$\epsilon_1\epsilon_2\epsilon_3 = \frac{\langle \mathbf{p}, \mathbf{n}\rangle}{p_z}, \tag{3.126}$$

or both p_z and $\langle \mathbf{p}, \mathbf{n}\rangle$ are zero.

Let us first consider the latter case. With (3.112)-(3.115) we can follow

$$\langle \mathbf{q}-\mathbf{p}, \mathbf{n}\rangle = \langle \mathbf{q}-\mathbf{p}, \mathbf{e}_3\rangle = \langle \mathbf{q}, \mathbf{n}\rangle = \langle \mathbf{q}, \mathbf{e}_3\rangle = 0.$$

This means, assuming $\mathbf{e}_3 \times \mathbf{n} \neq \mathbf{o}$, that $(\mathbf{q}-\mathbf{p})$, $\mathbf{q}$ and $\mathbf{e}_3 \times \mathbf{n}$ are collinear. Thus we can write

$$\mathbf{q}-\mathbf{p} = \epsilon_4\delta\frac{\mathbf{e}_3\times\mathbf{n}}{\|\mathbf{e}_3\times\mathbf{n}\|}, \quad \mathbf{q} = \epsilon_5 r\frac{\mathbf{e}_3\times\mathbf{n}}{\|\mathbf{e}_3\times\mathbf{n}\|}$$

with $\epsilon_4, \epsilon_5 \in \{-1, +1\}$. This implies

$$\langle \mathbf{p}, \mathbf{e}_3\times\mathbf{n}\rangle = (\epsilon_5 r - \epsilon_4\delta)\|\mathbf{e}_3\times\mathbf{n}\|. \tag{3.127}$$

If $\mathbf{p} = \mathbf{o}$, we follow $r = \delta$ and $\epsilon_4 = \epsilon_5$. Equation (3.123) provides $\epsilon_1 = -\epsilon_2$ and from (3.121) we get

$$n_z R^2 = \epsilon_3(R^2 - 2r^2). \tag{3.128}$$

If $n_z = 0$, (3.128) implies

$$R = r\sqrt{2}. \tag{3.129}$$

This is the only case in detection of conic sections of simple surfaces, which cannot occur by the application of rational input, i.e. if the underlying field $\mathbb{K}$ for the geometrical representation of objects is equal to $\mathbb{Q}$. So this special case of Villarceau circles in a torus-torus intersection can only occur if at least $\mathbb{K} \supset \mathbb{Q}[\sqrt{2}]$. From (3.125) we then get

$$\mathbf{q} = -\epsilon_2\epsilon_3 r(\mathbf{e}_3\times\mathbf{n}), \quad \mathbf{n_q} = \frac{1}{\sqrt{2}}(\mathbf{e}_3 + \epsilon_3\mathbf{n}),$$

where ϵ_2 and ϵ_3 are not determinate. This case results in four different Villarceau circles, see Figure 3.9(a).

Consider the case $n_z \neq 0$. Then we can determine ϵ_3 from (3.128) and $\mathbf{q}$ and $\mathbf{n_q}$ are given by

$$\mathbf{q} = \pm\frac{r}{\|\mathbf{e}_3\times\mathbf{n}\|}(\mathbf{e}_3\times\mathbf{n}), \quad \mathbf{n_q} = \frac{r}{R\|\mathbf{e}_3\times\mathbf{n}\|}(\mathbf{e}_3 + \epsilon_3\mathbf{n}).$$

Here we get two Villarceau circles embedded in the same plane, since ϵ_2 is still free of choice, see Figure 3.9(b).

Turning to the case $\mathbf{p} \neq \mathbf{o}$, we saw in (3.127) that we can express $\mathbf{p}$ by

$$\mathbf{p} = (\epsilon_5 r - \epsilon_4\delta)\frac{\mathbf{e}_3\times\mathbf{n}}{\|\mathbf{e}_3\times\mathbf{n}\|}.$$

Determining ϵ_4 and ϵ_5, $\mathbf{q}$ is given by

$$\mathbf{q} = \frac{r}{r - \epsilon_4\epsilon_5\delta}\mathbf{p}. \tag{3.130}$$

To compute $\mathbf{n_q}$ we distinguish two cases. Let again $n_z = 0$. Equation (3.124) yields then

$$R^2 = r^2 + \delta^2. \tag{3.131}$$

Since we have arbitrary signs, $\mathbf{n_q}$ computes to

$$\mathbf{n_q} = \frac{1}{R}(\delta \mathbf{e}_3 \pm r\mathbf{n}),$$

so this case provides two Villarceau circles with the same centre, see Figure 3.9(c).

Consider the case $n_z \neq 0$. Solving (3.124) for n_z leads to

$$n_z = \frac{1}{R^2}\big[\epsilon_3\sqrt{(R^2 - r^2)(R^2 - \delta^2)} + \epsilon_6 r\delta\big] \tag{3.132}$$

with $\epsilon_6 \in \{-1, +1\}$. Substituting n_z into (3.121) and with $p_x^2 + p_y^2 = (\epsilon_5 r - \epsilon_4\delta)^2$ we get

$$\epsilon_6 = \epsilon_1\epsilon_2\epsilon_3\epsilon_4\epsilon_5.$$

To determine ϵ_6 we transform (3.132) to

$$r^2 + \delta^2 - R^2(1 - n_z^2) = 2\epsilon_6 r\delta n_z. \tag{3.133}$$

Substituting n_z into (3.119) and (3.123) yields

$$\frac{1}{R^2}\epsilon_3(r - \epsilon_4\epsilon_5\delta)\big[\epsilon_1 r\sqrt{R^2 - \delta^2} - \epsilon_2\epsilon_4\epsilon_5\delta\sqrt{R^2 - r^2}\big] = \epsilon_5(r - \epsilon_4\epsilon_5\delta)\sqrt{1 - n_z^2},$$
$$\frac{1}{R^2}\epsilon_3 r\big[\epsilon_1 r\sqrt{R^2 - \delta^2} - \epsilon_2\epsilon_4\epsilon_5\delta\sqrt{R^2 - r^2}\big] = \epsilon_5 r\sqrt{1 - n_z^2}.$$

By subtraction of the equations we follow

$$\beta := \epsilon_1 r\sqrt{R^2 - \delta^2} - \epsilon_2\epsilon_4\epsilon_5\delta\sqrt{R^2 - r^2} > 0 \quad \text{and} \quad \epsilon_3 = \epsilon_5.$$

We now determine ϵ_1 which we need to compute the normal $\mathbf{n_q}$. Since β is always positive, we have either $\epsilon_1 = 1$ if $r > \delta$. Otherwise it must be $\epsilon_2\epsilon_4\epsilon_5 = -1$ and therewith $\epsilon_1 = -\epsilon_5\epsilon_6$. The normal $\mathbf{n_q}$ can then be expressed by

$$\mathbf{n_q} = \frac{1}{R}\big[\epsilon_1\frac{r}{r - \epsilon_4\epsilon_5\delta}(\mathbf{p} \times \mathbf{e}_3) + \sqrt{R^2 - r^2}\mathbf{e}_3\big].$$

See also Figure 3.9(d).

We note that the hitherto described cases have been derived under the assumption that $\mathbf{e}_3 \times \mathbf{n} \neq \mathbf{o}$. So now consider the case $\mathbf{e}_3 \times \mathbf{n} = \mathbf{o}$. Then (3.118) tells us that $\epsilon_3 = n_z$, (3.123) yields $\epsilon_2 = -\epsilon_1$ and $r = \delta$. From (3.122) we get $p_z = 0$ and from (3.121) we get $\|\mathbf{p}\| = 2r$. Since ϵ_1 is still indeterminate, we get two Villarceau circles with the same centre

$$\mathbf{q} = \frac{1}{2}\mathbf{p}$$

and normals

$$\mathbf{n_q} = \frac{1}{R}\big[\pm\frac{1}{2}(\mathbf{p} \times \mathbf{e}_3) + \sqrt{R^2 - r^2}\mathbf{e}_3\big],$$

see Figure 3.9(e).

We go back to (3.126) and consider the case where $\langle \mathbf{p}, \mathbf{n}\rangle \neq 0 \neq p_z$. We already determined the product $\epsilon_1\epsilon_2\epsilon_3$. From (3.123) we determine ϵ_1 and ϵ_2 separately, and therewith we get ϵ_3 as well. First we consider the case $\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n}\rangle = 0$, see Figure 3.9(f). Then $\epsilon_2 = -\epsilon_1$ and $r = \delta$. Since ϵ_1 is still indeterminate, we get two Villarceau circles with centres and orientations given in

(3.125). Otherwise we get one Villarceau circle, see Figure 3.9(g). To determine ϵ_1 and ϵ_2, we first determine the product $\epsilon_1\epsilon_2$ by squaring (3.123) which yields

$$\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle^2 - \alpha^2(2R^2 - r^2 - \delta^2) = 2\alpha^2\epsilon_1\epsilon_2\sqrt{(R^2 - r^2)(R^2 - \delta^2)}.$$

So $\epsilon_1\epsilon_2$ is equal to the sign of the left hand side. Depending on r and δ we can determine ϵ_1 respectively ϵ_2 by

$$\mathtt{sign}(\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle) \cdot \mathtt{sign}(\alpha) = \begin{cases} \epsilon_1, & r < \delta \\ \epsilon_2, & r > \delta \end{cases}$$

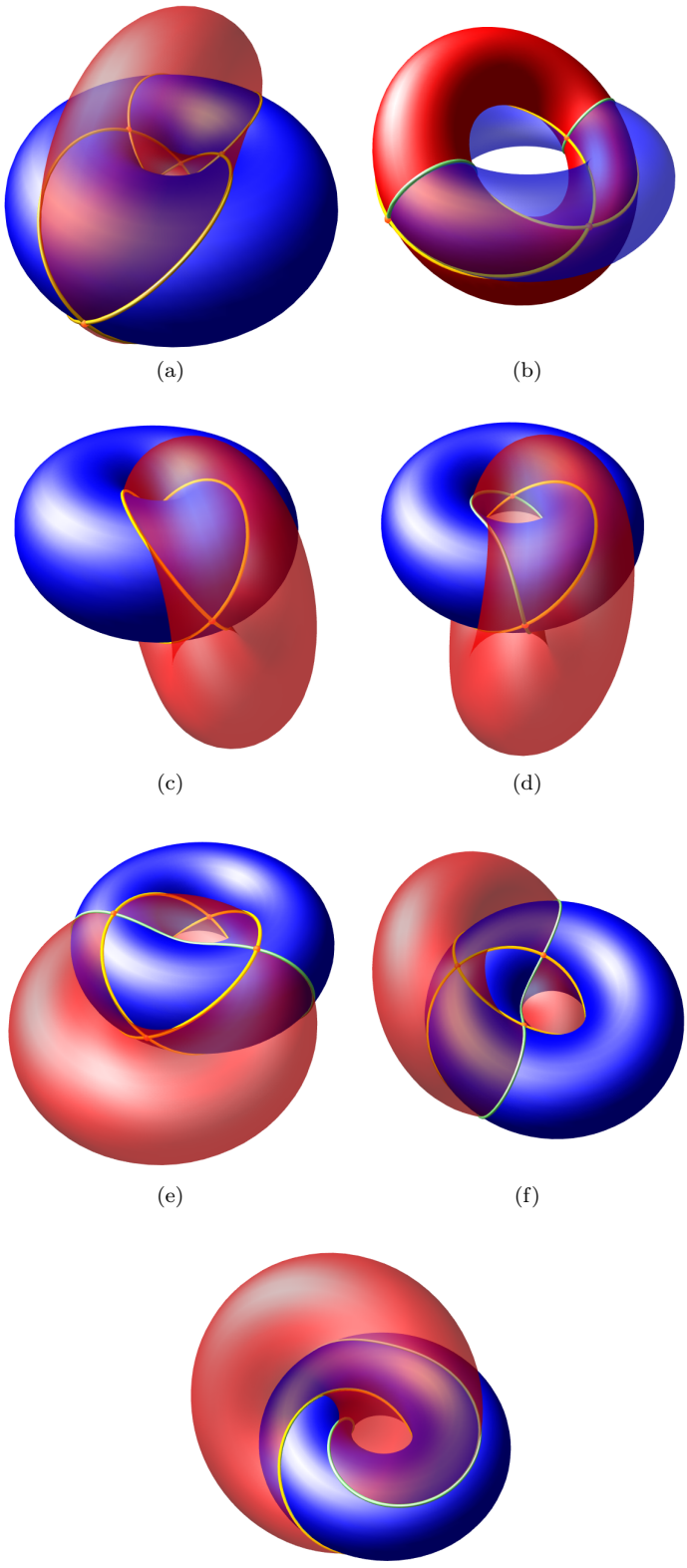
Since detecting Villarceau circles in torus-torus intersections is the most complex case, we provide Algorithm 4 as summary.

Algorithm 4: `ComputeVillarceauCircles`$(T_{r,R}(\mathbf{o}, \mathbf{e}_3), T_{\delta,\Delta}(\mathbf{p}, \mathbf{n}))$

Requires: Two tori $T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and $T_{\delta,\Delta}(\mathbf{p}, \mathbf{n})$
Returns: A set of Villarceau circles $\{C\}$

if $R = \Delta$ **then**
  if $\langle \mathbf{p}, \mathbf{n} \rangle = 0$ and $p_z = 0$ **then**
    if $\mathbf{e}_3 \times \mathbf{n} \neq \mathbf{o}$ **then**
      if $\mathbf{p} = \mathbf{o}$ **then**
        if $r = \delta$ **then**
          if $n_z = 0$ **then**
            if $R^2 = 2r^2$ **then**
              return $\{C_R(\pm_1\mathbf{e}_3 \times \mathbf{n}, \frac{r}{R}(\mathbf{e}_3 \pm_2 \mathbf{n}))\}$ // Figure 3.9(a)
            end if
          else // $n_z \neq 0$
            $\epsilon_3 \leftarrow \frac{R^2 - 2r^2}{n_z R^2}$
            if $|\epsilon_3| = 1$ **then**
              return $\{C_R(\frac{r}{\|\mathbf{e}_3 \times \mathbf{n}\|}(\mathbf{e}_3 \times \mathbf{n}), \pm\frac{r}{R\|\mathbf{e}_3 \times \mathbf{n}\|}(\mathbf{e}_3 + \epsilon_3\mathbf{n}))\}$ // Figure 3.9(b)
            end if
          end if
        end if
      else // $\mathbf{p} \neq \mathbf{o}$
        $\epsilon_{45} \leftarrow \frac{(r^2+\delta^2)\|\mathbf{e}_3 \times \mathbf{n}\|^2 - \langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle^2}{2r\delta\|\mathbf{e}_3 \times \mathbf{n}\|^2}$
        if $|\epsilon_{45}| = 1$ **then**
          $\epsilon_5 \leftarrow \mathtt{sign}(\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle) \cdot \mathtt{sign}(r - \epsilon_{45}\delta)$
          $\mathbf{q} = \frac{r}{r - \epsilon_{45}\delta}\mathbf{p}$
          if $n_z = 0$ **then**
            if $R^2 = r^2 + \delta^2$ **then**
              return $\{C_R(\mathbf{q}, \frac{1}{R}(\delta\mathbf{e}_3 \pm r\mathbf{n}))\}$ // Figure 3.9(c)
            end if
          else // $n_z \neq 0$
            $\epsilon_6 \leftarrow \frac{r^2 + \delta^2 - R^2\|\mathbf{e}_3 \times \mathbf{n}\|^2}{2r\delta n_z}$
            if $|\epsilon_6| = 1$ **then**
              $\epsilon_1 \leftarrow 1$
              if $r < \delta$ **then**
                $\epsilon_1 \leftarrow -\epsilon_5\epsilon_6$
              end if
              return $\{C_R(\mathbf{q}, \frac{1}{R}[\epsilon_1\frac{r}{r - \epsilon_{45}\delta}(\mathbf{p} \times \mathbf{e}_3) + \sqrt{R^2 - r^2}\mathbf{e}_3])\}$ // Figure 3.9(d)
            end if
          end if
        end if
      end if

else // $\mathbf{e}_3 \times \mathbf{n} = \mathbf{o}$
  if $r = \delta$ **and** $p_z = 0$ **and** $\|\mathbf{p}\|^2 = 4r^2$ **then**
    return $\{C_R(\frac{1}{2}\mathbf{p}, \frac{1}{R}[\pm\frac{1}{2}(\mathbf{p} + \mathbf{e}_3) + \sqrt{R^2 + r^2}\mathbf{e}_3])\}$ // Figure 3.9(e)
  end if
end if
else if $\langle \mathbf{p}, \mathbf{n} \rangle \neq 0$ **and** $p_z \neq 0$ **then**
  $\epsilon_{123} \leftarrow \frac{\langle \mathbf{p}, \mathbf{n} \rangle}{p_z}$
  if $|\epsilon_{123}| = 1$ **then**
    $\alpha \leftarrow \epsilon_{123} - n_z$
    $a \leftarrow \langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle^2 - \alpha^2(2R^2 - r^2 - \delta^2)$
    $b \leftarrow p_x^2 + p_y^2 - \alpha^2 R^2$
    $c \leftarrow p_z^2 + (1 + n_z^2)R^2 - r^2 - \delta^2$
    $d \leftarrow 4(R^2 - r^2)(R^2 - \delta^2)$
    if $a^2 = b^2 = \alpha^2 d$ **and** $c^2 = n_z^2 d$ **and** $\mathtt{sign}(b) = \mathtt{sign}(c)$ **then**
      if $\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle = 0$ **then**
        if $r = \delta$ **then**
          $\mathbf{q}^{\pm} = \frac{1}{\epsilon_{123} + n_z}[\epsilon_{123}(\mathbf{e}_3 \times \mathbf{p}) \times \mathbf{e}_3 \mp \sqrt{R^2 - r^2}(\mathbf{e}_3 \times \mathbf{n})]$
          return $\{C_R(\mathbf{q}^{\pm}, \frac{1}{R}[\pm(\mathbf{q}^{\pm} \times \mathbf{e}_3) + \sqrt{R^2 - r^2}\mathbf{e}_3])\}$ // Figure 3.9(f)
        end if
      else // $\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle \neq 0$
        $\epsilon_{12} \leftarrow \mathtt{sign}(a)$
        if $\mathtt{sign}(b) = \epsilon_{12}\epsilon_{123}$ **then**
          if $r < \delta$ **then**
            $\epsilon_1 \leftarrow \mathtt{sign}(\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle) \cdot \mathtt{sign}(\alpha)$
          else // $r > \delta$
            $\epsilon_1 \leftarrow \epsilon_{12} \cdot \mathtt{sign}(\langle \mathbf{p}, \mathbf{e}_3 \times \mathbf{n} \rangle) \cdot \mathtt{sign}(\alpha)$
          end if
          $\mathbf{q} \leftarrow \frac{1}{\alpha}[\epsilon_{123}(\mathbf{e}_3 \times \mathbf{p}) \times \mathbf{e}_3 - \epsilon_1\sqrt{R^2 - r^2}(\mathbf{e}_3 \times \mathbf{n})]$
          return $\{C_R(\mathbf{q}, \frac{1}{R}[\epsilon_1(\mathbf{q} \times \mathbf{e}_3) + \sqrt{R^2 - r^2}\mathbf{e}_3])\}$ // Figure 3.9(g)
        end if
      end if
    end if
  end if
end if



(a) (b) (c) (d) (e) (f) (g)

Figure 3.9: Villarceau circles in torus-torus intersection

3.7 Circle Detection in Parameter Space

In this section we give an alternative approach for detecting conic sections in a torus-simple surface intersection which is independent of the type of the simple surface. This leads to the advantage that no extensive case differentiation is needed anymore. We do so by analysing the intersection curve in the parameter space of the torus.

Let $T : \mathbb{V}^2 \to \mathbb{V}^3$ with

$$T(s,t) = \begin{pmatrix} \frac{[R(1+s^2)+r(1-s^2)](1-t^2)}{(1+s^2)(1+t^2)} \\ \frac{[R(1+s^2)+r(1-s^2)]2t}{(1+s^2)(1+t^2)} \\ \frac{2rs}{1+s^2} \end{pmatrix}, \quad s,t \in \mathbb{K}$$

be the rational parameter form of the torus in canonical position, see Section 2.2.3, and let $f : \mathbb{V}^3 \to \mathbb{K}$ be the algebraic function of the simple surface. Then

$$F(s,t) := f(T(s,t)) = 0$$

is the implicit equation of the intersection curve in the parameter space of the torus. To detect conic sections in the real space curve, we now try to locate their representatives in parameter space.

From Section 2.2.3 we know that profile circles are obtained by retaining the parameter s, cross-sectional circles are obtained by retaining the parameter t. This is equivalent to straight lines in parameter space, lying parallel to the axes, profile circles to the t-axis and cross-sectional circles to the s-axis. This means for the implicit function F, that linear terms can be separated. An intersection curve containing m profile circles and n cross-sectional circles, counted with multiplicity, has therewith the form

$$F(s,t) = G(s,t)\prod_{i=1}^{m}[s-s_i]\prod_{j=1}^{n}[t-t_j]$$

with some rational polynomial G. Here we already see the disadvantage of this approach, since the constants s_i, t_j do not necessarily belong to the underlying field $\mathbb{K}$. For example:

Consider the torus $T_{1,\frac{1}{3}}(\mathbf{o}, \mathbf{e}_3)$ and the plane $P((0,0,\frac{1}{6}), \mathbf{e}_3)$. The numerator N of the implicit function F computes to $N(s,t) = s^2 - 4s + 1$ which is irreducible under $\mathbb{Q}$, but decomposes into $(s-2-\sqrt{3})(s-2+\sqrt{3})$ under $\mathbb{Q}[\sqrt{3}]$. Substituting the solutions $s_{1,2} = 2 \pm \sqrt{3}$ into the parameter form of the torus yields

$$T(s_{1,2},t) = \begin{pmatrix} (1 \mp \frac{1}{6}\sqrt{3})\frac{1-t^2}{1+t^2} \\ (1 \mp \frac{1}{6}\sqrt{3})\frac{2t}{1+t^2} \\ \frac{1}{6} \end{pmatrix}$$

which corresponds to two profile circles $C_{1\mp\frac{1}{6}\sqrt{3}}((0,0,\frac{1}{6}), \mathbf{e}_3)$.

If nevertheless $\mathbb{K}$ is algebraically closed, we can find any profile and cross-sectional circle by factorisation of the implicit function F into linear terms of the above form and looking for real solutions. Detecting Villarceau circles in parameter space is a bit more complicated.

As we saw in Section 3.1.3, a Villarceau circle of a torus in canonical position is embedded in a plane $P(\mathbf{o}, \mathbf{n}_u)$ with

$$\mathbf{n}_u = \frac{1}{R}\left(r\frac{w^2-u^2}{w^2+u^2}, r\frac{2uw}{w^2+u^2}, \sqrt{R^2-r^2}\right)$$

and suitable parameters $u, w \in \overline{\mathbb{K}}$, not both zero. The corresponding implicit function is

$$f_P(\mathbf{x}) = \frac{r(w^2-u^2)x + 2ruwy + \sqrt{R^2-r^2}(w^2+u^2)z}{w^2+u^2}.$$

The implicit function $F(s,t) = f_P(T(s,t))$ can then be factorised into

$$F(s,t) = \frac{rH^-(s,t)H^+(s,t)}{R(1+s^2)(1+t^2)(w^2+u^2)} \quad \text{with}$$

$$H^-(s,t) = -\sqrt{R-r}(w-u)st + \sqrt{R+r}(w+u)t + \sqrt{R-r}(w+u)s + \sqrt{R+r}(w-u),$$
$$H^+(s,t) = \sqrt{R-r}(w+u)st - \sqrt{R+r}(w-u)t + \sqrt{R-r}(w-u)s + \sqrt{R+r}(w+u),$$

where the factors H^- and H^+ correspond to Villarceau circles of the sets Γ_T^{v-} and Γ_T^{v+}, respectively. Thus the shape of a Villarceau circle in parameter space equals the graph of a hyperbolic function

$$t = \frac{as+b}{cs+d}$$

with some constants $a, b, c, d \in \overline{\mathbb{K}}$. There are two exceptions, namely if $u = \pm w$. In these cases we get straight lines of the form

$$t = \mp\sqrt{\frac{R-r}{R+r}}s.$$

If the implicit function F of an intersection curve of a torus with a general simple surface factorises into

$$F(s,t) = G(s,t)\prod_{k=1}^{l}[a_k st + b_k t + c_k s + d_k]$$

with some constants $a_k, b_k, c_k, d_k \in \overline{\mathbb{K}}$ and rational polynomial G, Algorithm 5 decides, whether a Villarceau circle is contained. In this case, an exact representation of this circle is returned. This is done by comparing the coefficients a_k, b_k, c_k, d_k with the form H^- and H^+.

Summary

We found all sufficient conditions to detect any conic section in the intersection curve between a torus in canonical position and a simple surface. Therefore we just need the implicit function F of the intersection curve in parameter space of the torus, so we can avoid extensive case differentiation. On the other side, it is necessary that F decomposes into a form

$$F(s,t) = G(s,t)\prod_{i=1}^{m}[s-s_i]\prod_{j=1}^{n}[t-t_j]\prod_{k=1}^{l}[a_k st + b_k t + c_k s + d_k]$$

with some rational polynomial G and coefficients $s_i, t_j, a_k, b_k, c_k, d_k \in \overline{\mathbb{K}}$.

Algorithm 5 Detect Villarceau Circles in Parameter Space

Requires: A non-trivial polynomial $G(s,t) = ast + bt + cs + d$ and the radii R, r of the torus in canonical position

Returns: The corresponding Villarceau circle $C_R(\mathbf{q}, \mathbf{n_q})$, if existent

1: **if** $d = 0$ **then**
2: **if** $a = 0$ and $c^2(R+r) = b^2(R-r)$ **then**
3: $\mathbf{q} \leftarrow (r, 0, 0)$
4: $\mathbf{n_q} \leftarrow \frac{1}{R}(0, \mathtt{sign}(bc)r, \sqrt{R^2 - r^2})$
5: **return** $C_R(\mathbf{q}, \mathbf{n_q})$
6: **end if**
7: **else**
8: **if** $ab + cd = 0$ **and** $a^2(R+r) = d^2(R-r)$ **then**
9: $c_\phi \leftarrow \frac{2bd}{b^2+d^2}$
10: $s_\phi \leftarrow \frac{b^2-d^2}{b^2+d^2}$
11: $\mathbf{q} \leftarrow (rs_\phi, -rc_\phi, 0)$
12: $\mathbf{n_q} \leftarrow \frac{1}{R}(-\mathtt{sign}(ad)rc_\phi, -\mathtt{sign}(ad)rs_\phi, \sqrt{R^2 - r^2})$
13: **return** $C_R(\mathbf{q}, \mathbf{n_q})$
14: **end if**
15: **end if**

Chapter 4

Intersecting Two Objects

In this chapter we intersect two objects, where one object is always a torus and the other object is a simple surface like plane, sphere, cylinder, cone or another torus. We do not consider intersections between two quadrics, since these cases are already handled sufficiently by previous literature [6, 24, 32, 49, 60, 68, 69, 72].

Analogue to Kim [46] we provide algorithms for any kind of input, computing reference points for each intersection curve component. These reference points will become initial starting points for the curve tracing step, see Chapter 5. Furthermore we detect degeneracies like embedded conic sections and singularities.

In contrast to Kim's work we provide exact results, i.e. all degeneracies can be represented by exact number types. In Chapter 3 we already saw, that one algebraic field extension suffices to represent embedded conic sections. In this chapter we will see, that two algebraic field extensions suffice to represent any singularity, in other words: A singular point can be represented exactly by two-root numbers in the worst case. The exact representation of degeneracies is important, since those are highly responsible for the most instabilities of current CAD applications.

We represent reference points on regular intersection curve components by roots of algebraic equations. Thus we do not employ the same accurate representation as for singular points, but with methods of refinement we can guarantee that the exact points lie within any arbitrary small environment. This is sufficient for the further processing, since the tracing of regular curve components will be of numerical nature and therewith we only can give a guarantee that the exact intersection curve lies within a given error bound of the traced curve.

Following Kim's configuration space approach, see Section 2.3, we first discuss planar intersections of simple objects such as plane-plane intersections and plane-quadric intersections (Section 4.1). Then we consider intersections of lower dimensional objects like circle-line intersections, circle-conic intersections, etc. (Section 4.2). Intersections in C-space, involving a torus, will complete the algorithms introduced in Table 2.2 (Section 4.3). Finally we present algorithms for the computation of reference points and the detection of degeneracies for each torus-simple surface intersection separately (Section 4.4).

4.1 Planar Intersections

In this section we consider the intersections of planes and intersections of a plane with a quadric. Since the resulting intersection curves are planar and the involved objects are algebraic surfaces of degree two at most, we will get conics as results. This is the first step in Kim's configuration space approach to simplify computations. The second step will be the computation of intersection points of the computed conics, see Section 4.2.

4.1.1 Plane-Plane Intersection

Given two planes $P_1 = P(\mathbf{p}_1, \mathbf{n}_1)$ and $P_2 = P(\mathbf{p}_2, \mathbf{n}_2)$, the set of common points consists of

$$P_1 \cap P_2 = \big\{\mathbf{x} \in \mathbb{V}^3 \big| \langle \mathbf{x} - \mathbf{p}_1, \mathbf{n}_1 \rangle = \langle \mathbf{x} - \mathbf{p}_2, \mathbf{n}_2 \rangle = 0\big\}.$$

It is clear that two planes intersect if and only if they are not coplanar, i.e. $\mathbf{n}_1 \times \mathbf{n}_2 \neq \mathbf{o}$. Otherwise the intersection is empty ($\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \neq 0$) or the planes are coincident ($\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle = 0$).

Furthermore we know that the intersection of two planes consists of a line $L(\mathbf{q}, \mathbf{d})$ where the direction vector $\mathbf{d}$ is perpendicular to both plane normals $\mathbf{n}_1$ and $\mathbf{n}_2$. Since direction vectors of lines have not necessarily to be normalised, we can define

$$\mathbf{d} = \mathbf{n}_1 \times \mathbf{n}_2.$$

It is left to determine the suspension point $\mathbf{q}$.

From Figure 4.1 we follow directly

$$\begin{aligned} b &= \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{n}_2 \rangle, \\ \sin\phi &= \|\mathbf{n}_1 \times \mathbf{n}_2\|, \\ \mathbf{a} &= \frac{\mathbf{d} \times \mathbf{n}_1}{\|\mathbf{d}\|}. \end{aligned}$$

Note that the distance b is signed. It is positive (negative), if $\mathbf{p}_1$ lies in the positive (negative) half space of plane P_2. Therewith we yield $\mathbf{q}$ as

$$\begin{aligned} \mathbf{q} &= \mathbf{p}_1 - \frac{b}{\sin\phi}\mathbf{a} \\ &= \mathbf{p}_1 + \frac{\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_2 \rangle}{\|\mathbf{d}\|^2}(\mathbf{d} \times \mathbf{n}_1). \end{aligned}$$

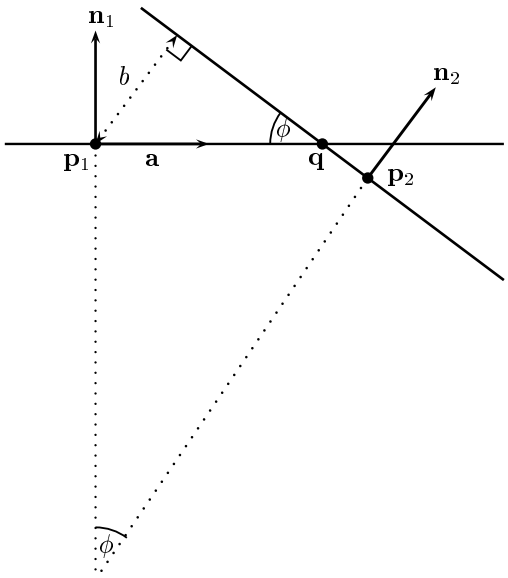


Figure 4.1: Two intersecting planes viewed in the direction $\mathbf{d}$

4.1.2 Sphere-Plane Intersection

Given a sphere $S = S_\delta(\mathbf{p}_1)$ and a plane $P = P(\mathbf{p}_2, \mathbf{n})$. If these objects intersect, the intersection curve consists of a circle $C = C_d(\mathbf{q}, \mathbf{n})$. There is no intersection, if S lies completely in one of the half spaces of P, i.e.

$$|\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{n} \rangle| > \delta.$$

In the case of equality S and P share the touching point

$$\mathbf{q} = \mathbf{p}_1 - \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{n} \rangle \mathbf{n}$$

which is the orthogonal projection of $\mathbf{p}_1$ on P. Otherwise $\mathbf{q}$ lies inside S and $\mathbf{q}$ becomes the centre of the resulting circle C, see Figure 3.2(a). As we already saw in Section 3.1.4, the radius d of C then computes to

$$d = \sqrt{\delta^2 - \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{n} \rangle^2}.$$

4.1.3 Cylinder-Plane Intersection

Given a cylinder $Y = Y_\delta(\mathbf{p}_1, \mathbf{n}_1)$ and a plane $P = P(\mathbf{p}_2, \mathbf{n}_2)$. There is no intersection of these two objects, if and only if Y lies completely in one of the half spaces of P, i.e.

$$\langle \mathbf{n}_1, \mathbf{n}_2 \rangle = 0 \quad \text{and} \quad |\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{n}_2 \rangle| > \delta.$$

Otherwise the intersection curve consists of a conic E, embedded in the plane P. To represent E, we need a local coordinate system on P in the form of two orthonormal vectors $\mathbf{a}$ and $\mathbf{b}$ with $\mathbf{a} \times \mathbf{b} = \mathbf{n}_2$, e.g. the first two columns of the matrix $M_{\mathbf{n}_2}$, see (2.1).

A point $\mathbf{x}$ on P can then be represented by

$$\mathbf{x} = \mathbf{p}_2 + u\mathbf{a} + v\mathbf{b} \quad \text{with} \quad u, v \in \mathbb{K}\,.$$

Substituting this equation into the algebraic function f_Y of the cylinder Y, see Section 2.2.1, we gain the polynomial representation of E

$$f_E(u, v) = au^2 + bv^2 + 2fuv + 2lu + 2mv + d$$

with

$$\begin{aligned}
a &= 1 - \langle \mathbf{a}, \mathbf{n}_1 \rangle^2, \\
b &= 1 - \langle \mathbf{b}, \mathbf{n}_1 \rangle^2, \\
f &= -\langle \mathbf{a}, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle, \\
l &= \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{a} \rangle - \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{a}, \mathbf{n}_1 \rangle, \\
m &= \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{b} \rangle - \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle, \\
d &= \|\mathbf{p}_2 - \mathbf{p}_1\|^2 - \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle^2 - \delta^2.
\end{aligned}$$

4.1.4 Cone-Plane Intersection

Given a cone $K = K_\delta(\mathbf{p}_1, \mathbf{n}_1)$ and a plane $P = P(\mathbf{p}_2, \mathbf{n}_2)$. We know that there is always an intersection between these two objects, and it consists of a conic E. Similar to the previous section we determine the polynomial representation of E by substituting the parameter form of P into the algebraic function f_K of K and we gain

$$f_E(u, v) = au^2 + bv^2 + 2fuv + 2lu + 2mv + d$$

with

$$\begin{aligned}
a &= 1 - (1 + \delta^2) \langle \mathbf{a}, \mathbf{n}_1 \rangle^2, \\
b &= 1 - (1 + \delta^2) \langle \mathbf{b}, \mathbf{n}_1 \rangle^2, \\
f &= -(1 + \delta^2) \langle \mathbf{a}, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle, \\
l &= \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{a} \rangle - (1 + \delta^2) \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{a}, \mathbf{n}_1 \rangle, \\
m &= \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{b} \rangle - (1 + \delta^2) \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle, \\
d &= \|\mathbf{p}_2 - \mathbf{p}_1\|^2 - (1 + \delta^2) \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle^2.
\end{aligned}$$

Algorithm 6 `PlanePlaneIntersection`$(P(\mathbf{p}_1, \mathbf{n}_1), P(\mathbf{p}_2, \mathbf{n}_2))$

Requires: Two planes $P(\mathbf{p}_1, \mathbf{n}_1)$ and $P(\mathbf{p}_2, \mathbf{n}_2)$
Returns: A line $L(\mathbf{q}, \mathbf{d})$

1: $\mathbf{d} \leftarrow \mathbf{n}_1 \times \mathbf{n}_2$
2: **if** $\|\mathbf{d}\| \neq 0$ **then**
3: $\quad \mathbf{q} \leftarrow \mathbf{p}_1 + \frac{\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_2 \rangle}{\|\mathbf{d}\|^2} (\mathbf{d} \times \mathbf{n}_1)$
4: $\quad$ **return** $L(\mathbf{q}, \mathbf{d})$
5: **end if**

Algorithm 7 `SpherePlaneIntersection`$(S_\delta(\mathbf{p}_1), P(\mathbf{p}_2, \mathbf{n}))$

Requires: A sphere $S_\delta(\mathbf{p}_1)$ and a plane $P(\mathbf{p}_2, \mathbf{n})$
Returns: A circle $C_d(\mathbf{q}, \mathbf{n})$

$c \leftarrow \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{n} \rangle$
if $\|c\| \leq \delta$ **then**
  $\mathbf{q} \leftarrow \mathbf{p}_1 - c\mathbf{n}$
  $d \leftarrow \sqrt{\delta^2 - c^2}$
  return $C_d(\mathbf{q}, \mathbf{n})$
end if

Algorithm 8 `CylinderPlaneIntersection`$(Y_\delta(\mathbf{p}_1, \mathbf{n}_1), P(\mathbf{p}_2, \mathbf{n}_2))$

Requires: A cylinder $Y_\delta(\mathbf{p}_1, \mathbf{n}_1)$ and a plane $P(\mathbf{p}_2, \mathbf{n}_2)$
Returns: A conic $E(\mathbf{p}_2, M_{\mathbf{n}_2})$

$c \leftarrow \langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{n}_2 \rangle$
if $\langle \mathbf{n}_1, \mathbf{n}_2 \rangle \neq 0$ **or** $\|c\| \leq \delta$ **then**
  $\mathbf{a} \leftarrow M_{\mathbf{n}_2}\mathbf{e}_1$
  $\mathbf{b} \leftarrow M_{\mathbf{n}_2}\mathbf{e}_2$
  $a \leftarrow 1 - \langle \mathbf{a}, \mathbf{n}_1 \rangle^2$
  $b \leftarrow 1 - \langle \mathbf{b}, \mathbf{n}_1 \rangle^2$
  $f \leftarrow -\langle \mathbf{a}, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle$
  $l \leftarrow \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{a} \rangle - \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{a}, \mathbf{n}_1 \rangle$
  $m \leftarrow \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{b} \rangle - \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle$
  $d \leftarrow \|\mathbf{p}_2 - \mathbf{p}_1\|^2 - \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle^2 - \delta^2$
  $E \leftarrow \begin{pmatrix} a & f & l \\ f & b & m \\ l & m & d \end{pmatrix}$
  return $E(\mathbf{p}_2, M_{\mathbf{n}_2})$
end if

Algorithm 9 `ConePlaneIntersection`$(K_\delta(\mathbf{p}_1, \mathbf{n}_1), P(\mathbf{p}_2, \mathbf{n}_2))$

Requires: A cone $K_\delta(\mathbf{p}_1, \mathbf{n}_1)$ and a plane $P(\mathbf{p}_2, \mathbf{n}_2)$
Returns: A conic $E(\mathbf{p}_2, M_{\mathbf{n}_2})$

$\mathbf{a} \leftarrow M_{\mathbf{n}_2}\mathbf{e}_1$
$\mathbf{b} \leftarrow M_{\mathbf{n}_2}\mathbf{e}_2$
$a \leftarrow 1 - (1+\delta^2)\langle \mathbf{a}, \mathbf{n}_1 \rangle^2$
$b \leftarrow 1 - (1+\delta^2)\langle \mathbf{b}, \mathbf{n}_1 \rangle^2$
$f \leftarrow -(1+\delta^2)\langle \mathbf{a}, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle$
$l \leftarrow \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{a} \rangle - (1+\delta^2)\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{a}, \mathbf{n}_1 \rangle$
$m \leftarrow \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{b} \rangle - (1+\delta^2)\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \langle \mathbf{b}, \mathbf{n}_1 \rangle$
$d \leftarrow \|\mathbf{p}_2 - \mathbf{p}_1\|^2 - (1+\delta^2)\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle^2$
$E \leftarrow \begin{pmatrix} a & f & l \\ f & b & m \\ l & m & d \end{pmatrix}$
return $E(\mathbf{p}_2, M_{\mathbf{n}_2})$

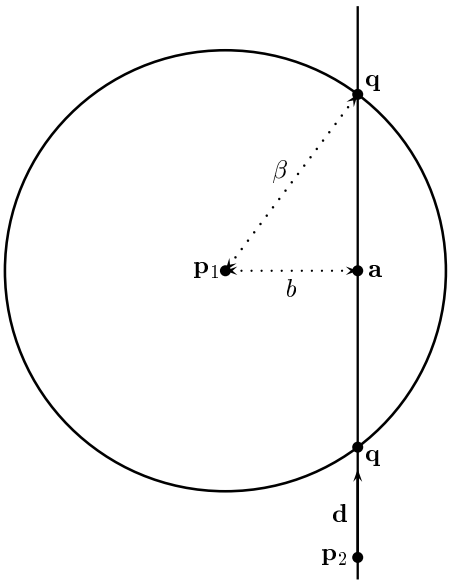


Figure 4.2: Intersecting a circle with a line

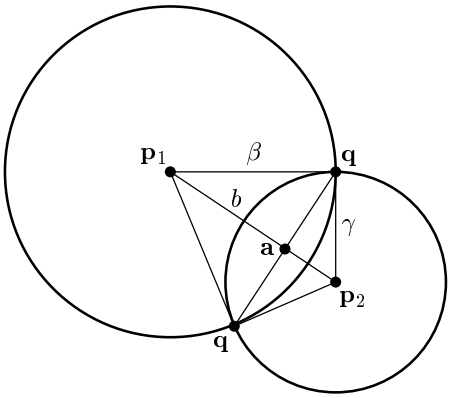


Figure 4.3: Intersecting two circles

4.2 Lower Dimensional Intersections

In this section we consider intersections of conics. This presumes that both conics to be examined have to share the same plane. This is already prepared by our algorithms, for example: We intersect a circle $C = C_\beta(\mathbf{p}_1, \mathbf{n}_1)$ with a cone $K = K_\delta(\mathbf{p}_2, \mathbf{n}_2)$. Therefore we first intersect the cone K with the plane $P = P(\mathbf{p}_1, \mathbf{n}_1)$ the circle C is embedded in, resulting in a conic E. Then we intersect this conic E with the circle C, knowing that both lie in the same plane. The remaining problem is, that both C and E have their own local coordinate system on P. So intersecting conics requires a coordinate transformation at first. This will be discussed in the following.

4.2.1 Circle-Line Intersection

Given a circle $C = C_\beta(\mathbf{p}_1, \mathbf{n})$ and a line $L = L(\mathbf{p}_2, \mathbf{d})$. We presume both objects to lie in the same plane, i.e.

$$\langle \mathbf{n}, \mathbf{d} \rangle = 0 \quad \text{and} \quad \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n} \rangle = 0.$$

From Figure 4.2 we follow

$$\begin{aligned} \mathbf{a} &= \mathbf{p}_2 + \frac{\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{d} \rangle}{\|\mathbf{d}\|} \cdot \frac{\mathbf{d}}{\|\mathbf{d}\|}, \\ b &= \|\mathbf{p}_1 - \mathbf{a}\| \\ &= \frac{\|(\mathbf{p}_1 - \mathbf{p}_2) \times \mathbf{d}\|}{\|\mathbf{d}\|}. \end{aligned}$$

If $b < \beta$, there is no intersection. In case of equality we get a single touching point $\mathbf{a}$. Otherwise we get two intersection points

$$\mathbf{q} = \mathbf{p}_2 + \frac{1}{\|\mathbf{d}\|^2} \big[\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{d} \rangle \pm \sqrt{d} \big] \mathbf{d}$$

with

$$d = \beta^2 \|\mathbf{d}\|^2 - \|(\mathbf{p}_1 - \mathbf{p}_2) \times \mathbf{d}\|^2.$$

4.2.2 Circle-Circle Intersection

Given two circles $C_1 = C_\beta(\mathbf{p}_1, \mathbf{n}_1)$ and $C_2 = C_\gamma(\mathbf{p}_2, \mathbf{n}_2)$. We presume both circles to lie in the same plane, i.e.

$$\mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{o} \quad \text{and} \quad \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle = 0.$$

If we additionally have $\mathbf{p}_2 = \mathbf{p}_1$, both circles either coincide ($\beta = \gamma$), or they are concentric and thus we have no intersection. So we can assume $\mathbf{p}_2 \neq \mathbf{p}_1$. From Figure 4.3 we follow

$$\begin{aligned} b &= \frac{\beta^2 - \gamma^2 + \|\mathbf{p}_2 - \mathbf{p}_1\|^2}{2\|\mathbf{p}_2 - \mathbf{p}_1\|}, \\ \mathbf{a} &= \mathbf{p}_1 + b\frac{\mathbf{p}_2 - \mathbf{p}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|}, \\ \mathbf{q} &= \mathbf{a} \pm \sqrt{\beta^2 - b^2}\frac{(\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{n}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|}. \end{aligned}$$

Together we get the intersection points

$$\mathbf{q} = \mathbf{p}_1 + \frac{\beta^2 - \gamma^2 + \|\mathbf{p}_2 - \mathbf{p}_1\|^2}{2\|\mathbf{p}_2 - \mathbf{p}_1\|^2}(\mathbf{p}_2 - \mathbf{p}_1) \pm \frac{\sqrt{d}}{2\|\mathbf{p}_2 - \mathbf{p}_1\|^2}(\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{n}_1$$

with

$$d = -[\|\mathbf{p}_2 - \mathbf{p}_1\|^2 - (\beta + \gamma)^2][\|\mathbf{p}_2 - \mathbf{p}_1\|^2 - (\beta - \gamma)^2].$$

Depending on d, we get two different intersection points if

$$|\beta - \gamma| < \|\mathbf{p}_2 - \mathbf{p}_1\| < \beta + \gamma.$$

In case of equality of one of the above relations we get one single touching point $\mathbf{a}$. Otherwise there is no intersection.

4.2.3 Circle-Conic Intersection

Given a circle $C = C_\beta(\mathbf{p}_1, \mathbf{n}_1)$ and a conic $E = E(\mathbf{p}_2, M_2)$, where E simultaneously denotes the coefficient matrix of the conic, see Section 2.2.2

$$E = \begin{pmatrix} a & f & l \\ f & b & m \\ l & m & d \end{pmatrix}.$$

We specify the local coordinate systems of both objects by

$$\begin{aligned} M_1 &= M_{\mathbf{n}_1}, \\ \mathbf{a}_i &= M_i \mathbf{e}_1, \ i = 1, 2 \\ \mathbf{b}_i &= M_i \mathbf{e}_2, \ i = 1, 2 \\ \mathbf{n}_2 &= M_2 \mathbf{e}_3, \end{aligned}$$

where $M_1 = \begin{pmatrix} \mathbf{a}_1, & \mathbf{b}_1, & \mathbf{n}_1 \end{pmatrix}$ and $M_2 = \begin{pmatrix} \mathbf{a}_2, & \mathbf{b}_2, & \mathbf{n}_2 \end{pmatrix}$ denote the orientation matrices of C and E, respectively. We further presume both objects to lie in the same plane, i.e.

$$\mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{o} \quad \text{and} \quad \langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle = 0.$$

To find the intersection points between two objects, we usually substitute the parameter form of one object into the implicit equation of the other one, we solve the resulting univariate polynomial and evaluate the parameter form at the computed roots. But therefor both objects must belong to the same coordinate system.

Let $\mathbf{a}_1$ and $\mathbf{b}_1$ the two orthogonal unit vectors, building the coordinate system S_C of the circle C, see Figure 4.4. A point $\mathbf{x} = (x, y, z) \in \mathbb{V}^3$ in the plane $P(\mathbf{p}_1, \mathbf{n}_1)$ can then be expressed by

$$\mathbf{x} = \mathbf{p}_1 + u\mathbf{a}_1 + v\mathbf{b}_1$$

with $\mathbf{x}' = (u, v) \in \mathbb{V}^2$. We can derive $\mathbf{x}'$ from

$$\begin{pmatrix} u \\ v \\ 0 \end{pmatrix} = M_1^{-1}(\mathbf{x} - \mathbf{p}_1),$$

see Section 2.2.2. Analogue, let $\mathbf{a}_2$ and $\mathbf{b}_2$ build the local coordinate system S_E for the conic E. Then we can determine $\mathbf{a}_2'$, $\mathbf{b}_2'$ and $\mathbf{p}'$ in the coordinate system S_C of the circle by neglecting the last coordinate of $M_1^{-1}\mathbf{a}_2$, $M_1^{-1}\mathbf{b}_2$ and $M_1^{-1}(\mathbf{p}_2 - \mathbf{p}_1)$, respectively.

We determine the transformation A from S_C to S_E, represented by its homogeneous transformation matrix. A point $\mathbf{x}_E$ in coordinate system S_E can then be expressed by a point $\mathbf{x}_C$ in coordinate system S_C, where $\mathbf{x}_C = A\mathbf{x}_E$. For a point $\mathbf{x}_E$ lying on conic E it holds

$$0 = \mathbf{x}_E^T E \mathbf{x}_E = (A^{-1}\mathbf{x}_C)^T E (A^{-1}\mathbf{x}_C) = {\mathbf{x}_C}^T (A^{-T} E A^{-1}) \mathbf{x}_C.$$

Thus the conic E in coordinate system S_E is also represented by a conic $E' = A^{-T}EA^{-1}$ in coordinate system S_C.

To determine the transformation A consider Figure 4.4(b). A consists of a rotational and a translational part. We write

$$A = \begin{pmatrix} R & \mathbf{p}' \\ 0 & 1 \end{pmatrix} \quad \text{with} \quad R = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}.$$

The inverse transformation computes then to

$$A^{-1} = \begin{pmatrix} R^T & -R^T\mathbf{p}' \\ 0 & 1 \end{pmatrix}.$$

We can avoid the computation of the angle ϕ, since we derive directly

$$\begin{aligned} \cos\phi &= \langle \mathbf{a}_1', \mathbf{a}_2' \rangle = \langle \mathbf{a}_1, \mathbf{a}_2 \rangle, \\ \sin\phi &= -\cos\theta = -\langle \mathbf{a}_1, \mathbf{b}_2 \rangle. \end{aligned}$$

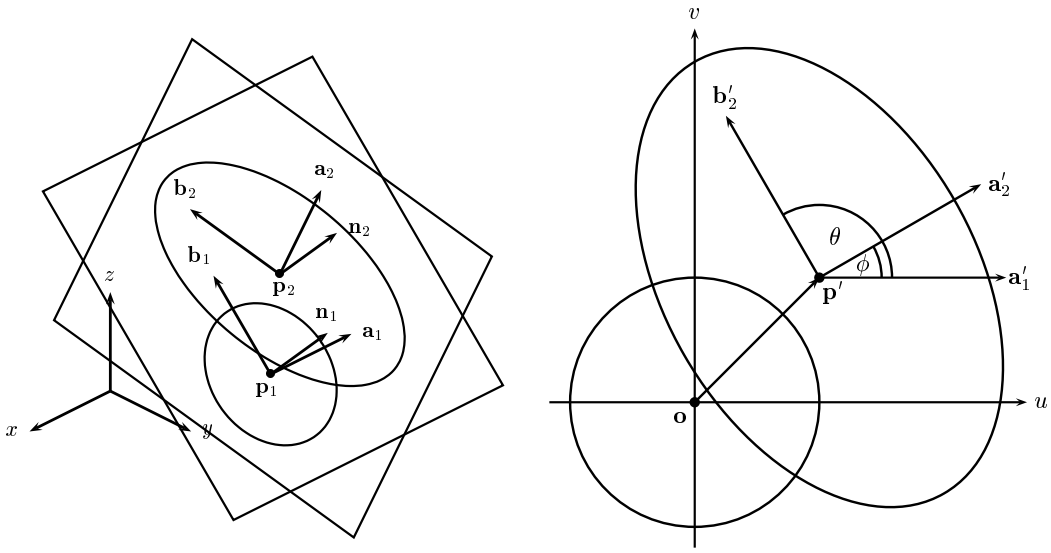
Once we have determined conic E' in the coordinate system of the circle C, we substitute the implicit polynomial equation of E' into the parameter form of C, yielding

$$f(t) = c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0 \tag{4.1}$$

with

$$\begin{aligned} c_4 &= a'\beta^2 - 2l'\beta + d', \\ c_3 &= -4f'\beta^2 + 4m'\beta, \\ c_2 &= -2a'\beta^2 + 4b'\beta^2 + 2d', \\ c_1 &= 4f'\beta^2 + 4m'\beta, \\ c_0 &= a'\beta^2 + 2l'\beta + d'. \end{aligned}$$

A root t_0 of this quartic polynomial corresponds to an intersection point $\mathbf{q}_0$ between C and E', which can be computed by evaluating the parameter form of C at t_0. Since there are at most four different roots, the intersection between a circle and a conic can have up to four intersection points. Furthermore we can state that singular intersection points or touching points correspond



(a) Circle and conic in real affine space

(b) Circle and conic in circle's coordinate system

Figure 4.4: Circle-conic intersection

to roots with higher multiplicity. To determine the complexity of singular points, we consider the following:

Let f be a polynomial with $deg(f) \leq 4$ and a root t_0 with multiplicity of at least two. Then we can write $f = g^k \cdot h$ with $g(t_0) = 0$, $deg(g) \geq 1$ and $k \geq 2$. It follows immediately that $deg(g) \leq 2$. That means for the root t_0 that we can represent it by an one-root number. If f has another root t_1 with higher multiplicity, we distinguish between two cases: Firstly t_1 is also a root of g. Then both roots lie in the same extension field, see Algorithm 12. Otherwise t_1 must be a root of h and f decomposes into $f = g^2 \cdot h^2$. Then g and h are linear terms and therewith t_0 and t_1 are rational.

To find all roots of a quartic polynomial f, we first factorise f into

$$f = \prod_{i=1}^{n} g_i^{k_i}$$

with Yun's square-free factorisation algorithm, see [74]. Then the roots of the factors g_i with multiplicity $k_i \geq 2$ can be computed by a quadratic formula. The remaining roots can be found by a root isolation algorithm like Descartes' rule of signs, see [40]. We assume in our algorithms that the two predicates `SquarefreeFactorisation` and `RootIsolator` are available. Moreover we presume that the roots, returned by `RootIsolator`, are given as `AlgebraicReal` numbers, ordered by their respective intervals.

As implied in Section 2.2.2, we cannot evaluate the parameter form of a circle at infinity. Thus we will eventually miss intersections. Going over to homogeneous coordinates, equation (4.1) becomes

$$f(t, w) = c_4 t^4 + c_3 t^3 w + c_2 t^2 w^2 + c_1 t w^3 + c_0 w^4.$$

A point at infinity can be represented by $t_\infty = (\gamma, 0) \in \mathbb{P}$ with $\gamma \neq 0$. So $c_4 = 0$ is equivalent to an intersection at point $(-\beta, 0) \in \mathbb{V}^2$. Is even $c_3 = 0$, f has a root at infinity with higher multiplicity and thus the intersection at $(-\beta, 0)$ is tangential.

Algorithm 10 `CircleLineIntersection`$(C_\beta(\mathbf{p}_1, \mathbf{n}), L(\mathbf{p}_2, \mathbf{d}))$

Requires: A circles $C_\beta(\mathbf{p}_1, \mathbf{n})$ and a line $L(\mathbf{p}_2, \mathbf{d})$
Returns: A set of intersection points $\{\mathbf{q}\}$

if $\langle \mathbf{n}, \mathbf{d} \rangle = 0$ **and** $\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n} \rangle = 0$ **then**
  $c \leftarrow \|\mathbf{d}\|^2$
  $d \leftarrow \beta^2 c^2 - \|(\mathbf{p}_1 - \mathbf{p}_2) \times \mathbf{n}\|^2$
  if $d = 0$ **then**
    return $\{\mathbf{p}_2 + \frac{\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{d} \rangle}{c} \mathbf{d}\}$
  else if $d > 0$ **then**
    $\mathbf{a} \leftarrow \mathbf{p}_2 + \frac{\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{d} \rangle}{c} \mathbf{d}$
    $\mathbf{b} \leftarrow \frac{\sqrt{d}}{c} \mathbf{d}$
    return $\{\mathbf{a} + \mathbf{b}, \mathbf{a} - \mathbf{b}\}$
  end if
end if

Algorithm 11 `CircleCircleIntersection`$(C_\beta(\mathbf{p}_1, \mathbf{n}_1), C_\gamma(\mathbf{p}_2, \mathbf{n}_2))$

Requires: Two circles $C_\beta(\mathbf{p}_1, \mathbf{n}_1)$ and $C_\gamma(\mathbf{p}_2, \mathbf{n}_2)$
Returns: A set of intersection points $\{\mathbf{q}\}$

if $\mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{o}$ **and** $\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle = 0$ **then**
  $c \leftarrow \|\mathbf{p}_2 - \mathbf{p}_1\|^2$
  $d \leftarrow -[c - (\beta + \gamma)^2][c - (\beta - \gamma)^2]$
  if $d = 0$ **then**
    return $\{\mathbf{p}_1 + \frac{\beta^2 - \gamma^2 + c}{2c}(\mathbf{p}_2 - \mathbf{p}_1)\}$
  else if $d > 0$ **then**
    $\mathbf{a} \leftarrow \mathbf{p}_1 + \frac{\beta^2 - \gamma^2 + c}{2c}(\mathbf{p}_2 - \mathbf{p}_1)$
    $\mathbf{b} \leftarrow \frac{1}{2c}\sqrt{d}(\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{n}_1$
    return $\{\mathbf{a} + \mathbf{b}, \mathbf{a} - \mathbf{b}\}$
  end if
end if

Algorithm 12 `SolveQuadratic`(f)

Requires: A polynomial $f = c_2 t^2 + c_1 t + c_0$
Returns: A set of roots $\{t\}$

if $c_2 \neq 0$ **then**
  $d \leftarrow c_1^2 - 4 c_2 c_0$
  if $d = 0$ **then**
    return $\{-\frac{c_1}{2c_2}\}$
  else if $d > 0$ **then**
    return $\{-\frac{c_1 + \sqrt{d}}{2c_2}, -\frac{c_1 - \sqrt{d}}{2c_2}\}$
  end if
else if $c_1 \neq 0$ **then**
  return $\{-\frac{c_0}{c_1}\}$
end if

Algorithm 13 `SolveQuartic`(f)

Requires: A polynomial f with $\deg(f) \leq 4$
Returns: A set of simple roots $\{t_s\}$ and a set of roots with higher multiplicity $\{t_d\}$

$\{(g_i, k_i)\} \leftarrow$ `SquarefreeFactorisation`(f)
for all i **do**
  if $k_i = 1$ **then**
    $\{t_s\} \leftarrow$ `RootIsolator`(g_i)
  else
    $\{t_d\} \leftarrow$ `SolveQuadratic`(g_i)
  end if
end for
return $\{t_s\}, \{t_d\}$

Algorithm 14 `CircleConicIntersection`$(C_\beta(\mathbf{p}_1, \mathbf{n}_1), E(\mathbf{p}_2, M_2))$

Requires: A circle $C = C_\beta(\mathbf{p}_1, \mathbf{n}_1)$ and a conic $E(\mathbf{p}_2, M_2)$
Returns: A set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

if $\mathbf{n}_1 \times (M_2\mathbf{e}_3) = \mathbf{o}$ **and** $\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle = 0$ **then**
  $\mathbf{a}_1 \leftarrow M_{\mathbf{n}_1}\mathbf{e}_1$
  $\mathbf{p} \leftarrow M_{\mathbf{n}_1}^{-1}(\mathbf{p}_2 - \mathbf{p}_1)$
  $c_\phi \leftarrow \langle \mathbf{a}_1, M_2\mathbf{e}_1 \rangle$
  $s_\phi \leftarrow -\langle \mathbf{a}_1, M_2\mathbf{e}_2 \rangle$
  $A \leftarrow \begin{pmatrix} c_\phi & -s_\phi & p_x \\ s_\phi & c_\phi & p_y \\ 0 & 0 & 1 \end{pmatrix}$
  $E' \leftarrow A^{-T}EA^{-1}$
  $\mathbf{c} \leftarrow (1 - t^2, 2t, 1 + t^2)$
  $f \leftarrow \mathbf{c}E'\mathbf{c}^T$
  $\{t_s\}, \{t_d\} \leftarrow$ `SolveQuartic`(f)
  for all $t \in \{t_s\}$ **do**
    $\{\mathbf{q}_r\} \leftarrow$ `IntersectionPoint`$(t, \mathfrak{P}_C)$
  end for
  for all $t \in \{t_d\}$ **do**
    $\{\mathbf{q}_s\} \leftarrow$ `IntersectionPoint`$(t, \mathfrak{P}_C)$
  end for
  if `coeff`$(f, 4) = 0$ **then**
    if `coeff`$(f, 3) = 0$ **then**
      $\{\mathbf{q}_s\} \leftarrow$ `IntersectionPoint`$(\infty, \mathfrak{P}_C)$
    else
      $\{\mathbf{q}_r\} \leftarrow$ `IntersectionPoint`$(\infty, \mathfrak{P}_C)$
    end if
  end if
end if
return $\{\mathbf{q}_r\}, \{\mathbf{q}_s\}$

4.3 Torus Involved Intersections

As we see in Table 2.2, there is a type of intersections involving a torus and a lower dimensional object like a point, a line or a circle. We will see that, although a torus is an algebraic surface with a higher degree than a quadric, we can even here reduce these intersections to solving an univariate polynomial of degree four at most. Similar to previous algorithms a set of regular respective singular intersection points will be returned. The only exception are torus-point intersections. Here we just can determine the region where the given point is lying in.

All the following algorithms assume that the torus lies in canonical position.

4.3.1 Torus-Point Intersection

Given a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and a point $\mathbf{p}$. The torus subdivides the space into three regions T, T_- and T_+, see Section 2.2, where

- $\mathbf{p} \in T$ means, $\mathbf{p}$ lies on the surface,
- $\mathbf{p} \in T_-$ means, $\mathbf{p}$ lies inside the torus,
- $\mathbf{p} \in T_+$ means, $\mathbf{p}$ lies outside the torus.

We determine the region $\mathbf{p}$ is lying in by evaluating the algebraic function of the torus at the point $\mathbf{p}$. Computing the sign of $f_T(\mathbf{p})$, the values of $-1, 0, 1$ are equivalent to $\mathbf{p}$ lies inside, on or outside T, respectively.

4.3.2 Torus-Line Intersection

Given a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and a line $L = L(\mathbf{p}, \mathbf{d})$. Substituting the parameter form of L into the implicit algebraic equation of T yields

$$f(t) = c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0$$

with

$$\begin{aligned}
c_4 &= \|\mathbf{d}\|^4, \\
c_3 &= 4\|\mathbf{d}\|^2 \langle \mathbf{p}, \mathbf{d} \rangle, \\
c_2 &= 2\|\mathbf{d}\|^2(\|\mathbf{p}\|^2 + R^2 - r^2) + 4\langle \mathbf{p}, \mathbf{d} \rangle^2 - 4R^2(d_x^2 + d_y^2), \\
c_1 &= 4\langle \mathbf{p}, \mathbf{d} \rangle(\|\mathbf{p}\|^2 + R^2 - r^2) - 8R^2(p_x d_x + p_y d_y), \\
c_0 &= (\|\mathbf{p}\|^2 + R^2 - r^2)^2 - 4R^2(p_x^2 + p_y^2).
\end{aligned}$$

Analogue to previous sections we solve the polynomial and save the intersection points as pairs of a root and a parameter form.

4.3.3 Torus-Circle Intersection

Given a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and a circle $C = C_\beta(\mathbf{p}, \mathbf{n})$. Again we perform two steps: First we intersect T with the plane $P = P(\mathbf{p}, \mathbf{n})$ the circle is embedded in. This results in a planar algebraic curve of degree four. Then we intersect this algebraic curve with C and gain the intersection points we are looking for. According to Bezout's theorem, the second step might yield up to eight intersection points. However, we will show that the problem reduces to a quartic univariate polynomial and we just get up to four solutions, compare also [46].

To compute the intersection between T and P, we substitute the parameter form of P into the implicit algebraic function f_T (2.5) of T. Therefore we determine two orthogonal unit vectors lying in P, e.g. the first two columns of the matrix $M_\mathbf{n}$, see (2.1), and denote these with $\mathbf{a}$ and $\mathbf{b}$. The parameter form of P then becomes

$$P(u, v) = \mathbf{p} + u\mathbf{a} + v\mathbf{b}.$$

Substituting into f_T yields

$$f_T(P(u,v)) = \left[u^2+v^2+2u\langle \mathbf{p},\mathbf{a}\rangle+2v\langle \mathbf{p},\mathbf{b}\rangle+\|\mathbf{p}\|^2+R^2-r^2\right]^2-4R^2\left[(p_x+ua_x+vb_x)^2+(p_y+ua_y+vb_y)^2\right].$$

Expanding this yields a quartic bivariate polynomial. However, we can simplify the polynomial by substituting

$$u^2 + v^2 - \beta^2 = 0$$

which comes from the implicit form of the circle C. Therewith we gain

$$f_T(P(u,v)) = au^2 + bv^2 + 2fuv + 2lu + 2mv + d$$

with

$$\begin{aligned}
a &= 4\langle \mathbf{p},\mathbf{a}\rangle - 4R^2(a_x^2 + a_y^2),\\
b &= 4\langle \mathbf{p},\mathbf{b}\rangle - 4R^2(b_x^2 + b_y^2),\\
f &= 4\langle \mathbf{p},\mathbf{a}\rangle\langle \mathbf{p},\mathbf{b}\rangle - 4R^2(a_xb_x + a_yb_y),\\
l &= 2\langle \mathbf{p},\mathbf{a}\rangle(\|\mathbf{p}\|^2 + \beta^2 + R^2 - r^2) - 4R^2(p_xa_x + p_ya_y),\\
m &= 2\langle \mathbf{p},\mathbf{b}\rangle(\|\mathbf{p}\|^2 + \beta^2 + R^2 - r^2) - 4R^2(p_xb_x + p_yb_y),\\
d &= (\|\mathbf{p}\|^2 + \beta^2 + R^2 - r^2)^2 - 4R^2(p_x^2 + p_y^2).
\end{aligned}$$

This corresponds to a conic E. Thus we can use the algorithm `CircleConicIntersection` to compute the intersection points between E and C, corresponding to the intersection points between T and C.

4.4 Torus-Simple Surface Intersections

This section considers the intersections between a torus and a simple surface like plane, sphere, cylinder, cone, or another torus. Thereby we restrict ourselves to the computations of singularities and regular initial starting points for each intersection curve component. We stay close to Kim's configuration space approach [46] and since our higher level algorithms are based on his results, we omit to reinvestigate the categorisation of topologically different types of intersection curves. Instead we concentrate on the feasibility of all necessary computations with exact arithmetic. We will see that this is possible with some additional effort, gaining exact and simply representable results.

Furthermore we assume that algorithms to compute conic section are available, see Chapter 3. We denote them by `ComputeProfileCircles`, `ComputeCrossSectionalCircles` and `ComputeVillarceauCircles`, where a torus and a simple surface are given as parameters and a set of circles, which may be empty, is returned.

Algorithm 15 `TorusPointIntersection`$(T_{r,R}(\mathbf{o},\mathbf{e}_3),\mathbf{p})$

Requires: A torus $T_{r,R}(\mathbf{o},\mathbf{e}_3)$ with algebraic function $f_T(\mathbf{x})$ and a point $\mathbf{p}$

Returns: $\begin{cases} -1, & \mathbf{p} \text{ lies inside } T \\ 0, & \mathbf{p} \text{ lies on } T \\ 1, & \mathbf{p} \text{ lies outside } T \end{cases}$

1: **return** `sign`$(f_T(\mathbf{p}))$

Algorithm 16 `TorusLineIntersection`$(T_{r,R}(\mathbf{o},\mathbf{e}_3),L(\mathbf{p},\mathbf{d}))$

Requires: A torus $T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a line $L = L(\mathbf{p},\mathbf{d})$
Returns: A set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

1: $a \leftarrow \|\mathbf{p}\|^2$
2: $b \leftarrow \|\mathbf{d}\|^2$
3: $c \leftarrow \langle \mathbf{p},\mathbf{d} \rangle$
4: $d \leftarrow a + R^2 - r^2$
5: $c_4 \leftarrow b^2$
6: $c_3 \leftarrow 4bc$
7: $c_2 \leftarrow 2bd + 4c^2 - 4R^2(d_x^2 + d_y^2)$
8: $c_1 \leftarrow 4cd - 8R^2(p_x d_x + p_y d_y)$
9: $c_0 \leftarrow d^2 - 4R^2(p_x^2 + p_y^2)$
10: $\{t_s\},\{t_d\} \leftarrow$ `SolveQuartic`$(c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0)$
11: **for all** $t \in \{t_s\}$ **do**
12: $\quad \{\mathbf{q}_r\} \leftarrow$ `IntersectionPoint`$(t,\mathfrak{P}_L)$
13: **end for**
14: **for all** $t \in \{t_d\}$ **do**
15: $\quad \{\mathbf{q}_s\} \leftarrow$ `IntersectionPoint`$(t,\mathfrak{P}_L)$
16: **end for**
17: **return** $\{\mathbf{q}_r\},\{\mathbf{q}_s\}$

Algorithm 17 `TorusCircleIntersection`$(T_{r,R}(\mathbf{o},\mathbf{e}_3),C_\beta(\mathbf{p},\mathbf{n}))$

Requires: A torus $T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a circle $C = C_\beta(\mathbf{p},\mathbf{n})$
Returns: A set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

1: $\alpha \leftarrow \|\mathbf{p}\|^2$
2: $\gamma \leftarrow \langle \mathbf{p}, M_\mathbf{n}\mathbf{e}_1 \rangle$
3: $\delta \leftarrow \langle \mathbf{p}, M_\mathbf{n}\mathbf{e}_2 \rangle$
4: $\epsilon \leftarrow a + \beta^2 + R^2 - r^2$
5: $a \leftarrow 4\gamma - 4R^2(a_x^2 + a_y^2)$
6: $b \leftarrow 4\delta - 4R^2(b_x^2 + b_y^2)$
7: $f \leftarrow 4\gamma\delta - 4R^2(a_x b_x + a_y b_y)$
8: $l \leftarrow 2\gamma\epsilon - 4R^2(p_x a_x + p_y a_y)$
9: $m \leftarrow 2\delta\epsilon - 4R^2(p_x b_x + p_y b_y)$
10: $d \leftarrow \epsilon^2 - 4R^2(p_x^2 + p_y^2)$
11: $E \leftarrow E(\mathbf{p},M_\mathbf{n})$ with $E = \begin{pmatrix} a & f & l \\ f & b & m \\ l & m & d \end{pmatrix}$
12: $\{\mathbf{q}_r\},\{\mathbf{q}_s\} \leftarrow$ `CircleConicIntersection`(C,E)
13: **return** $\{\mathbf{q}_r\},\{\mathbf{q}_s\}$

4.4.1 Torus-Plane Intersection

Given a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and a plane $P = P(\mathbf{p}, \mathbf{n})$. Following Kim's C-space approach we consider the torus as the envelope surface of a moving ball along a circular trajectory and the plane as an obstacle. The trajectory is precisely the main circle $C_R = C_R(\mathbf{o}, \mathbf{e}_3)$ of the torus T. The obstacle is bounded by two offsets P^O and P^I of the plane P, where

$$P^O = P(\mathbf{p} + r\mathbf{n}, \mathbf{n}) \quad \text{and} \quad P^I = P(\mathbf{p} - r\mathbf{n}, \mathbf{n}),$$

i.e. the obstacle becomes $P^O_- \cap P^I_+$, where P_- and P_+ denotes the negative and positive halfspace of P, respectively, see Section 2.2.

The C-space approach prescribes the following:

1. Intersect the trajectory C_R with the obstacle $P^O_- \cap P^I_+$.
2. Each connected component in C-space corresponds to a connected intersection curve component in affine space.
3. For each connected circle arc C in $C_R \cap P^O_- \cap P^I_+$ do:
   a. Determine a representative point $\mathbf{p}$ on C (prefer tangential intersection points).
   b. Intersect the plane P with the cross-sectional circle C_r of T with centre $\mathbf{p}$.
   c. Save the intersection points $\mathbf{q}_i$ of $P \cap C_r$ as initial starting points for the corresponding intersection curve component.

In step 3.a we prefer tangential intersection points $\mathbf{p}$ since in that case $\mathbf{q}_i$ are singular points and the corresponding intersection curve components, which can be traced from the points $\mathbf{q}_i$, are singular as well. For more details we refer to [46].

In case of a regular circle arc C without any tangential intersection point we need to determine a point $\mathbf{p}$ on C such that we can guarantee that the cross-sectional circle C_r with centre $\mathbf{p}$ intersects with the plane P. So let $C(t)$ be a regular circle arc of C_R with a rational parametrisation and end points $\mathbf{a} = C(t_1)$ and $\mathbf{b} = C(t_2)$. Then the balls $B_r(\mathbf{a})$ and $B_r(\mathbf{b})$ touch the plane P in two points $\mathbf{a}'$ and $\mathbf{b}'$ which are precisely the projections of $\mathbf{a}$ and $\mathbf{b}$ onto the plane P under the direction of the normal $\mathbf{n}$. From Kim [46] we know that the intersection curve component γ, corresponding to $C(t)$, consists of a closed loop. Points on this loop result from the intersection of cross-sectional circles between $\mathbf{a}$ and $\mathbf{b}$ with the plane P, i.e.

$$\gamma = \bigcup_{\hat{t}_1 \trianglelefteq t \trianglelefteq \hat{t}_2} \left\{ C_r(C(t), \mathbf{n}_{C(t)}) \cap P \right\} \quad \text{with} \quad t_1 \lhd \hat{t}_1 \lhd \hat{t}_2 \lhd t_2 \quad \text{and} \quad \mathbf{n}_{C(t)} = \frac{1}{R}(C(t) \times \mathbf{e}_3).$$

The relation $t_1 \lhd t_2$ means that the circle arc starts with point $C(t_1)$, continues in counter clockwise direction and ends with point $C(t_2)$, even if $t_1 > t_2$. In this case the circle arc passes the point $C(\infty)$ where a sign change in the parameter happens.

Since $\mathbf{a}'$ and $\mathbf{b}'$ lie inside the area of P, surrounded by the intersection curve γ, there are t'_1 and t'_2 such that

- $t_1 \lhd \hat{t}_1 \lhd t'_1 \lhd t'_2 \lhd \hat{t}_2 \lhd t_2$,
- $\mathbf{a}' \in P(C(t'_1), \mathbf{n}_{C(t'_1)})$,
- $\mathbf{b}' \in P(C(t'_2), \mathbf{n}_{C(t'_2)})$.

This means we can choose any arbitrary t_0 with $t'_1 \lhd t_0 \lhd t'_2$, and we can be sure that the cross-sectional circle $C_r(C(t_0), \mathbf{n}_{C(t_0)})$ will intersect with the plane P at two different intersection points. The problem is to determine the parameters t'_1 and t'_2.

A point $\mathbf{x} \in \mathbb{V}(\mathbb{K})^3$ on the xy-plane can be written as

$$\begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \sqrt{x^2 + y^2} \begin{pmatrix} \frac{1-t^2}{1+t^2} \\ \frac{2t}{1+t^2} \\ 0 \end{pmatrix}.$$

If we want to solve for the parameter t, we must consider two cases:

- $y = 0$: Depending on the sign of x we conclude

$$t_0 = \begin{cases} 0, & x > 0 \\ \infty, & x < 0. \end{cases}$$

  Technically we represent infinity by setting a flag connected to a variable. Working with homogeneous parameters $\mathfrak{t} = (t, w)$, this is equivalent to $w = 0$.

- $y \neq 0$: The explicit solution becomes

$$t_0 = \frac{-x + \sqrt{x^2 + y^2}}{y},$$

  which corresponds to a root of the polynomial $f(t) = t^2 + 2\frac{x}{y}t - 1$. From the constant term of f we follow, that f has always one positive and one negative root. Geometrically this means that the points on a circle, represented by these roots, lie on the same line going through the origin. Depending on the sign of y we choose the right root.

Although we have an exact representation of the parameter t, we prefer the representation as algebraic real due to generality. We will see later that end points of a circle arc may be defined by either parameters of more complex number types or elements of vector spaces over algebraically extended fields. Nevertheless we want to choose a rational number inside an interval bounded by more complex number types to simplify further computations. Algebraic reals are the most general number type for this occasion. Since this is a quasi inverse tangent computation, we name the predicate to compute the circle parameter to a given point `atan`, see Algorithm 18.

Algorithm 18 `atan`($\mathbf{x}$)

Requires: $\mathbf{x} = (x, y, z) \in \mathbb{V}(\mathbb{K})^3$ with $x^2 + y^2 \neq 0$
Returns: The corresponding circle parameter $t \in \overline{\mathbb{K}} \cup \{\infty\}$

```
1: s_x ← sign(x)
2: s_y ← sign(y)
3: if s_y = 0 then
4:    if s_x = 1 then
5:       return 0
6:    else
7:       return ∞
8:    end if
9: else // s_y ≠ 0
10:   if s_x = 0 then
11:      return s_y
12:   end if
13:   [t_1, t_2] ← RootIsolator(t^2 + 2(x/y)t − 1) // t_1 < t_2
14:   if s_y = 1 then
15:      return t_2
16:   else
17:      return t_1
18:   end if
19: end if
```

It will also happen that the point $\mathbf{x}$ consists of a more complex number type, e.g. one-root numbers. Let us denote the first two components of $\mathbf{x}$ by $x = x_0 + x_1\sqrt{c}$ and $y = y_0 + y_1\sqrt{c}$ with some $c \in \mathbb{K}^+$. In this case we represent the parameter t by a suitable root of the quartic polynomial

$$f(t) = t^4 + 4\frac{x_0y_0 - x_1y_1c}{y_0^2 - y_1^2c}t^3 + [4\frac{x_0^2 - x_1^2c}{y_0^2 - y_1^2c} - 2]t^2 - 4\frac{x_0y_0 - x_1y_1c}{y_0^2 - y_1^2c}t + 1.$$

The roots of f have the form

$$\frac{-x+\sqrt{x^2+y^2}}{y}, \frac{-x-\sqrt{x^2+y^2}}{y}, \frac{-x_0+x_1\sqrt{c}+\sqrt{x^2+y^2}}{y_0-y_1\sqrt{c}}, \frac{-x_0+x_1\sqrt{c}-\sqrt{x^2+y^2}}{y_0-y_1\sqrt{c}},$$

where the product of the first pair of roots as well as the product of the latter two roots equals to -1. So again we have pairs of roots representing two points in opposite direction, and we follow that we always have two positive and two negative roots. Depending on the sign of y we distinguish them, since in case of $y > 0$ only the positive roots are relevant, in case of $y < 0$ analogously the negative ones. Solving f with a `RootIsolator` we get four algebraic reals t_1, t_2, t_3, t_4 in increasing order. Each algebraic real t_i consists of the defining polynomial f and an enclosing interval $I_i = [l_i, u_i]$ for $i = 1, 2, 3, 4$. Furthermore we know that

$$t_1 \leq u_1 < \frac{u_1 + l_2}{2} < l_2 \leq t_2 < 0 < t_3 \leq u_3 < \frac{u_3 + l_4}{2} < l_4 \leq t_4.$$

In the case of $y > 0$ let $\mathbf{p} = (\cos(m), \sin(m), 0)$ with $m = \frac{u_3+l_4}{2}$. If $\frac{x}{y} < \frac{p_x}{p_y}$ then t_4 is the root corresponding to $\mathbf{x}$, otherwise it is t_3. In the case of $y < 0$ we compute $\mathbf{p} = (\cos(m), \sin(m), 0)$ with $m = \frac{u_1+l_2}{2}$ and take root t_2 if $\frac{x}{y} < \frac{p_x}{p_y}$, otherwise we take t_1. For a summary see Algorithm 19.

Let $t_i, i = 1, 2$, be two algebraic reals defined by polynomials f_i and enclosing intervals $I_i = [l_i, u_i]$. To determine a rational parameter t with $t_1 \lhd t \lhd t_2$ we first have to ensure that the intervals do not overlap. We refine both intervals until either $u_1 < l_2$ or $l_1 > u_2$. This is done in the `less` predicate, see Algorithm 2. In the latter case we simply set $t = \infty$. Otherwise we may set $t = \frac{u_1+l_2}{2}$, see Algorithm 20.

Our algorithm `TorusPlaneIntersection`, see Algorithm 22, is strictly based on [46]. Summarising the results we have the following case analysis for the intersection of the circle C_R with the offset surfaces P^O and P^I:

1. $C_R \subset P^O_- \cap P^I_+$, i.e. the circle C_R lies completely inside the obstacle:
   a. C_R and P are coplanar: The TPI curve consists of two profile circles (Figure 3.3(a)).
   b. Otherwise: The TPI curve consists of two regular closed loops. Starting points can be found by intersecting P with any arbitrary cross-sectional circle of T (Figure 4.5(a)).
2. C_R intersects tangentially with both P^O and P^I: The TPI curve consists of two Villarceau circles (Figure 3.3(c)).
3. C_R intersects tangentially with either P^O or P^I: The TPI curve is a singular curve with one singular point $\mathbf{q}_s$ (Figure 4.5(b)).
4. C_R intersects with either P^O or P^I at two regular points, i.e. there is one circular arc C inside the obstacle: The TPI curve consists of one regular closed loop. Starting points can be found by intersecting P with a suitable cross-sectional circle of T, which lies on C (Figure 4.5(c)).
5. C_R intersects with both P^O and P^I at two regular points each, i.e. there are two circular arcs C_1 and C_2 inside the obstacle:
   a. P splits T vertically in exact two halves: The TPI curve consists of two cross-sectional circles (Figure 3.3(b)).
   b. Otherwise: The TPI curve consists of two regular closed loops. Starting points can be found by intersecting P with any arbitrary profile circle of T (Figure 4.5(d)).
6. $C_R \subset P^O_+ \cup P^I_-$, i.e. the circle C_R lies completely outside the obstacle: There is no intersection.

Consider case 3 and let C_R intersect tangentially with P^O at a point $\mathbf{q}$. To determine the singular point $\mathbf{q}_s$ we make use of the following lemma, which generalises tangential intersections between arbitrary objects:

Definition 1. *Given a differentiable space curve* $\mathbf{a} : \mathbb{R} \to \mathbb{R}^3$ *and a radius* r*. Then we call* $F = \partial(\bigcup B_r(\mathbf{a}(t)))$ *the envelope surface of a moving ball with radius* r *along the space curve* $\mathbf{a}(t)$*. We call further* $\mathbf{a}(t)$ *the trajectory of* F *and* $C_r(\mathbf{p}, \mathbf{n})$ *with* $\mathbf{p} = \mathbf{a}(t_0)$ *and* $\mathbf{n} \times \dot{\mathbf{a}}(t_0) = 0$ *the cross-sectional circle of* F *at point* $\mathbf{p}$*.*

Lemma 1. *a) Given a surface* F *with trajectory* $\mathbf{a}(t)$ *and another surface* G *with an offset surface* O*, which evolves by offsetting surface* G *about a given distance* r*. Let* $\mathbf{a}(t_0)$ *be an intersection point between the trajectory* $\mathbf{a}(t)$ *of* F *and the offset surface* O *of* G*. Then the ball* $B_r(\mathbf{a}(t_0))$ *has tangential intersections* $\mathbf{q}$ *with surface* G*.*

b) If $\mathbf{a}(t_0)$ *is a tangential intersection, all tangential intersections* $\mathbf{q}$ *with surface* G *lie on the cross-sectional circle* C *of* F *at point* $\mathbf{a}(t_0)$*.*

Proof. a) Since $\mathbf{a}(t_0) \in O$ there must be a $\mathbf{q} \in G$ with $\|\mathbf{q} - \mathbf{a}(t_0)\| = r$ and $\nabla G|_{\mathbf{q}} \times (\mathbf{q} - \mathbf{a}(t_0)) = 0$ ($\mathbf{q}$ is the orthogonal projection of $\mathbf{a}(t_0)$ onto G). Since $\nabla B_r|_{\mathbf{p}} \times (\mathbf{p} - \mathbf{a}(t_0)) = 0$ and $\|\mathbf{p} - \mathbf{a}(t_0)\| = r$ for all $\mathbf{p} \in B_r$ we have $\mathbf{q} \in B_r \cap G$ with $\nabla G|_{\mathbf{q}} \times \nabla B_r|_{\mathbf{q}} = 0$.

b) Since $\mathbf{q}$ is the orthogonal projection of $\mathbf{a}(t_0)$ onto G and O is the offset surface of G we have $\nabla G|_{\mathbf{q}} \times \nabla O|_{\mathbf{a}(t_0)} = 0$. Since $\mathbf{a}(t_0)$ is a tangential intersection we have

$$0 = \langle \dot{\mathbf{a}}(t_0), \nabla O|_{\mathbf{a}(t_0)} \rangle = \langle \dot{\mathbf{a}}(t_0), \nabla G|_{\mathbf{q}} \rangle = \langle \dot{\mathbf{a}}(t_0), \nabla B_r|_{\mathbf{q}} \rangle = \langle \dot{\mathbf{a}}(t_0), \mathbf{q} - \mathbf{a}(t_0) \rangle.$$

So $\mathbf{q}$ lies in the normal plane of $\mathbf{a}(t_0)$ and so we have $\mathbf{q} \in C$. □

Now we can directly follow, that the singular point $\mathbf{q}_s$ is the orthogonal projection of the tangential intersection point $\mathbf{q}$ onto the plane P. Depending on the offset surface involved, it computes to

$$\mathbf{q}_s = \begin{cases} \mathbf{q} - r\mathbf{n}, & \mathbf{q} \in C_R \cap P^O \\ \mathbf{q} + r\mathbf{n}, & \mathbf{q} \in C_R \cap P^I \end{cases} .$$

To shorten the algorithm `TorusPlaneIntersection` we combine a plane-plane intersection with a circle-line intersection to a new algorithm `CirclePlaneIntersection`.

Algorithm 19 `atan`(**x**)

Requires: $\mathbf{x} = (x, y, z) \in \mathbb{V}(\mathbb{K}(\sqrt{c}))^3$ with $x^2 + y^2 \neq 0$ and $x = x_0 + x_1\sqrt{c}, y = y_0 + y_1\sqrt{c}$
Returns: The corresponding circle parameter $t \in \overline{\mathbb{K}} \cup \{\infty\}$

$s_x \leftarrow \texttt{sign}(x)$
$s_y \leftarrow \texttt{sign}(y)$
if $s_y = 0$ **then**
  if $s_x = 1$ **then**
    return 0
  else
    return ∞
  end if
else // $s_y \neq 0$
  if $s_x = 0$ **then**
    return s_y
  end if
  $a \leftarrow \frac{x}{y}$ // $a = a_0 + a_1\sqrt{c}$
  if $\texttt{sign}(a_1) = 0$ **then**
    return $\texttt{atan}(s_x|a_0|, s_y)$
  end if
  $b \leftarrow a_0^2 - a_1^2 r$
  if $b = 0$ **then**
    return $\texttt{atan}(2s_x|a_0|, s_y)$
  end if
  $[t_1, t_2, t_3, t_4] \leftarrow \texttt{RootIsolator}(t^4 + 4a_0bt^3 + (4b - 2)t^2 - 4a_0bt + 1)$ // $t_1 < t_2 < t_3 < t_4$
  if $s_y = 1$ **then**
    $m \leftarrow \frac{u_3 + l_4}{2}$
    if $\texttt{less}(a, \frac{1-m^2}{2m})$ **then**
      return t_4
    else
      return t_3
    end if
  else // $s_y = -1$
    $m \leftarrow \frac{u_1 + l_2}{2}$
    if $\texttt{less}(a, \frac{1-m^2}{2m})$ **then**
      return t_2
    else
      return t_1
    end if
  end if
end if

Algorithm 20 `RationalBetween`(t_1, t_2)

Requires: Two algebraic reals $t_1, t_2 \in \overline{\mathbb{K}} \cup \{\infty\}$ with $t_1 \lhd t_2$
Returns: A rational number $t \in \mathbb{K} \cup \{\infty\}$ with $t_1 \lhd t \lhd t_2$

if $t_1 = \infty$ **then**
  if $t_2 = \infty$ **then**
    return ∞ // $t_1 = \infty = t_2$
  else // $t_2 \neq \infty$
    if $\texttt{lower}(t_2) > 0$ **then**
      return 0 // $t_1 = \infty \lhd 0 < \texttt{lower}(t_2) \leq t_2$
    else
      return $\texttt{lower}(t_2) - 1$ // $t_1 = \infty \lhd \texttt{lower}(t_2) - 1 < \texttt{lower}(t_2) \leq t_2$
    end if
  end if
else // $t_1 \neq \infty$
  if $t_2 = \infty$ **then**
    if $\texttt{upper}(t_1) < 0$ **then**
      return 0 // $t_1 \leq \texttt{upper}(t_1) < 0 < \infty = t_2$
    else
      return $\texttt{upper}(t_1) + 1$ // $t_1 \leq \texttt{upper}(t_1) < \texttt{upper}(t_1) + 1 < \infty = t_2$
    end if
  else // $t_2 \neq \infty$
    if $\texttt{less}(t_1, t_2)$ **then**
      return $\frac{\texttt{upper}(t_1)+\texttt{lower}(t_2)}{2}$ // $t_1 \leq \texttt{upper}(t_1) < \frac{\texttt{upper}(t_1)+\texttt{lower}(t_2)}{2} < \texttt{lower}(t_2) \leq t_2$
    else
      return ∞ // $t_1 \lhd \infty \lhd t_2$
    end if
  end if
end if

Algorithm 21 `CirclePlaneIntersection`$(C_\beta(\mathbf{p}_1, \mathbf{n}_1), P(\mathbf{p}_2, \mathbf{n}_2))$

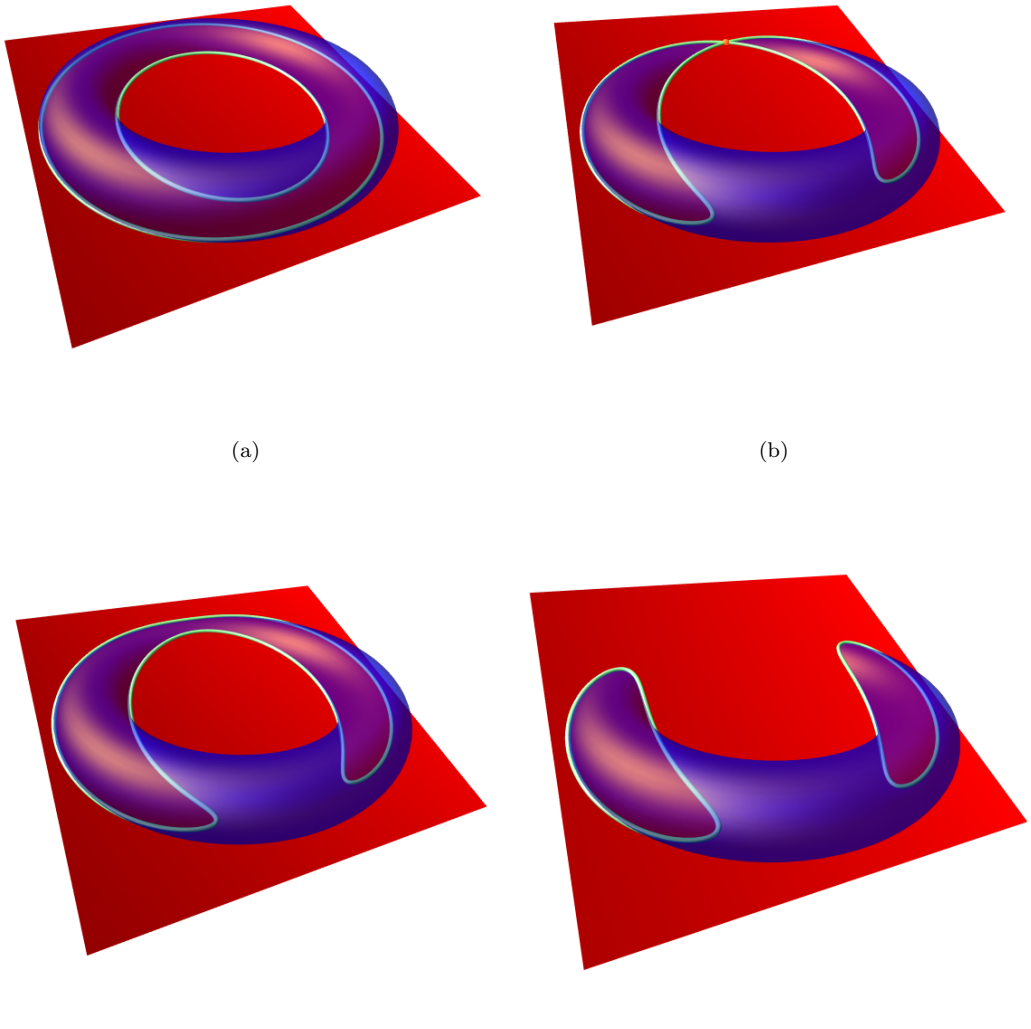
Requires: A circle $C = C_\beta(\mathbf{p}_1, \mathbf{n}_1)$ and a plane $P = P(\mathbf{p}_2, \mathbf{n}_2)$
Returns: A set of intersection points $\{\mathbf{q}\}$

if $\mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{o}$ **then**
  $L \leftarrow \texttt{PlanePlaneIntersection}(P(\mathbf{p}_1, \mathbf{n}_1), P)$
  return $\texttt{CircleLineIntersection}(C, L)$
end if

Algorithm 22 `TorusPlaneIntersection`$(T_{r,R}(\mathbf{o},\mathbf{e}_3), P(\mathbf{p},\mathbf{n}))$

Requires: A torus $T = T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a plane $P = P(\mathbf{p},\mathbf{n})$
Returns: A set of conic sections $\{C\}$, a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

1: **if** $\mathbf{e}_3 \times \mathbf{n} = \mathbf{o}$ **then**
2: **if** $|p_z| \leq r$ **then** // Figure 3.3(a)
3: **return** `ComputeProfileCircles`(T, P)
4: **end if**
5: **else**
6: $C_R \leftarrow C_R(\mathbf{o},\mathbf{e}_3)$
7: $\{\mathbf{q}^O\} \leftarrow$ `CirclePlaneIntersection`$(C_R, P(\mathbf{p}+r\mathbf{n},\mathbf{n}))$
8: $\{\mathbf{q}^I\} \leftarrow$ `CirclePlaneIntersection`$(C_R, P(\mathbf{p}-r\mathbf{n},\mathbf{n}))$
9: **if** $|\{\mathbf{q}^O\}| = 0$ **and** $|\{\mathbf{q}^I\}| = 0$ **then**
10: **if** $\langle \mathbf{p}+r\mathbf{n},\mathbf{n}\rangle > 0$ **and** $\langle \mathbf{p}-r\mathbf{n},\mathbf{n}\rangle < 0$ **then** // Figure 4.5(a)
11: $\{\mathbf{q}_r\} \leftarrow$ `CirclePlaneIntersection`$(C_r(R\mathbf{e}_1,\mathbf{e}_2), P)$
12: **end if**
13: **else if** $|\{\mathbf{q}^O\}| = 1$ **and** $|\{\mathbf{q}^I\}| = 1$ **then** // Figure 3.3(c)
14: $\{C\} \leftarrow$ `ComputeVillarceauCircles`(T, P)
15: $\{\mathbf{q}_s\} \leftarrow \{\mathbf{q}_1 - r\mathbf{n}, \mathbf{q}_2 + r\mathbf{n}\},\ \mathbf{q}_1 \in \{\mathbf{q}^O\},\ \mathbf{q}_2 \in \{\mathbf{q}^I\}$
16: **return** $\{C\}, \{\mathbf{q}_s\}$
17: **else if** $|\{\mathbf{q}^O\}| = 1$ **then** // Figure 4.5(b)
18: $\{\mathbf{q}_s\} \leftarrow \mathbf{q} - r\mathbf{n},\ \mathbf{q} \in \{\mathbf{q}^O\}$
19: **else if** $|\{\mathbf{q}^I\}| = 1$ **then**
20: $\{\mathbf{q}_s\} \leftarrow \mathbf{q} + r\mathbf{n},\ \mathbf{q} \in \{\mathbf{q}^I\}$
21: **else if** $|\{\mathbf{q}^O\} \cup \{\mathbf{q}^I\}| = 2$ **then** // Figure 4.5(c)
22: $\hat{\mathbf{q}}_i \leftarrow \mathbf{q}_i - \langle \mathbf{q}_i - \mathbf{p},\mathbf{n}\rangle \mathbf{n},\ \mathbf{q}_i \in \{\mathbf{q}^O\} \cup \{\mathbf{q}^I\},\ i = 1,2$
23: $\mathbf{q} \leftarrow C_R($`RationalBetween`$($`atan`$(\hat{\mathbf{q}}_1),$ `atan`$(\hat{\mathbf{q}}_2)))$
24: $\{\mathbf{q}_r\} \leftarrow$ `CirclePlaneIntersection`$(C_r(R\mathbf{q}, \mathbf{e}_3 \times \mathbf{q}), P)$
25: **else**
26: **if** $\langle \mathbf{p},\mathbf{n}\rangle = 0$ **and** $n_z = 0$ **then** // Figure 3.3(b)
27: **return** `ComputeCrossSectionalCircles`(T, P)
28: **else** // Figure 4.5(d)
29: $\{\mathbf{q}_r\} \leftarrow$ `CirclePlaneIntersection`$(C_{R+r}(\mathbf{o},\mathbf{e}_3), P)$
30: **end if**
31: **end if**
32: **end if**
33: **return** $\{\mathbf{q}_r\}, \{\mathbf{q}_s\}$



(a) (b)

(c) (d)

Figure 4.5: Torus-plane intersections

4.4.2 Torus-Sphere Intersection

Given a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and a sphere $S = S_\delta(\mathbf{p})$. In case of $\delta \leq r$ we build the obstacle from the torus T, bounded by two offset surfaces $T^O = T_{r+\delta,R}(\mathbf{o}, \mathbf{e}_3)$ and $T^I = T_{r-\delta,R}(\mathbf{o}, \mathbf{e}_3)$. The trajectory of the sphere S consists of the point $\mathbf{p}$. Note that, if $\delta = r$, it happens that T^I degenerates into the main circle $C_R = C_R(\mathbf{o}, \mathbf{e}_3)$ of T. In case of $\delta > r$ we consider the torus T as envelope surface of a moving ball with radius r along a trajectory, which consists of the main circle C_R. The sphere S becomes an obstacle, bounded by two offset surfaces $S^O = S_{\delta+r}(\mathbf{p})$ and $S^I = S_{\delta-r}(\mathbf{p})$. In the following we will handle both cases separately, resulting in two algorithms `TorusSphereIntersectionI` and `TorusSphereIntersectionII`.

Case $\delta \leq r$

Although we just consider ring tori in our algorithms, i.e. $r < R$, the offset surface T^O may become a horn torus, when $r + \delta = R$ or even a spindle torus, when $r + \delta > R$. The case analysis becomes more difficult due to the self intersection part of T^O. To stay consistent to Kim's thesis we name this self intersection part T^D. In addition we denote the vertices of T^D by $\mathbf{q}^\pm = (0, 0, z^\pm)$ with $z^\pm = \pm\sqrt{(r+\delta)^2 - R^2}$. In case of $r+\delta = R$ the vertices degenerates into one single point $\mathbf{q}^\pm = \mathbf{o}$. To distinguish the part T^D from T^O itself, we notice that a point $\mathbf{p}$ lies on T^D if and only if $\mathbf{p}$ lies on T^O and $\|\mathbf{p}\| \leq z^+$. Let $S^D = S_{z^+}(\mathbf{o})$ be the sphere enclosing the part T^D. Then we can write

$$\mathbf{p} \in T^D \Leftrightarrow \mathbf{p} \in T^O \cap (S^D \cup S^D_-) \Leftrightarrow f_{T^O}(\mathbf{p}) = 0 \wedge f_{S^D}(\mathbf{p}) \leq 0.$$

In this way we determine if $\mathbf{p}$ is one of the vertices $\mathbf{q}^\pm$, namely if $f_{T^O}(\mathbf{p}) = 0$ and $f_{S^D}(\mathbf{p}) = 0$.

We define T^D_- for the interior region of T^D and T^D_+ for the exterior region. To specify a condition for the case $\mathbf{p} \in T^D_-$, we notice that the sign of the algebraic function f_{T^O} changes, passing T^D. So inside the self intersecting part of a spindle torus, the algebraic function is positive. Therefore we specify a point $\mathbf{p} \in T^D_-$ by $f_{T^O}(\mathbf{p}) > 0$ and $f_{S^D}(\mathbf{p}) < 0$.

Figure 4.6 shows the partition of a spindle torus into separate regions. Table 4.1 lists all conditions to specify the corresponding region for a given point $\mathbf{x}$. We may use these results even for ring tori and horn tori if we admit the regions T^D, T^D_- and $\{\mathbf{q}^\pm\}$ to be empty.

In the following we summarise Kim's results for this case:

1. $\mathbf{p}$ is a vertex of T^D: The TSI curve consists of one profile circle (Figure 4.9(a)).
2. $\mathbf{p} \in T^O$: The TSI curve consists of a single touching point $\mathbf{q}_s$ (Figure 4.9(b)).
3. $\mathbf{p} \in T^D$: The TSI curve consists of a singular curve with a singular point $\mathbf{q}_s$ (Figure 4.9(c)).
4. $\mathbf{p} \in T^I$:
   a. $\delta = r$: The TSI curve consists of a cross-sectional circle (Figure 4.9(d)).
   b. Otherwise: The TSI curve consists of a single touching point $\mathbf{q}_s$ (Figure 4.9(e)).
5. $\mathbf{p} \in T^O_- \cap T^I_+ \cap T^D_+$: The TSI curve consists of one regular closed loop. Starting points can be found by intersecting S with a suitable cross-sectional circle of T (Figure 4.9(f)).
6. $\mathbf{p} \in T^D_-$:
   a. $\mathbf{p}$ lies on the z-axis: The TSI curve consists of two profile circles (Figure 3.4(a)).
   b. Otherwise: The TSI curve consists of two regular closed loops. Starting points can be found by intersecting S with any arbitrary cross-sectional circle of T (Figure 4.9(g)).
7. $\mathbf{p} \in T^O_+ \cup T^I_-$: There is no intersection.

Consider case 1. In Figure 4.7(a) we see that all conditions for profile circles are fulfilled, since $p_x = p_y = 0$ and $p_z^2 + R^2 = (\delta + r)^2$, see Section 3.3.1. Since we have equality in the latter condition,

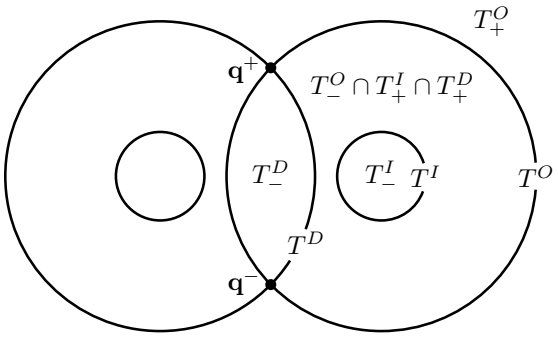


Figure 4.6: Regions of a spindle torus (cross-section)

region	condition
T^I	$f_{T^I}(\mathbf{x}) = 0$
T^D	$f_{T^O}(\mathbf{x}) = 0 \wedge f_{S^D}(\mathbf{x}) < 0$
T^O	$f_{T^O}(\mathbf{x}) = 0 \wedge f_{S^D}(\mathbf{x}) > 0$
$\{q^\pm\}$	$f_{T^O}(\mathbf{x}) = 0 \wedge f_{S^D}(\mathbf{x}) = 0$
T_-^I	$f_{T^I}(\mathbf{x}) < 0$
T_+^O	$f_{T^O}(\mathbf{x}) > 0 \wedge f_{S^D}(\mathbf{x}) > 0$
T_-^D	$f_{T^O}(\mathbf{x}) > 0 \wedge f_{S^D}(\mathbf{x}) < 0$
$T_-^O \cap T_+^I \cap T_+^D$	$f_{T^O}(\mathbf{x}) < 0 \wedge f_{T^I}(\mathbf{x}) > 0$

Table 4.1: Classification of regions of a spindle torus

the two resulting profile circles coincide to a singular one, where all points of this profile circle are touching points. We represent this circle by

$$C_\beta(\mathbf{q}, \mathbf{e}_3) \quad \text{with} \quad \beta = \frac{R\delta}{\delta + r} \quad \text{and} \quad \mathbf{q} = \frac{r}{\delta + r}\mathbf{p}.$$

To compute the touching point $\mathbf{q}_s$ in case 2 consider Figure 4.7(b) where we can see that

$$\mathbf{q}_s = \frac{1}{r+\delta}[r\mathbf{p} + \delta\mathbf{a}] \quad \text{with} \quad \mathbf{a} = \frac{R}{\sqrt{p_x^2 + p_y^2}} \begin{pmatrix} p_x \\ p_y \\ 0 \end{pmatrix}.$$

Since $f_{T^O}(\mathbf{p}) = 0$ we further substitute

$$2R\sqrt{p_x^2 + p_y^2} = \|\mathbf{p}\|^2 + R^2 - (r+\delta)^2 = f_{S^D}(\mathbf{p}) > 0.$$

So we get for the point $\mathbf{a}$

$$\mathbf{a} = \frac{2R^2}{f_{S^D}(\mathbf{p})} \begin{pmatrix} p_x \\ p_y \\ 0 \end{pmatrix},$$

which even has rational components. Fortunately there is no difference to case 3, since we keep the right sign using $f_{S^D}(\mathbf{p})$. In case 4.b, however, the touching point $\mathbf{q}_s$ computes to

$$\mathbf{q}_s = \frac{1}{r-\delta}[r\mathbf{p} - \delta\mathbf{a}] \quad \text{with} \quad \mathbf{a} = \frac{2R^2}{\|\mathbf{p}\|^2 + R^2 - (r-\delta)^2} \begin{pmatrix} p_x \\ p_y \\ 0 \end{pmatrix}.$$

In case 5 we get a regular closed loop as intersection curve, which is symmetric to the plane P containing the z-axis and the point $\mathbf{p}$. Therefore the cross-sectional circle of T, which is embedded in that plane and nearest to S, intersects with S. Its centre $\mathbf{q}$ is that point on the main circle C_R, which is nearest to $\mathbf{p}$, namely

$$\mathbf{q} = \frac{R}{\sqrt{p_x^2 + p_y^2}}(\mathbf{e}_3 \times \mathbf{p}) \times \mathbf{e}_3.$$

The normal of the plane P computes then to

$$\mathbf{n}_\mathbf{q} = \frac{1}{R}(\mathbf{q} \times \mathbf{e}_3) = \frac{1}{\sqrt{p_x^2 + p_y^2}}(\mathbf{e}_3 \times \mathbf{p}).$$

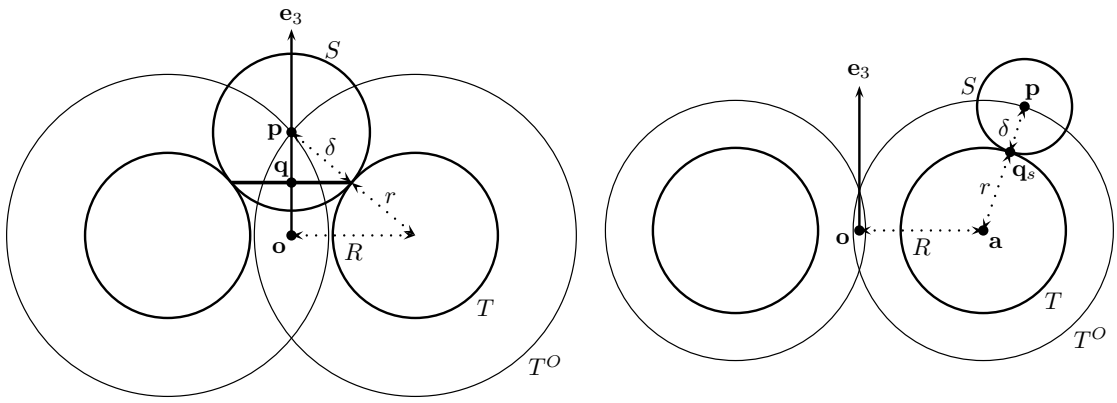


Figure 4.7: Tangential intersections between torus and sphere (cross-section)

So the suitable cross-sectional circle, which will be intersected with S, is $C_r(\mathbf{q}, \mathbf{n_q})$. Note that all computations from this point on will take place in an algebraic extension of the field $\mathbb{K}$, namely $\mathbb{K}(\sqrt{p_x^2 + p_y^2})$. However, due to generic programming we can use the same algorithms for this intersection.

Case $\delta > r$

Let $C_R = C_R(\mathbf{o}, \mathbf{e}_3)$ be the main circle and simultaneously the trajectory of the torus T. Let further $S^O = S_{\delta+r}(\mathbf{p})$ and $S^I = S_{\delta-r}(\mathbf{p})$ be the offset surfaces of the sphere S. These surfaces partition the space into five regions: The exterior region S_+^O, the outer bound of the obstacle S^O, the obstacle itself $S_-^O \cap S_+^I$, the inner bound S^I and the interior region S_-^I. Depending on the relative position of the torus T, we may have the following types of intersection curves:

1. C_R is embedded in S^O: The TSI curve consists of one profile circle (Figure 4.10(a)).
2. C_R is embedded in S^I: The TSI curve consists of one profile circle (Figure 4.10(b)).
3. $C_R \subset S_-^O \cap S_+^I$, i.e. the circle C_R lies completely inside the obstacle:
   a. $p_x^2 + p_y^2 = 0$: The TSI curve consists of two profile circles (Figure 3.4(a)).
   b. Otherwise: The TSI curve consists of two regular closed loops. Starting points can be found by intersecting S with any arbitrary cross-sectional circle of T (Figure 4.10(c)).
4. C_R intersects tangentially with both S^O and S^I: The TSI curve consists of two Villarceau circles (Figure 4.10(d)+3.4(c)).
5. C_R intersects tangentially with either S^O or S^I: The TSI curve is a singular curve with one singular point $\mathbf{q}_s$ (Figure 4.10(e)+(f)).
6. C_R intersects with either S^O or S^I at two regular points, i.e. there is one circular arc C inside the obstacle: The TSI curve consists of one regular closed loop. Starting points can be found by intersecting S with a suitable cross-sectional circle of T, which lies on C (Figure 4.10(g)).
7. C_R intersects with both S^O and S^I at two regular points each, i.e. there are two circular arcs C_1 and C_2 inside the obstacle:
   a. $\mathbf{p}$ lies in the main plane of T and fulfils the condition $\|\mathbf{p}\|^2 = R^2 + r^2 - \delta^2$: The TSI curve consists of two cross-sectional circles (Figure 3.4(b)).

b. Otherwise: The TSI curve consists of two regular closed loops. Starting points can be found by intersecting S with any arbitrary profile circle of T (Figure 4.10(h)).

8. $C_R \subset S_+^O \cup S_-^I$, i.e. the circle C_R lies completely outside the obstacle: There is no intersection.

In case 1 we have the same constellation as in the former case 1. Again the conditions for profile circles are fulfilled and we can compute the resulting profile circle in the same way, see Figure 4.7a+4.8. Case 2 is very similar. Here the profile circle is represented by

$$C_\beta(\mathbf{q}, \mathbf{e}_3) \quad \text{with} \quad \beta = \frac{R\delta}{\delta - r} \quad \text{and} \quad \mathbf{q} = \frac{r}{\delta - r}\mathbf{p}.$$

In case 5 we have a tangential intersection point $\mathbf{q}$ between the main circle C_R and one of the offset surfaces of S. From Lemma 1 we know, that in this case we have a singular point $\mathbf{q}_s$ between torus T and sphere S, namely the orthogonal projection of $\mathbf{q}$ onto the surface S. Thus we compute $\mathbf{q}_s$ to

$$\mathbf{q}_s = \begin{cases} \frac{1}{\delta + r}[\delta\mathbf{q} + r\mathbf{p}], & \mathbf{q} \in S^O \\ \frac{1}{\delta - r}[\delta\mathbf{q} - r\mathbf{p}], & \mathbf{q} \in S^I \end{cases}.$$

As before we get in case 6 an intersection curve symmetric to the plane containing the z-axis and the point $\mathbf{p}$. However, the centre $\mathbf{q}$ of the cross-sectional circle is not necessarily the nearest point to $\mathbf{p}$, since the radius δ may be large enough, such that we should prefer the cross-sectional circle on the opposite site. To decide which cross-sectional circle we should choose, we check if the point $\mathbf{q}$ lies inside the obstacle, i.e. $\mathbf{q} \in S_-^O \cap S_+^I$. In this case it should hold

$$f_{S^O}(\mathbf{q}) f_{S^I}(\mathbf{q}) < 0.$$

Otherwise we take the opposite point, which is simply $-\mathbf{q}$. Note again that the computations afterwards take place in an algebraic extension field.

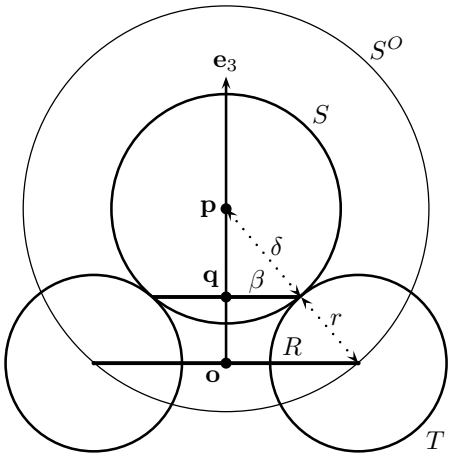


Figure 4.8: Tangential intersections between torus and sphere (cross-section)

Algorithm 23 `CircleSphereIntersection`$(C_\beta(\mathbf{p}_1,\mathbf{n}), S_\delta(\mathbf{p}_2))$

Requires: A circle $C = C_\beta(\mathbf{p}_1,\mathbf{n})$ and a sphere $S = S_\delta(\mathbf{p}_2)$
Returns: A set of intersection points $\{\mathbf{q}\}$

if $\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}\rangle \le \delta$ **then**
  $C' \leftarrow$ `SpherePlaneIntersection`$(S, P(\mathbf{p}_1,\mathbf{n}))$
  return `CircleCircleIntersection`(C, C')
end if

Algorithm 24 `TorusSphereIntersectionI`$(T_{r,R}(\mathbf{o},\mathbf{e}_3), S_\delta(\mathbf{p}))$

Requires: A torus $T = T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a sphere $S = S_\delta(\mathbf{p})$ with $\delta \le r$
Returns: A set of conic sections $\{C\}$, a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

$s_O \leftarrow$ `TorusPointIntersection`$(T_{r+\delta}(\mathbf{o},\mathbf{e}_3), \mathbf{p})$
$s_I \leftarrow$ `TorusPointIntersection`$(T_{r-\delta}(\mathbf{o},\mathbf{e}_3), \mathbf{p})$
$a \leftarrow \|\mathbf{p}\|^2 + R^2 - (r+\delta)^2$
$s_D \leftarrow$ `sign`(a)
if $s_O = 0$ **then**
  if $s_D = 0$ **then** // Figure 4.9(a)
    return $\{C_{\frac{R\delta}{r+\delta}}(\frac{r}{r+\delta}\mathbf{p}, \mathbf{e}_3)\}$
  else // Figure 4.9(b)+(c)
    $\{\mathbf{q}_s\} \leftarrow \frac{1}{r+\delta}\big[r\mathbf{p} + \frac{2\delta R^2}{a}(\mathbf{e}_3 \times \mathbf{p}) \times \mathbf{e}_3\big]$
  end if
else if $s_I = 0$ **then**
  if $\delta = r$ **then** // Figure 4.9(d)
    return $\{C_r(\mathbf{p}, \frac{1}{R}(\mathbf{p} \times \mathbf{e}_3))\}$
  else // Figure 4.9(e)
    $\{\mathbf{q}_s\} \leftarrow \frac{1}{r-\delta}\big[r\mathbf{p} - \frac{2\delta R^2}{a+4r\delta}(\mathbf{e}_3 \times \mathbf{p}) \times \mathbf{e}_3\big]$
  end if
else if $s_O < 0$ **and** $s_I > 0$ **then** // Figure 4.9(f)
  $\mathbf{n_q} \leftarrow \frac{1}{\sqrt{p_x^2+p_y^2}}(\mathbf{e}_3 \times \mathbf{p})$
  $\mathbf{q} \leftarrow R(\mathbf{n_q} \times \mathbf{e}_3)$
  $\{\mathbf{q}_r\} \leftarrow$ `CircleSphereIntersection`$(C_r(\mathbf{q}, \mathbf{n_q}), S)$
else if $s_O > 0$ **and** $s_D < 0$ **then**
  if $p_x^2 + p_y^2 = 0$ **then** // Figure 3.4(a)
    return `ComputeProfileCircles`(T, S)
  else // Figure 4.9(g)
    $\{\mathbf{q}_r\} \leftarrow$ `CircleSphereIntersection`$(C_r(R\mathbf{e}_1, \mathbf{e}_2), S)$
  end if
end if
return $\{\mathbf{q}_r\}, \{\mathbf{q}_s\}$

Algorithm 25: `TorusSphereIntersectionII`$(T_{r,R}(\mathbf{o},\mathbf{e}_3), S_\delta(\mathbf{p}))$

Requires: A torus $T = T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a sphere $S = S_\delta(\mathbf{p})$ with $\delta < r$
Returns: A set of conic sections $\{C\}$, a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

if $|p_z| \le \delta + r$ **then**
  $C^O \leftarrow$ `SpherePlaneIntersection`$(S_{\delta+r}(\mathbf{p}), P(\mathbf{o},\mathbf{e}_3))$ // $C^O = C_\beta((p_x, p_y, 0), \mathbf{e}_3)$
  if $p_x^2 + p_y^2 > (\beta + R)^2$ **then** // $C_R \subset S_+^O$
    return

else if $C^O = C_R$ **then** // Figure 4.10(a)
  return $\{C_{\frac{R\delta}{\delta+r}}(\frac{r}{r+\delta}\mathbf{p}, \mathbf{e}_3)\}$
else
  $\{\mathbf{q}^O\} \leftarrow$ `CircleCircleIntersection`(C^O, C_R)
end if
if $|p_z| \leq \delta - r$ **then**
  $C^I \leftarrow$ `SpherePlaneIntersection`$(S_{\delta-r}(\mathbf{p}), P(\mathbf{o}, \mathbf{e}_3))$ // $C^I = C_\gamma((p_x, p_y, 0), \mathbf{e}_3)$
  if $R < \gamma$ **and** $p_x^2 + p_y^2 < (\gamma - R)^2$ **then** // $C_R \subset S_-^I$
    return
  else if $C^I = C_R$ **then** // Figure 4.10(b)
    return $\{C_{\frac{R\delta}{\delta-r}}(\frac{r}{r-\delta}\mathbf{p}, \mathbf{e}_3)\}$
  else
    $\{\mathbf{q}^I\} \leftarrow$ `CircleCircleIntersection`(C^I, C_R)
  end if
end if
if $|\{\mathbf{q}^O\} \cup \{\mathbf{q}^I\}| = 0$ **then**
  if $R \leq \beta$ **then**
    if $p_x^2 + p_y^2 = 0$ **then** // Figure 3.4(a)
      return `ComputeProfileCircles`(T, S)
    else // Figure 4.10(c)
      $\{\mathbf{q}_r\} \leftarrow$ `CircleSphereIntersection`$(C_r(R\mathbf{e}_1, \mathbf{e}_2), S)$
    end if
  end if
else if $|\{\mathbf{q}^O\}| = 1$ **and** $|\{\mathbf{q}^I\}| = 1$ **then** // Figure 4.10(d)+(c)
  $\{C\} \leftarrow$ `ComputeVillarceauCircles`(T, S)
  $\{\mathbf{q}_s\} \leftarrow \{\frac{1}{\delta+r}(\delta\mathbf{q}_1 + r\mathbf{p}), \frac{1}{\delta-r}(\delta\mathbf{q}_2 - r\mathbf{p})\},\ \mathbf{q}_1 \in \{\mathbf{q}^O\},\ \mathbf{q}_2 \in \{\mathbf{q}^I\}$
  return $\{C\}, \{\mathbf{q}_s\}$
else if $|\{\mathbf{q}^O\}| = 1$ **then** // Figure 4.10(e)
  $\{\mathbf{q}_s\} \leftarrow \frac{1}{\delta+r}(\delta\mathbf{q} + r\mathbf{p}),\ \mathbf{q} \in \{\mathbf{q}^O\}$
else if $|\{\mathbf{q}^I\}| = 1$ **then** // Figure 4.10(f)
  $\{\mathbf{q}_s\} \leftarrow \frac{1}{\delta-r}(\delta\mathbf{q} - r\mathbf{p}),\ \mathbf{q} \in \{\mathbf{q}^I\}$
else if $|\{\mathbf{q}^O\} \cup \{\mathbf{q}^I\}| = 2$ **then** // Figure 4.10(g)
  $\mathbf{n_q} \leftarrow \frac{1}{\sqrt{p_x^2+p_y^2}}(\mathbf{e}_3 \times \mathbf{p})$
  $\mathbf{q} \leftarrow R(\mathbf{n_q} \times \mathbf{e}_3)$
  if $f_{S^O}(\mathbf{q}) f_{S^I}(\mathbf{q}) \geq 0$ **then**
    $\mathbf{q} \leftarrow -\mathbf{q}$
  end if
  $\{\mathbf{q}_r\} \leftarrow$ `CircleSphereIntersection`$(C_r(\mathbf{q}, \mathbf{n_q}), S)$
else
  if $p_z = 0$ **and** $p_x^2 + p_y^2 = R^2 + r^2 - \delta^2$ **then** // Figure 3.4(b)
    return `ComputeCrossSectionalCircles`(T, S)
  else // Figure 4.10(h)
    $\{\mathbf{q}_r\} \leftarrow$ `CircleSphereIntersection`$(C_{R+r}(\mathbf{o}, \mathbf{e}_3), S)$
  end if
end if
end if
return $\{\mathbf{q}_r\}, \{\mathbf{q}_s\}$

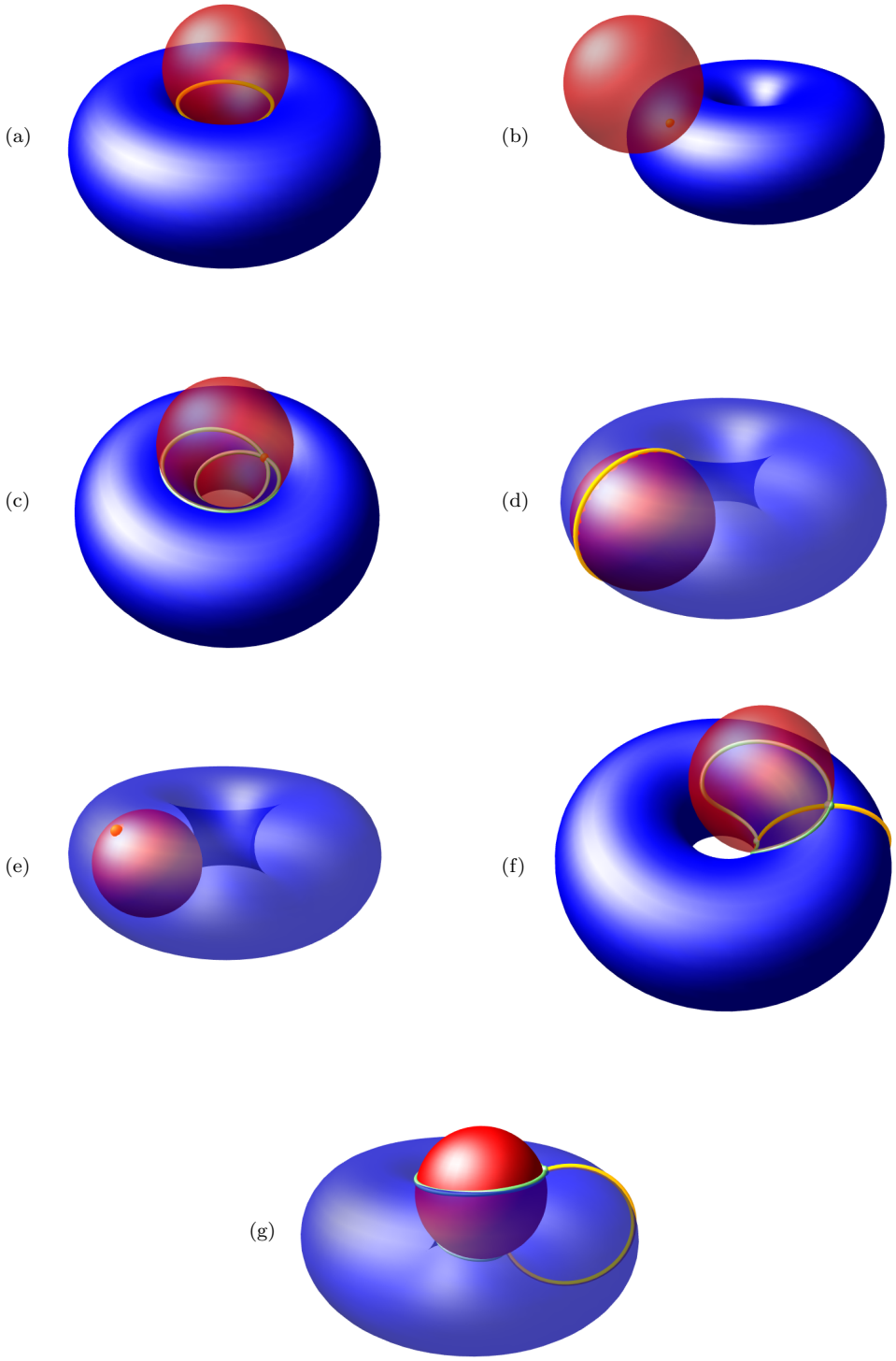


Figure 4.9: Torus-sphere intersections ($\delta \leq r$)

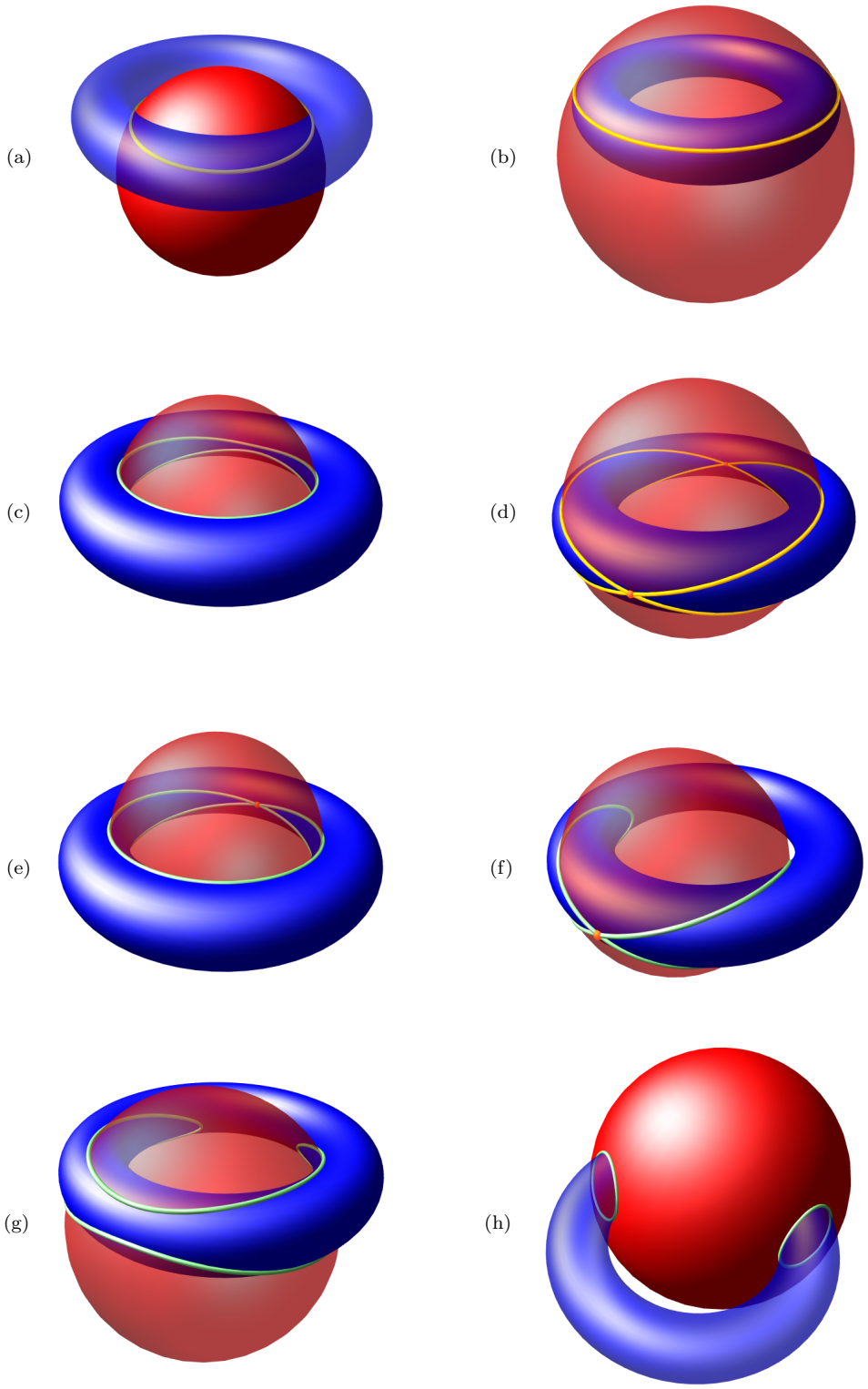


Figure 4.10: Torus-sphere intersections ($\delta < r$)

4.4.3 Torus-Cylinder Intersection

Given a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and a cylinder $Y = Y_\delta(\mathbf{p}, \mathbf{n})$. In case of $\delta \geq r$ we consider the torus T as envelope surface of a moving ball with radius r along a trajectory, which consists of the main circle $C_R = C_R(\mathbf{o}, \mathbf{e}_3)$. The cylinder Y becomes an obstacle, bounded by two offset surfaces $Y^O = Y_{\delta+r}(\mathbf{p}, \mathbf{n})$ and $Y^I = Y_{\delta-r}(\mathbf{p}, \mathbf{n})$. Note that, if $\delta = r$, it happens that Y^I degenerates into the line $L = L(\mathbf{p}, \mathbf{n})$. In case of $\delta < r$ we build the obstacle from the torus T, bounded by two offset surfaces $T^O = T_{r+\delta,R}(\mathbf{o}, \mathbf{e}_3)$ and $T^I = T_{r-\delta,R}(\mathbf{o}, \mathbf{e}_3)$. The trajectory of the cylinder Y consists of the line L. In the following we will treat both cases separately, resulting in two algorithms `TorusCylinderIntersectionI` and `TorusCylinderIntersectionII`.

Case $\delta \geq r$

The offset surfaces Y^O and Y^I partition the space into five regions: The exterior region Y^O_+, the outer bound of the obstacle Y^O, the obstacle itself $Y^O_- \cap Y^I_+$, the inner bound Y^I and the interior region Y^I_-. Denote the intersections between the trajectory C_R and the closed obstacle $\overline{Y^O_- \cap Y^I_+} = (Y^O_- \cap Y^I_+) \cup Y^O \cup Y^I$ by circle arcs C_i of the circle C_R. A circle arc C_i has tangential intersections with the obstacle, if C_R has tangential intersections with either Y^O or Y^I or both. In this case we call C_i a singular circle arc, otherwise we call it regular. In case of $\delta = r$ the intersections with Y^I, which becomes geometrically the line L, have always higher multiplicity, since we keep on treating Y^I as algebraic surface of degree two. Thus the intersections with Y^I counts twice, i.e. if a circle arc C_i has k intersections with L, not necessarily tangential, then the corresponding intersection curve component consists of a singular curve with $2k$ singular points.

We summarise Kim's results [46] for torus-cylinder intersections:

1. $C_R \subset Y^O_- \cap Y^I_+$, i.e. the circle C_R lies completely inside the obstacle:
   a. $p_x = p_y = n_x = n_y = 0$: The TYI curve consists of up to two profile circles (Figure 3.5(a)).
   b. Otherwise: The TYI curve consists of two regular closed loops. Starting points can be found by intersecting Y with any arbitrary cross-sectional circle of T (Figures 4.13(a)+(b)).
2. $C_R \cap \overline{Y^O_- \cap Y^I_+}$ decomposes into circle arcs C_i:
   a. $\delta < r$ and C_i has k tangential intersections with $Y^O \cup Y^I$: The corresponding TYI curve component consists of a singular curve with k singular points (Figures 4.13(c)+(d)).
   b. $\delta = r$ and C_i has k intersections with Y^I and m tangential intersections with Y^O: The corresponding TYI curve component consists of a singular curve with $2k + m$ singular points (Figure 4.13(e)).
   c. C_i has regular intersections with $Y^O \cup Y^I$: The corresponding TYI curve component consists of one regular closed loop. Starting points can be found by intersecting T with a suitable profile line of Y (Figures 4.13(f)-(h)).
3. $C_R \subset Y^O_+ \cup Y^I_-$: There is no intersection.

In our algorithms we will detect any conic sections in advance, so it suffices to consider singular and regular circle arcs inside the obstacle. The singular arcs are easy to handle, since we just need to compute the singular points on the corresponding intersection curve component, which become starting points for the tracing step. To find these singular points, consider Lemma 1 from Section 4.4.1. Let $\mathbf{q}$ be a tangential intersection point between the circle arc C_i and the obstacle $\overline{Y^O_- \cap Y^I_+}$. Let further be $C_r = C_r(\mathbf{q}, \mathbf{n_q})$ the cross-sectional circle of T at the point $\mathbf{q}$. Then the lemma states that the singular intersection points $\mathbf{q}_s$ between T and Y, which lie on the intersection curve component corresponding to C_i, are precisely the tangential intersections between C_r and Y. So for each tangential intersection point on the trajectory C_R we intersect the respective cross-sectional circle with the cylinder Y and gain the set of all singular intersection points between T and Y. From this we can directly follow a statement about the complexity of singularities.

Lemma 2. *A singularity* $\mathbf{q}_s$ *of the intersection curve between a torus and a cylinder can be represented by two-root numbers, i.e.*

$$\mathbf{q}_s \in \mathbb{V}(\mathbb{K}(\sqrt{\alpha})(\sqrt{\beta}))^3 \quad \textit{with} \quad \alpha \in \mathbb{K}, \beta \in \mathbb{K}(\sqrt{\alpha}).$$

Proof. Intersecting a circle with a torus results in solving a quartic polynomial. According to the description from above, this has to be done twice, where the roots of the first polynomial effects the coefficients of the second one. Since singular points computes both times to roots with higher multiplicity, these roots can be expressed by one-root numbers. So we have to extend the original field twice by square root expressions, where the second root is an element of the first field extension. □

In Sections 4.4.1 and 4.4.2 we have already seen, that singularities in TPI and TSI curves are even rational. The general procedure to compute singular points in TKI and TTI curves is similar to this section. Thus we can widen the statement of Lemma 2 onto singularities of any torus-simple surface intersection.

We now concentrate on regular circle arcs. First of all we have to identify such arcs, since all we know are the regular and tangential intersections of the trajectory C_R with the offset surfaces Y^O and Y^I. Therefore we first sort the intersection points along the trajectory, consider pairs of adjacent regular points and check, whether an arbitrary point between two points lies inside or outside the obstacle. In details:

1. Intersect the main plane $P = P(\mathbf{o}, \mathbf{e}_3)$ of the torus T with the offset surfaces Y^O and Y^I of the cylinder Y, resulting in two conics E^O and E^I.

2. Intersect the trajectory C_R of T with both conics E^O and E^I, resulting in a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$. For each point keep the corresponding offset surface in mind.

3. Sort all intersection points along the trajectory according to their parameter t in increasing order (remember that all points of circle-conic intersections are represented by the `IntersectionPoint` data type, see Section 2.4.3). Keep in mind whether the points are singular or regular.

4. For each pair of adjacent regular points $\mathbf{q}_1$ and $\mathbf{q}_2$ with respective circle parameters t_1 and t_2 and common homogeneous transformation $\mathfrak{P}$ (since both points belong to the same circle) do:

   a. If $\mathbf{q}_1$ and $\mathbf{q}_2$ come from intersections with different surfaces, then the circle arc, bounded by $\mathbf{q}_1$ and $\mathbf{q}_2$, is regular and lies inside the obstacle.

   b. Otherwise compute the midpoint $\mathbf{q}_m = \texttt{dehom}(\mathfrak{P}(\frac{\texttt{upper}(t_1)+\texttt{lower}(t_2)}{2}))$.

   c. Check if $\mathbf{q}_m$ lies inside the obstacle, i.e. $f_{Y^O}(\mathbf{q}_m) < 0$ and $f_{Y^I}(\mathbf{q}_m) > 0$.

   d. If $\mathbf{q}_m$ passes the test, then the circle arc, bounded by $\mathbf{q}_1$ and $\mathbf{q}_2$, is regular and lies inside the obstacle.

Note that in step 3 we need the functionality of comparing parameters in form of algebraic reals and one-root numbers. The algebraic reals come from regular intersection points, the one-root numbers come from singular ones. Comparisons among themselves are provided, see Section 2.4.1. To compare an algebraic real α with an one-root number β, we first refine α until β lies outside the isolating interval of α. Then we compare β with the boundary of the isolating interval, which are rational numbers.

Once we found all regular circle arcs inside the obstacle, we need to compute a regular starting point on each corresponding intersection curve component. Therefor we consider the projection of the whole scene onto the main plane $P = P(\mathbf{p}, \mathbf{n})$ of the cylinder Y, such that Y becomes a circle $C_\delta = C_\delta(\mathbf{p}, \mathbf{n})$ and the trajectory C_R of the torus T becomes an ellipse E, see Figure 4.11.

Let $C(t)$ be a regular circle arc of C_R with endpoints $\mathbf{a}_1 = C(t_1)$ and $\mathbf{a}_2 = C(t_2)$. We assume $t_1 \lhd t_2$, i.e. the circle arc is oriented counter clockwise, seen in direction of the negative z-axis.

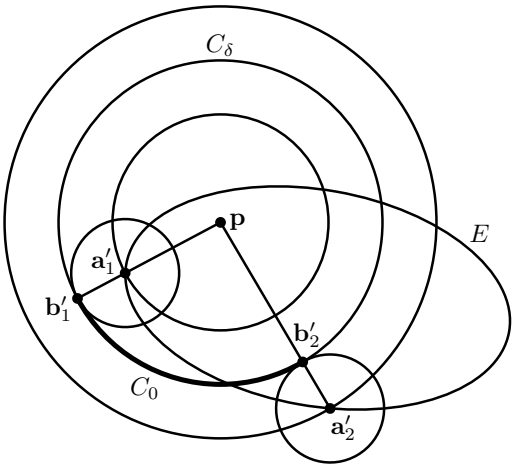


Figure 4.11: Projection onto the main plane of the cylinder

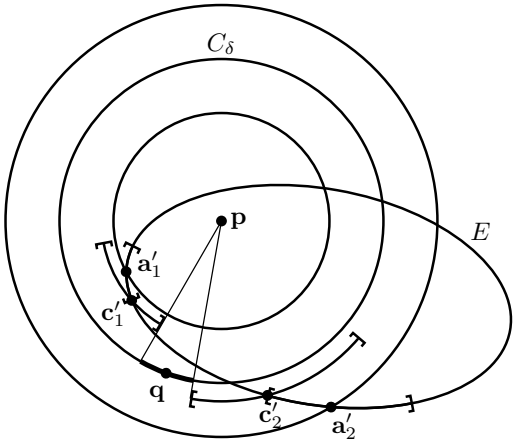


Figure 4.12: Finding a rational point on a circle arc

Then $C(t)'$ is the corresponding connected component of E with endpoints $\mathbf{a}'_1$ and $\mathbf{a}'_2$. As we saw in Lemma 1, the intersection point of a ball at an endpoint $\mathbf{a}_i,\ i = 1, 2$ with cylinder Y is precisely the orthogonal projection of $\mathbf{a}_i$ onto Y. Denote these points by $\mathbf{b}_i$ and the projection onto P by $\mathbf{b}'_i$. Then $\mathbf{p}$, $\mathbf{a}'_i$ and $\mathbf{b}'_i$ are collinear. Furthermore the balls $B_r(C(t))$ for $t_1 \lhd t \lhd t_2$ become disks on P and each disk $D_r(C(t)')$ intersects with C_δ. Since $C(t)'$ is continuous we have a connected circle arc

$$C_0 = \bigcup_{t_1 \lhd t \lhd t_2} (C_\delta \cap D_r(C(t)')) = C_\delta \cap (\bigcup_{t_1 \lhd t \lhd t_2} D_r(C(t)')).$$

So each profile line of Y, going through a point on C_0, intersects with T and we can use at least two of the resulting intersection points as initial starting points for the corresponding intersection curve component. In the tracing step we will automatically eliminate redundant regular starting points, thus we may add all intersection points to the set of regular starting points.

The last problem we need to solve is, that we cannot evaluate the endpoints $\mathbf{a}_1$ and $\mathbf{a}_2$ of the regular circle arc $C(t)$ exactly, since the respective parameters t_1 and t_2 are algebraic reals in general. By merging the intersections of both offset surfaces, we can not even guarantee that the respective enclosing intervals $I_1 = [l_1, u_1]$ and $I_2 = [l_2, u_2]$ do not overlap. Thus we first refine both intervals until they become disjoint. Then it holds $t_1 \trianglelefteq u_1 \lhd l_2 \trianglelefteq t_2$. Since u_1 and l_2 are rational we can now evaluate $\mathbf{c}_1 = C(u_1)$ and $\mathbf{c}_2 = C(l_2)$. Therewith we shortened the circle arc $C(t)$. Denote the projections of the new endpoints onto P by $\mathbf{c}'_i$ for $i = 1, 2$, then we can compute the respective parameters for the circle C_δ by $\hat{t}_i = \texttt{atan}(\mathbf{c}'_i)$, which are again algebraic reals, see Figure 4.12. Doing the same as before, i.e. isolating the enclosing intervals, computing a rational parameter $\hat{t}$ in between by the Algorithm `RationalBetween` and evaluating C_δ at this parameter, we finally gain a rational point $\mathbf{q}$, which lies on the circle arc C_0. Additionally we know, that a profile line of Y, going through $\mathbf{q}$, intersects at least twice with the corresponding intersection curve component.

Depending on the relative position of the projected circle E to the centre $\mathbf{p}$ of the cylinder's main circle C_δ, we have to consider four cases:

1. The ellipse E degenerates into a straight line through the origin, i.e. the points $\mathbf{p}$, $\mathbf{c}'_1$ and $\mathbf{c}'_2$ are collinear and we cannot generally find a rational parameter $\hat{t}$ between $\hat{t}_1$ and $\hat{t}_2$. This case should be intercepted in advance.

2. $\hat{t}_1 = \hat{t}_2$, i.e. $\mathbf{p}$, $\mathbf{c}'_1$ and $\mathbf{c}'_2$ are collinear again, but here we can choose any other rational point on the circle arc $C(t)$ as new endpoint, e.g. the midpoint $\mathbf{c}_m = C(\frac{u_1+l_2}{2})$, since in contrast to the first case, we know that $\hat{t}_m = \texttt{atan}(\mathbf{c}'_m) \neq \hat{t}_i,\ i = 1, 2$.

3. The point $\mathbf{p}$ lies inside E, i.e. we have always $\hat{t}_1 \neq \hat{t}_2$. If E and C_δ have the same orientation, we have $\hat{t}_1 \lhd \hat{t}_2$, otherwise we just swap $\hat{t}_1$ with $\hat{t}_2$.

4. The point $\mathbf{p}$ lies outside E and we have $\hat{t}_1 \neq \hat{t}_2$. Then we know that the angle between the endpoints of the circle arc C_0 is less than π. If $\langle \mathbf{n} \times (\mathbf{c}_1' - \mathbf{p}), \mathbf{c}_2' - \mathbf{p} \rangle > 0$ then we have $\hat{t}_1 \lhd \hat{t}_2$, otherwise we swap again.

Case 1 can be handled by intersecting the torus T with two profile lines of the cylinder Y, lying in the main plane of T. These lines split the intersection curve into two symmetric halves, one at the upper part on the torus and one at the lower part. The other cases are summarised in the Algorithm `RationalBetweenProjectedCircle`, which computes the rational parameter $\hat{t}$, discussed above.

Algorithm 28 computes conic sections and starting points for a torus-cylinder intersection. In the first part (up to line 13) we detect and compute conic sections. In the case of cross-sectional circles and Villarceau circles we additionally specify all singular points on the curve. Then we compute all intersections of the main circle C_R with the offset surfaces Y^O and Y^I. In the part from line 19 we handle the case where C_R lies completely inside the obstacle. Lines 25ff computes all singular points from the tangential intersections between C_R and the offset surfaces. In line 30 we construct all regular circle arcs inside the obstacle as described above. Due to the readability of this algorithm and the following ones we omit going into details.

The next part handles the case where the projection of the main circle C_R onto the main plane of the cylinder Y would result in a degenerate line through the origin, see above. The last part from line 34 computes starting points on regular intersection curve components. Therefor we transform the main circle C_R of the torus T into the local coordinate system of the cylinder Y. For each regular circle arc C_i with endpoints in form of circle parameters t_1, t_2 we find a rational point $\mathbf{q}$ on the main circle C_δ of the cylinder. Finally we intersect the profile line of Y, going through $\mathbf{q}$, with the torus T to compute the starting points on the corresponding intersection curve component. Although we may get more than one point, we save all points, since it would be too expensive to decide which point belongs to the intersection curve component. Anyway, redundant starting points will be detected and deleted during the tracing step later on.

Case $\delta < r$

In this case we consider the torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ as an obstacle, bounded by two offset surfaces $T^O = T_{r+\delta,R}(\mathbf{o}, \mathbf{e}_3)$ and $T^I = T_{r-\delta,R}(\mathbf{o}, \mathbf{e}_3)$. In case of a self intersection of T^O, i.e. when $r + \delta > R$, we denote the self intersection part by T^D, see Section 4.4.2. The trajectory of the cylinder $Y = Y_\delta(\mathbf{p}, \mathbf{n})$ consists of the line $L = L(\mathbf{p}, \mathbf{n})$. We have an intersection between T and Y, if and only if L intersects with the obstacle $\overline{T_-^O \cap T_+^I}$. Let $L_i = L \cap \overline{T_-^O \cap T_+^I}$ denote the connected line segments of L inside the obstacle. Depending on the position of each line segment L_i we may have the following cases:

1. L_i passes through $\mathbf{q}^+$ and $\mathbf{q}^-$: The TYI curve consists of up to two profile circles (Figure 3.5(a)).

2. L_i passes through neither $\mathbf{q}^+$ nor $\mathbf{q}^-$:

   a. $L_i \subset T_+^D$, i.e. the line segment L_i has no intersection with T^D:

      i. L_i has no tangential intersection with $T^O \cup T^I$: The corresponding TYI curve component consists of one regular closed loop. Starting points can be found by intersecting Y with a suitable cross-sectional circle of T (Figures 4.14(a)+(b)).

      ii. L_i has k tangential intersections with $T^O \cup T^I$: The corresponding TYI curve component consists of a singular curve with k singular points (Figure 4.14(c)).

   b. $L_i \cap T_-^D \neq \emptyset$, i.e. the line segment L_i intersects transversely with T^D:

      i. L_i has no tangential intersection with $T^O \cup T^I$: The corresponding TYI curve component consists of two regular closed loops. Starting points can be found by intersecting Y with any arbitrary cross-sectional circle of T (Figure 4.14(d)).

ii. L_i has k tangential intersections with T^I: The corresponding TYI curve component consists of two singular curves or one regular closed loop and a singular curve with k singular points. Starting points for the regular closed loop can be found by intersecting Y with any arbitrary cross-sectional circle of T (Figures 4.14(e)-(f)).

c. L_i intersects tangentially with T^D:

i. L_i has no tangential intersection with T^I: The corresponding TYI curve component consists of a singular curve with one singular point (Figure 4.14(h)).

ii. L_i has k tangential intersections with T^I: The corresponding TYI curve component consists of a singular curve with $k+1$ singular points (Figure 4.14(i)).

3. L_i passes through either $\mathbf{q}^+$ or $\mathbf{q}^-$:

a. $L_i \subset T^D_+$:

i. L_i has no tangential intersection with T^I: The corresponding TYI curve component consists of a singular curve with two singular points (Figure 4.14(j)).

ii. L_i has k tangential intersections with T^I: The corresponding TYI curve component consists of a singular curve with $k+2$ singular points.

b. $L_i \cap T^D_- \neq \emptyset$:

i. L_i has no tangential intersection with T^I: The corresponding TYI curve component consists of two regular closed loops. Starting points can be found by intersecting Y with any arbitrary cross-sectional circle of T (Figure 4.14(k)).

ii. L_i has k tangential intersections with T^I: The corresponding TYI curve component consists of one regular closed loop and a singular curve with k singular points. Starting points for the regular closed loop can be found by intersecting Y with any arbitrary cross-sectional circle of T.

c. L_i intersects tangentially with T^D:

i. L_i has no tangential intersection with T^I: The corresponding TYI curve component consists of a singular curve with one singular point (Figure 4.14(l)).

ii. L_i has k tangential intersections with T^I: The corresponding TYI curve component consists of a singular curve with $k+1$ singular points.

Since singular intersection curve components can be traced from their singular points and these points can be computed from the tangential intersections between the trajectory L and the offset surfaces T^O and T^I, see lines 10ff in Algorithm 29, we may concentrate on finding starting points on regular closed loops, see cases 2.a.i, 2.b and 3.b.

Consider a regular connected line segment L_i which has no intersection with the self intersecting part T^D of the torus T. Denote the endpoints of L_i by $\mathbf{a}_1 = L_i(t_1)$ and $\mathbf{a}_2 = L_i(t_2)$. We know that L_i corresponds to one regular closed loop in the intersection curve. Starting points can be found by intersecting Y with a suitable cross-sectional circle of T. Similar to the previous section we project the scene onto the main plane $P = P(\mathbf{o}, \mathbf{e}_3)$ of the torus T. The line L becomes a line again or it degenerates into a point, if L is orthogonal to P. The balls $B_\delta(\mathbf{a}_i)$ with $i = 1, 2$ intersect with T at two touching points $\mathbf{b}_i$, which are the projections of the endpoints $\mathbf{a}_i$ onto the surface T. Again it is clear, that each triple $\mathbf{o}, \mathbf{a}'_i$ and $\mathbf{b}'_i$ for $i = 1, 2$, where $\mathbf{a}'$ denotes the projection of the point $\mathbf{a}$ onto P, are collinear. Furthermore the balls $B_\delta(L_i(t))$ with $t_1 \leq t \leq t_2$ intersect with T. Thus we have a circle arc C_0 of the main circle C_R of the torus T, bounded by the endpoints $C_R(\texttt{atan}(\mathbf{a}_i))$, where each cross-sectional circle on this circle arc intersects with the cylinder Y. In the case that the line L crosses the z-axis or L is orthogonal to P, the circle arc C_0 degenerates into a single point $\mathbf{q}$, see line 23. Otherwise we choose $\mathbf{q}$ to be a rational point on C_0, depending on the orientation of the line L relative to the main circle C_R. The suitable cross-sectional circle becomes then $C_r(\mathbf{q}, \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q}))$. Note that in the degenerate case all further computations take place in an algebraically extended field.

Consider now a regular connected line segment L_i with $L_i \cap T^D_- \neq \emptyset$, see Figures 4.14(d)+(k). In this case we have $\delta > R - r$ and the cylinder Y fills the hole of the torus T. Thus we have

two regular closed loops in the intersection curve, corresponding to L_i, one at the upper part of T and one at the lower part. Since both loops surround the hole, we can choose any arbitrary cross-sectional circle of T to intersect with Y, and it will cross both loops, see lines 14ff.

If additionally L_i has tangential intersections with T^I, the regular closed loops become singular curves, see Figures 4.14(e)-(g).

Algorithm 26 `RationalBetweenProjectedCircle`$(C_\beta(\mathbf{p},\mathbf{n}), t_1, t_2)$

Requires: A circle $C = C_\beta(\mathbf{p},\mathbf{n})$ and two algebraic reals $t_1, t_2 \in \overline{\mathbb{K}} \cup \{\infty\}$ with $t_1 \lhd t_2$ and respective enclosing intervals $I_1 = [l_1, u_1]$ and $I_2 = [l_2, u_2]$

Returns: A rational number $t \in \mathbb{K} \cup \{\infty\}$ such that a ray from the origin with direction $(\cos(t), \sin(t), 0)$ intersects with the projected circle arc of C, bounded by t_1 and t_2

1: **while** $I_1 \cap I_2 \neq \emptyset$ **do**
2: `refine`(t_1)
3: `refine`(t_2)
4: **end while**
5: $\mathbf{c}_1' \leftarrow (\mathbf{e}_3 \times C(\texttt{upper}(t_1))) \times \mathbf{e}_3$
6: $\mathbf{c}_2' \leftarrow (\mathbf{e}_3 \times C(\texttt{lower}(t_2))) \times \mathbf{e}_3$
7: $\hat{t}_1 \leftarrow \texttt{atan}(\mathbf{c}_1')$
8: $\hat{t}_2 \leftarrow \texttt{atan}(\mathbf{c}_2')$
9: **if** $(p_x n_x + p_y n_y)^2 + n_z^2(p_x^2 + p_y^2) < \beta^2$ **then**
10: **if** $n_z > 0$ **then**
11: **return** `RationalBetween`$(\hat{t}_1, \hat{t}_2)$
12: **else**
13: **return** `RationalBetween`$(\hat{t}_2, \hat{t}_1)$
14: **end if**
15: **end if**
16: **if** $\hat{t}_1 = \hat{t}_2$ **then**
17: $\mathbf{c}_1' \leftarrow (\mathbf{e}_3 \times C(\frac{\texttt{upper}(t_1)+\texttt{lower}(t_2)}{2})) \times \mathbf{e}_3$
18: $\hat{t}_1 \leftarrow \texttt{atan}(\mathbf{c}_1')$
19: **end if**
20: **if** $\langle \mathbf{n} \times (\mathbf{c}_1' - \mathbf{p}), \mathbf{c}_2' - \mathbf{p} \rangle > 0$ **then**
21: **return** `RationalBetween`$(\hat{t}_1, \hat{t}_2)$
22: **else**
23: **return** `RationalBetween`$(\hat{t}_2, \hat{t}_1)$
24: **end if**

Algorithm 27 `CircleCylinderIntersection`$(C_\beta(\mathbf{p}_1,\mathbf{n}_1), Y_\delta(\mathbf{p}_2,\mathbf{n}_2))$

Requires: A circle $C = C_\beta(\mathbf{p}_1,\mathbf{n}_1)$ and a cylinder $Y = Y_\delta(\mathbf{p}_2,\mathbf{n}_2)$

Returns: A set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

1: **if** $\langle \mathbf{n}_1, \mathbf{n}_2 \rangle \neq 0$ **or** $\langle \mathbf{p}_2 - \mathbf{p}_1, \mathbf{n}_1 \rangle \leq \delta$ **then**
2: $E \leftarrow$ `CylinderPlaneIntersection`$(Y, P(\mathbf{p}_1, \mathbf{n}_1))$
3: **return** `CircleConicIntersection`(C, E)
4: **end if**

Algorithm 28 `TorusCylinderIntersectionI`$(T_{r,R}(\mathbf{o},\mathbf{e}_3), Y_\delta(\mathbf{p},\mathbf{n}))$

Requires: A torus $T = T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a cylinder $Y = Y_\delta(\mathbf{p},\mathbf{n})$ with $\delta \geq r$
Returns: A set of conic sections $\{C\}$, a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

if $p_x = p_y = n_x = n_y = 0$ **and** $|\delta - R| \leq r$ **then** // Figure 3.5(a)
  return `ComputeProfileCircles`(T, Y)
else if $p_z = n_z = 0$ **and** $\delta = r$ **and** $\|\mathbf{p}\|^2 = R^2 + \langle \mathbf{p},\mathbf{n}\rangle^2$ **then** // Figure 3.5(b)
  $\{C\} \leftarrow$ `ComputeCrossSectionalCircles`(T, Y)
  $\mathbf{q} \leftarrow \mathbf{p} - \langle \mathbf{p},\mathbf{n}\rangle\mathbf{n}$
  $\{\mathbf{q}_s\} \leftarrow \{\mathbf{q} + r\mathbf{e}_3, \mathbf{q} - r\mathbf{e}_3\}$
  return $\{C\}, \{\mathbf{q}_s\}$
else if $\delta = R$ **and** $R^2 n_z^2 = R^2 - r^2$ **and** $\|\mathbf{p} \times \mathbf{n}\|^2 = r^2$ **and** $R^2 p_z^2 = \langle \mathbf{p},\mathbf{n}\rangle^2 (R^2 - r^2)$ **then**
  $\{C\} \leftarrow$ `ComputeVillarceauCircles`(T, Y) // Figure 3.5(c)
  $\mathbf{q} \leftarrow \mathbf{p} - \langle \mathbf{p},\mathbf{n}\rangle\mathbf{n}$
  $\{\mathbf{q}_s\} \leftarrow \{\frac{r+R}{r}\mathbf{q}, \frac{r-R}{r}\mathbf{q}\}$
  return $\{C\}, \{\mathbf{q}_s\}$
else
  $Y^O \leftarrow Y_{\delta+r}(\mathbf{p},\mathbf{n})$
  $Y^I \leftarrow Y_{\delta-r}(\mathbf{p},\mathbf{n})$
  $C_R \leftarrow C_R(\mathbf{o},\mathbf{e}_3)$
  $\{\mathbf{p}_r\}, \{\mathbf{p}_s\} \leftarrow$ `CircleCylinderIntersection`(C_R, Y^O)
  $\{\mathbf{p}_r\}, \{\mathbf{p}_s\} \leftarrow$ `CircleCylinderIntersection`(C_R, Y^I)
  if $\{\mathbf{p}_r\} \cup \{\mathbf{p}_s\} = \emptyset$ **then**
    $\mathbf{q} \leftarrow R\mathbf{e}_1$
    if $f_{Y^O}(\mathbf{q}) f_{Y^I}(\mathbf{q}) < 0$ **then** // Figures 4.13(a)+(b)
      $\{\mathbf{q}_r\} \leftarrow$ `CircleCylinderIntersection`$(C_r(\mathbf{q},\mathbf{e}_2), Y)$
    end if
  else
    for all $\mathbf{p}_s =$ `IntersectionPoint`$(t, \mathfrak{P}) \in \{\mathbf{p}_s\}$ **do** // Figures 4.13(c)-(e)
      $\mathbf{q} \leftarrow$ `dehom`$(\mathfrak{P}(t))$
      $\mathbf{n}_\mathbf{q} \leftarrow \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q})$
      $\{\mathbf{q}_s\} \leftarrow$ `CircleCylinderIntersection`$(C_r(\mathbf{q}, \mathbf{n}_\mathbf{q}), Y)$
    end for
    $\{C_i\} \leftarrow C_R \cap \overline{Y_-^O \cap Y_+^I}$
    if $|\{C_i\}| > 0$ **and** $p_z = n_z = 0$ **then** // Figures 4.13(f)
      $\{\mathbf{q}_r\} \leftarrow$ `TorusLineIntersection`$(T, L(\mathbf{p} + \delta(\mathbf{n} \times \mathbf{e}_3), \mathbf{n}))$
      $\{\mathbf{q}_r\} \leftarrow$ `TorusLineIntersection`$(T, L(\mathbf{p} - \delta(\mathbf{n} \times \mathbf{e}_3), \mathbf{n}))$
    else // Figures 4.13(g)+(h)
      $C \leftarrow C_R(M_\mathbf{n}^{-1}(-\mathbf{p}), M_\mathbf{n}^{-1}\mathbf{n})$
      for all $C_i = [t_1, t_2] \in \{C_i\}$ **do** // t_1, t_2 regular intersections
        $\mathbf{q} \leftarrow C_\delta($`RationalBetweenProjectedCircle`$(C, t_1, t_2))$
        $\{\mathbf{q}_r\} \leftarrow$ `TorusLineIntersection`$(T, L(\mathbf{p} + M_\mathbf{n}\mathbf{q}, \mathbf{n}))$
      end for
    end if
  end if
end if
return $\{\mathbf{q}_r\}, \{\mathbf{q}_s\}$

Algorithm 29 `TorusCylinderIntersectionII`$(T_{r,R}(\mathbf{o},\mathbf{e}_3), Y_\delta(\mathbf{p},\mathbf{n}))$

Requires: A torus $T = T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a cylinder $Y = Y_\delta(\mathbf{p},\mathbf{n})$ with $\delta < r$
Returns: A set of conic sections $\{C\}$, a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

1: **if** $p_x = p_y = n_x = n_y = 0$ **and** $|\delta - R| \leq r$ **then** // Figure 3.5(a)
2: **return** `ComputeProfileCircles`(T, Y)
3: **else**
4: $T^O \leftarrow T_{r+\delta,R}(\mathbf{o},\mathbf{e}_3)$
5: $T^I \leftarrow T_{r-\delta,R}(\mathbf{o},\mathbf{e}_3)$
6: $L \leftarrow L(\mathbf{p},\mathbf{n})$
7: $\{\mathbf{p}_r\}, \{\mathbf{p}_s\} \leftarrow$ `TorusLineIntersection`(T^O, L)
8: $\{\mathbf{p}_r\}, \{\mathbf{p}_s\} \leftarrow$ `TorusLineIntersection`(T^I, L)
9: **if** $\{\mathbf{p}_r\} \cup \{\mathbf{p}_s\} \neq \emptyset$ **then**
10: **for all** $\mathbf{p}_s =$ `IntersectionPoint`$(t, \mathfrak{P}) \in \{\mathbf{p}_s\}$ **do** // Figures 4.14(c)+(e)-(j)+(l)
11: $\mathbf{q} \leftarrow$ `dehom`$(\mathfrak{P}(t))$
12: $\{\mathbf{q}_s\} \leftarrow$ `TorusCircleIntersection`$(T, C_\delta(\mathbf{q},\mathbf{n}))$
13: **end for**
14: **if** $L \cap T^D_- \neq \emptyset$ **then** // Figures 4.14(d)-(f)+(k)
15: $\{\mathbf{q}_r\} \leftarrow$ `CircleCylinderIntersection`$(C_r(R\mathbf{e}_1,\mathbf{e}_2), Y)$
16: **end if**
17: $\{L_i\} \leftarrow L \cap \overline{T^O_- \cap T^I_+ \cap T^D_+}$
18: **for all** $L_i = [t_1, t_2] \in \{L_i\}$ **do** // Figures 4.14(a)+(b)+(f)+(j)
19: $\mathbf{a} \leftarrow$ `dehom`$(\mathfrak{P}_L($`upper`$(t_1)))$
20: $\mathbf{b} \leftarrow$ `dehom`$(\mathfrak{P}_L($`lower`$(t_2)))$
21: $s \leftarrow$ `sign`$(\langle \mathbf{a} \times \mathbf{b}, \mathbf{e}_3 \rangle)$
22: **if** $s = 0$ **then**
23: $\mathbf{q} \leftarrow \frac{R}{\sqrt{a_x^2 + a_y^2}} (\mathbf{e}_3 \times \mathbf{a}) \times \mathbf{e}_3$
24: **else if** $s > 0$ **then**
25: $\mathbf{q} \leftarrow C_R($`RationalBetween`$($`atan`$(\mathbf{a}),$ `atan`$(\mathbf{b})))$
26: **else**
27: $\mathbf{q} \leftarrow C_R($`RationalBetween`$($`atan`$(\mathbf{b}),$ `atan`$(\mathbf{a})))$
28: **end if**
29: $\mathbf{n}_\mathbf{q} \leftarrow \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q})$
30: $\{\mathbf{q}_r\} \leftarrow$ `CircleCylinderIntersection`$(C_r(\mathbf{q}, \mathbf{n}_\mathbf{q}), Y)$
31: **end for**
32: **return** $\{\mathbf{q}_r\}, \{\mathbf{q}_s\}$
33: **end if**
34: **end if**

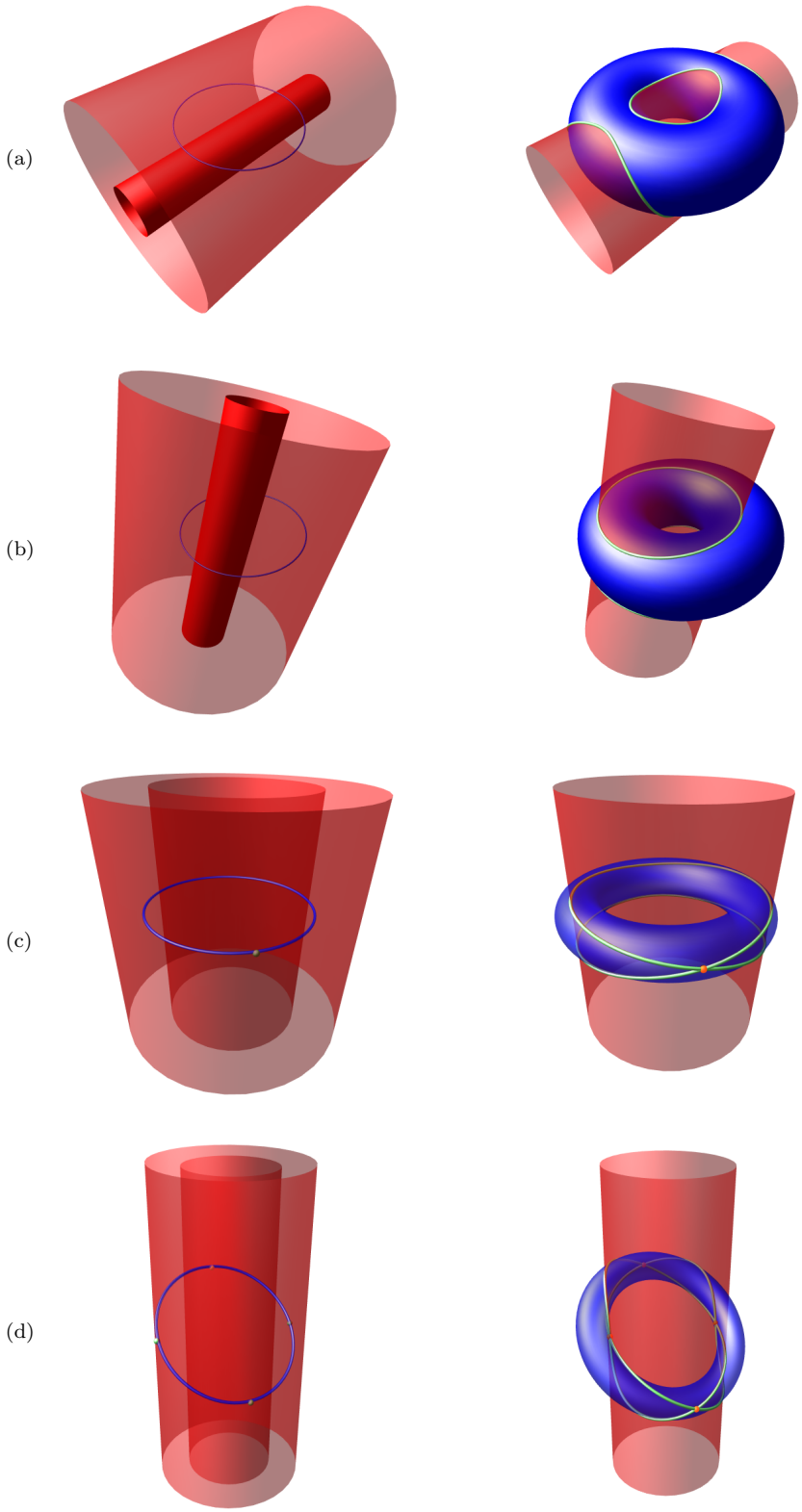


Figure 4.13: Torus-cylinder intersections ($\delta \geq r$)

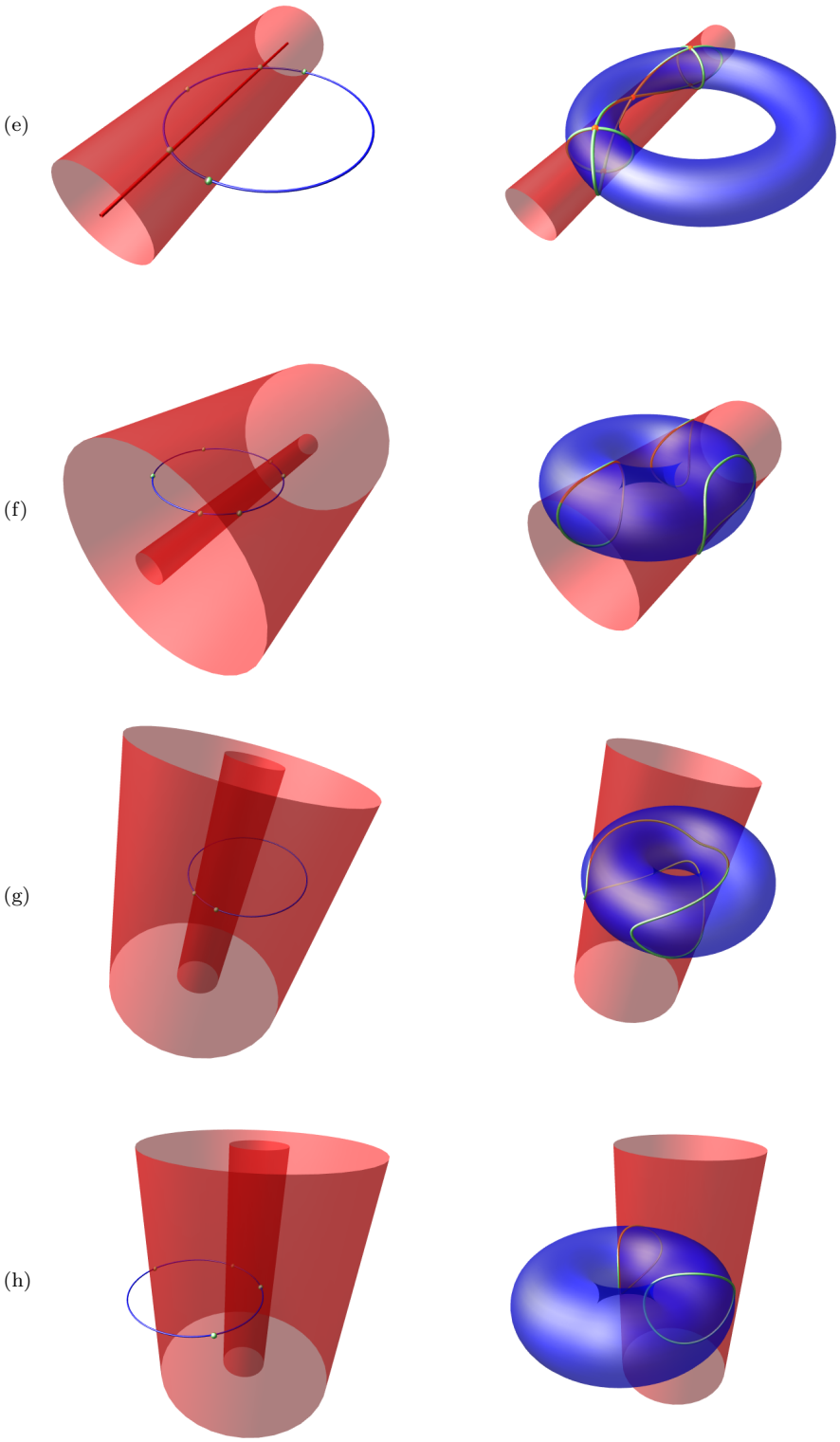


Figure 4.13: Torus-cylinder intersections ($\delta \geq r$) - (*cont.*)

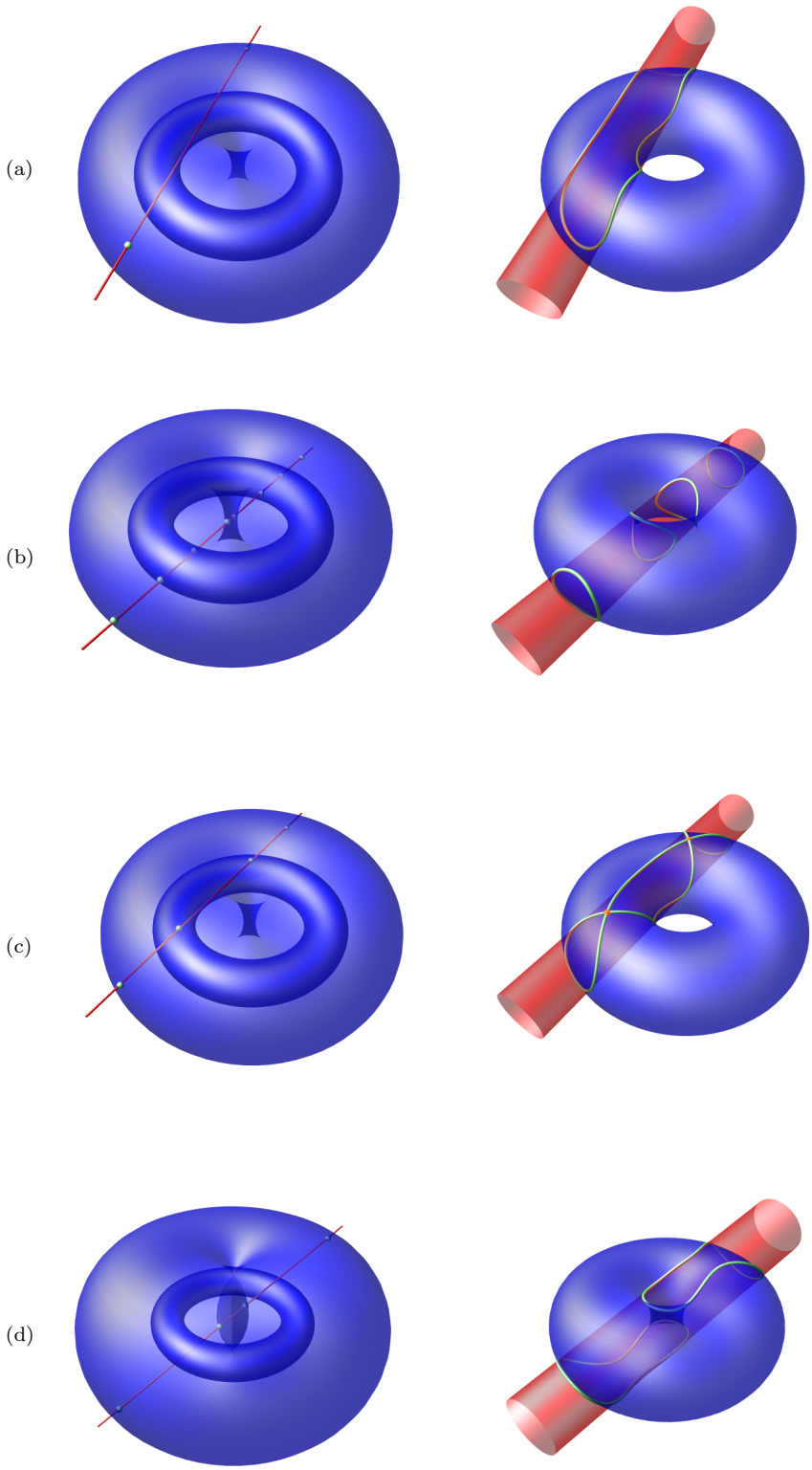


Figure 4.14: Torus-cylinder intersections ($\delta < r$)

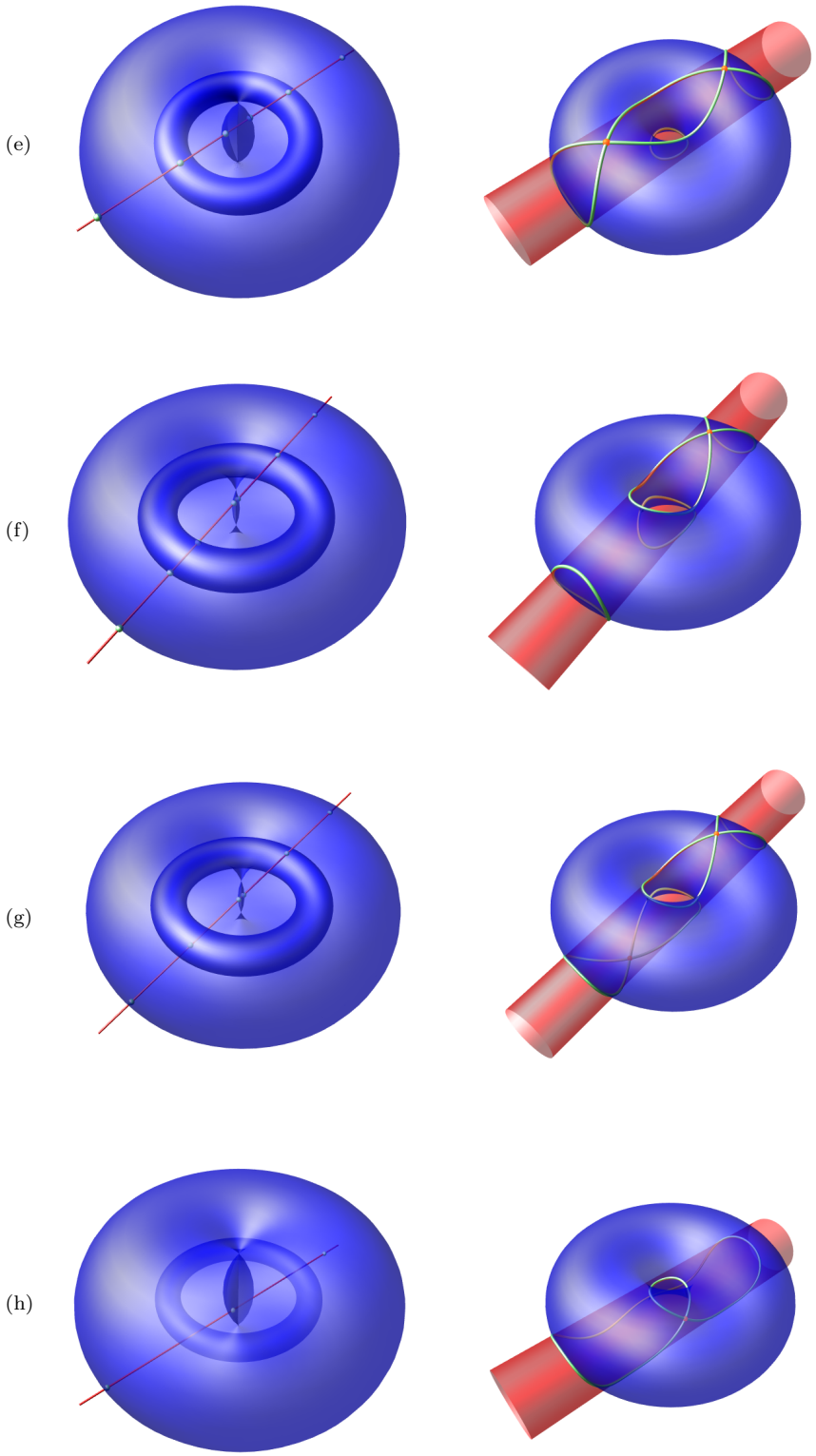


Figure 4.14: Torus-cylinder intersections ($\delta < r$) - (*cont.*)

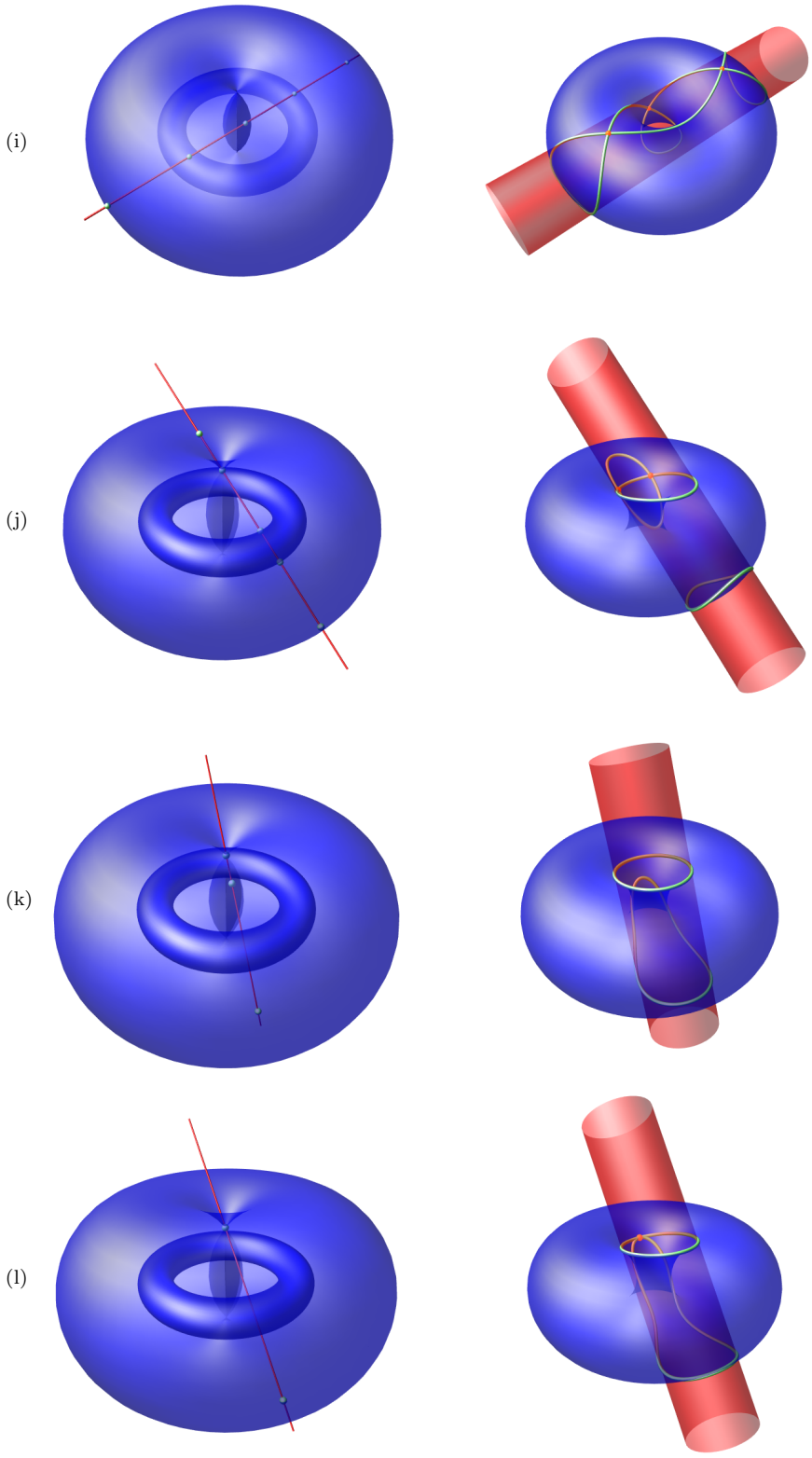


Figure 4.14: Torus-cylinder intersections ($\delta < r$) - (*cont.*)

4.4.4 Torus-Cone Intersection

Given a torus $T = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and a cone $K = K_\delta(\mathbf{p}, \mathbf{n})$. We consider the torus as envelope surface of a moving ball with radius r along the trajectory $C_R = C_R(\mathbf{o}, \mathbf{e}_3)$. Since we cannot distinguish algebraically between the upper and lower part of a cone, we consider only full cones in our algorithms, in contrast to Kim's thesis. Nevertheless, we can use his results due to symmetry reasons. The cone becomes an obstacle, bounded by three offset surfaces with distance r, see Figure 4.15. We gain the two inner offset surfaces by separating and moving the two half cones of K about $\pm\frac{r}{\sin\theta}$ in direction $\mathbf{n}$ (θ is the half angle of the cone with the relation $\delta = \tan\theta$). The outer offset surface results from moving the two half cones in opposite directions, such that they overlap. We name the central overlapping part K^C, which belongs to the obstacle as well. Treating K^C as an own surface, we denote the interior by K^C_- and the exterior by K^C_+. Thus the interior of the whole obstacle consists of $(K^O_+ \cap K^I_-) \cup (K^O_- \cap K^I_+) \cup K^C_- \cup K^C$. Furthermore we build an offset surface around the singular point $\mathbf{p}$ of the cone K, resulting in a sphere $K^B = S_r(\mathbf{p})$. By the union of K^B with the upper and lower part of K^C we gain a special central region which we denote by K^D similar to previous sections. The obstacle of the cone can be completely specified by the three surfaces $K^O = K_\delta(\mathbf{p} + \frac{r}{\sin\theta}\mathbf{n}, \mathbf{n})$, $K^I = K_\delta(\mathbf{p} - \frac{r}{\sin\theta}\mathbf{n}, \mathbf{n})$ and K^B. In Table 4.2 we summarise all conditions to specify each region of the obstacle.

We have two possibilities to specify the angle of the cone: First by the half angle θ respective the rational substitution t_θ such that

$$\delta = \tan\theta = \frac{2t_\theta}{1 - t_\theta^2} \quad \text{and} \quad \sin\theta = \frac{2t_\theta}{1 + t_\theta^2}, \tag{4.2}$$

which are rational as well. The second possibility is to specify δ, but then we would gain

$$\sin\theta = \frac{\delta}{\sqrt{1 + \delta^2}}.$$

Doing so, further computations would require algebraic extensions, whereas the first case provides rational representations of the offset surfaces K^O and K^I. On the other hand this case cannot handle a cone with a full angle of $\pi/2$, which could be simply specified by setting $\delta = 1$. The implementation is highly dependent on this choice. Due to running time aspects we decided to use the first technique, so when we write $\sin\theta$ we mean the rational substitution (4.2).

The trajectory C_R can have up to ten intersections with $K^O \cup K^I \cup K^B$, i.e. four intersections with each of the offset surfaces K^O and K^I and two intersections with K^B. This fact and the high number of different regions, the obstacle can be partitioned in, makes the classification of the topological type of the whole intersection curve between a torus and a cone unapparent. Kim considers already just subtypes of not necessarily independent intersection curve components. Since we consider even a full cone instead of a half cone, the number of possible topological types in a closer environment of the apex $\mathbf{p}$ of the cone exceeds the feasibility in this thesis. Nevertheless we present all types of topologically different intersection curve components and discuss several approaches of computing starting points on each of these component types. Furthermore we consider the most special configurations.

We group all topologically different TKI curve components into seven types:

Type A: Regular closed loops which surround the medial axis of the cone K. Starting points on these loops can be found by intersecting the torus T with any arbitrary profile line of the cone K.

Type B: Regular closed loops which surround the hole of the torus T. Starting points on these loops can be found by intersecting the cone K with any arbitrary cross-sectional circle of the torus T.

Type C: Regular closed loops, where starting points can be found by intersecting the cone K with a suitable cross-sectional circle of the torus T.

Type D: Regular closed loops, where starting points can be found by intersecting the torus T with a suitable profile line of the cone K.

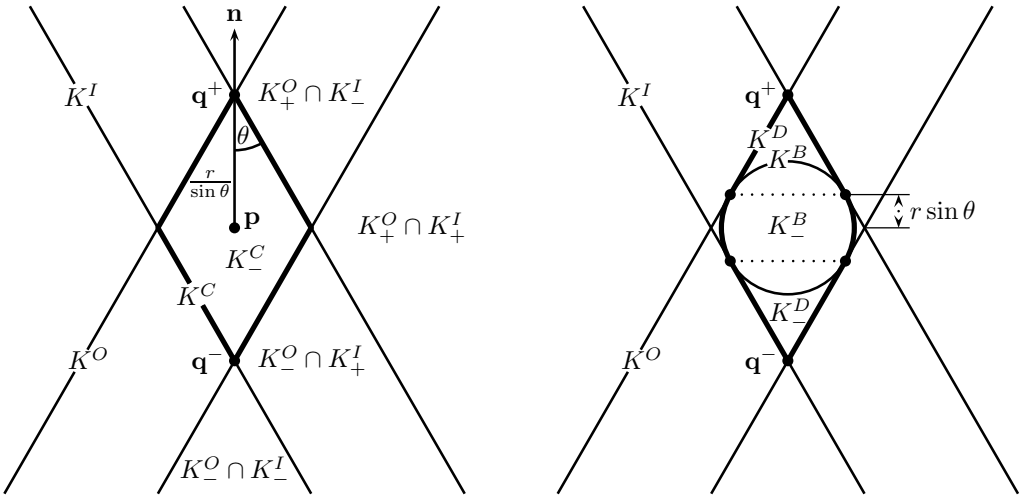


Figure 4.15: Regions of a cone obstacle (cross-section)

region	condition
K^O	$f_{K^O}(\mathbf{x}) = 0 \wedge \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \notin [r\sin\theta, \frac{r}{\sin\theta}]$
K^I	$f_{K^I}(\mathbf{x}) = 0 \wedge \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \notin [-\frac{r}{\sin\theta}, -r\sin\theta]$
K^B	$f_{K^B}(\mathbf{x}) = 0$
K^C	$\begin{cases} f_{K^O}(\mathbf{x}) = 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [0, \frac{r}{\sin\theta}] \\ f_{K^I}(\mathbf{x}) = 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [-\frac{r}{\sin\theta}, 0] \end{cases}$
K^D	$\begin{cases} f_{K^O}(\mathbf{x}) = 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [r\sin\theta, \frac{r}{\sin\theta}] \\ f_{K^B}(\mathbf{x}) = 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [-r\sin\theta, r\sin\theta] \\ f_{K^I}(\mathbf{x}) = 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [-\frac{r}{\sin\theta}, -r\sin\theta] \end{cases}$
q^+	$f_{K^O}(\mathbf{x}) = 0 \wedge \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle = \frac{r}{\sin\theta}$
q^-	$f_{K^I}(\mathbf{x}) = 0 \wedge \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle = -\frac{r}{\sin\theta}$
$K_-^O \cap K_-^I$	$f_{K^O}(\mathbf{x}) < 0 \wedge f_{K^I}(\mathbf{x}) < 0$
$K_+^O \cap K_+^I$	$f_{K^O}(\mathbf{x}) > 0 \wedge f_{K^I}(\mathbf{x}) > 0$
$K_-^O \cap K_+^I$, $K_+^O \cap K_-^I$	$f_{K^O}(\mathbf{x}) f_{K^I}(\mathbf{x}) < 0$
K_-^C	$f_{K^O}(\mathbf{x}) < 0 \wedge f_{K^I}(\mathbf{x}) < 0 \wedge \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [-\frac{r}{\sin\theta}, \frac{r}{\sin\theta}]$
K_-^D	$\begin{cases} f_{K^O}(\mathbf{x}) < 0 \wedge f_{K^I}(\mathbf{x}) < 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [r\sin\theta, \frac{r}{\sin\theta}] \\ f_{K^B}(\mathbf{x}) < 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [-r\sin\theta, r\sin\theta] \\ f_{K^O}(\mathbf{x}) < 0 \wedge f_{K^I}(\mathbf{x}) < 0, & \langle \mathbf{x}-\mathbf{p}, \mathbf{n}\rangle \in [-\frac{r}{\sin\theta}, -r\sin\theta] \end{cases}$

Table 4.2: Classification of regions of a cone obstacle

Type E: Singular curves with k singular points on it. These curves can be traced from their singular points.

Type F: Isolated singularities or touching points.

Type G: Singular curves, where each point of these curves is a touching point between T and K.

In the following we will give the necessary conditions for the correlation between the relative position of circle arcs in C-space and the types of TKI curve components.

Let $C_i = C_R \cap (\overline{K_+^O \cap K_-^I} \cup \overline{K_-^O \cap K_+^I} \cup K_-^C)$ with endpoints on $K^O \cup K^I$ be a circle arc of C_R which lies inside the obstacle. We consider first the most special configurations. Then we can exclude them while considering the more general ones.

Conic sections

Although we already treated conic sections, see Section 3.5, we consider a special configuration of profile circles of T, which occurs when C_R is embedded in $K^O \cup K^I$. In this case the TKI curve consists of one circle of Type G. If C_R is embedded in K^I, the torus T lies in the interior of the cone K, see Figure 4.16(a). Otherwise T lies outside K, see Figure 4.16(b).

C_i passes through $\mathbf{q}^+$ or $\mathbf{q}^-$

1. $C_i \subset K_+^D$:
   a. C_i has no tangential intersection with $K^O \cup K^I \cup K^B$: The corresponding TKI curve component is of Type E with two singular points (Figure 4.17(a)).
   b. C_i has k tangential intersections with $K^O \cup K^I$: The corresponding TKI curve component is of Type E with $k+2$ singular points.
2. $C_i \cap K_-^D \neq \emptyset$:
   a. C_i has no tangential intersection with $K^O \cup K^I \cup K^B$: The corresponding TKI curve component consists of two curves of Type A (Figure 4.17(b)).
   b. C_i has k tangential intersections with $K^O \cup K^I$: The corresponding TKI curve component consists of one curve of Type E with k singular points and one curve of Type A.
3. C_i intersects tangentially with K^D: The corresponding TKI curve component is of Type E with one singular point (Figures 4.17(c)+(d)).

If C_i passes through both $\mathbf{q}^+$ and $\mathbf{q}^-$, we consider each point separately. To distinguish the cases, we consider the tangent $\mathbf{n_q}$ of the circle C_R at the point $\mathbf{q}^\pm$, which computes to $\mathbf{n_q} = \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q}^\pm)$. In case 1 the angle between $\mathbf{n_q}$ and $\mathbf{n}$ is greater than θ, i.e. $|\langle \mathbf{n_q}, \mathbf{n} \rangle| < \cos\theta$. In case 2 the angle is smaller, and in case of equality we have a tangential intersection, see case 3.

Tangential intersections with K^B

If C_i intersects tangentially with K^B, the corresponding TKI curve component is always singular with one singular point which is the apex $\mathbf{p}$ of the cone K. There are three different types of resulting curves depending on the location of the touching point $\mathbf{q} = C_i \cap K^B$:

4. $\mathbf{q} \in K_-^D$, i.e. $\langle \mathbf{q}-\mathbf{p}, \mathbf{n} \rangle \in [-\frac{r}{\sin\theta}, -r\sin\theta] \cup [r\sin\theta, \frac{r}{\sin\theta}]$: The corresponding TKI curve component is of Type F (Figure 4.18(a)).
5. $\mathbf{q} \in K_-^C / K_-^D$, i.e. $\langle \mathbf{q}-\mathbf{p}, \mathbf{n} \rangle \in [-r\sin\theta, r\sin\theta]$: The corresponding TKI curve component is of Type E (Figure 4.18(b)).

6. $\mathbf{q} \in K^C$, i.e. $\langle \mathbf{q} - \mathbf{p}, \mathbf{n} \rangle = \pm r \sin\theta$: The corresponding TKI curve component is of Type E (Figures 4.18(c)+(d)). In this case the singular point $\mathbf{p}$ has two branches, whereas all singular points, we have discussed until now, have four branches (except touching points). We call such a singularity a *cusp*.

Tangential intersections with $K^O \cup K^I$

A tangential intersection point on C_R connects two circle arcs C_i and C_j, which may become two separate arcs or which may be merged to one circle arc, if C_R moved slightly in a suitable direction. Thus singular points on intersection curves may connect some loops. Since we considered the cases 1-3 and the cases 4-6 separately, it is possible in some configurations, that singular curve components merge to one singular curve component, see Figures 4.19(a)+(b), where the number of singular points do not change. In the Algebraic Part we need not to know, how the singular points are connected to each other or how many distinct singular intersection curve components we have, since in the Numeric Part this will be resolved automatically by tracing the curve. Thus we can at last concentrate on regular curves.

Regular circle arcs

The circle arc C_i is a regular circle arc which has no tangential intersection with $K^O \cup K^I \cup K^B$.

7. $C_i = C_R$, i.e. the whole trajectory lies in the interior of the obstacle: The intersection curve consists of up to four intersection curve components of Type B (Figures 4.20(a)-(c)).

8. $C_i \cap (K^D_- \cup K^D) = \emptyset$: The corresponding TKI curve component is of Type D (Figures 4.20(d)+(e)).

9. $C_i \cap K^D_- \neq \emptyset$: We may have up to four TKI curve components of Type A (Figures 4.20(f)+(g)).

10. $C_k = C_i \cap K^D_+$ has both endpoints on K^D: The TKI curve component, corresponding to C_k, is of Type C (Figure 4.20(h)).

The whole TKI curve evolves from the combination of some of the cases 1-10, where singular points may connect regular closed curves as well as singular curve components to one singular curve component. Therefore we can not determine the exact number of regular closed loops in cases 7 and 9. On the other hand, if there are no such circle arcs, we can be sure, that there are neither Type A nor Type B curves, since all cases provide necessary conditions for the presence of the different curve types.

In the Algorithm 31 we first detect any conic sections in the TKI curve. In the case of profile circles we can even stop the algorithm, since no other intersection curve components occur. For a simpler reading of the algorithm we use the predicates `DetectProfileCircles`, `DetectCrossSectionalCircles` and `DetectVillarceauCircles`, which check the necessary conditions and return any occurring conic section in a set. Since the different types of singular curves will be detected in the tracing step, we can completely concentrate on regular intersection curve components (cases 7-10). Alone by the examination of the intersections of C_R with the offset surfaces K^O and K^I including K^D we are able to detect each intersection curve component of the Type A-D. We will mostly get more starting points than we actually need, since intersections of two different profile lines of K with the torus T may lie on the same intersection curve component. However, during the tracing step the redundant points will be detected and erased anyway.

Algorithm 30 `CircleConeIntersection`$(C_\beta(\mathbf{p}_1, \mathbf{n}_1), K_\delta(\mathbf{p}_2, \mathbf{n}_2))$

Requires: A circle $C = C_\beta(\mathbf{p}_1, \mathbf{n}_1)$ and a cone $K = K_\delta(\mathbf{p}_2, \mathbf{n}_2)$
Returns: A set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

1: $E \leftarrow$ `ConePlaneIntersection`$(K, P(\mathbf{p}_1, \mathbf{n}_1))$
2: **return** `CircleConicIntersection`(C, E)

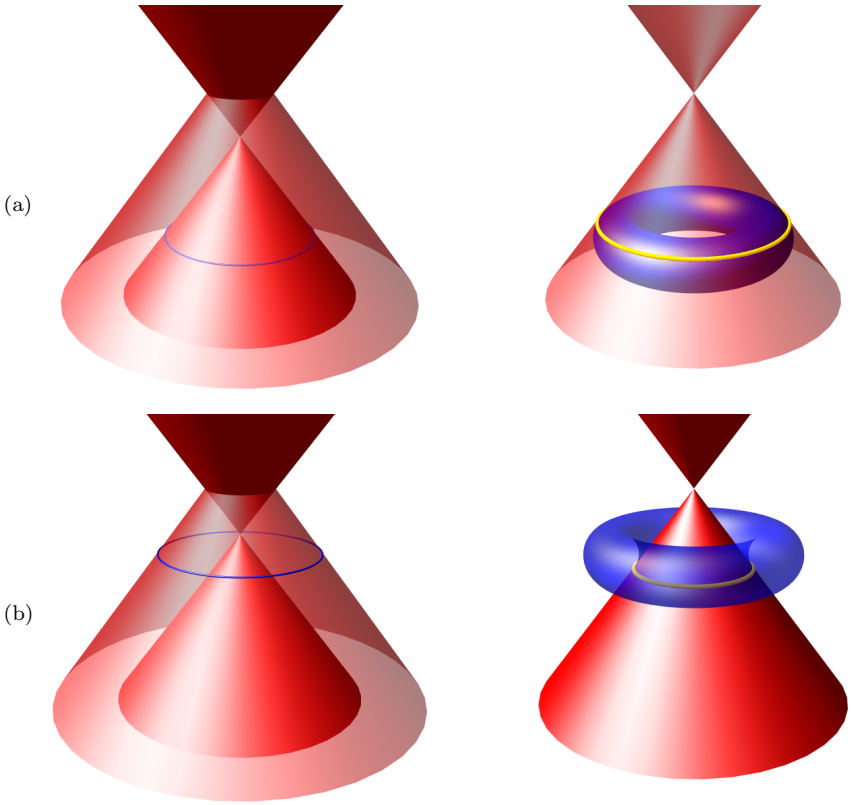


Figure 4.16: Torus-cone intersections (singular circles)

Algorithm 31 `TorusConeIntersection`$(T_{r,R}(\mathbf{o},\mathbf{e}_3), K_\delta(\mathbf{p},\mathbf{n}))$

Requires: A torus $T = T_{r,R}(\mathbf{o},\mathbf{e}_3)$ and a cone $K = K_\delta(\mathbf{p},\mathbf{n})$ with $\delta = \tan\theta$
Returns: A set of conic sections $\{C\}$, a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

$\{C\} \leftarrow$ `DetectProfileCircles`(T,K) // Figure 3.6a
if $\{C\} = \emptyset$ **then**
  $\{C\} \leftarrow$ `DetectCrossSectionalCircles`(T,K) // Figure 3.6b
  $\{C\} \leftarrow$ `DetectVillarceauCircles`(T,K) // Figure 3.6c
  $C_R \leftarrow C_R(\mathbf{o},\mathbf{e}_3)$
  $\{\mathbf{p}_r\},\{\mathbf{p}_s\} \leftarrow$ `CircleConeIntersection`$(C_R, K_\delta(\mathbf{p} + \frac{r}{\sin\theta}\mathbf{n},\mathbf{n}))$
  $\{\mathbf{p}_r\},\{\mathbf{p}_s\} \leftarrow$ `CircleConeIntersection`$(C_R, K_\delta(\mathbf{p} - \frac{r}{\sin\theta}\mathbf{n},\mathbf{n}))$
  for all $\mathbf{p}_s =$ `IntersectionPoint`$(t,\mathfrak{P}) \in \{\mathbf{p}_s\}$ **do** // Figures 4.17(a)+(c)+(d)+4.18(c)
    $\mathbf{q} \leftarrow$ `dehom`$(\mathfrak{P}(t))$
    $\mathbf{n_q} \leftarrow \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q})$
    $\{\mathbf{q}_s\} \leftarrow$ `CircleConeIntersection`$(C_r(\mathbf{q},\mathbf{n_q}), K)$
  end for
  $\{\mathbf{p}_b\} \leftarrow$ `CircleSphereIntersection`$(C_R, S_\delta(\mathbf{p}))$
  if $|\{\mathbf{p}_b\}| = 1$ **then** // Figure 4.18
    $\{\mathbf{q}_s\} \leftarrow \{\mathbf{p}\}$
  end if
  $[\mathbf{p}_a] \leftarrow \{\mathbf{p}_r\} \cup \{\mathbf{p}_s\} \cup \{\mathbf{p}_b\}$ // sorted by the corresponding circle parameters t
  if $[\mathbf{p}_a] = \emptyset$ **or** C_R has exactly one tangential intersection with K^C **then**
    $\mathbf{q} \leftarrow R\mathbf{e}_1$
    if $f_{K^O}(\mathbf{q}) f_{K^I}(\mathbf{q}) < 0$ **then**
      $\{\mathbf{q}_r\} \leftarrow$ `CircleConeIntersection`$(C_r(\mathbf{q},\mathbf{e}_2), K)$
    end if
  else
    $\{C_i\} \leftarrow C_R \cap (\overline{K_+^O \cap K_-^I} \cup \overline{K_-^O \cap K_+^I} \cup K_-^C)$
    $C \leftarrow C_R(M_\mathbf{n}^{-1}(-\mathbf{p}), M_\mathbf{n}^{-1}\mathbf{n})$
    for all $C_i = [t_1,t_2] \in \{C_i\}$ **do** // t_1, t_2 are regular intersections
      if $C_i \cap K_-^D = \emptyset$ **then**
        if $p_z = n_z = 0$ **then**
          $\mathbf{q} \leftarrow C_R($`RationalBetween`$(t_1,t_2))$
          $\mathbf{d} \leftarrow$ `sign`$(\langle \mathbf{q}-\mathbf{p}, \mathbf{n}\times\mathbf{e}_3\rangle)\tan\theta(\mathbf{n}\times\mathbf{e}_3) +$ `sign`$(\langle \mathbf{q}-\mathbf{p},\mathbf{n}\rangle)\mathbf{n}$
        else
          $\mathbf{q} \leftarrow C($`RationalBetween`$(t_1,t_2))$
          $\mathbf{q}_n \leftarrow C_R($`RationalBetweenProjectedCircle`$(C,t_1,t_2))$
          $\mathbf{d} \leftarrow \frac{\tan\theta}{R} M_\mathbf{n}\mathbf{q}_n +$ `sign`$(\langle \mathbf{q}-\mathbf{p},\mathbf{n}\rangle)\mathbf{n}$
        end if
        $\{\mathbf{q}_r\} \leftarrow$ `TorusLineIntersection`$(T, L(\mathbf{p},\mathbf{d}))$
      else
        $C_k = [t_3,t_4] \leftarrow C_i \cap K_+^D$
        if t_3, t_4 are intersections with K^D **then**
          $\mathbf{q} \leftarrow C_R($`RationalBetween`$(t_3,t_4))$
          $\mathbf{n_q} \leftarrow \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q})$
          $\{\mathbf{q}_r\} \leftarrow$ `CircleConeIntersection`$(C_r(\mathbf{q},\mathbf{n_q}), K)$
        end if
      end if
    end for
    if $C_R \cap K_-^D \neq \emptyset$ **then**
      $\{\mathbf{q}_r\} \leftarrow$ `TorusLineIntersection`$(T, L(\mathbf{p}, M_\mathbf{n}(\tan\theta\mathbf{e}_1 + \mathbf{e}_3)))$
    end if
  end if
end if
return $\{C\},\{\mathbf{q}_r\},\{\mathbf{q}_s\}$

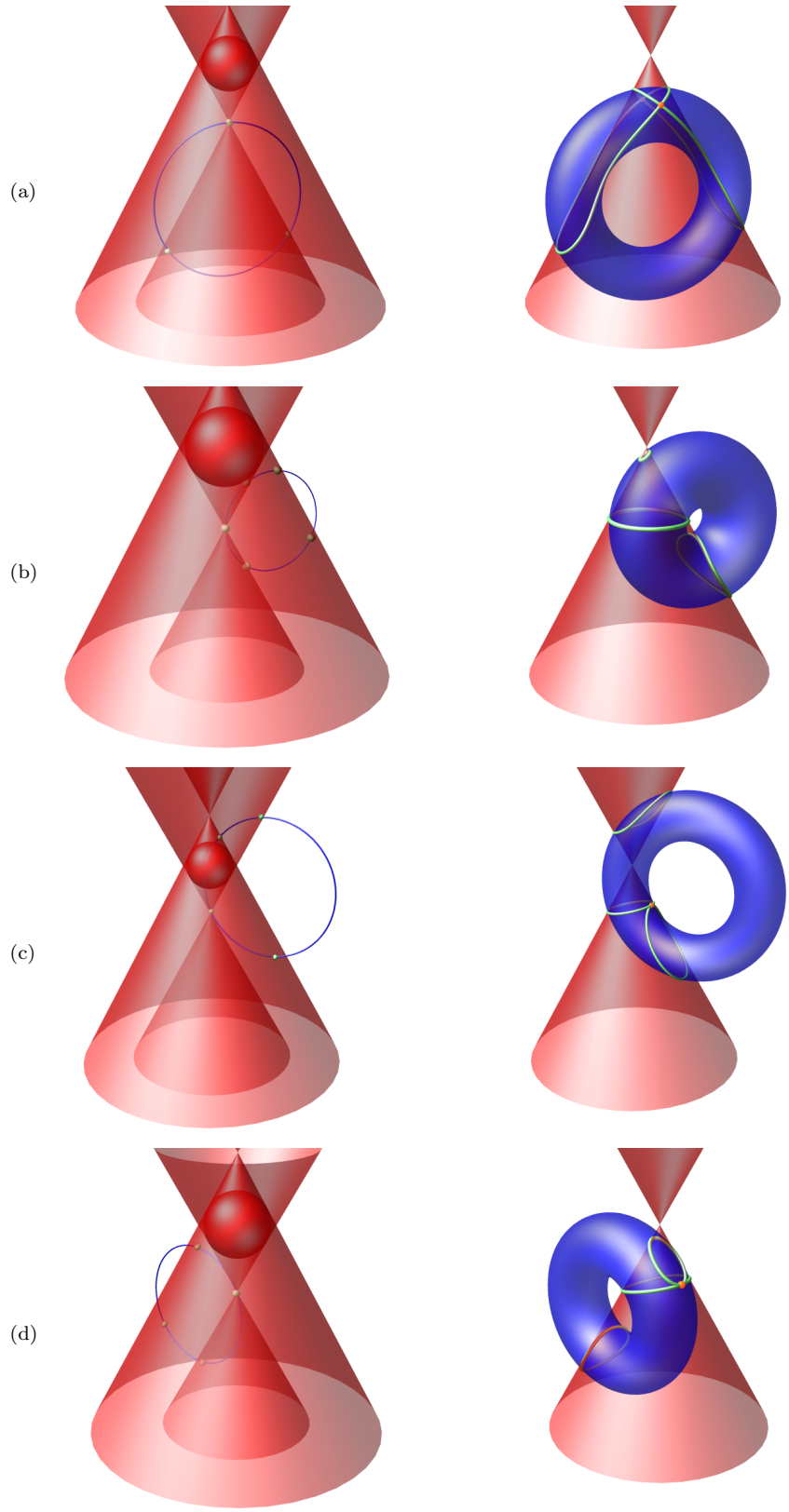


Figure 4.17: Torus-cone intersections (involving $\mathbf{q}^{\pm}$)

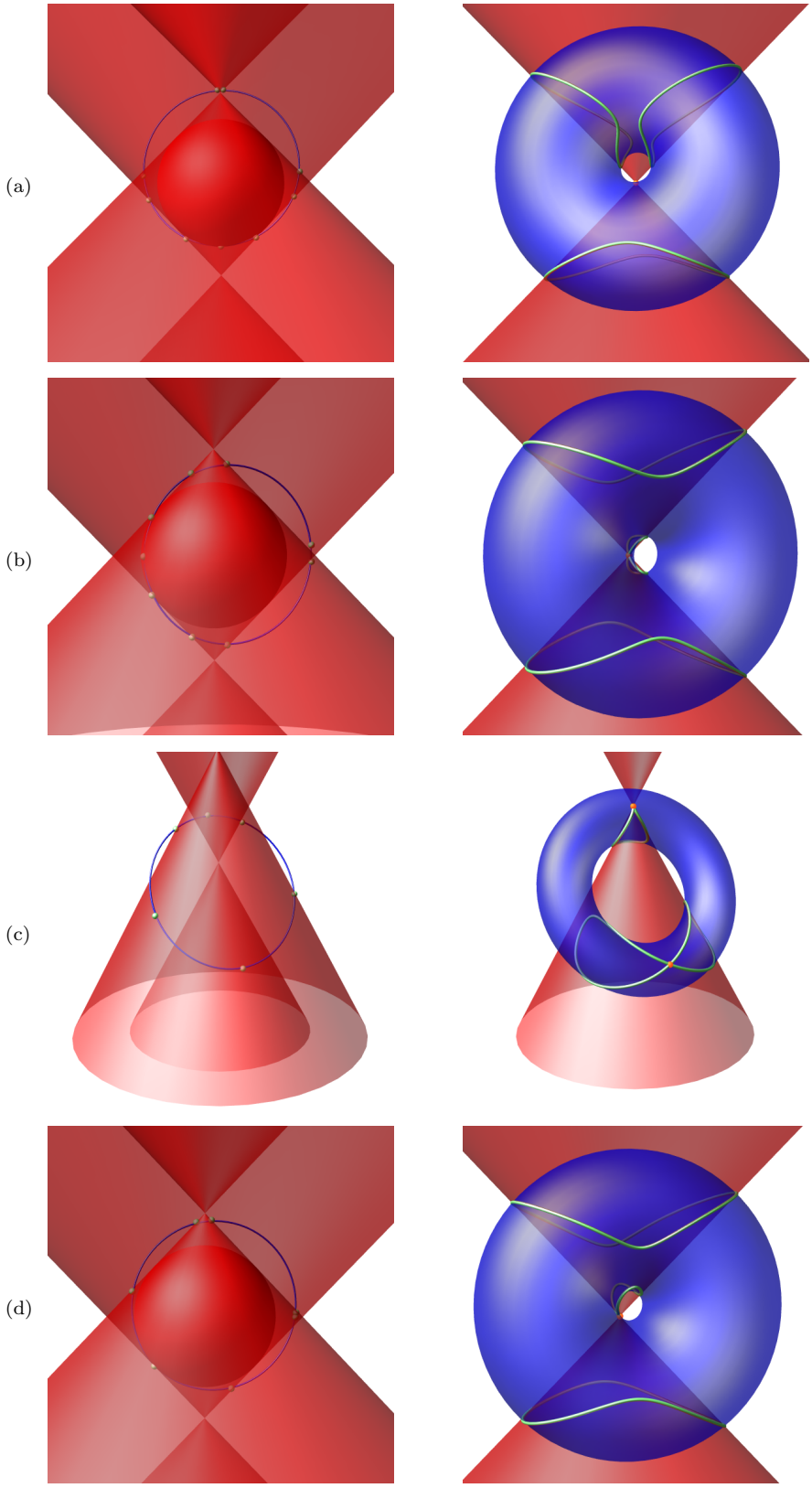


Figure 4.18: Torus-cone intersections (tangential to K^B)

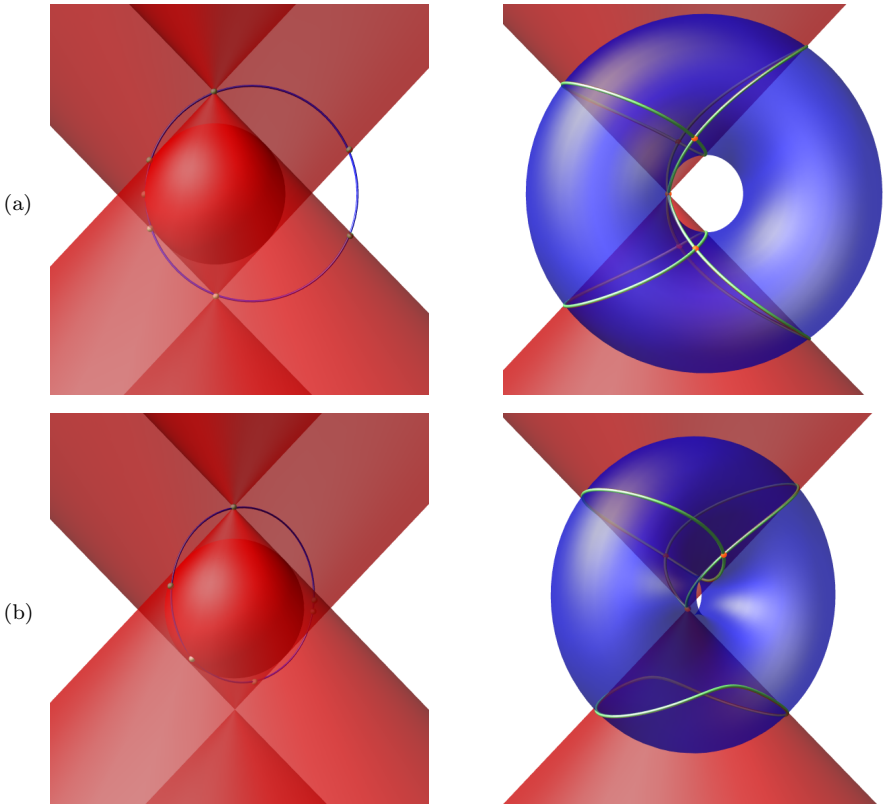


Figure 4.19: Torus-cone intersections (connected singular curves)

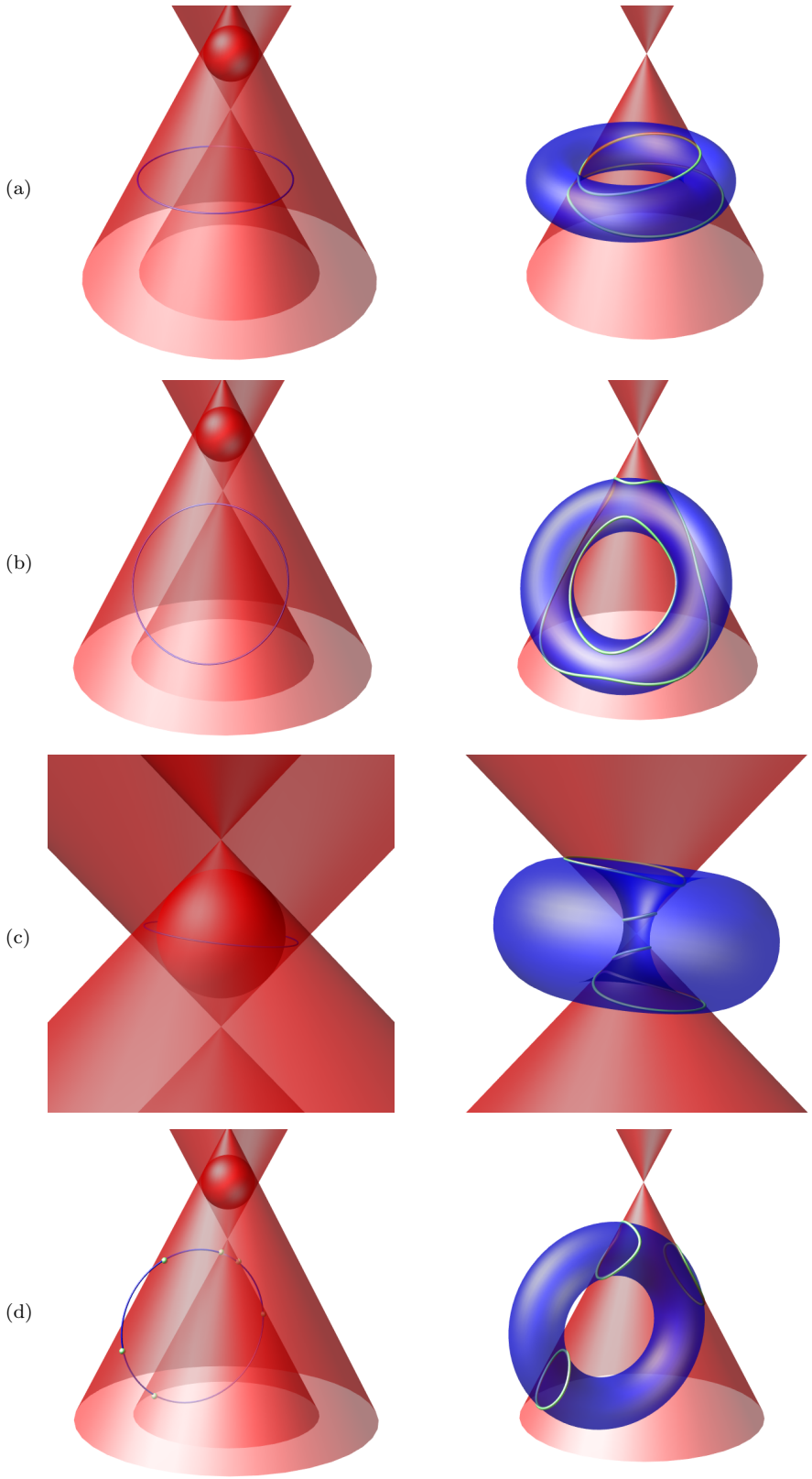


Figure 4.20: Torus-cone intersections (regular circle arcs)

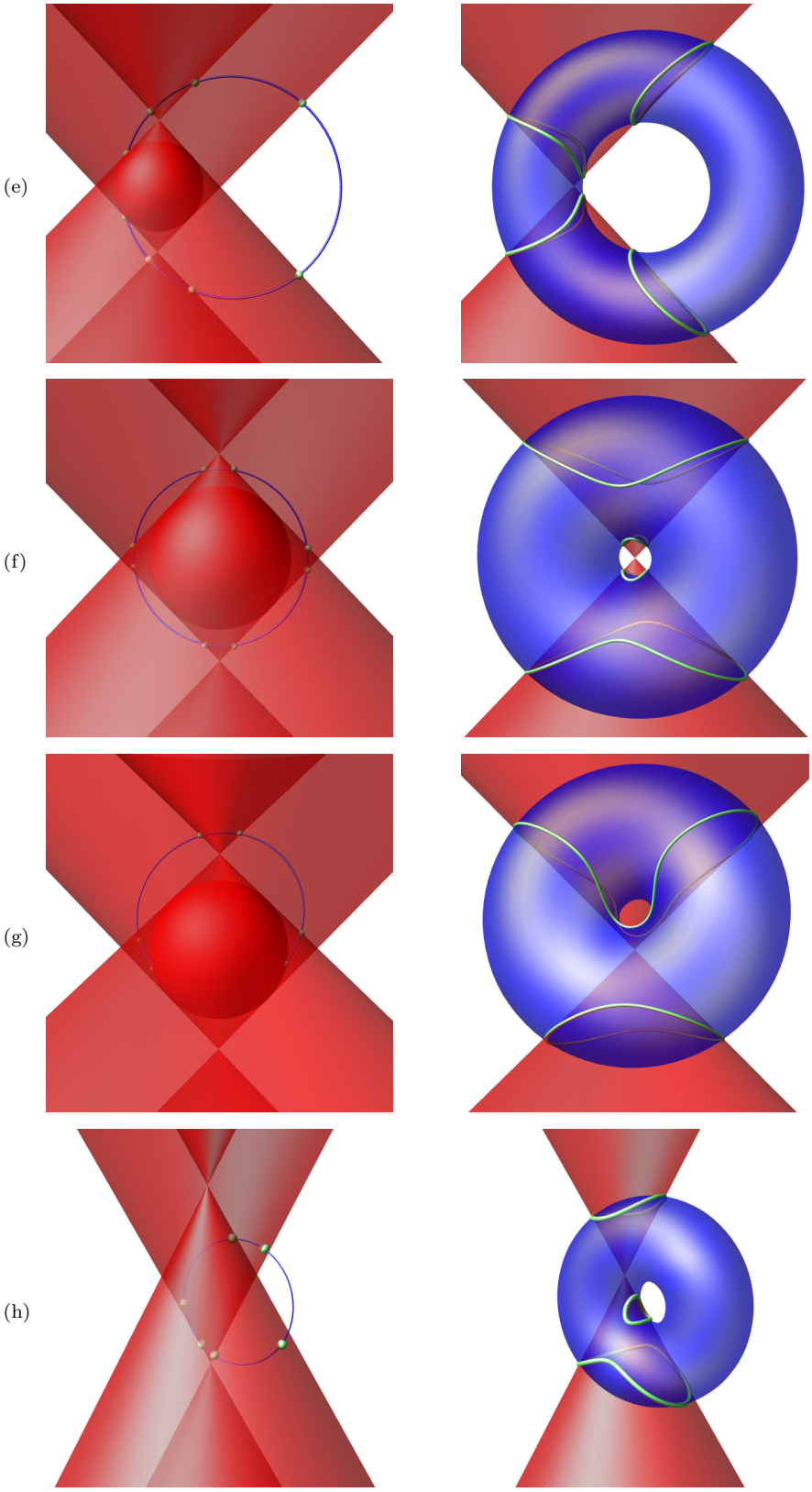


Figure 4.20: Torus-cone intersections (regular circle arcs) - (*cont.*)

4.4.5 Torus-Torus Intersection

Given two tori $T_1 = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and $T_2 = T_{\delta,\Delta}(\mathbf{p}, \mathbf{n})$. Without loss of generality we assume $r \geq \delta$, so we consider the main circle $C_\Delta = C_\Delta(\mathbf{p}, \mathbf{n})$ of T_2 as trajectory and the offset surfaces $T^O = T_{r+\delta,R}(\mathbf{o}, \mathbf{e}_3)$ and $T^I = T_{r-\delta,R}(\mathbf{o}, \mathbf{e}_3)$ of T_1 as boundaries of the obstacle in C-space. If $r + \delta \geq R$ the offset surface T^O becomes a horn respective spindle torus. We specify the different regions as in Figure 4.6.

Similar to Section 4.4.4 we group all topologically different TTI curve components into seven types:

Type A: Regular closed loops which surround the hole of the torus T_1. Starting points on these loops can be found by intersecting the torus T_2 with any arbitrary cross-sectional circle of the torus T_1.

Type B: Regular closed loops which surround the hole of the torus T_2. Starting points on these loops can be found by intersecting the torus T_1 with any arbitrary cross-sectional circle of the torus T_2.

Type C: Regular closed loops, where starting points can be found by intersecting the torus T_1 with a suitable cross-sectional circle of the torus T_2.

Type D: Regular closed loops, where starting points can be found by intersecting the torus T_2 with a suitable cross-sectional circle of the torus T_1.

Type E: Singular curves with k singular points on it. These curves can be traced from their singular points.

Type F: Isolated singularities or touching points.

Type G: Singular curves, where each point of these curves is a touching point between T_1 and T_2.

In the following we will give the necessary conditions for the correlation between the relative position of circle arcs in C-space and the types of TTI curve components.

Let $C_i = C_\Delta \cap \overline{T^O_- \cap T^I_+}$ with endpoints on $T^O \cup T^I$ be a circle arc of C_Δ which lies inside the obstacle. We consider first the most special configurations. Then we can exclude them while considering the more general ones.

Conic sections

We consider the cases, where C_Δ is a conic section of $T^O \cup T^I$, i.e. C_Δ is a profile, a cross-sectional or a Villarceau circle of either T^O or T^I. In these cases the TTI curve has components of Type G. We consider each case separately.

1. C_Δ is a profile circle: In this case T_1 intersects with T_2 along a singular profile circle of T_1 and T_2. If C_Δ is a profile circle of T^I, T_2 lies in the interior of T_1. Otherwise T_2 lies outside T_1 (Figures 4.21(a)+(b)).

2. C_Δ is a cross-sectional circle: If C_Δ is a cross-sectional circle of T^I, T_2 lies in the interior of T_1 and intersects with T_1 along a singular cross-sectional circle of T_1, which is a profile circle of T_2 (Figure 4.21(c)). If C_Δ is a cross-sectional circle of T^O, we distinguish between the cases, if T^O is a ring torus, a horn torus or a spindle torus.

   In the former case, T_2 lies outside T_1 and intersects with T_1 along the singular cross-sectional circle $C_r(\mathbf{p}, \mathbf{n})$ of T_1, which is a profile circle of T_2 (Figure 4.21(d)).

   If T^O is a horn torus, T_2 intersects with T_1 along the singular cross-sectional circle $C_r(\mathbf{p}, \mathbf{n})$ of T_1 and the singular cross-sectional circle $C_\delta(\mathbf{o}, \mathbf{e}_3)$ of T_2. Both circles are connected by the singular point $\mathbf{q} = \frac{\delta}{\Delta}\mathbf{p}$ (Figure 4.21(e)).

If T^O is a spindle torus, T_2 intersects with T_1 along the singular cross-sectional circle $C_r(\mathbf{p},\mathbf{n})$ of T_1 and two closed loops, which are connected to the circle by the singular points $\mathbf{q}_1$ and $\mathbf{q}_2$ (Figure 4.21(f)). These points can be computed by intersecting T_1 with the cross-sectional circles of T_2 at the points $\mathbf{q}^+$ and $\mathbf{q}^-$, i.e. the vertices of the self-intersecting part T^D, resulting in

$$\mathbf{q}_{1,2} = \frac{1}{\Delta}(\delta\mathbf{p} \pm r\sqrt{\Delta^2 - R^2}\mathbf{e}_3).$$

3. C_Δ is a Villarceau circle: The resulting TTI curve in this case is a singular quartic curve of Type G (Figures 4.21(g)+(h)). Since each point $\mathbf{q}$ of this curve is a touching point between T_1 and T_2, the normals $\mathbf{n}_1$ and $\mathbf{n}_2$ of the surfaces T_1 and T_2 at point $\mathbf{q}$ are collinear. Thus we cannot use our approach for tracing a regular intersection curve component, see Section 5.1. Fortunately we can parametrise the curve as follows:

   Since C_Δ is a Villarceau circle of either T^O or T^I, the torus T_2 fulfils the following conditions:

$$\begin{aligned}\Delta &= R,\\ \langle\mathbf{p},\mathbf{n}\rangle &= 0,\\ \langle\mathbf{p},\mathbf{e}_3\rangle &= 0,\\ \|\mathbf{p}\| &= r \pm \delta,\\ R^2 n_z^2 &= R^2 - (r\pm\delta)^2,\end{aligned}$$

   see Sections 3.1.3 and 3.2.3. We rotate the scene such that $\mathbf{p}$ lies on the positive x-axis. The corresponding transformation matrix is

$$A = \begin{pmatrix} \frac{p_x}{\sqrt{p_x^2+p_y^2}} & \frac{p_y}{\sqrt{p_x^2+p_y^2}} & 0\\ -\frac{p_y}{\sqrt{p_x^2+p_y^2}} & \frac{p_x}{\sqrt{p_x^2+p_y^2}} & 0\\ 0 & 0 & 1\end{pmatrix}. \tag{4.3}$$

   Whereas T_1 is invariant under this transformation, torus T_2 transfers into

$$T_2' = T_{\delta,\Delta}((r\pm\delta,0,0),(0,\epsilon\frac{r\pm\delta}{R},n_z))$$

   with $\epsilon = \mathtt{sign}(\langle\mathbf{p},(\mathbf{n}\times\mathbf{e}_3)\rangle)$. Substituting the parameter form (2.6) of T_1 into the implicit form (2.5) of T_2' yields

$$f(s,t) = \frac{16r(r\pm\delta)[R(1+s^2)+r(1-s^2)][\sqrt{R+r\pm\delta}t + \epsilon\sqrt{R-r\mp\delta}s]^2}{(1+s^2)^2(1+t^2)}.$$

   We neglect the case $r-\delta=0$, since this would expect identical tori. Thus it suffices to consider the last bracket of the numerator of f, since all other terms are nonzero. We follow that f vanishes if and only if

$$s = -\epsilon\sqrt{\frac{R+r\pm\delta}{R-r\mp\delta}}t \quad\text{with}\quad t\in\mathbb{K}\cup\{\infty\}. \tag{4.4}$$

   Substituting this into the parameter form of T_1 yields

$$\mathfrak{P}(t) = \begin{pmatrix} [-R(R\pm\delta)+r(r\pm\delta)]t^4 + [\pm 2R\delta]t^2 + [R(R\mp\delta)-r(r\pm\delta)]\\ [2R(R\pm\delta)-2r(r\pm\delta)]t^3 + [2R(R\mp\delta)-2r(r\pm\delta)]t\\ [-2\epsilon rRn_z]t^3 + [-2\epsilon rRn_z]t\\ [R+r\pm\delta]t^4 + [2R]t^2 + [R-r\mp\delta]\end{pmatrix},$$

   which is the homogeneous parameter form of the intersection curve between T_1 and T_2', see Section 2.4.3. To gain the homogeneous parameter form of the TTI curve between T_1 and T_2, we simply apply the inverse transformation A^{-1} to $\mathfrak{P}$.

C_i passes through $\mathbf{q}^+$ or $\mathbf{q}^-$

4. $C_i \subset T_+^D$:

   a. C_i has no tangential intersection with $T^O \cup T^I$: The corresponding TTI curve component is of Type E with two singular points (Figure 4.22(a)).

   b. C_i has k tangential intersections with $T^O \cup T^I$: The corresponding TTI curve component is of Type E with $k+2$ singular points (Figure 4.22(b)).

5. $C_i \cap T_-^D \neq \emptyset$:

   a. C_i has no tangential intersection with $T^O \cup T^I$: The corresponding TTI curve component consists of two curves of Type A (Figure 4.22(c)).

   b. C_i has k tangential intersections with $T^O \cup T^I$: The corresponding TTI curve component consists of one curve of Type E with k singular points and one curve of Type A (Figure 4.22(d)).

6. C_i intersects tangentially with T^D: The corresponding TTI curve component is of Type E with one singular point (Figure 4.25(d)).

If C_i passes through both $\mathbf{q}^+$ and $\mathbf{q}^-$, we consider each point separately. To distinguish the cases, we consider the tangent $\mathbf{n_q}$ of the circle C_Δ at the point $\mathbf{q}^\pm$, which computes to $\mathbf{n_q} = \frac{1}{\Delta}(\mathbf{n} \times \mathbf{q}^\pm - \mathbf{p})$. In case 4 the angle between $\mathbf{n_q}$ and $\mathbf{e}_3$ is greater than the half opening angle of T^D, i.e. $|\langle \mathbf{n_q}, \mathbf{e}_3 \rangle| < \frac{R}{r+\delta}$. In case 5 the angle is smaller, and in case of equality we have a tangential intersection, see case 6.

If $\delta + r = R$, the offset surface T^O becomes a horn torus. In this case $\mathbf{q}^+$ and $\mathbf{q}^-$ coincide to the origin $\mathbf{o}$ and the set T_-^D is empty. Thus case 5 can not occur. If C_i passes through $\mathbf{o}$, we distinguish between case 4 ($|\langle \mathbf{n_o}, \mathbf{e}_3 \rangle| < 1$) and case 6 ($|\langle \mathbf{n_o}, \mathbf{e}_3 \rangle| = 1$). In the former case nothing changes, whereas in the latter case we get a TTI curve component of Type G which is a profile circle of T_1 and a cross-sectional circle of T_2. Depending on the radius Δ we may have one of the following cases:

7. $\Delta > R$: The TTI curve consists of the singular circle $C_\delta(\mathbf{o}, \mathbf{e}_3)$ (Figure 4.23(a)).

8. $\Delta = R$: The TTI curve consists of the two singular circles $C_\delta(\mathbf{o}, \mathbf{e}_3)$ and $C_r(\mathbf{p}, \mathbf{n})$, which are connected by the singular point $\mathbf{q} = \frac{\delta}{\Delta}\mathbf{p}$ (Figure 4.21(e)).

9. $\Delta < R$: The TTI curve consists of the singular circle $C_\delta(\mathbf{o}, \mathbf{e}_3)$ and two closed loops which are connected to the circle by the singular points

$$\mathbf{q}_{1,2} = \frac{\delta}{R}(\mathbf{p} \pm \sqrt{R^2 - \Delta^2}\mathbf{n}).$$

   If additionally $r = \Delta$, the closed loops are connected by a third singular point $\mathbf{q}_3 = \frac{R+\Delta}{\Delta}\mathbf{p}$ (Figure 4.23(c)). Otherwise see Figure 4.23(b) ($r < \Delta$) and Figure 4.23(d) ($r > \Delta$).

Tangential intersections with T^I while $r = \delta$

If $r = \delta$, the offset surface T^I degenerates into the circle $C_R = C_R(\mathbf{o}, \mathbf{e}_3)$. Thus there are at most two intersections between C_Δ and C_R. Let $\mathbf{q}$ be one of those intersections. Since the surface normal of T^I at $\mathbf{q}$ is undefined, singular points evolve from the intersection between the cross-sectional circle of T_2 at $\mathbf{q}$ and T_1, see Lemma 1 in Section 4.4.1. Due to symmetry these singular points evolve also from the intersection between the cross-sectional circle of T_1 at $\mathbf{q}$ and T_2. Both cross-sectional circles are great circles of the sphere $S_r(\mathbf{q})$ and thus we either gain two singular points or the circles coincide. The latter case occurs when the tangents of C_R and C_Δ at $\mathbf{q}$ are collinear. This means for the TTI curve:

10. $(\mathbf{q} \times \mathbf{e}_3) \times ((\mathbf{q} - \mathbf{p}) \times \mathbf{n}) = \mathbf{o}$: The TTI curve consists of the singular cross-sectional circle $C_r = C_r(\mathbf{q}, \frac{1}{R}(\mathbf{q} \times \mathbf{e}_3))$ and two closed loops, which are connected to C_r by the singular points $\mathbf{q}_1$ and $\mathbf{q}_2$ (Figure 3.8(a)). To compute these points it does not suffice to intersect the cross-sectional circle of T_2 at $\mathbf{q}$ with T_1, since the resulting set is precisely the circle C_r and thus not zero-dimensional. Instead we consider again the TTI curve in parameter space.

    Therefor we rotate the scene such that $\mathbf{q}$ lies on the positive x-axis by applying matrix A (4.3) from above. This does not change anything for torus T_1. The point $\mathbf{p}$ transforms into

$$\mathbf{p}' = A\mathbf{p} = (\sqrt{p_x^2 + p_y^2}, 0, p_z) = (R + \Delta\cos(\phi), 0, \Delta\sin(\phi))$$

    with some suitable angle $\phi \in [0, 2\pi]$. The normal $\mathbf{n}' = A\mathbf{n}$ becomes then

$$\mathbf{n}' = (-\sin(\phi), 0, \cos(\phi)).$$

    We substitute the parameter form (2.6) of T_1 into the implicit form (2.5) of $T_2' = T_{r,\Delta}(\mathbf{p}', \mathbf{n}')$ which yields

$$f(s,t) = \frac{16t^2[R(1+s^2) + r(1-s^2)]g(s,t)}{(1+s^2)^2(1+t^2)^2}$$

    with a quartic polynomial g. The factor t^2 in the numerator represents the singular cross-sectional circle. The other brackets are positive. Thus the remaining TTI curve components must be represented by the polynomial g. Since we are only interested in the connecting points of the remaining curve components with the singular circle, we set $t = 0$ and get

$$g(s,0) = r\Delta\big[(\Delta + R\cos(\phi))s^2 - 2R\sin(\phi)s - (\Delta + R\cos(\phi))\big],$$

    which has two solutions $s_{1,2}$. We evaluate the parameter form of T_1 with the parameters $s_{1,2}$ and $t = 0$, substitute the trigonometric functions backwards and apply the inverse transformation of A, which yields altogether the singular connection points

$$\mathbf{q}_{1,2} = \frac{1}{\Delta\|\mathbf{p}\|}\big[2R^2\frac{\Delta\|\mathbf{p}\| \mp rp_z}{\|\mathbf{p}\|^2 + R^2 - \Delta^2}(p_x, p_y, 0) \pm \frac{r}{2}(\|\mathbf{p}\|^2 - R^2 + \Delta^2)\mathbf{e}_3\big].$$

    To avoid different algebraic extensions, we made use of the fact, that $\mathbf{p}$ lies on a torus $T_{R,\Delta}(\mathbf{o}, \mathbf{e}_3)$, see (3.101) in Section 3.6.4. Thus the expression $\sqrt{p_x^2 + p_y^2}$ is rational.

11. Otherwise: The TTI curve component is of Type E with two singular points. These points can be computed by intersecting T_1 with the cross-sectional circle of T_2 at $\mathbf{q}$ (Figures 4.24(a)+(b)).

Tangential intersections with T^D

There are three different types of resulting curves depending on the location of the circle arc C_i:

12. $C_i \subset T_+^D$: The corresponding TTI curve component is of Type E with one singular point (Figure 4.25(a)).

13. $C_i \subset T_-^D$: The corresponding TTI curve component is of Type F (Figure 4.25(b)).

14. C_i intersects traversally with T^D at the tangential intersection point $\mathbf{q}$: The corresponding TTI curve component is of Type E with one singular point (Figure 4.25(c)). In this case the singular point is a cusp, see 6 in Section 4.4.4.

Regular circle arcs

Similar to Section 4.4.4 we neglect the consideration of those configurations which involve tangential intersections of the trajectory with the obstacle. Instead we are more interested in regular curve components and how to find their initial starting points. Therefore let C_i be a regular circle arc inside the obstacle which has no tangential intersection with $T^O \cup T^I$.

15. $C_i = C_\Delta$, i.e. the whole trajectory lies in the interior of the obstacle: The TTI curve consists of two intersection curve components of Type B (Figure 4.26(a)).

16. $C_i \cap (T_-^D \cup T^D) = \emptyset$: The corresponding TTI curve component is of Type D (Figure 4.26(b)).

17. $C_i \cap T_-^D \neq \emptyset$: We may have up to two TTI curve components of Type A (Figure 4.26(c)).

18. $C_k = C_i \cap T_+^D$ has both endpoints on T^D: The TTI curve component, corresponding to C_k, is of Type C (Figure 4.26(d)).

The whole TTI curve, if it is not one of the special cases 1-3 and 7-9, evolves from the combination of some of the cases 4-6 and 10-18, where singular points may connect regular closed curves as well as singular curve components to one singular curve component, see Section 4.4.4.

In the Algorithm 32 we first detect any conic sections in the TTI curve. Analogue to Section 4.4.4 we assume that the predicates for detection and computation of those are available. Additionally any homogeneous parameter forms of parametric curves are detected and computed by the predicate `ComputeParameterForms`. These forms are returned in a set $\{\mathfrak{P}\}$, see Section 2.4.3.

Algorithm 32: `TorusTorusIntersection`$(T_{r,R}(\mathbf{o}, \mathbf{e}_3), T_{\delta,\Delta}(\mathbf{p}, \mathbf{n}))$

Requires: Two tori $T_1 = T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ and $T_2 = T_{\delta,\Delta}(\mathbf{p}, \mathbf{n})$ with $r \geq \delta$
Returns: A set of conic sections $\{C\}$, a set of homogeneous parameter forms $\{\mathfrak{P}\}$, a set of regular intersection points $\{\mathbf{q}_r\}$ and a set of singular intersection points $\{\mathbf{q}_s\}$

$\{\mathfrak{P}\} \leftarrow$ `ComputeParameterForms`(T_1, T_2) // Figures 4.21(g)+(h)
if $\{\mathfrak{P}\} = \emptyset$ **then**
  $\{C\} \leftarrow$ `DetectProfileAndProfileCircles`(T_1, T_2) // Figures 3.7(a)+4.21(a) +(b)
  if $\{C\} = \emptyset$ **then**
    if $\langle \mathbf{e}_3, \mathbf{n} \rangle = 0$ **then**
      if $\langle \mathbf{p}, \mathbf{n} \rangle = 0$ **and** $(\delta - R)^2 + p_z^2 = r^2$ **and** $p_x^2 + p_y^2 = \Delta^2$ **then** // Figure 3.7(b)
        $\{C\} \leftarrow \{C_\delta((0, 0, p_z), \mathbf{e}_3)\}$
        if $p_z = 0$ **then** // Figure 4.23(a)
          if $\Delta = R$ **then** // Figure 4.21(e)
            $\{C\} \leftarrow \{C_r(\mathbf{p}, \mathbf{n})\}$
            $\{\mathbf{q}_s\} \leftarrow \{\frac{\delta}{\Delta}\mathbf{p}\}$
          else if $\Delta < R$ **then** // Figures 4.23(b)-(d)
            $\{\mathbf{q}_s\} \leftarrow \{\frac{\delta}{R}(\mathbf{p} \pm \sqrt{R^2 - \Delta^2}\mathbf{n})\}$
            if $r = \Delta$ **then** // Figure 4.23(c)
              $\{\mathbf{q}_s\} \leftarrow \{\frac{R+\Delta}{\Delta}\mathbf{p}\}$
            end if
          end if
          return $\{C\}, \{\mathbf{q}_s\}$
        end if
      else
        if $p_z = 0$ **and** $(r - \Delta)^2 + \langle \mathbf{p}, \mathbf{n} \rangle^2 = \delta^2$ **and** $\|\mathbf{p} \times \mathbf{n}\|^2 = R^2$ **then** // Figure 3.7(b)
          $\{C\} \leftarrow \{C_r(\mathbf{p} - \langle \mathbf{p}, \mathbf{n} \rangle \mathbf{n}, \mathbf{n})\}$
          if $\langle \mathbf{p}, \mathbf{n} \rangle = 0$ **then** // Figures 4.21(c) +(d)
            if $r + \delta > R$ **then** // Figure 4.21(f)
              $\{\mathbf{q}_s\} \leftarrow \{\frac{1}{\Delta}(\delta\mathbf{p} \pm r\sqrt{\Delta^2 - R^2}\mathbf{e}_3)\}$
            end if
            return $\{C\}, \{\mathbf{q}_s\}$
          end if
        end if
      end if
    end if
    $\{C\} \leftarrow$ `DetectProfileAndVillarceauCircles`(T_1, T_2) // Figure 3.7(c)
    $\{C\} \leftarrow$ `DetectProfileAndVillarceauCircles`(T_2, T_1)

$\{C\} \leftarrow$ `DetectCrossSectionalAndCrossSectionalCircles`(T_1, T_2) // Figure 3.8(a)
$\{C\} \leftarrow$ `DetectCrossSectionalAndVillarceauCircles`(T_1, T_2) // Figure 3.8(b)
$\{C\} \leftarrow$ `DetectCrossSectionalAndVillarceauCircles`(T_2, T_1)
$\{C\} \leftarrow$ `DetectVillarceauAndVillarceauCircles`(T_1, T_2) // Figure 3.9
$C_\Delta \leftarrow C_\Delta(\mathbf{p}, \mathbf{n})$
$\{\mathbf{p}_r\}, \{\mathbf{p}_s^O\} \leftarrow$ `TorusCircleIntersection`$(T_{r+\delta,R}(\mathbf{o}, \mathbf{e}_3), C_\Delta)$
$\{\mathbf{p}_r\}, \{\mathbf{p}_s^I\} \leftarrow$ `TorusCircleIntersection`$(T_{r-\delta,R}(\mathbf{o}, \mathbf{e}_3), C_\Delta)$
for all $\mathbf{p}_s =$ `IntersectionPoint`$(t, \mathfrak{P}) \in \{\mathbf{p}_s^O\} \cup \{\mathbf{p}_s^I\}$ **do** // Figures 4.22+4.24+4.25
  $\mathbf{q} \leftarrow$ `dehom`$(\mathfrak{P}(t))$
  $\mathbf{n_q} \leftarrow \frac{1}{\Delta}(\mathbf{n} \times (\mathbf{q} - \mathbf{p}))$
  if $\mathbf{p}_s \in \{\mathbf{p}_s^I\}$ **and** $r = \delta$ **and** $(\mathbf{e}_3 \times \mathbf{q}) \times \mathbf{n_q} = \mathbf{o}$ **then** // Figure 3.8(a)
    $\{\mathbf{q}_s\} \leftarrow \{\frac{1}{\Delta\|\mathbf{p}\|}\left[2R^2 \frac{\Delta\|\mathbf{p}\| \mp r p_z}{\|\mathbf{p}\|^2 + R^2 - \Delta^2}(p_x, p_y, 0) \pm \frac{r}{2}(\|\mathbf{p}\|^2 - R^2 + \Delta^2)\mathbf{e}_3\right]\}$
  else
    $\{\mathbf{q}_s\} \leftarrow$ `TorusCircleIntersection`$(T_1, C_\delta(\mathbf{q}, \mathbf{n_q}))$
  end if
end for
$[\mathbf{p}_a] \leftarrow \{\mathbf{p}_r\} \cup \{\mathbf{p}_s^O\} \cup \{\mathbf{p}_s^I\}$ // sorted by the corresponding circle parameters t
if $[\mathbf{p}_a] = \emptyset$ **or** C_Δ has exactly one tangential intersection with T^D **then** // Figure 4.26(a)
  $\mathbf{q} \leftarrow \mathbf{p} + \Delta M_\mathbf{n} \mathbf{e}_1$
  $\mathbf{n_q} \leftarrow M_\mathbf{n} \mathbf{e}_2$
  if $f_{T^O}(\mathbf{q}) f_{T^I}(\mathbf{q}) < 0$ **then**
    $\{\mathbf{q}_r\} \leftarrow$ `TorusCircleIntersection`$(T_1, C_\delta(\mathbf{q}, \mathbf{n_q}))$
  end if
else
  $\{C_i\} \leftarrow C_\Delta \cap \overline{T_-^O \cap T_+^I}$
  for all $C_i = [t_1, t_2] \in \{C_i\}$ **do** // t_1, t_2 are regular intersections
    if $C_i \cap T_-^D = \emptyset$ **then** // Figure 4.26(b)
      if $\langle \mathbf{p}, \mathbf{n} \rangle = n_z = 0$ **then**
        $\mathbf{a} \leftarrow$ `dehom`$(\mathfrak{P}_{C_\Delta}($`upper`$(t_1)))$
        $\mathbf{q} \leftarrow \frac{R}{\sqrt{a_x^2 + a_y^2}}(\mathbf{e}_3 \times \mathbf{a}) \times \mathbf{e}_3$
      else
        $\mathbf{q} \leftarrow C_R($`RationalBetweenProjectedCircle`$(C, t_1, t_2))$
      end if
      $\mathbf{n_q} \leftarrow \frac{1}{R}(\mathbf{e}_3 \times \mathbf{q})$
      $\{\mathbf{q}_r\} \leftarrow$ `TorusCircleIntersection`$(T_{\delta,\Delta}(\mathbf{o}, \mathbf{e}_3), C_r(M_\mathbf{n}^{-1}(\mathbf{q} - \mathbf{p}), M_\mathbf{n}^{-1}\mathbf{n_q}))$
    else
      $C_k = [t_3, t_4] \leftarrow C_i \cap T_+^D$
      if t_3, t_4 are intersections with T^D **then** // Figure 4.26(d)
        $\mathbf{q} \leftarrow C_\Delta($`RationalBetween`$(t_3, t_4))$
        $\mathbf{n_q} \leftarrow \frac{1}{\Delta}(\mathbf{n} \times (\mathbf{q} - \mathbf{p}))$
        $\{\mathbf{q}_r\} \leftarrow$ `TorusCircleIntersection`$(T_1, C_\delta(\mathbf{q}, \mathbf{n_q}))$
      end if
    end if
  end for
  if $C_R \cap T_-^D \neq \emptyset$ **then** // Figure 4.26(c)
    $\{\mathbf{q}_r\} \leftarrow$ `TorusCircleIntersection`$(T_{\delta,\Delta}(\mathbf{o}, \mathbf{e}_3), C_r(M_\mathbf{n}^{-1}(R\mathbf{e}_1 - \mathbf{p}), M_\mathbf{n}^{-1}\mathbf{e}_2))$
  end if
end if
end if
return $\{C\}, \{\mathfrak{P}\}, \{\mathbf{q}_r\}, \{\mathbf{q}_s\}$

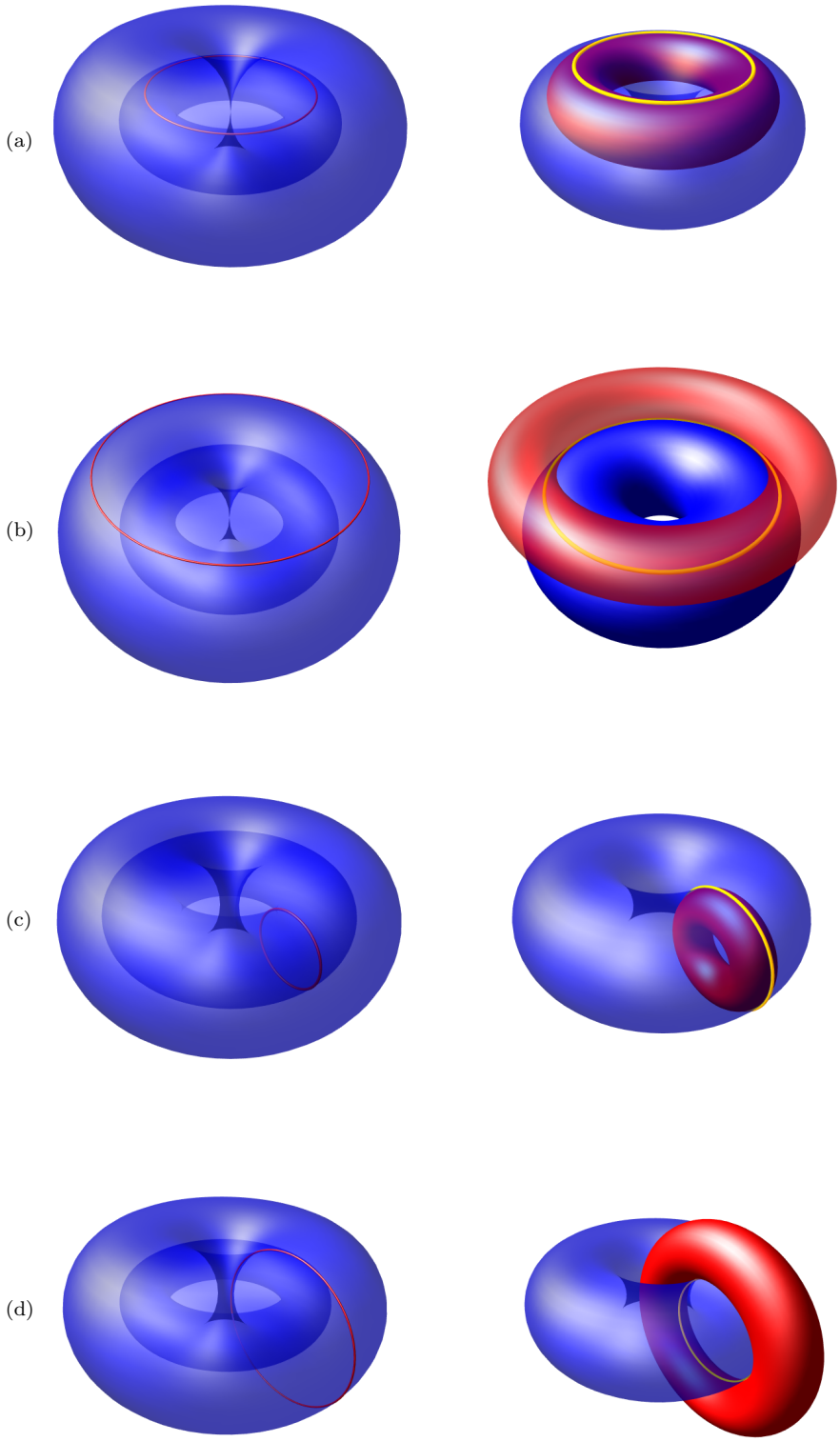


Figure 4.21: Torus-torus intersections (conic sections)

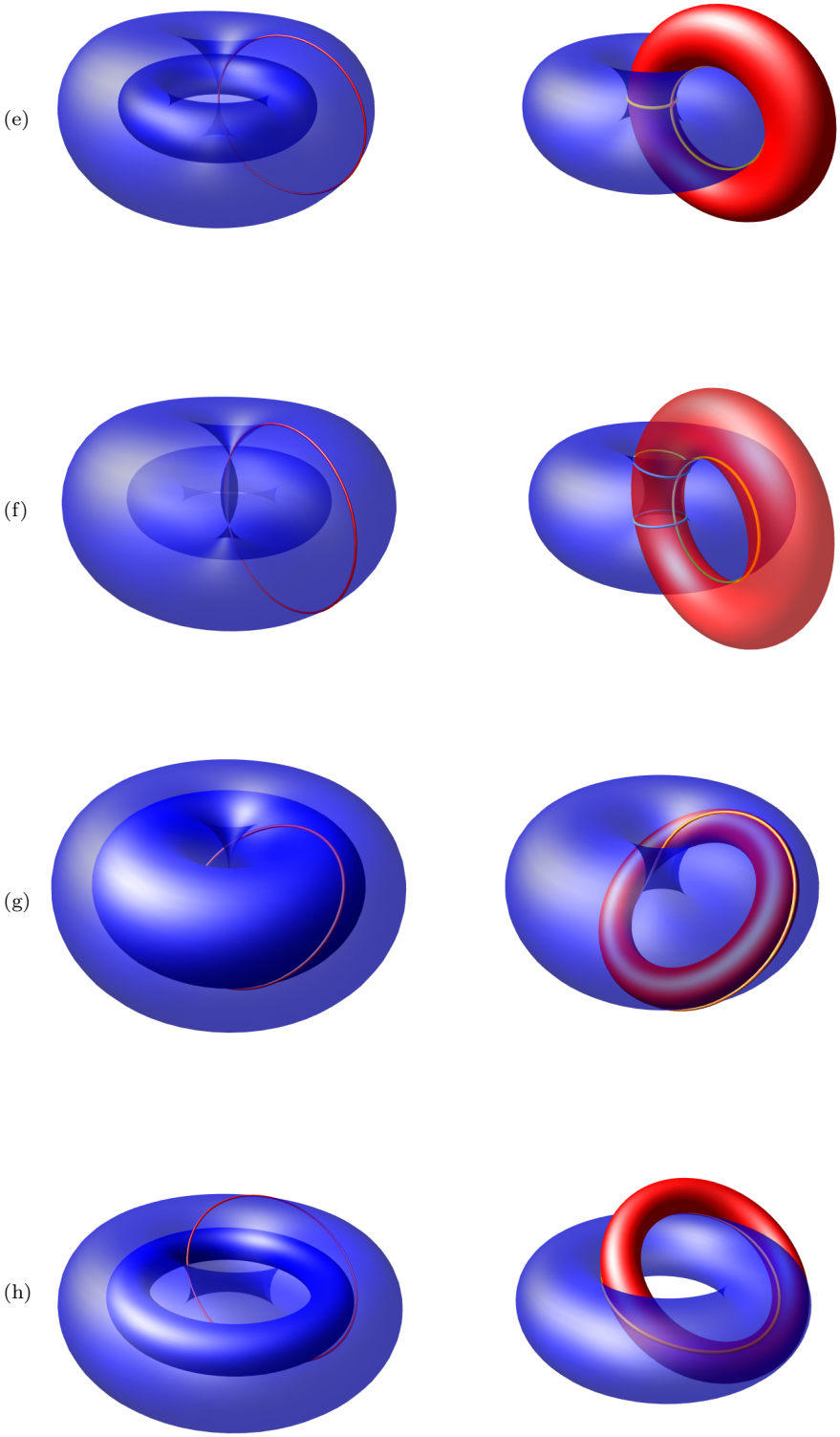


Figure 4.21: Torus-torus intersections (conic sections) - (*cont.*)

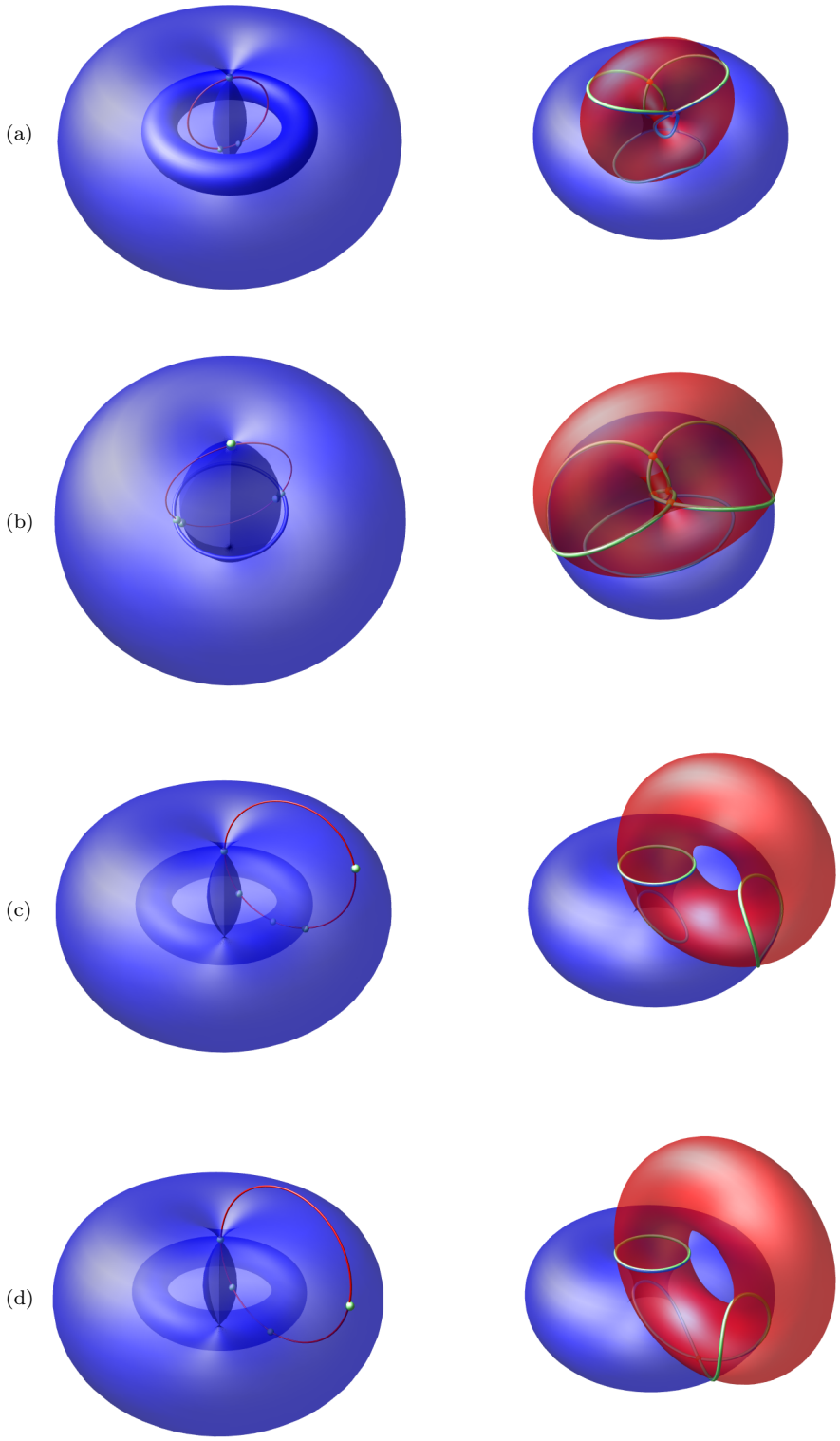


Figure 4.22: Torus-torus intersections (involving $\mathbf{q}^{\pm}$)

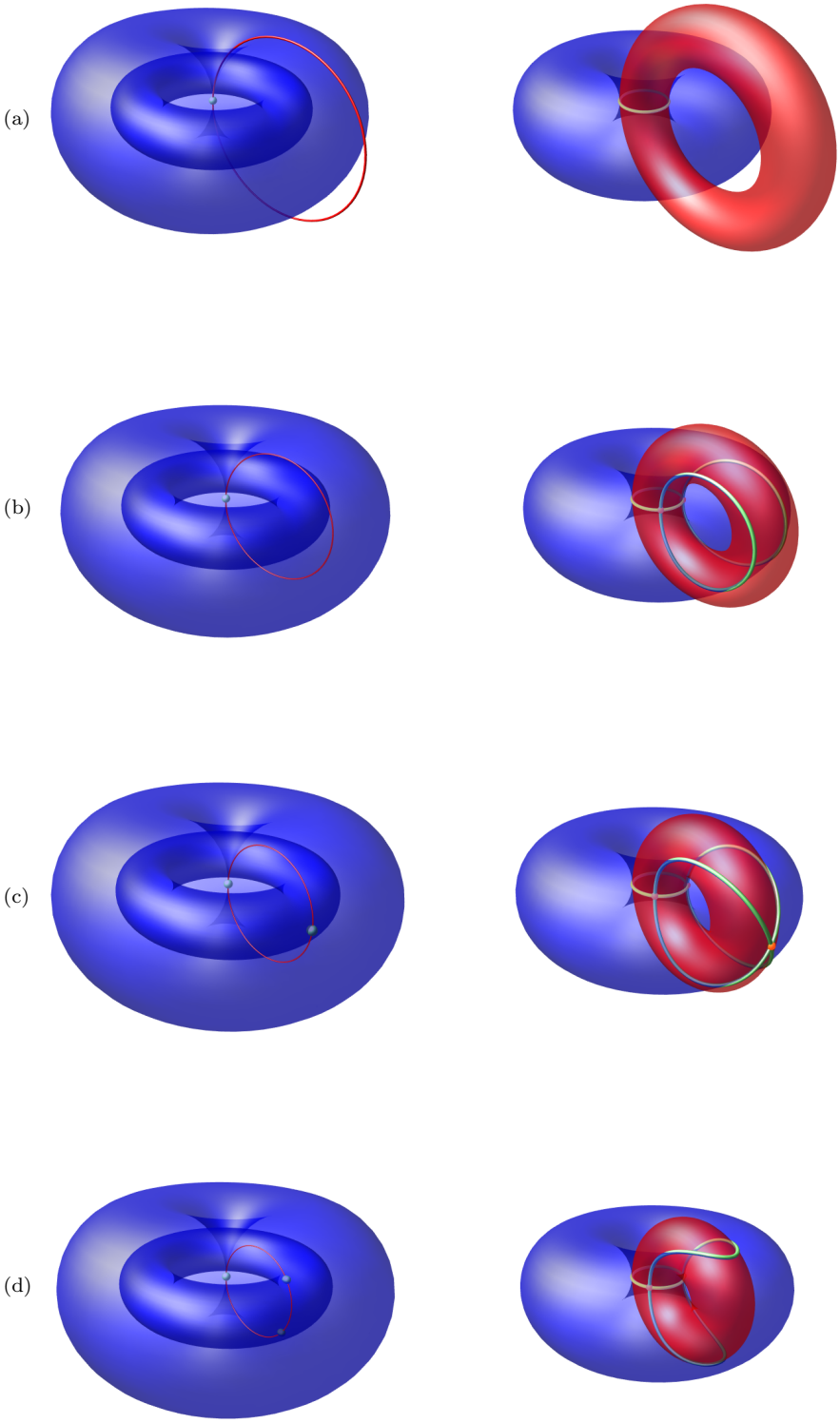
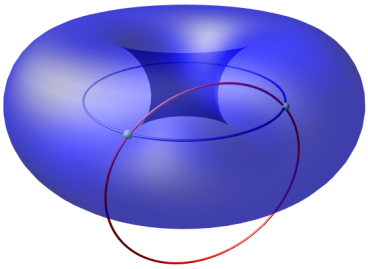
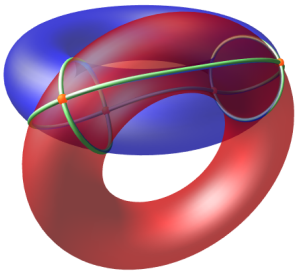


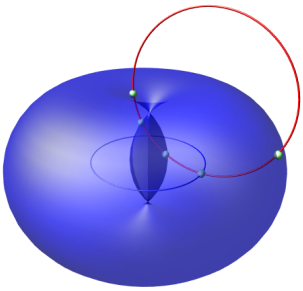
Figure 4.23: Torus-torus intersections (with horn torus T^O)

(a)





(b)



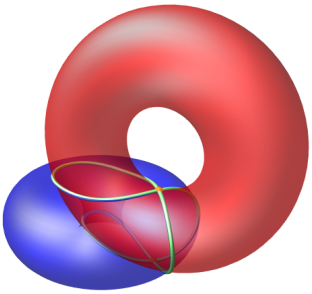


Figure 4.24: Torus-torus intersections (tangential to T^I while $r = \delta$)

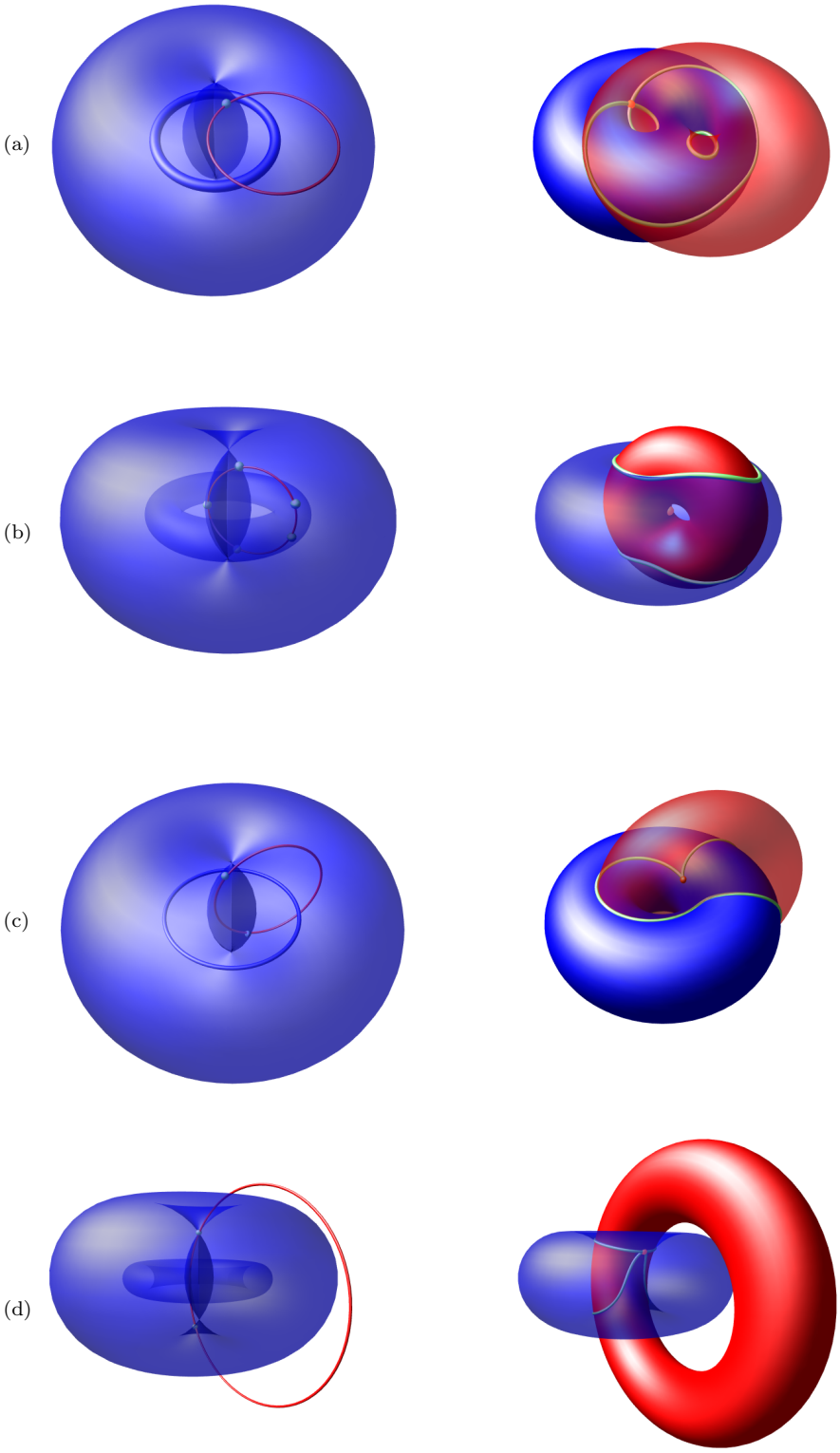


Figure 4.25: Torus-torus intersections (tangential to T^D)

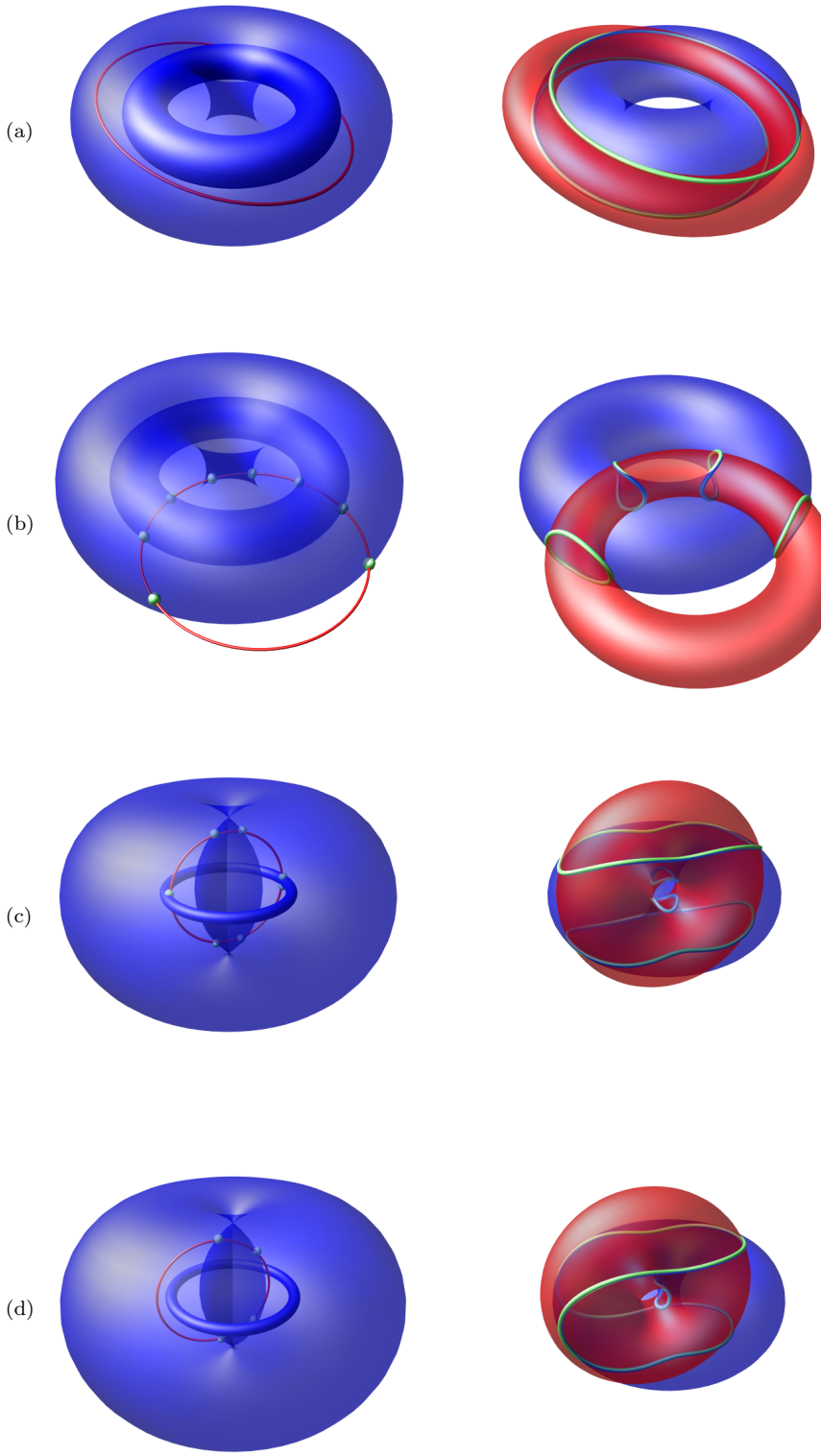


Figure 4.26: Torus-torus intersections (regular circle arcs)

Numeric Part

Chapter 5

Curve Tracing

In this chapter we compute an approximation of the intersection curve between two objects. Like in a typical marching method we therefor trace all intersection curve components from their initial starting points we already computed in the Algebraic Part before. For curves with a parametrisation this can be easily done by the evaluation of suitable parameter values, resulting in a list of points which are then connected by a linear spline, see Section 5.2. For curves without parametrisation we use a predictor-corrector step method. In general curve-tracing algorithms this method works as follows:

Given a continuous space curve α with starting point $\mathbf{q}_0$ and ending point $\mathbf{q}_e$. One step of the predictor-corrector method consists of:

- The predictor step, which makes a short step from the current curve point $\mathbf{q}_i$ into the tangential direction $\mathbf{n}(\mathbf{q}_i)$ of the curve, i.e. $\mathbf{p} = \mathbf{q}_i + \delta\mathbf{n}(\mathbf{q}_i)$, where δ is a suitable step size.

- The corrector step, which goes from point $\mathbf{p}$ back to the curve, resulting in the next curve point $\mathbf{q}_{i+1}$.

This scheme continues until the ending point $\mathbf{q}_e$ is reached. This method has to deal with the following problems:

- Computing the tangential direction $\mathbf{n}$ at each iteration point $\mathbf{q}_i$: In the case of singularities, this tangential direction vanishes, i.e. the method will loose the track.

- Determining the step size δ: If δ is too small, the progress of the tracing is slow and requires too much computation time. If δ is too big, we may miss some parts of the curve or the approximation will become inaccurate.

- The corrector step: This step uses numerical methods to find the next curve point $\mathbf{q}_{i+1}$. However, sometimes these methods do not converge.

In the following we will show that we can handle these problems. We give proofs for the correctness of our numerical methods, i.e. each curve component is traced correctly without missing parts or jumping on other curve components. Furthermore we verify the results, i.e. our numerical methods are able to handle any input and the resulting approximation lies arbitrarily close to the real curve.

We split this chapter into three parts: The regular case, where we trace regular intersection curve components which have no singularities. In this part we deal with the second and third problem. The case of parametrised curves as already mentioned above. The singular case, where we examine the local neighbourhood of singularities to solve the first problem. Here we use algebraical methods, since singular points are given exactly. We then lead back the tracing of singular curves to the regular case.

5.1 Regular Curves

In this section we approximate the intersection curve of two algebraic surfaces by a piecewise linear curve, using a predictor-corrector step method. In our implementation we use the data types `leda::bigfloat` respective `CORE::BigFloat`, which represent variable sized floating point numbers. Therefore the following computations take place in $\mathbb{R}$.

Let $f, g : \mathbb{R}^3 \to \mathbb{R}$ be the defining implicit functions of two algebraic surfaces F and G. Furthermore let $\mathbf{q}$ be a point on the intersection curve between F and G. A tangent direction to the curve at $\mathbf{q}$ is given by

$$\omega(\mathbf{q}) = \nabla f(\mathbf{q}) \times \nabla g(\mathbf{q}).$$

Note that the derivatives of f and g are defined since algebraic functions are polynomials and therewith elements of C^∞. The case $\omega(\mathbf{q}) = 0$ happens if and only if $\mathbf{q}$ is a singular point, i.e. the tangent planes of F and G at $\mathbf{q}$ are coplanar. In this section we consider regular curves and thus we can define the normalised vector field

$$\mathbf{n}(\mathbf{q}) = \frac{\nabla f(\mathbf{q}) \times \nabla g(\mathbf{q})}{\|\nabla f(\mathbf{q}) \times \nabla g(\mathbf{q})\|}. \tag{5.1}$$

In the following we consider a single step of the predictor-corrector method. Starting at a curve point $\mathbf{q}_i$, the predictor step becomes $\mathbf{p}_0 = \mathbf{q}_i + \delta\mathbf{n}(\mathbf{q}_i)$, where δ denotes the step size. For the step size δ we have to take three directives into account:

1. δ should be as big as possible for saving computation time.
2. δ depends somehow on the quality of the curve approximation.
3. δ has to be small enough for not missing some parts of the curve or accidentally jumping on other curve components.

Relating to directive 3 we use the interval test of [64], where interval arithmetic is used to decide, whether a given environment contains a single curve component. This interval test proceeds as follows:

Let $\mathbf{q}_i$ be the current approximation point on the curve, $\mathbf{p}_0$ the predicted point in tangential direction and δ the step size. Let further be $S = S_\delta(\mathbf{q}_i)$ the sphere around $\mathbf{q}_i$ containing $\mathbf{p}_0$ and

$$\mathbf{B} = [q_{ix} - \delta, q_{ix} + \delta] \times [q_{iy} - \delta, q_{iy} + \delta] \times [q_{iz} - \delta, q_{iz} + \delta]$$

the bounding box of S. With interval arithmetic we compute the interval

$$I = \langle \mathbf{n}(\mathbf{q}_i), \mathbf{n}(\mathbf{B}) \rangle.$$

Denote $\underline{X}$ as the lower bound and $\overline{X}$ as the upper bound of the interval X. Then it holds:

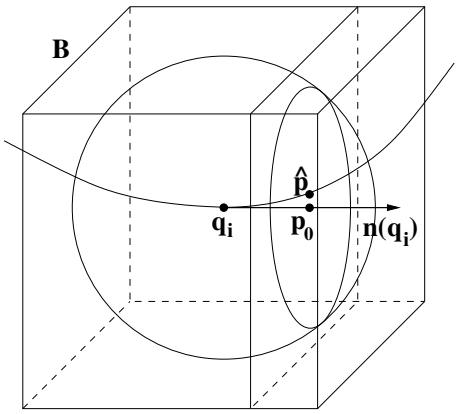


Figure 5.1: Interval test

Lemma 3. *If $\underline{I} > \frac{1}{2}\sqrt{2}$, the part of the intersection curve within S consists of a single component.*

The proof is based on a plane sweep perpendicular to $\mathbf{n}(\mathbf{q}_i)$. For a detailed description we refer to [64]. If the test fails we decrease the step size and try again. It is obvious that this procedure stops due to the continuity of f and g. If we pass the test, the relation $\underline{I} > \frac{1}{2}\sqrt{2}$ implies that the vector field (5.1) inside S does not deviate from $\mathbf{n}(\mathbf{q}_i)$ for more than $\pi/4$. Hence it follows directly

Corollary 1. *The intersection curve lies within the cone $K = K_1(\mathbf{q}_i, \mathbf{n}(\mathbf{q}_i))$ with apex $\mathbf{q}_i$, direction $\mathbf{n}(\mathbf{q}_i)$ and half angle $\pi/4$.*

However, we work with approximations, e.g. the curve points $\mathbf{q}_i$ lie within an environment around the exact curve, bounded by a given parameter ν. Thus we consider rather the offset surface of the cone K, created by expanding the surface about ν in the outward direction.

For the corrector step back to the intersection curve we use a Newton method. Since we work on two algebraic surfaces with implicit functions $f, g : \mathbb{R}^3 \to \mathbb{R}$, the Jacobian, which is involved by Newton methods, would become an 2×3 matrix. To proceed even though, we had to use a pseudo inverse matrix for the Newton iterations. Instead we add a third equation which keeps the Newton iteration on a plane perpendicular to the predictor step direction, see Figure 5.1. Thus we have more control over the area, in which we are searching for the next curve point. Additionally we save computation time, since our Jacobian becomes a square matrix whose inverse is easily to be computed, see below.

Let $\mathbf{n} := \mathbf{n}(\mathbf{q}_i)$ be the direction of the predictor step and $\mathbf{p}_0 = \mathbf{q}_i + \delta\mathbf{n}$ the predicted point. Let further $\mathbf{F} : \mathbb{R}^3 \to \mathbb{R}^3$ be a map defined by

$$\mathbf{F}(\mathbf{p}) = \begin{pmatrix} f(\mathbf{p}) \\ g(\mathbf{p}) \\ \langle \mathbf{p} - \mathbf{p}_0, \mathbf{n} \rangle \end{pmatrix} \quad \text{for} \quad \mathbf{p} \in \mathbb{R}^3.$$

A Newton iteration consists of

$$\mathbf{p}_{i+1} = \mathbf{p}_i - J_{\mathbf{F}}^{-1}(\mathbf{p}_i)\mathbf{F}(\mathbf{p}_i) \tag{5.2}$$

with the Jacobian

$$J_{\mathbf{F}} = \begin{pmatrix} \nabla f^T \\ \nabla g^T \\ \mathbf{n}^T \end{pmatrix}$$

and its inverse

$$J_{\mathbf{F}}^{-1} = \frac{1}{\langle \mathbf{n}, \nabla f \times \nabla g \rangle} \begin{pmatrix} \nabla g \times \mathbf{n} & \mathbf{n} \times \nabla f & \nabla f \times \nabla g \end{pmatrix}.$$

To be sure that the Newton iterations (5.2) converge, we make use of the Kantorowitsch lemma [41] which states:

Lemma 4 (Kantorowitsch)**.** *Let $\mathbf{F} : U \subset \mathbb{R}^3 \to \mathbb{R}^3$ be a map with*

$$\|J_{\mathbf{F}}(\mathbf{x}) - J_{\mathbf{F}}(\mathbf{y})\| \leq L_U \|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in U, \tag{5.3}$$

where $L_U > 0$ denotes the Lipschitz constant of $\mathbf{F}$ in the environment U. Let further be $\mathbf{p}_0 \in U$ and $\mathbf{p}_1 = \mathbf{p}_0 - \mathbf{h}_0$ with $\mathbf{h}_0 = J_{\mathbf{F}}^{-1}(\mathbf{p}_0)\mathbf{F}(\mathbf{p}_0)$ such that the ball $B = B_{\|\mathbf{h}_0\|}(\mathbf{p}_1) \subset U$. Set

$$\alpha_0 = L_B \|J_F^{-1}(\mathbf{p}_0)\| \|\mathbf{h_0}\|,$$

where $L_B > 0$ denotes the Lipschitz constant of $\mathbf{F}$ in B. If

$$\alpha_0 \leq \frac{1}{2}, \tag{5.4}$$

then there exists an unique $\hat{\mathbf{p}} \in B$ with $\mathbf{F}(\hat{\mathbf{p}}) = 0$. Furthermore the iteration (5.2) converge to $\hat{\mathbf{p}}$.

The proof is based on the Banach fixed point theorem. For a detailed description we refer to [57]. In the following we make estimations on the Lipschitz constants L_U and L_B additionally the Kantorowitsch condition (5.4). Since $\mathbf{F}$ consists of polynomials, the Lipschitz constant L_U exists for each bounded environment U and we can apply the Kantorowitsch lemma to our problem. However, instead of computing the probably smaller constant L_B at each step of our algorithm, we estimate L_U once for each object.

Let us denote the partial derivatives of a function f by $\partial_i f, i = 1..3$. Then we estimate

$$\begin{aligned}
|\partial_i f(\mathbf{x}) - \partial_i f(\mathbf{y})| &= \Big| \int_0^1 \langle \nabla \partial_i f(t\mathbf{x} + (1-t)\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle dt \Big| \\
&= \Big| \sum_{j=1}^{3} (x_j - y_j) \int_0^1 \partial_{ij} f(t\mathbf{x} + (1-t)\mathbf{y}) dt \Big| \\
&\leq \Big[\sum_{j=1}^{3} \int_0^1 |\partial_{ij} f(t\mathbf{x} + (1-t)\mathbf{y})| dt \Big] \|\mathbf{x} - \mathbf{y}\|. \qquad (5.5)
\end{aligned}$$

Since quadrics are algebraic surfaces of degree two, i.e.

$$f_Q(\mathbf{x}) = a_9 x^2 + a_8 y^2 + a_7 z^2 + 2a_6 xy + 2a_5 xz + 2a_4 yz + 2a_3 x + 2a_2 y + 2a_1 z + a_0,$$

their Hessian matrices have the form

$$H_Q(\mathbf{x}) = 2 \begin{pmatrix} a_9 & a_6 & a_5 \\ a_6 & a_8 & a_4 \\ a_5 & a_4 & a_7 \end{pmatrix} \quad \text{with} \quad a_i \in \mathbb{R}.$$

Thus we have

$$|\partial_{ij} f_Q(\mathbf{x})| \leq 2 \max_{k=4..9} \{|a_k|\} =: c_Q \quad \forall \mathbf{x} \in \mathbb{R}^3.$$

For a torus $T_{r,R}(\mathbf{o}, \mathbf{e}_3)$ in canonical position with implicit function

$$f_T(\mathbf{x}) = \left(\|\mathbf{x}\|^2 + R^2 - r^2 \right)^2 - 4R^2 \left(x^2 + y^2 \right)$$

we get

$$H_T(\mathbf{x}) = 4 \begin{pmatrix} \|\mathbf{x}\|^2 - R^2 - r^2 & 0 & 0 \\ 0 & \|\mathbf{x}\|^2 - R^2 - r^2 & 0 \\ 0 & 0 & \|\mathbf{x}\|^2 + R^2 - r^2 \end{pmatrix} + 8\mathbf{x} \circ \mathbf{x}^T,$$

where the operator $\circ$ denotes the dyadian product. Since we stay near the surface of the torus, say in an ϵ-environment, we can estimate

$$|\partial_{ij} f_T(\mathbf{x})| \leq 8((R + r + \epsilon)^2 + R(r + \epsilon)) =: c_T \quad \forall \mathbf{x} \in B_{R+r+\epsilon}(\mathbf{o}).$$

It is obvious that this upper bound c_T does not change for a torus in arbitrary position. Thus the integral of (5.5) is bounded and we can estimate further

$$\begin{aligned}
\|J_{\mathbf{F}}(\mathbf{x}) - J_{\mathbf{F}}(\mathbf{y})\|_2 &\leq \sqrt{3} \|J_{\mathbf{F}}(\mathbf{x}) - J_{\mathbf{F}}(\mathbf{y})\|_\infty \\
&= \sqrt{3} \max_{j=1..3} \sum_{i=1}^{3} |\partial_i F_j(\mathbf{x}) - \partial_i F_j(\mathbf{y})| \\
&= \sqrt{3} \max \Big\{ \sum_{i=1}^{3} |\partial_i f(\mathbf{x}) - \partial_i f(\mathbf{y})|, \sum_{i=1}^{3} |\partial_i g(\mathbf{x}) - \partial_i g(\mathbf{y})| \Big\} \\
&\leq 16 \max \{c_f, c_g\} \|\mathbf{x} - \mathbf{y}\|_2,
\end{aligned}$$

where the constants c_f and c_g are defined by either c_Q or c_T, depending on the object type of F and G, respectively. With the definition

$$L := 16 \max \{c_f, c_g\} \qquad (5.6)$$

the Lipschitz condition (5.3) holds.

We now consider condition (5.4). If the interval test (Lemma 3) holds, we have

$$I = \langle \mathbf{n}, \frac{\nabla f(\mathbf{B}) \times \nabla g(\mathbf{B})}{\|\nabla f(\mathbf{B}) \times \nabla g(\mathbf{B})\|} \rangle > \frac{1}{2}\sqrt{2}.$$

This holds for all $\mathbf{x} \in \mathbf{B}$, especially for $\mathbf{p}_0$. For the Jacobian inverse at $\mathbf{p}_0$ we make the estimation

$$\begin{aligned}
\|J_{\mathbf{F}}^{-1}\|_2 &\leq \frac{1}{|\langle \mathbf{n}, \nabla f \times \nabla g \rangle|} \left[\|\nabla g \times \mathbf{n}\|^2 + \|\mathbf{n} \times \nabla f\|^2 + \|\nabla f \times \nabla g\|^2 \right]^{\frac{1}{2}} \\
&\leq \frac{\sqrt{2}}{\|\nabla f \times \nabla g\|} \left[\|\nabla g\|^2 + \|\nabla f\|^2 + \|\nabla f \times \nabla g\|^2 \right]^{\frac{1}{2}} \\
&\leq \frac{\sqrt{2}}{\underline{\|\nabla f(\mathbf{B}) \times \nabla g(\mathbf{B})\|}} \left[\overline{\|\nabla g(\mathbf{B})\|}^2 + \overline{\|\nabla f(\mathbf{B})\|}^2 + \overline{\|\nabla f(\mathbf{B}) \times \nabla g(\mathbf{B})\|}^2 \right]^{\frac{1}{2}} \\
&=: A. \qquad (5.7)
\end{aligned}$$

With the mean value theorem we get

$$\begin{aligned}
|f(\mathbf{p}_0)| &= |f(\mathbf{q}_i) + \langle \nabla f(\mathbf{q}_i + t\mathbf{n}), \mathbf{p}_0 - \mathbf{q}_i \rangle| \quad \text{for a suitable} \quad t \in [0, \delta] \\
&\leq \max_{\mathbf{x} \in \mathbf{B}} \|\nabla f(\mathbf{x})\| \cdot \delta \\
&= \overline{\|\nabla f(\mathbf{B})\|} \cdot \delta.
\end{aligned}$$

For the first iteration of the Newton method (5.2) we estimate

$$\begin{aligned}
\|J_{\mathbf{F}}^{-1}\mathbf{F}(\mathbf{p}_0)\|_2 &\leq \frac{1}{|J_{\mathbf{F}}|} \left[|f(\mathbf{p}_0)|^2 \|\nabla g \times \mathbf{n}\|^2 + |g(\mathbf{p}_0)|^2 \|\mathbf{n} \times \nabla f\|^2 \right]^{\frac{1}{2}} \\
&\leq 2 \frac{\overline{\|\nabla g(\mathbf{B})\|} \cdot \overline{\|\nabla f(\mathbf{B})\|}}{\underline{\|\nabla f(\mathbf{B}) \times \nabla g(\mathbf{B})\|}} \cdot \delta \\
&=: C \cdot \delta. \qquad (5.8)
\end{aligned}$$

With (5.6)-(5.8) we have

$$\alpha_0 = L_B \|J_{\mathbf{F}}^{-1}\| \|J_{\mathbf{F}}^{-1}\mathbf{F}(\mathbf{p}_0)\| \leq LAC \cdot \delta$$

which can be made arbitrarily small by decreasing the step size δ.

If condition (5.4) holds as well, we perform some Newton iterations until the distance between the latter two iterates $\mathbf{p}_{j-1}$ and $\mathbf{p}_j$ is less than the given upper bound $\nu < \delta$. Lemma 4 states then, that the exact solution $\hat{\mathbf{p}}$ lies inside the ball $B_{\|\mathbf{p}_j - \mathbf{p}_{j-1}\|}(\mathbf{p}_j)$. Corollary 1 states that $\hat{\mathbf{p}}$ lies inside the cone with apex $\mathbf{q}_i$, direction $\mathbf{n}$ and half angle $\pi/4$. So if we consider the sphere $S_{\gamma_i}(\mathbf{q}_i)$ with $\gamma_i = \sqrt{2}\delta + \nu$ in the interval test, we can state that the exact curve is completely contained in the union of all spheres around the approximation points $\mathbf{q}_i$ of the curve, i.e. $U = \bigcup_i S_{\gamma_i}(\mathbf{q}_i)$. Thus the quality of approximation of the whole curve is depending on the step size δ and the Newton termination condition ν. If ν is much less than δ, more Newton iterations have to be performed. For a bigger ν, the predictor step direction $\mathbf{n}$ is less accurate. We made some good experiences with a value of $\nu = \frac{1}{100}\delta$. For a simpler computation we set $\gamma_i = 2\delta$.

To follow the directive 1, concerning the step size, we increase δ after each iteration, regarding the given approximation bound ϵ. Therewith we have an heuristic for an automatic and adaptive step size control.

In Algorithm 33 we summarise the results of this section. Due to generality with Section 5.3.2 we hand over a list of possible ending points. Whereas in case of regular loops we know, that the ending point $\mathbf{q}_e$ is equal to the starting point $\mathbf{q}_0$, in case of singular curve components the curve topology is first developed during the tracing. The parameter $\{\mathbf{q}_r\}$ is a list of the remaining initial starting points which are no ending points. This list results from the computation of redundant starting points in the Algebraic Part. We decide, whether a starting point $\mathbf{q}_r$ is redundant in order that the encounter of $\mathbf{q}_r$ inside an interval box $\mathbf{B}$ indicates that $\mathbf{q}_r$ already belongs to the curve component which is traced and thus can be deleted, see lines 29-33. Lines 9-17 determine the step size δ, fulfilling the interval test (Lemma 3) and the Kantorowitsch condition (5.4). Lines 20-26 are the Newton method, which terminates when the iterates are close to the intersection curve. In the last loop we check, if a possible ending point is reached.

Algorithm 33 `TraceRegularCurve`$(f, g, \mathbf{q}_0, \{\mathbf{q}_e\}, \{\mathbf{q}_r\}, \epsilon)$

Requires: Two algebraic functions f and g, a starting point $\mathbf{q}_0$, a set of possible ending points $\{\mathbf{q}_e\}$, a set of remaining initial starting points $\{\mathbf{q}_r\}$ and the upper approximation bound ϵ
Returns: A linear spline $[\mathbf{v}]$

$\delta \leftarrow \frac{1}{2}\epsilon$
$L \leftarrow 16 \max\{c_f, c_g\}$
$\mathbf{q} \leftarrow \mathbf{q}_0$
loop
  if $\delta < \frac{1}{2}\epsilon$ **then**
    $\delta \leftarrow 2\delta$
  end if
  $\mathbf{n} \leftarrow \frac{\nabla f(\mathbf{q}) \times \nabla g(\mathbf{q})}{\|\nabla f(\mathbf{q}) \times \nabla g(\mathbf{q})\|}$
  repeat
    $\mathbf{B} \leftarrow [q_x - \delta, q_x + \delta] \times [q_y - \delta, q_y + \delta] \times [q_z - \delta, q_z + \delta]$
    $\mathbf{F} \leftarrow \nabla f(\mathbf{B})$
    $\mathbf{G} \leftarrow \nabla g(\mathbf{B})$
    $\mathbf{H} \leftarrow \mathbf{F} \times \mathbf{G}$
    $I \leftarrow \langle \mathbf{n}, \frac{\mathbf{H}}{\|\mathbf{H}\|} \rangle$
    $\alpha \leftarrow 2\sqrt{2}\delta \frac{\overline{\|\mathbf{F}\| \cdot \|\mathbf{G}\|}}{\underline{\|\mathbf{H}\|^2}} \left(\overline{\|\mathbf{F}\|}^2 + \overline{\|\mathbf{G}\|}^2 + \overline{\|\mathbf{H}\|}^2 \right)^{\frac{1}{2}}$
    $\delta \leftarrow \frac{1}{2}\delta$
  until $\underline{I} > \frac{1}{2}\sqrt{2}$ **and** $\alpha \leq \frac{1}{2}$
  $\mathbf{p}_0 \leftarrow \mathbf{q} + \delta\mathbf{n}$
  $\mathbf{p} \leftarrow \mathbf{p}_0$
  repeat
    $\mathbf{v} \leftarrow \left(f(\mathbf{p}), g(\mathbf{p}), \langle \mathbf{p} - \mathbf{p}_0, \mathbf{n} \rangle \right)^T$
    $\mathbf{w} \leftarrow \nabla f(\mathbf{p}) \times \nabla g(\mathbf{p})$
    $A \leftarrow \begin{pmatrix} \nabla g(\mathbf{p}) \times \mathbf{n} & \mathbf{n} \times \nabla f(\mathbf{p}) & \mathbf{w} \end{pmatrix}$
    $\mathbf{d} \leftarrow \frac{1}{\langle \mathbf{n}, \mathbf{w} \rangle} A\mathbf{v}$
    $\mathbf{p} \leftarrow \mathbf{p} - \mathbf{d}$
  until $\|\mathbf{d}\| \leq \frac{1}{100}\delta$
  $\mathbf{q} \leftarrow \mathbf{p}$
  $[\mathbf{v}] \leftarrow \mathbf{q}$
  for all $\mathbf{q}_r \in \{\mathbf{q}_r\}$ **do**
    if $\mathbf{q}_r \in \mathbf{B}$ **then**
      $\{\mathbf{q}_r\} \leftarrow \{\mathbf{q}_r\} / \mathbf{q}_r$
    end if
  end for
  for all $\mathbf{q}_e \in \{\mathbf{q}_e\}$ **do**
    if $\mathbf{q}_e \in \mathbf{B}$ **and** $\langle \mathbf{n}, \mathbf{q}_e - \mathbf{q} \rangle > 0$ **then**
      $\{\mathbf{q}_e\} \leftarrow \{\mathbf{q}_e\} / \mathbf{q}_e$
      $[\mathbf{v}] \leftarrow \mathbf{q}_e$
      return $[\mathbf{v}]$
    end if
  end for
end loop

5.2 Parametric Curves

In opposite to regular curves, tracing parametric curves has the advantage that the curve is explicitly given. Thus we can omit time consuming Newton schemes to move back to the intersection curve. On the other hand we need a new heuristic to determine the next curve point, since the parametric curves are not given by arc length parameters.

A parametric curve consists of a homogeneous parameter form

$$\mathfrak{P}(t) = \begin{pmatrix} f_x(t) \\ f_y(t) \\ f_z(t) \\ f_w(t) \end{pmatrix} \quad \text{with} \quad f_x, f_y, f_z, f_w \in \mathbb{K}[x],$$

see Section 2.4.3. We assume that f_w has no roots and thus the dehomogenising step is well defined. Furthermore the degree of f_w is greater or equal the degrees of the other polynomials, since the curve is bounded. A point $\mathbf{q}$ on the curve at a given parameter t computes to

$$\mathbf{q}(t) = \texttt{dehom}(\mathfrak{P}(t)) = \left(\frac{f_x(t)}{f_w(t)}, \quad \frac{f_y(t)}{f_w(t)}, \quad \frac{f_z(t)}{f_w(t)} \right).$$

Since the parameter domain includes infinity, see case 3 in Section 4.4.5, we could evaluate $\mathbf{q}_\infty$ symbolically by considering the respective leading coefficients of the polynomials. However, evaluations in the closer environment of $\mathbf{q}_\infty$, seen geometrically, become unstable. Hence we prefer to split the curve into the parts $\mathfrak{P}(t)$ and $\mathfrak{P}(\frac{1}{t})$ with $t \in [-1, 1]$ for each part. By the substitution of $t' = \frac{1}{t}$ into the given polynomials we gain a second homogeneous parameter form, which we denote by *reciprocal form*.

Let $d = \deg(f_w)$ be the maximum degree of the polynomials of $\mathfrak{P}$. A representative polynomial f' of the reciprocal form $\mathfrak{P}'$ is then given by

$$f'(t) = \sum_{i=0}^{d} \texttt{coeff}(f, d-i) t^i,$$

where the predicate `coeff` may return zero, if the respective monomial of f does not exist. Analogously let $\mathbf{q}'(t) = \texttt{dehom}(\mathfrak{P}'(t))$ be a point on the reciprocal curve part.

Since we have an exact parametrisation of the curve, we derive the normalised tangential direction $\mathbf{n}$ of the curve from

$$\mathbf{n} = \frac{\frac{d}{dt}\mathbf{q}}{\|\frac{d}{dt}\mathbf{q}\|} = \frac{(\frac{d}{dt}\mathbf{f})f_w - \mathbf{f}(\frac{d}{dt}f_w)}{\|(\frac{d}{dt}\mathbf{f})f_w - \mathbf{f}(\frac{d}{dt}f_w)\|} \quad \text{with} \quad \mathbf{f} = \Big(f_x, \quad f_y, \quad f_z \Big).$$

Let $I = [t_1, t_2]$ an interval and $\mathbf{q}_i = \mathbf{q}(t_i)$ for $i = 1, 2$ the curve points at the interval bounds with $\beta = \|\mathbf{q}_1 - \mathbf{q}_2\|$. By the same argument we followed Corollary 1 from Lemma 3, we state

Corollary 2. *If* $\frac{1}{\beta}\langle \mathbf{n}(I), \mathbf{q}_1 - \mathbf{q}_2 \rangle > \frac{1}{2}\sqrt{2}$*, the curve lies within the sphere* $S = S_{\frac{1}{2}\beta}(\frac{1}{2}(\mathbf{q}_1 + \mathbf{q}_2))$.

Let ϵ be the approximation error bound. The restriction of $\beta < 2\epsilon$ guarantees that the exact curve lies inside the ϵ-environment of the linear spline approximation. By an adaptive step size control we try to maximise the step size to spare computation time.

We note that the only parametric curves, which occur in our approach, come from case 3 in Section 4.4.5. In parameter space of the torus in canonical position, these curves are represented by straight lines through the origin, see (4.4). Hence we omit the examination of self-intersections and branches which lie very close.

Algorithm 34 summarises the results of this section.

Algorithm 34 `TraceParametricCurve`$(\mathfrak{P}, \epsilon)$

Requires: A homogeneous parameter form $\mathfrak{P}$ and the upper approximation bound ϵ
Returns: A linear spline $[\mathbf{v}]$

$\mathbf{q} \leftarrow$ the real curve parametrisation
$\mathbf{n} \leftarrow$ the normalised tangential direction
$\mathbf{q}' \leftarrow$ the reciprocal curve parametrisation
$\mathbf{n}' \leftarrow$ the normalised tangential direction of the reciprocal form
$t \leftarrow -1$
$\delta \leftarrow 1$
repeat
  while $\|\mathbf{q}(t+\delta) - \mathbf{q}(t)\| < 2\epsilon$ **do**
    $\delta \leftarrow 2\delta$
  end while
  while $\|\mathbf{q}(t+\delta) - \mathbf{q}(t)\| > 2\epsilon$ **do**
    $\delta \leftarrow \frac{1}{2}\delta$
  end while
  while $\langle \mathbf{n}([t, t+\delta]), \mathbf{q}(t+\delta) - \mathbf{q}(t)\rangle < \frac{1}{2}\sqrt{2}\|\mathbf{q}(t+\delta) - \mathbf{q}(t)\|$ **do**
    $\delta \leftarrow \frac{1}{2}\delta$
  end while
  $[\mathbf{v}] \leftarrow \mathbf{q}(t)$
  $t \leftarrow t + \delta$
until $t > 1$
$[\mathbf{v}] \leftarrow \mathbf{q}(1)$
$t \leftarrow 1$
repeat
  while $\|\mathbf{q}'(t) - \mathbf{q}'(t-\delta)\| < 2\epsilon$ **do**
    $\delta \leftarrow 2\delta$
  end while
  while $\|\mathbf{q}'(t) - \mathbf{q}'(t-\delta)\| > 2\epsilon$ **do**
    $\delta \leftarrow \frac{1}{2}\delta$
  end while
  while $\langle \mathbf{n}'([t-\delta, t]), \mathbf{q}'(t) - \mathbf{q}'(t-\delta)\rangle < \frac{1}{2}\sqrt{2}\|\mathbf{q}'(t) - \mathbf{q}'(t-\delta)\|$ **do**
    $\delta \leftarrow \frac{1}{2}\delta$
  end while
  $[\mathbf{v}] \leftarrow \mathbf{q}'(t)$
  $t \leftarrow t - \delta$
until $t < -1$
$[\mathbf{v}] \leftarrow \mathbf{q}'(-1)$
return $[\mathbf{v}]$

5.3 Handling Singularities

In Section 5.1 we defined the normalised vector field $\mathbf{n}$ (5.1) by the cross product of the gradients of the two surfaces F and G. At a singular point $\mathbf{q}$ the vector $\mathbf{n}(\mathbf{q})$ vanishes, since F and G share the same tangent plane at $\mathbf{q}$. Thus we cannot determine the direction of the intersection curve at singular points in this way. Instead we consider higher order derivatives of F and G at $\mathbf{q}$ by algebraic methods. Since singular points are given exactly by two-root numbers at most, we can perform the determination of curve directions exactly as well. This is important for the distinction of different types of singularities. During the classification of the intersection curve topology, see [46], six different types of singular points occur:

Type A: Singular points. connecting traversal regular curve components (Figures 3.3(c), 3.4(c), 3.5(c), 3.6(c), 3.7(c), 3.8(b), 3.9(a)-(f), 4.5(b), 4.9(c), 4.10(e)+(f), 4.13(c)-(e), 4.14(c)+(e)-(j), 4.17(a), 4.18(b), 4.19(a)+(b), 4.22(a)+(b), 4.23(c), 4.24, 4.25(a)).

Type B: Singular points. connecting tangential regular curve components (Figures 4.14(l), 4.17(c)+(d), 4.25(d)).

Type C: Singular points. connecting a regular curve component with a singular circle, where each point of the circle is a touching point between the two involved surfaces (Figures 3.5(b), 3.8(a), 4.21(f), 4.23(b)-(d)).

Type D: Singular points connecting two singular circles (Figure 4.21(e)).

Type E: Isolated singular points or touching points (Figures 4.18(a), 4.25(b)).

Type F: Cusps (Figures 4.18(c)+(d), 4.19(b), 4.25(c)).

This section splits into two parts: First the examination of the local environment of the singular point $\mathbf{q}$ by algebraic methods. In this part we already distinguish between simple singularities like Types A and B, isolated singularities (Type E) and higher order singularities like Types C and D. Furthermore we compute the directions of the intersection curve at the singular point $\mathbf{q}$. In the second part we make an estimation on the first step size in each computed intersection curve direction to determine an initial starting point on each branch of the singular point $\mathbf{q}$. From the set of these initial starting points, we trace each curve component using the routine `TraceRegularCurve` from Section 5.1.

5.3.1 Local environment examination

Let F and G be the two algebraic surfaces with defining implicit functions f and g, respectively. Let further $\mathbf{q}$ be a singular point on the intersection curve of F and G with

$$\nabla f(\mathbf{q}) \times \nabla g(\mathbf{q}) = 0. \tag{5.9}$$

Let $\phi : U \subset \mathbb{R} \to \mathbb{R}^3$ be the parametrisation of the intersection curve in the local environment of $\mathbf{q}$ such that

$$f(\phi(\tau)) = g(\phi(\tau)) \equiv 0 \quad \forall \tau \in U$$

and $\phi(\tau_0) = \mathbf{q}$ for an appropriate $\tau_0 \in U$. Then the second derivative of f computes to

$$\begin{aligned} 0 \equiv \frac{d^2}{d\tau^2} f(\phi) &= \frac{d}{d\tau}\left[\langle \frac{\partial f}{\partial \phi}, \frac{d\phi}{d\tau}\rangle\right] \\ &= \langle \frac{d}{d\tau}\frac{\partial f}{\partial \phi}, \frac{d\phi}{d\tau}\rangle + \langle \frac{\partial f}{\partial \phi}, \frac{d^2\phi}{d\tau^2}\rangle \\ &= \langle \frac{\partial^2 f}{\partial \phi_i \partial \phi_j} \cdot \frac{d\phi}{d\tau}, \frac{d\phi}{d\tau}\rangle + \langle \frac{\partial f}{\partial \phi}, \frac{d^2\phi}{d\tau^2}\rangle \\ &= \mathbf{w}^T H_f \mathbf{w} + (\nabla f)^T \left(\frac{d^2\phi}{d\tau^2}\right). \end{aligned} \tag{5.10}$$

The matrix H_f denotes the Hessian matrix of f and $\mathbf{w} := \frac{d\phi}{d\tau}$ denotes the tangential direction of the parametrisation ϕ. Vector $\mathbf{w}$, evaluated at τ_0, becomes the direction of the intersection curve at the singular point $\mathbf{q}$. Analogue, we have for the function g

$$0 \equiv \frac{d^2}{d\tau^2} g(\phi) = \mathbf{w}^T H_g \mathbf{w} + (\nabla g)^T \left(\frac{d^2\phi}{d\tau^2} \right). \tag{5.11}$$

Since there is always a torus involved in our intersections and a torus is a regular surface, i.e. the gradient does not vanish, we may assume without loss of generality, that $\nabla f(\mathbf{p}) \neq 0$ for all points $\mathbf{p}$ on the surface F. From (5.9) we follow then $\nabla g(\mathbf{q}) = \lambda \nabla f(\mathbf{q})$ with either $\lambda = 0$ in case of $\nabla g(\mathbf{q}) = 0$, what can only occur at the apex of the cone, or otherwise

$$\lambda = \frac{\langle \nabla f(\mathbf{q}), \nabla f(\mathbf{q}) \rangle}{\langle \nabla f(\mathbf{q}), \nabla g(\mathbf{q}) \rangle}.$$

With (5.10), (5.11) and setting $A := H_g(\mathbf{q}) - \lambda H_f(\mathbf{q})$ we get

$$\mathbf{w}^T A \mathbf{w} = 0. \tag{5.12}$$

Let $\mathbf{u}$ and $\mathbf{v}$ be two linear independent vectors, lying in the tangent plane at $\mathbf{q}$, with $\mathbf{u} \times \mathbf{v} = \nabla f(\mathbf{q}) =: \mathbf{n}$. Then we can write

$$\mathbf{w} = \alpha \mathbf{u} + \beta \mathbf{v}$$

with appropriate $\alpha, \beta \in \mathbb{R}$, not both zero. From (5.12) we get

$$\alpha^2 \mathbf{u}^T A \mathbf{u} + 2\alpha\beta \mathbf{u}^T A \mathbf{v} + \beta^2 \mathbf{v}^T A \mathbf{v} = 0 \tag{5.13}$$

what can be solved for $\mathfrak{t} = (\alpha, \beta) \in \mathbb{P}^1$ as follows

$$(\alpha, \beta) = (-\mathbf{u}^T A \mathbf{v} \pm \sqrt{(\mathbf{u}^T A \mathbf{v})^2 - (\mathbf{u}^T A \mathbf{u})(\mathbf{v}^T A \mathbf{v})}, \mathbf{u}^T A \mathbf{u}).$$

The expression under the square root can be simplified to

$$(\mathbf{v} \times \mathbf{u})^T (A\mathbf{u} \times A\mathbf{v}) = -\mathbf{n}^T adj(A) \mathbf{n} =: D, \tag{5.14}$$

where $adj(A)$ denotes the adjugate matrix of A, see [71]. Equation (5.14) shows that the number of solutions does not depend on the choice of $\mathbf{u}$ and $\mathbf{v}$.

This leads to the cases:

1. $\mathbf{u}^T A \mathbf{u} = \mathbf{v}^T A \mathbf{v} = 0$, i.e. the vectors $\mathbf{u}$ and $\mathbf{v}$ are already solutions of (5.12):
   a. $\mathbf{u}^T A \mathbf{v} = 0$: Equation (5.12) holds for all vectors in the tangent plane at $\mathbf{q}$. This means that the second derivatives of f and g are equivalent. Thus the singularity at $\mathbf{q}$ is of order three at least (Types C and D).
   b. $\mathbf{u}^T A \mathbf{v} \neq 0$: From (5.13) we get either $\alpha = 0$ or $\beta = 0$, which coincides with the two solutions $\mathbf{w}_1 = \mathbf{u}$ and $\mathbf{w}_2 = \mathbf{v}$. Thus $\mathbf{q}$ is of Type A.
2. W.l.o.g. let be $\mathbf{u}^T A \mathbf{u} \neq 0$: We restrict (5.13) by setting $\beta = 1$ and solve for α which yields

$$\alpha_{1,2} = -\frac{\mathbf{u}^T A \mathbf{v} \pm \sqrt{D}}{\mathbf{u}^T A \mathbf{u}}.$$

   Depending on the sign of D we have
   a. $D < 0$: There is no solution for α and the singular point $\mathbf{q}$ is isolated (Type E).
   b. $D = 0$: There is only one solution α which means that all branches of the singular point $\mathbf{q}$ share the same tangential direction $\mathbf{w} = \alpha \mathbf{u} + \mathbf{v}$ (Types B and F).
   c. $D > 0$: There are two solutions $\alpha_{1,2}$ and hence two different tangential directions $\mathbf{w}_1 = \alpha_1 \mathbf{u} + \mathbf{v}$ and $\mathbf{w}_2 = \alpha_2 \mathbf{u} + \mathbf{v}$ (Type A).

Since conic sections can be represented efficiently, it suffices to trace the remaining curve components. In the case that the singular point $\mathbf{q}$ is embedded in a conic section, we neglect the corresponding tangential direction $\mathbf{w}$. Since singular points as well as conic sections are given exactly by some algebraically extended number types, we are able to decide correctly whether a singular point is embedded in a conic section.

Let $\mathbf{q}$ be a singular point and $C = C_r(\mathbf{p}, \mathbf{n}) \in \{C\}$ a conic section of the intersection curve. The point $\mathbf{q}$ is embedded in C, if and only if

$$\langle \mathbf{q} - \mathbf{p}, \mathbf{n} \rangle = 0 \quad \text{and} \quad \|\mathbf{q} - \mathbf{p}\|^2 = r^2. \tag{5.15}$$

If $\mathbf{q}$ and C are defined by numbers of different algebraic extensions, say $\mathbb{K}(\sqrt{c})(\sqrt{d})$ and $\mathbb{K}(\sqrt{e})$, respectively, we check conditions (5.15) in the extension field $\mathbb{K}(\sqrt{c})(\sqrt{d})(\sqrt{e})$, where we resolve appearing radicals by using repeated squaring. If the conditions hold, the tangential direction $\mathbf{w}$ with $\langle \mathbf{w}, \mathbf{n} \rangle = 0$ corresponds to C and hence can be neglected.

Figure 4.21(e) shows the only case, where a singular point is embedded in two conic sections. In this case nothing is to do since there is no part left to be traced. In such a way we distinguish between points of Type C and Type D in case 1.a.

To compute the tangential directions of the remaining intersection curve branches at Type C points, we notice, that all these intersection curves are symmetric to the plane, the conic section is embedded in, see Figures 3.5(b), 3.8(a), 4.21(f), 4.23(b)-(d). Thus both tangential directions $\mathbf{w}_{1,2}$ at the singular point $\mathbf{q}$ are orthogonal to each other and the direction, tangential to the remaining curve branches, is $\mathbf{w} = \mathbf{n}$, where $\mathbf{n}$ is the normal of the conic section.

So far, we can classify a given singular point $\mathbf{q}$ into one of the Types A-F, where we cannot distinguish between Types B and F at this point. Moreover we have a tangential direction of each intersection curve branch at $\mathbf{q}$, which needs to be traced. All necessary decisions and computations, as are required therefor, perform exactly.

5.3.2 First step size estimation

To be able to trace intersection curve branches from a singular point $\mathbf{q}$, we need to do a first step away from $\mathbf{q}$, such that the interval test, see Lemma 3 in Section 5.1, can be applied. Singular points have either no branches in case of touching points or at least two branches. Thus the interval test always fails at singular points. So we are looking for an environment U of $\mathbf{q}$ with the following properties:

1. There is exactly one singular point $\mathbf{q}$ inside U.
2. There are as many intersection points $\mathbf{p}_i$ between the intersection curve and the boundary of U as branches outgoing from $\mathbf{q}$.
3. The distance between the intersection curve and the linear approximation $\overline{\mathbf{q}\mathbf{p}_i}$ inside U never exceeds the given approximation error bound ϵ.
4. There are no closed loops inside U.

Once we found such an environment U, we use the algorithm `TraceRegularCurve` to trace each intersection curve branch between the points $\mathbf{p}_i$, where we distinguish between starting and ending points. Since we trace curves in the direction of the normalised vector field (5.1), starting points are those points $\mathbf{p}_i$ with

$$\langle \mathbf{p}_i - \mathbf{q}, \mathbf{n}(\mathbf{p}_i) \rangle > 0.$$

The remaining points are ending points.

Properties 1 and 3 can easily be maintained by defining

$$U = B_\delta(\mathbf{q}) \quad \text{with} \quad \delta = \min\{\epsilon, \frac{1}{2}\|\mathbf{q} - \mathbf{q}_j\|\},$$

where $\mathbf{q}_j$ denote the other singular points. Regarding the remaining properties we consider the parameter space of the torus.

W.l.o.g. let be F the surface of the torus with rational parameter form $T(s,t)$, see Section 3.7, and g the algebraic function of the other surface, see Section 2.2. Then

$$h(s,t) := g(T(s,t)) = 0$$

is the implicit equation of the intersection curve in the parameter space of the torus. Similar to Section 5.3.1 let

$$\phi : V \subset \mathbb{R} \to \mathbb{R}^3$$

be the parametrisation of the intersection curve in the local environment U of $\mathbf{q}$ such that

$$\begin{aligned} g(\phi(V)) &\equiv 0, \\ \phi(\tau_0) &= \mathbf{q} \end{aligned}$$

for an appropriate $\tau_0 \in V$. Analogue, let

$$\Phi : W \subset \mathbb{R} \to \mathbb{R}^2$$

be the parametrisation of the intersection curve in parameter space with

$$h(\Phi(W)) \equiv 0, \tag{5.16}$$

$$T(\Phi(W)) \equiv \phi(V), \tag{5.17}$$

$$T(\Phi(\sigma_0)) = T(s_0,t_0) = \mathbf{q} = \phi(\tau_0), \tag{5.18}$$

where

$$s_0 = \begin{cases} \infty & \text{, if } \|\mathbf{q}\| = R - r \\ \dfrac{2Rq_z}{\|\mathbf{q}\|^2 - (R-r)^2} & \text{, otherwise} \end{cases}$$

$$t_0 = \begin{cases} 0 & \text{, if } q_y = 0 \wedge q_x \geq 0 \\ \infty & \text{, if } q_y = 0 \wedge q_x < 0 \\ \dfrac{\|\mathbf{q}\|^2 + R^2 - r^2 - 2Rq_x}{2Rq_y} & \text{, if } q_y \neq 0 \end{cases}$$

are the parameters of the singular point $\mathbf{q}$ in parameter space. To avoid evaluations at infinity we may substitute $\hat{s} := \frac{1}{s}$ and $\hat{t} := \frac{1}{t}$, respectively. This results in some sign changes in the parameter form T, see Section 2.2.3. Thus we can disregard these cases.

From (5.17) we get by differentiation

$$\frac{d\phi}{d\tau} = \frac{d}{d\sigma} T(\Phi) = J_T \cdot \frac{d\Phi}{d\sigma}, \tag{5.19}$$

where $J_T = (\frac{\partial T_i}{\partial \Phi_j})_{i,j}, i = 1..3, j = 1..2$ denotes the Jacobian matrix of T. Therewith we have a direct correlation between tangents in parameter space and tangents in real affine space. Let $\mathbf{w}_{1,2}$ be the tangential directions of the intersection curve from Section 5.3.1, we get the tangents of the intersection curve at the singular point (s_0,t_0) in parameter space by

$$\mathbf{v}_i := \frac{d\Phi}{d\sigma}\Big|_{\sigma_0} = J_T^{inv}\Big|_{(s_0,t_0)} \mathbf{w}_i \quad , i = 1..2, \tag{5.20}$$

where $J_T^{inv} := (J_T^T J_T)^{-1} J_T^T$ denotes a pseudo inverse of J_T. Due to this pseudo inverse the equal sign in (5.20) means equal up to a constant factor. However, we are just interested in the directions, not their lengths. In case we have only one single direction $\mathbf{w}$, we set $\mathbf{v}_1 = J_T^{inv}\big|_{(s_0,t_0)} \mathbf{w}$ and $\mathbf{v}_2 = (v_{1y}, -v_{1x})$ to ensure linear independence.

By the substitution

$$\begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{pmatrix} \begin{pmatrix} \tilde{s} \\ \tilde{t} \end{pmatrix} + \begin{pmatrix} s_0 \\ t_0 \end{pmatrix}$$

we get a representation of the intersection curve $\tilde{h}(\tilde{s},\tilde{t})$ where the singular point lies at the origin and the outgoing branches are tangential to the coordinate axes.

Now let B be a box around the origin such that its image under $\tilde{T}$ is a subset of the environment U. Furthermore set

$$h^+(\tilde{s}) := \tilde{h}(\tilde{s}, \tilde{s}),$$
$$h^o(\tilde{s}) := \tilde{h}(\tilde{s}, 0),$$
$$h^-(\tilde{s}) := \tilde{h}(\tilde{s}, -\tilde{s}).$$

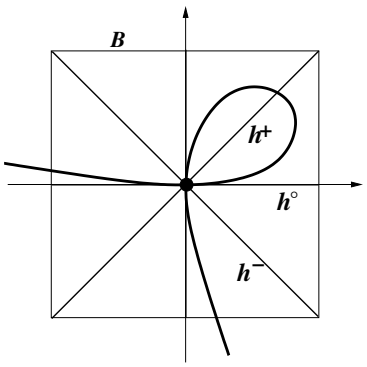


Figure 5.2: Singular loop

If d is the degree of the singularity $\mathbf{q}$, we divide h^+, h^o and h^- by $\tilde{s}^d$. If the remaining terms have no further root inside B, we can conclude that no singular loop lies inside B, see Figure 5.2. Otherwise we decrease the box B and try again. Fortunately we can further exclude regular loops inside B, since in each possible topological case involving singularities, all occurring regular curve components take course either around the hole of the torus or around its tube, see [46], i.e. in parameter space of the torus these components are unbounded. Roughly speaking they cross infinity.

It remains to consider property 2. With the algebraic examination of the singular point $\mathbf{q}$ alone we cannot distinguish between singularities with tangential intersection curve branches and cusps, since we just know the number of tangents of the intersection curve. However, by means of the intersections between the intersection curve and the boundary of B we identify the following cases, see Figure 5.3:

1. We have two tangents and there is exactly one intersection point at the left, right, upper and lower boundary, respectively.

2. We have one tangent and there are exactly two intersection points on the left boundary and two points on the right one.

3. We have one tangent and there are exactly two intersection points at either the left or the right boundary, i.e. the singularity is a cusp.

If none of the above cases are present, we refine the box B and try again. Thus we cannot only identify cusps which are heavy to handle algebraically, we even found the starting and ending points $\tilde{\mathbf{p}}_i$ in parameter representation. To gain the starting and ending points $\mathbf{p}_i$ in real affine space, we just evaluate $\tilde{T}$ at the points $\tilde{\mathbf{p}}_i$.

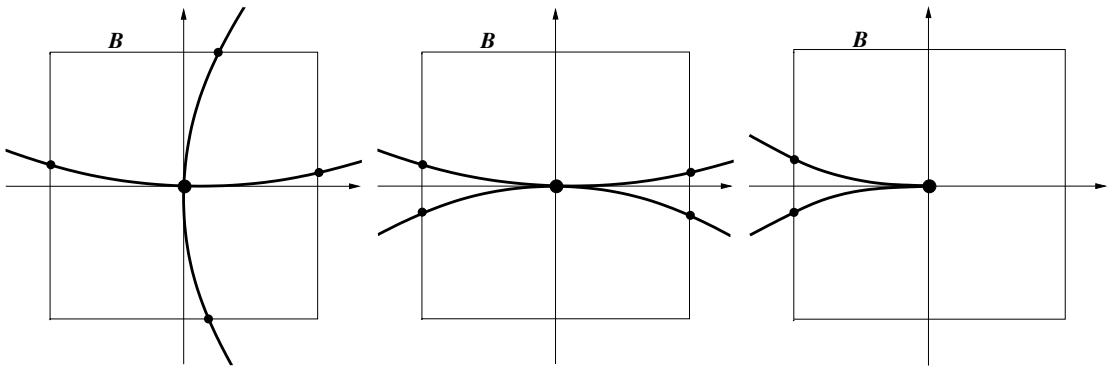


Figure 5.3: Several singular point topologies

Chapter 6

Conclusion and Outlook

In this thesis we provided algorithms to compute the intersection curve between a torus and a simple surface like a plane, a sphere, a circular cylinder, a circular cone and another torus. We presumed that the objects were given exactly by a geometrical representation over a field $\mathbb{K}$. We presented all necessary and sufficient conditions for the detection of conic sections in the intersection curve, where the required computations could be performed exactly. We showed, that just one algebraic field extension $\mathbb{K}(\sqrt{\alpha})$ suffices to represent any conic section. In contrast to other approaches, we could avoid the expensive analysis of sophisticated algebraic equation systems, regarding singularities of the intersection curve, and there was no need to transform the objects into a generic configuration. Instead, we were able to detect each singularity directly by just solving quadratic equations. Moreover, we proofed, that any singularity can be represented exactly by double nested field extensions $\mathbb{K}(\sqrt{\beta})(\sqrt{\gamma})$ in the worst case. Regarding the intersection curve itself, we provided algorithms for the correct tracing of any curve component like regular closed loops, curve components, connected by singularities, and completely tangential curve components. For the tangential curve components we even provided rational parametrisations. The numerical methods, applied during the curve tracing, are robust and fast, resulting in a topologically correct approximation of the real intersection curve with respect to any arbitrary given error bound. However, there are still several open problems:

The representation of traced intersection curve components. Our curves are given by a set of linear splines, which is the simplest approximation scheme. Some applications may demand higher order continuity on the intersection curve, for instance, for a smooth rendering which requires accurate normals. One could think of NURBS approximations like in [3].

The expansion of the set of objects. We considered a small group of objects, consisting of natural quadrics and tori. This ensured that offset surfaces in configuration space belonged to the same class of objects, what made it easier to handle these offset surfaces algebraically. However, this is not a necessary condition for the configuration space approach itself as long as the intersection of objects in configuration space is more practicable than intersecting the corresponding objects in real affine space or the parameter space of one of the objects.

The extension to a boundary computation. Our goal is to develop a framework for the exact boundary computation of curved objects. This framework could then serve as a kernel for a CSG to B-rep conversion in CAD/CAM applications. Therefore data structures are required, which store the result of a single Boolean operation. The remaining work is then of combinatorial nature. Such a data structure for polyhedra was introduced by Nef [59] and extended by various authors [11, 22, 33, 38]. An implementation in the context of CGAL is provided by Granados et al. [36]. A main task is to extend this structure onto curved objects.

The intersection of three objects, which is a major task for the computation of the arrangement of more than two objects. In our context a 3D arrangement consists of lower dimensional

cells like vertices, edges and faces. Considering the arrangement of two objects, vertices are given by singularities and initial starting points, edges are given by intersection curve components and faces are given by patches of the given objects. Possibly some artificial edges might be constructed to guarantee the connectivity between the vertices. However, intersecting three objects yields vertices which are not detected by the examination of the pairwise intersections, namely the intersection points between two transversal intersection curve components of different object pairs. Together with the common intersection points of three objects, the set of necessary cells for an arrangement computation is complete.

The idea to gain these intersection points is to solve the equation system given by the algebraic functions of three objects. In a generic case this results in a zero-dimensional set of points. The two major approaches to solve algebraic equation systems are first the use of Gröbner bases [13, 21, 48] and second using resultants [15, 53, 54, 55]. For a resultant-based approach, specialised on quadrics, see Chionh et al. [17].

A computer algebra system, using Gröbner bases, is SINGULAR[1] from Greuel et al. [37]. Some experiments showed a good performance related to our problem, even though Gröbner based approaches suffer from exploding running times at even small polynomial systems. However, we could not find a guarantee that SINGULAR detects all solutions up to any arbitrary approximation level. Thus we made also some experiments with resultant-based approaches, which mainly proceed as follows:

Given a system of algebraic functions of three objects, eliminate two variables by some resultant scheme to gain an univariate polynomial in each coordinate direction. Roots of these polynomials represent the corresponding coordinates of the common intersection points. Since these roots are algebraic, the coordinates are given by axis parallel stripes, each stripe containing the exact coordinate of an intersection point. Intersecting these stripes yields boxes, some containing the exact intersection points. To identify the non-empty boxes, compute two additional univariate polynomials in generic directions, e.g. assure that no two intersection point coordinates coincide to one root with higher degree. By comparing the set of boxes, the empty ones can be eliminated. Details can be found in Seidel and Wolpert [67].

The remaining problem is the detection and exclusion of common intersection curve components of all three objects, since the approach above only works with zero-dimensional solution sets.

[1] SINGULAR was developed at the Technical University of Kaiserslautern, see http://www.mathematik.uni-kl.de/ftp/pub/Math/Singular/UNIX/

List of Figures

List of Algorithms

List of Tables

Bibliography

[1] John Abbott. Quadratic interval refinement for real roots, 2006.

[2] C. Bajaj and M.-S. Kim. Generation of configuration space obstacles: The case of a moving sphere. *IEEE J. of Robotics and Automation*, 4(1):94–99, 1988.

[3] C. Bajaj and G. Xu. NURBS approximation of surface/surface intersection curves. *Advances in Computational Mathematics*, 2(1):1–21, 1994.

[4] C. L. Bajaj, C. M. Hoffman, R. E. Lynch, and J. E. H. Hopcroft. Tracing surface intersections. *Comput. Aided Geom. Des.*, 5(4):285–307, 1988.

[5] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface/surface intersection. *Comput. Aided Geom. Des.*, 7(1-4):257–280, 1990.

[6] Eric Berberich. *Exact Arrangements of Quadric Intersection Curves.* PhD thesis, Universität des Saarlandes, 2004.

[7] Eric Berberich, Arno Eigenwillig, Michael Hemmer, Susan Hert, Kurt Mehlhorn, and Elmar Schömer. A computational basis for conic arcs and boolean operations on conic polygons. In *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*, pages 174–186, London, UK, 2002. Springer-Verlag.

[8] Eric Berberich, Michael Hemmer, Lutz Kettner, Elmar Schömer, and Nicola Wolpert. An exact, complete and efficient implementation for computing planar maps of quadric intersection curves. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 99–106, New York, NY, USA, 2005. ACM.

[9] Eric Berberich and Michael Kerber. Exact arrangements on tori and Dupin cyclides. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 59–66, New York, NY, USA, 2008. ACM.

[10] Eric Berberich, Michael Kerber, and Michael Sagraloff. Exact geometric-topological analysis of algebraic surfaces. In *SCG '08: Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 164–173, New York, NY, USA, 2008. ACM.

[11] H. Bieri and W. Nef. Elementary set operations with d-dimensional polyhedra. In *Proceedings on International Workshop on Computational Geometry on Computational Geometry and its Applications*, pages 97–112, New York, NY, USA, 1988. Springer-Verlag New York, Inc.

[12] I. C. Braid. The synthesis of solids bounded by many faces. *Commun. ACM*, 18(4):209–216, 1975.

[13] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, University of Innsbruck, 1965.

[14] C. Burnikel, J. Könemann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Exact geometric computation in LEDA. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 418–419, New York, NY, USA, 1995. ACM.

[15] John F. Canny. *The complexity of robot motion planning*. MIT Press, Cambridge, MA, USA, 1988.

[16] Gerolamo Cardano. *Artis magnae, sive de regulis algebraicis*. Nürnberg, Germany, 1545.

[17] Eng-Wee Chionh, Ronald N. Goldman, and James R. Miller. Using multivariate resultants to find the intersection of three quadric surfaces. *ACM Trans. Graph.*, 10(4):378–400, 1991.

[18] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, pages 134–183, London, UK, 1975. Springer-Verlag.

[19] George E. Collins and Alkiviadis G. Akritas. Polynomial real root isolation using Descarte's rule of signs. In *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 272–275, New York, NY, USA, 1976. ACM.

[20] George E. Collins and Rudiger Loos. Real zeroes of polynomials. *Computer algebra: symbolic and algebraic computation (2nd ed.)*, pages 83–94, 1983.

[21] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 1997.

[22] Katrin Dobrindt, Kurt Mehlhorn, and Mariette Yvinec. A complete and efficient algorithm for the intersection of a general and a convex polyhedron. In *WADS '93: Proceedings of the Third Workshop on Algorithms and Data Structures*, pages 314–324, London, UK, 1993. Springer-Verlag.

[23] C. Dupin. *Applications de Géométrie et de Méchanique*. Bachelier, Paris, 1822.

[24] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Towards the robust intersection of implicit quadrics. In *Proc. of Workshop on Uncertainty in Geometric Computations*, Sheeld, UK, 2001.

[25] Arno Eigenwillig. *Exact Arrangement Computation for Cubic Curves*. PhD thesis, Universität des Saarlandes, 2003.

[26] Arno Eigenwillig and Michael Kerber. Exact and efficient 2d-arrangements of arbitrary algebraic curves. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 122–131, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[27] Arno Eigenwillig, Michael Kerber, and Nicola Wolpert. Fast and exact geometric analysis of real algebraic plane curves. In *ISSAC '07: Proceedings of the 2007 international symposium on Symbolic and algebraic computation*, pages 151–158, New York, NY, USA, 2007. ACM.

[28] Arno Eigenwillig, Lutz Kettner, Elmar Schömer, and Nicola Wolpert. Complete, exact, and efficient computations with cubic curves. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 409–418, New York, NY, USA, 2004. ACM.

[29] Arno Eigenwillig, Lutz Kettner, Elmar Schömer, and Nicola Wolpert. Exact, efficient, and complete arrangement computation for cubic curves. *Comput. Geom. Theory Appl.*, 35(1):36–73, 2006.

[30] I. Emiris and E. Tsigaridas. Comparison of fourth-degree algebraic numbers and applications to geometric predicates. Technical Report ECG-TR-302206-03, INRIA Sophia-Antipolis, 2003.

[31] Andreas Fabri, Geert-Jan Giezeman, Lutz Kettner, Stefan Schirra, and Sven Schönherr. The CGAL kernel: A basis for geometric computation. In *FCRC '96/WACG '96: Selected papers from the Workshop on Applied Computational Geormetry, Towards Geometric Engineering*, pages 191–202, London, UK, 1996. Springer-Verlag.

[32] R. T. Farouki, C. Neff, and M. A. O'Conner. Automatic parsing of degenerate quadric-surface intersections. *ACM Trans. Graph.*, 8(3):174–203, 1989.

[33] Steven Fortune. Polyhedral modelling with exact arithmetic. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, pages 225–234, New York, NY, USA, 1995. ACM.

[34] Nicola Geismann, Michael Hemmer, and Elmar Schömer. Computing a 3-dimensional cell in an arrangement of quadrics: exactly and actually! In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*, pages 264–273, New York, NY, USA, 2001. ACM.

[35] Ronald N. Goldman and James R. Miller. Combining algebraic rigor with geometric robustness for the detection and calculation of conic sections in the intersection of two natural quadric surfaces. In *SMA '91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 221–231, New York, NY, USA, 1991. ACM.

[36] Miguel Granados, Peter Hachenberger, Susan Hert, Lutz Kettner, Kurt Mehlhorn, and Michael Seel. Boolean operations on 3d selective nef complexes: Data structure, algorithms and implementation. In *Proc. 11th Annu. Euro. Sympos. Alg., volume 2832 of LNCS*, pages 654–666. Springer, 2003.

[37] Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR — a computer algebra system for polynomial computations. *Symbolic computation and automated reasoning*, pages 227–233, 2001.

[38] Peter Hachenberger, Lutz Kettner, and Kurt Mehlhorn. Boolean operations on 3d selective nef complexes: Data structure, algorithms, optimized implementation and experiments. *Comput. Geom. Theory Appl.*, 38(1-2):64–99, 2007.

[39] Michael Hemmer. *Exact Computation of the Adjacency Graph of an Arrangement of Quadrics.* PhD thesis, Johannes Gutenberg-Universität, Mainz, 2007.

[40] Jeremy R. Johnson, Werner Krandick, Kevin Lynch, David G. Richardson, and Anatole D. Ruslanov. High-performance implementations of the Descartes method. In *ISSAC '06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 154–161, New York, NY, USA, 2006. ACM.

[41] L.V. Kantorowitsch. *Funktionalanalysis und Angewandte Mathematik.* Uspechi Mat. Nauk 3, 1948.

[42] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A core library for robust numeric and geometric computation. In *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*, pages 351–359, New York, NY, USA, 1999. ACM.

[43] B. Kearfott. Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, 2(1):95–112, 1996.

[44] Michael Kerber. *Analysis of Real Algebraic Plane Curves.* PhD thesis, Universität des Saarlandes, 2006.

[45] J. Keyser, T. Culver, M. Foskey, D. Manocha, and S. Krishnan. ESOLID - a system for exact boundary evaluation. *Computer Aided Design*, 36(2):175–193, 2004.

[46] Ku-Jin Kim. *Torus and Simple Surface Intersection Based on a Configuration Space Approach.* PhD thesis, Department of Computer Science and Engineering, POSTECH, February 1998.

[47] Shankar Krishnan and Dinesh Manocha. An efficient surface intersection algorithm based on lower-dimensional formulation. *ACM Trans. Graph.*, 16(1):74–106, 1997.

[48] D. Lazard. Solving zero-dimensional algebraic systems. *J. Symb. Comput.*, 13(2):117–131, 1992.

[49] S. Lazard, L. M. Peñaranda, and S. Petitjean. Intersecting quadrics: An efficient and exact implementation. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 419–428, New York, NY, USA, 2004. ACM Press.

[50] Joshua Levin. A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. *Commun. ACM*, 19(10):555–563, 1976.

[51] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[52] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, 1979.

[53] F.S. Macaulay. On some formula in elimination. *Proceedings of London Mathematical Society*, 1(33):3–27, May 1902.

[54] Dinesh Manocha. Computing selected solutions of polynomial equations. In *ISSAC '94: Proceedings of the international symposium on Symbolic and algebraic computation*, pages 1–8, New York, NY, USA, 1994. ACM.

[55] Dinesh Manocha and John F. Canny. Multipolynomial resultant algorithms. *J. Symb. Comput.*, 15(2):99–122, 1993.

[56] Kurt Mehlhorn and Stefan Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[57] Günter Meinardus. Über eine Verallgemeinerung einer Ungleichung von L. V. Kantorowitsch. *Numerische Mathematik*, 5:14–23, 1963.

[58] James R. Miller and Ronald N. Goldman. Geometric algorithms for detecting and calculating all conic sections in the intersection of any 2 natural quadric surfaces. *CVGIP: Graphical Model and Image Processing*, 57(1):55–66, 1995.

[59] W. Nef. *Beiträge zur Theorie der Polyeder*. Herbert Lang, Bern, 1978.

[60] M.A. O'Connor. Natural quadrics: Projections and intersections. *IBM J. Res. Dev.*, 33(4):417–446, 1989.

[61] Koji Ouchi and John Keyser. Handling degeneracies in exact boundary evaluation. In *SM '04: Proceedings of the ninth ACM symposium on Solid modeling and applications*, pages 321–326, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[62] Les Piegl. On NURBS: a survey. *IEEE Comput. Graph. Appl.*, 11(1):55–71, 1991.

[63] Les Piegl and Wayne Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[64] Simon Plantinga and Gert Vegter. Contour generators of evolving implicit surfaces. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 23–32, New York, NY, USA, 2003. ACM Press.

[65] A. A. G. Requicha and H. B. Voelcker. Solid modeling: A historical summary and contemporary assessment. *IEEE Comput. Graph. Appl.*, 2(2):9–24, 1982.

[66] Elmar Schömer and Nicola Wolpert. An exact and efficient approach for computing a cell in an arrangement of quadrics. *Comput. Geom. Theory Appl.*, 33(1-2):65–97, 2006.

[67] Raimund Seidel and Nicola Wolpert. On the exact computation of the topology of real algebraic curves. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 107–115, New York, NY, USA, 2005. ACM.

[68] Ching-Kuang Shene and John K. Johnstone. On the planar intersection of natural quadrics. In *SMA '91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 233–242, New York, NY, USA, 1991. ACM.

[69] Ching-Kuang Shene and John K. Johnstone. On the lower degree intersections of two natural quadrics. *ACM Trans. Graph.*, 13(4):400–424, 1994.

[70] Volker Stahl. *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations.* PhD thesis, University of Linz, Austria, 1995.

[71] Gilbert Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, 3 edition, 1988.

[72] Nicola Wolpert. *An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics.* PhD thesis, Universität des Saarlandes, 2002.

[73] C. Yap and T. Dubé. The exact computation paradigm. In *Computing in Euclidian Geometry.* World Scientific Press, 1994.

[74] David Y.Y. Yun. On square-free decomposition algorithms. In *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 26–35, New York, NY, USA, 1976. ACM Press.

[75] Antoine Joseph François Yvon-Villarceau. Théorème sur le tore. *Nouvelles Annales de Mathématiques*, 7:345–347, 1848.