Reliable computation of planar and spatial quadric arrangements.

von

Michael Hemmer

Saarbrücken
July 30, 2004

Diplomarbeit

nach einem Thema von Priv. Doz. Dr. Elmar Schömer in der Naturwissenschaftlich-Technischen Fakultät I, Fachrichtung 6.2 - Informatik, der Universität des Saarlandes, Saarbrücken.

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Saarbrücken im April 2002

Contents

Danksagung

Mein Dank gilt Herrn Priv. Doz. Dr. Elmar Schömer für die vergabe dieser Diplomarbeit. Herr Schömer hat mich wärend der ganzen Zeit hervorragend betreut und hatte für die auftretenden Probleme stets ein offenes Ohr.
Ferner möchte ich mich bei Nicola Wolpert, Christian Lennerz und Thomas Warken für die gute Zusammenarbeit bedanken.
Ganz besonders bedanken möchte ich mich bei meinen Eltern, die mich während meines Studiums stets unterstützt haben.

Chapter 1

Introduction

Efficient algorithms for the calculation of arrangements are a well known area of research in computational geometry. For a good and brief overview see [**?**]. The ones dealing with arrangements of hyperplanes can be implemented with exact arithmetic because they only have to deal with linear algebraic primitives. The situation becomes more difficult if the hyperplanes are substituted by arbitrary algebraic surfaces. In this context it can be very hard to implement the necessary algebraic primitives in an efficient and robust way because now we have to deal with algebraic numbers instead of rational arithmetic.

We consider two and three-dimensional arrangements of quadric. Our goal is to compute an exact topological description of the arrangements. This problem typically arises in solid modeling (see [**?**]) when performing boolean operations for quadric surfaces, which play an important role in the design of mechanical parts. Thus CAD systems dealing with curved objects in two and three dimensions have been available since the 60's. But they often use numerical procedures for tracing the intersection curves and then approximate them as a spline-curve. These algorithms are very sensitive to approximation and rounding errors. Without an exact representation of the resulting curves and without exact arithmetic it is difficult to detect degenerate configurations which are frequent in the design of geometric objects. Thus none of these systems is complete or exact.

The question of complete and exact implementations has only been addressed recently and there already is some work in computational geometry that deals with planar arrangements of algebraic curves. MAPC [**?**] for instance provides a set of classes for manipulating algebraically defined points and curves in the plane. But their algorithms are not complete, they handle some but not all degeneracies. Further on they only have an implementation of the naïve $O(n^2)$ algorithm for

computing an arrangement of n curves in the plane. But the $O((n+s)\log(n))$[1] Bentley-Ottmann [**?**] sweep-line algorithm is known to work for x-monotone segments, thus it would be nice to have one for any type of algebraic curves. CGAL's [**?**] planar map class already supports the computation of arrangements of circular arcs and line segments. Very recently, the implementation was extended to conic arcs by Halperin and Wein [**?**]. However, the sweep method is not yet available.

The goal in the first part of this master thesis is to provide the necessary theory for the implementation of an sweep-line algorithm for conic segments. First we give a x-monotone parameterization for conic arcs, which is essential for our considerations. Then we give an unambiguous representation for all types of intersection points. Further on we propose implementations for all predicates needed in the sweep-line algorithm. Thus we were able to extend the Bentley-Ottmann sweep-line algorithm to conic arcs, see [**?**], without any further assumptions.

The second part of this master thesis deals with 3-dimensional quadric arrangements. Up to now there are two approaches published that deal with spatial quadric arrangements.

The first is ESOLID [**?**] by D. Manocha, it performs accurate boundary evaluation of low-degree curved solids, and especially with quadric surfaces. The advantage of their approach is that they only have to work within one parameter space for each surface. But they can only give an implicit representation of the intersection curve of degree 8, while the minimal degree of such curves is 4, see [**?**]. Therefore they have some problems to handle degenerated situation and it it is explicitly stated that degeneracies are not treated by ESOLID.

The next approach has been introduced by N. Wolpert **??**. At first the spatial arrangement is reduced to a planar arrangement of algebraic curves. Then the planar arrangement can be determined by extended local box hit counting or by applying explicit solutions. The advantage of this approach is that it holds the minimal algebraic degree of the intersection curves and can detect all degeneracies. But for each pair of curves it has to perform several costly resultant computations, and does not take any advantage of non-degenerated cases, thus we have to consider this as the naïve approach.

The approach presented in this master thesis, works directly in 3-space. It uses an exact parameterization of the spatial intersection curves based on Levin [**?**] and S.Lazard [**?**], while holding the minimal algebraic degree of 4. We are able to give an unambiguous representation for every appearing intersection point, thus we especially have an equality test for intersection points. Thus we are able to handle situations as the intersection of more than three surfaces in one point. For singular points we do not have an algorithmic solution now, as there is a huge variety of cases that have to be considered. But we very optimistic to give an complete solution in the near future.

[1] n is the number of segment and s the number intersections

Chapter 2

Conics

Quadrics in 2D are the so called conics. In this chapter we want to give a general overview of conics. We will classify them by transforming them into canonical form, and we will look for some parameterizations.

2.1 Definition

Every conic is defined as the zero-set in $\mathbb{R}^2$ of a quadratic implicit equation P in the variables x_1 and x_2 i.e.,

$$\alpha_1 x_1^2 + \alpha_2 x_2^2 + 2\alpha_3 x_1 x_2 + 2\alpha_4 x_1 + 2\alpha_5 x_2 + \alpha_6 = 0 \tag{2.1}$$

with $\alpha_i \in \mathbb{R}$ and some $\alpha_i \neq 0$ for $i \leq 3$. We can rewrite equation (1) by using homogeneous coordinates

$$X^T P X = 0,$$

where $X = (x_1, x_2, 1)^T$ and P is the symmetric 3×3 matrix

$$P = \begin{pmatrix} \alpha_1 & \alpha_3 & \alpha_4 \\ \alpha_3 & \alpha_2 & \alpha_5 \\ \alpha_4 & \alpha_5 & \alpha_6 \end{pmatrix}.$$

It is also common to write equation (1) as

$$x P_u x + 2p^T x + \alpha_6 = 0, \tag{2.2}$$

where $p = (\alpha_4, \alpha_5)$ and P_u denotes the upper 2×2 matrix of P, and is called the principal submatrix of P. We will use the same notation P for a quadratic implicit equation and its associated matrix.

2.2 Classification

We will classify a quadric by its *canonical form*. To achieve this, we apply a principal axis transformation. P_u is a symmetric matrix. Therefore, P_u always has two real eigenvalues λ_1, λ_2. Let R_u be the matrix of the eigenvectors of P_u. We can choose R_u such that $det(R_u) = 1$ and $R_u^T = R_u^{-1}$, which means that we just rotate the coordinate system, if we substitute $x = R_u y$ in (2) and receive:

$$y^T B y + 2 y b^T y + \alpha_6 = 0 \tag{2.3}$$

where

$$b = R_u^T p = (b_1, b_2)^T,\ B = R_u^T P_u R_u = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

If $b \neq 0$, we apply a translation $y = z + t$, to get rid of the linear term. Geometrically, this means that we move the center of the quadric into the origin. The form of t depends on the eigenvalues and the vector b. Let us assume that $b \neq 0$, because in the case $b = 0$, P already is in canonical form. We also know that there is at least one quadratic term. Therefore, we can assume that $\lambda_1 \neq 0$.
1.case: $\lambda_2 \neq 0$
We choose

$$t = (-b_1/\lambda_1, -b_2/\lambda_2)$$

and receive the canonical form:

$$\lambda_1 z_1^2 + \lambda_2 z_2^2 + c = 0, \text{ where } c = t^T B t + 2 b^T t + \alpha_6$$

2.case: $\lambda_2 = 0$ and $b_2 = 0$
We choose

$$t = (-b_1/\lambda_1, 0)$$

and receive the canonical form:

$$\lambda_1 z_1^2 + c = 0, \text{ where } c = t^T B t + 2 b^T t + \alpha_6$$

3.case: $\lambda_2 = 0$ and $b_2 \neq 0$
In this case we have no chance to get rid of the linear term $2 b_2 z_2$ because of $\lambda_2 = 0$, but let us try

$$t' = (-b_1/\lambda_1, 0)$$

and we receive

$$\lambda_1 z_1^2 + 2 b_2 z_2 + c = 0$$

where $c = \alpha_6 - b_1^2/\lambda_1$. Therefore we choose

$$t = (-b_1/\lambda_1, -c/2 b_2)$$

to get rid of c and receive

$$\lambda_1 z_1^2 + 2b_2 z_2 = 0$$

So it turns out, that there are 3 basic types of conics:

$$\begin{array}{lccccccccl} \textbf{I} & \lambda_1 z_1^2 & + & \lambda_2 z_2^2 & + & c & = & 0 & : & \lambda_1 \neq 0, \lambda_2 \neq 0 \\ \textbf{II} & \lambda_1 z_1^2 & & & + & c & = & 0 & : & \lambda_1 \neq 0 \\ \textbf{III} & \lambda_1 z_1^2 & + & 2b_2 z_2 & & & = & 0 & : & \lambda_1 \neq 0, b_2 \neq 0 \end{array}$$

Note that we can multiply each equation by some constant without changing the zero-set of the equation, therefore we can assume $\lambda_1 > 0$.
Now we are able to classify the conic sections

Type	λ_1	λ_2	b_2	c	Class	Equation
I	> 0	> 0	$-$	> 0	empty set	$a^2x_1^2 + b^2x_2^2 + 1 = 0$
I	> 0	> 0	$-$	< 0	ellipse	$a^2x_1^2 + b^2x_2^2 - 1 = 0$
I	> 0	< 0	$-$	$\neq 0$	hyperbola	$a^2x_1^2 - b^2x_2^2 \pm 1 = 0$
I	> 0	< 0	$-$	$= 0$	two lines intersecting	$a^2x_1^2 - b^2x_2^2 = 0$
I	> 0	> 0	$-$	$= 0$	two complex lines	$a^2x_1^2 + b^2x_2^2 = 0$
II	> 0	$-$	$-$	> 0	empty set	$x_1^2 + a^2 = 0$
II	> 0	$-$	$-$	< 0	two parallel lines	$x_1^2 - a^2 = 0$
II	> 0	$-$	$-$	$= 0$	double line	$x_1^2 = 0$
III	> 0	$-$	$\neq 0$	$-$	parabola	$x_1^2 = 2px_2$

Note that we only used linear transformations to get the canonical form of the conic. We can easily express them by using the homogeneous representation.
We define the rotation by the 3×3 matrix

$$R = \begin{pmatrix} R_u & 0 \\ 0 & 1 \end{pmatrix}$$

and the translation t can also be defined by a 3×3 matrix

$$T = \begin{pmatrix} I & t \\ 0 & 1 \end{pmatrix}$$

This defines a transformation

$$U = TR = \begin{pmatrix} R_u & t \\ 0 & 1 \end{pmatrix},$$

that sends the conic into canonical form P''. Thus our transformation goes as follows:

$$X^T P X = X^T R^T P' R X = X^T R^T T^T P'' T R X = X^T U^T P'' U X$$

2.3 Parameterization

Suppose, that we have a parameterization of the canonical form $Z(t) = (z_1(t), z_2(t), 1)$ for some $t \in I \subseteq \mathbb{R}$. In this case we can easily give a parameterization of the original quadric by sending it back into the original coordinate frame

$$X(t) = U^{-1}Z(t) = R^{-1}T^{-1}Z(t).$$

Let us assume that we also scaled the coordinate frame such that the matrix P is $\in \{-1, 0, +1\}^{3\times 3}$, which we also stored in the matrix U. Thus the three conic types are:

$$\begin{array}{lll} \textbf{I} & P = & \begin{pmatrix} 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & c \end{pmatrix} \quad : \quad c \in \{-1, 0, +1\} \\ \textbf{II} & P = & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & c \end{pmatrix} \quad : \quad c \in \{-1, 0, +1\} \\ \textbf{III} & P = & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & \pm 1 \\ 0 & \pm 1 & 0 \end{pmatrix} \end{array}$$

2.3.1 Ellipse

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Because of the scaling the ellipse looks like a circle, so we can parameterize it like a circle. So there are several ways to parameterize an ellipse but it depends on your application, which one might be the best:
The first parameterization is to define the circle by the trigonometric functions cos and sin:

$$(cos(\varphi), sin(\varphi), 1), \text{ where } \varphi \in [0, 2\pi)$$

The advantage of this representation is that you do not need to consider two conic arcs, so you have a bijection between a point on the ellipse and the parameter φ. But this parameterization involves π which is a transcendental number, so this parameterization is not suitable for computing in exact arithmetic. The next one seems to get rid of this problem by substituting $\varphi = 2arctan(t)$ in the parameterization above.

$$((1 - t^2)/(1 + t^2), 2t/(1 + t^2), 1), \text{ where } t \in \mathbb{R}$$

But notice that the parameter range goes from minus to plus infinity, and again you might have some numerical problems.

At first glance the next parameterization might look not very clever, because you have to consider two conic arcs.

$$(t, \pm\sqrt{1-t^2}, 1), \text{ where } t \in [-1, 1]$$

But it is a straight forward way to get a parameterization and might also work for non-canonical forms.

2.3.2 Hyperbola

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

As in the section above there are three parameterizations one should know, with basically the same advantages and disadvantages:

$$(\pm cosh(\varphi), sinh(\varphi), 1), \text{ where } \varphi \in \mathbb{R}$$

$$((1+t^2)/(1-t^2), 2t/(1-t^2), 1), \text{ where } t \in \mathbb{R}$$

$$(\pm\sqrt{t^2+1}, t, 1), \text{ where } t \in \mathbb{R}$$

2.3.3 Parabola

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

If we look at the implicit equation

$$x_1^2 - 2x_2 = 0$$

it is obvious to choose this parameterization

$$(t, t^2/2, 1), \text{ where } t \in \mathbb{R}$$

Chapter 3

Sweep-line algorithm for conic arcs

The first sweep-line algorithm for computing an arrangement of segments in the plane was presented by J.L. Bentley and T.A. Ottmann [**?**] in 1979. The sweep-line algorithm takes as input a set $\mathbf{S}$ of n segments and produces as output a graph $\mathbf{G}(\mathbf{S})$ induced by the set of segments. The algorithm acts like a vertical line that sweeps from left to right while scanning the y-order of all segments that cross the sweep line. This is a continues problem, but topologically the y-order only changes for discrete points. These points are endpoints of segments or points where segments cross, we call these points event-points. For each event point of the arrangement the line stops and updates the data structure. There are two main internal structures, the X-structure and the Y-structure. The X-structure stores all known event points, that are right of the sweep line, in a priority queue. The Y-structure stores all segments, that cross the sweep line, according to their momentary order along the sweep line.
Originally the algorithm needed the assumption, that the input set of segments contains no overlapping segments. But the implementation in LEDA[**?**] removed this assumption and indeed makes no assumption about the input. This means, for example, that segments may be vertical or have zero length, several segments may overlap or intersect at a single point.
Bentley and Ottmann observed that their algorithm generalizes to curved x-monotone segments. The only thing which has to be recognized, is that segments do not necessarily change their order at a common sweep point. S. Hert and K. Mehlhorn have already implemented a sweep-line algorithm for circular arcs. Our goal is to show a possible extension for their algorithm to conic arcs.

1. We need a representation for x-monotone conic arcs
2. Obviously, we have to be able to compute the intersection between two conics and have to find an unambiguous representation for intersection points.

3. We have to show how to implement the predicates.

3.1 X-monotone parameterization

For the sweep-line algorithm it is advantageous to have a x-monotone parameterization of a quadric Q. To do so, we can rewrite Q as the zero set of a quadratic polynomial in y and with coefficients in x

$$Q = a(x)y^2 + b(x)y + c(x) = 0, \tag{3.1}$$

where

$$\begin{aligned} a(x) &= \alpha_2, \\ b(x) &= 2\alpha_3 x + 2\alpha_5, \\ c(x) &= \alpha_1 x^2 + 2\alpha_4 x + \alpha_6. \end{aligned}$$

case 1: $\alpha_2 \neq 0$
In this case we can receive the x-monotone parameterization by simply solving equation **??**.

$$\begin{aligned} y &= \frac{-b(x) \pm \sqrt{b(x)^2 - 4a(x)c(x)}}{2a(x)} \\ &= \frac{-(2\alpha_3 x + 2\alpha_5) \pm \sqrt{(2\alpha_3 x + 2\alpha_5)^2 - 4\alpha_2(\alpha_1 x^2 + 2\alpha_4 x + \alpha_6)}}{2\alpha_2} \\ &= \frac{-2\alpha_3 x - 2\alpha_5 \pm \sqrt{4\alpha_3^2 x^2 + 8\alpha_3 x\alpha_5 + 4\alpha_5^2 - 4\alpha_2\alpha_1 x^2 - 8\alpha_2\alpha_4 x - 4\alpha_2\alpha_6}}{2\alpha_2} \\ &= \frac{-b(x) \pm \sqrt{h(x)}}{2\alpha_2}, \end{aligned}$$

where $h(x) = h_2x^2 + h_1x + h_0$ is a quadratic polynomial and let us call

$$\mathbb{D}_Q = \{x \ \in \mathbb{R} |\ h(x) \geq 0\}$$

the domain of Q.

Note that we receive two arcs. Let us assume that $\alpha_2 > 0$ thus we can refer to the arc with the plus sign at the square root as the upper or positive arc and to the one with the minus sign as the lower or negative arc.
We can also classify a conic using the x-monotone parameterization. It depends on the behavior of the polynomial h what kind of conic we get. So we have to look for the degree, the leading coefficient and the real roots of h.

h_2	h_1	h_0	$roots(h)$	classification
> 0	$-$	$-$	none	hyperbola
> 0	$-$	$-$	one double root	two intersecting lines
> 0	$-$	$-$	two distinct roots	hyperbola
< 0	$-$	$-$	none	empty set
< 0	$-$	$-$	one double root	singular point
< 0	$-$	$-$	two distinct roots	ellipse
$= 0$	$\neq 0$	$-$	one root	parabola
$= 0$	$= 0$	< 0	none	two complex parallel lines
$= 0$	$= 0$	$= 0$	none	one double line
$= 0$	$= 0$	> 0	none	two parallel lines

case 2: $\alpha_2 = 0, b(x) \neq 0$
If $\alpha_2 = 0$, we receive a linear equation in y, which in general is solved by:

$$y = \frac{-c(x)}{b(x)} = \frac{-(\alpha_1 x^2 + 2\alpha_4 x + \alpha_6)}{2\alpha_3 x + 2\alpha_5}$$

In this case we need only one arc, but this parameterization will fail if b and c have a common root. This means that the conic has a vertical line at this root, which can not be x-monotone parameterized at all. To do a classification, we have to look at $deg(c)$, $deg(b)$ and $gcd(c, b)$.

Let α, β, $\gamma \in \mathbb{R}$

$deg(c)$	$deg(b)$	$gcd(c, b)$	parameterization	classification
2	1	$const$	$y = \alpha x + \beta + \frac{\gamma}{b(x)}$	hyperbola
1	1	$const$	$y = \beta + \frac{\gamma}{b(x)}$	hyperbola
0	1	$const$	$y = \frac{\gamma}{b(x)}$	hyperbola
2	0	$const$	$y = \frac{c(x)}{\alpha}$	parabola
2	1	$\sim b(x)$	$y = \alpha x + \beta$	two lines; one vertical
1	1	$\sim b(x)$	$y = \beta$	a horizontal and a vertical line
2	-1	$\sim c(x)$	$-$	two vertical lines

In all other cases, $deg(c) = 1$, $deg(b) = 0$ for instance, Q is not a conic anymore.

3.2 Intersection of two conics

In this section, we will discuss the intersection of two conics. Our goal is to give an unambiguous representation of the intersection of the two conics. Let

$$\begin{aligned} Q_1 &= \alpha_1 x^2 + \alpha_2 y^2 + 2\alpha_3 xy + 2\alpha_4 x + 2\alpha_5 y + \alpha_6, \\ Q_2 &= \beta_1 x^2 + \beta_2 y^2 + 2\beta_3 xy + 2\beta_4 x + 2\beta_5 y + \beta_6, \end{aligned}$$

$\alpha_i, \beta_i \in \mathbb{Q}$. For a better overview, let us rewrite these polynomials as polynomials in y and with coefficients in x.

$$\begin{aligned} Q_1 &= a_1 y^2 + b_1 y + c_1 \\ Q_2 &= a_2 y^2 + b_2 y + c_2 \end{aligned}$$

a_i, b_i and c_i are polynomials in x, with $deg(a_i) \leq 0$, $deg(b_i) \leq 1$ and $deg(c_i) \leq 2$. To intersect these two conics, let us now reconsider the general parameterization of Q_1, where $a_1 = \alpha_2 \neq 0$.

$$y = \frac{-b_1 \pm \sqrt{b_1^2 - 4a_1c_1}}{2a_1}$$

Now, we plug this parameterization into Q_2 and receive:

$$\frac{2a_2b_1^2 - 4a_2a_1c_1 - 2b_2a_1b_1 + 4c_2a_1^2 \pm (-2a_2b_1 + 2b_2a_1)\sqrt{b_1^2 - 4a_1c_1}}{4a_1^2} = 0$$

To get rid of the root, we square this equation in a suitable way and receive:

$$\frac{a_2b_1^2c_2 + a_2^2c_1^2 - a_2c_1b_2b_1 - 2a_2a_1c_1c_2 - b_2a_1b_1c_2 + c_2^2a_1^2 + b_2^2a_1c_1}{a_1^2} = 0$$

Because of $a_1^2 = \alpha_2^2 \neq 0$ we can ignore the denominator and we can simplify the equation:

$$r = a_2b_1^2c_2 + a_2^2c_1^2 - a_2c_1b_2b_1 - 2a_2a_1c_1c_2 - b_2a_1b_1c_2 + c_2^2a_1^2 + b_2^2a_1c_1 = 0$$

r is the so called **resultant** of Q_1, Q_2 with respect to y, see section(**??**). From there we know that, if $\xi \in \mathbb{C}$ is a root of multiplicity k, then the sum of the multiplicities of the intersection points $\in \mathbb{PC}^2$ at ξ equals k.
In the case of ellipses, hyperbolas and parabolas, it directly follows that these intersection points are $\in \mathbb{R}^2$, if and only if $\xi \in \mathbb{D}_{Q_1} \cap \mathbb{D}_{Q_2}$. As r is a polynomial in x with $deg(r) \leq 4$, it is obvious that we can only have up to four intersection points or infinitely many in the case of $r \equiv 0$. But in general a conic consists of

two arcs, therefore we do not know whether an intersection point at a root ξ lies on the upper or lower arc. For a higher multiplicity of ξ, we also have to take care of the number of the intersection points and their multiplicities. Depending on this problem, we also have to find a good representation of the roots.

Lemma 3.1: A degree four polynomial p either has four simple roots or all roots of p are of the form $\alpha + \beta\sqrt{\delta}$ with $\alpha, \beta, \delta \in \mathbb{Q}$.

Proof. Let $g = \gcd(p, p')$ be the greatest common divisor of p and its derivative.

1. If $\deg g = 0$, p has only simple roots. We can use Uspensky's algorithm, see section **??**, to compute isolating intervals [1] for them.

2. If $\deg g = 1$, p has two simple and one double root. Let $\xi \in Q$ be the root of g. Then $p = q(x-\xi)^2$ and hence $q = p/(x-\xi)^2$ is a rational polynomial. Thus all roots of p are in $\mathbb{Q}[\sqrt{\delta}]$.

3. If $\deg g = 2$, p either has two double roots or a simple root and a triple root. In either case the roots of p are precisely the roots of $q = p/g$ and hence are in $\mathbb{Q}[\sqrt{\delta}]$.

4. If $\deg g = 3$, p has the four-fold root ξ. Let $p = x^4 + a_3x^3 + \dots$, then $\xi = -a_3/4 \in \mathbb{Q}$.

□

In the case of a multiple root lemma **??** shows, that all roots of r are of the form $\alpha + \beta\sqrt{\delta}$, with $\alpha, \beta, \delta \in \mathbb{Q}$.

Definition 3.2: We call a number σ of the form $\alpha + \beta\sqrt{\delta}$, with $\alpha, \beta, \delta \in \mathbb{Q}$ a **one-root-expression**.

We are able to represent one-root-expressions easily by `LEDA reals` . In the case of simple roots we use algebraic numbers[2] to represent a root. In order to determine the intersection points, we have to distinguish these cases.

1. **case:** $r \equiv 0$

   If $r \equiv 0$, we know that Q_1 and Q_2 have a common factor. We can compute this factor by $gcd(Q_1, Q_2)$. This factor is rational and is either of degree 2 or 1.

   (a) If $deg(gcd(Q_1, Q_2)) = 2$, then Q_1 and Q_2 are associated and represent the same conic and the situation is clear.

[1] We call $\mathbf{I}$ an isolating interval of a polynomial $P(x)$, if $\mathbf{I}$ contains exactly one root of $P(x)$.
[2] Algebraic numbers are defined as the root within an isolating interval of some polynomial.

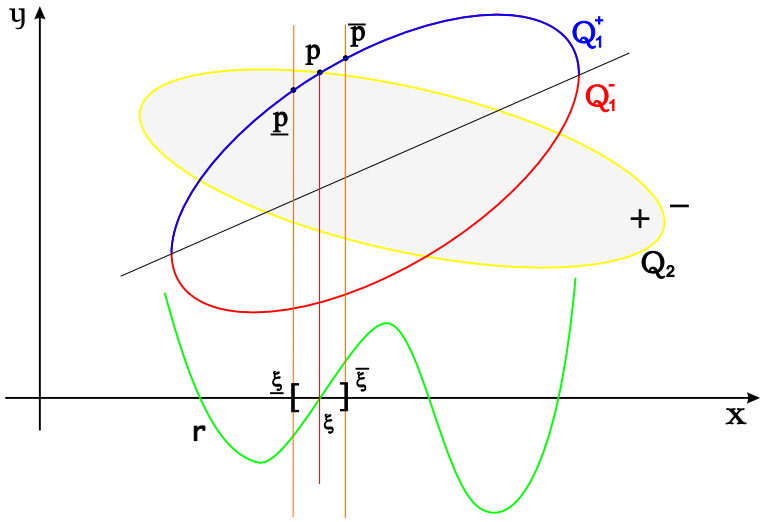

Figure 3.1: Computation of an intersection point in the case of a simple root.

(b) If $deg(gcd(Q_1, Q_2)) = 1$, this factor represents a common line $l := gcd(Q_1, Q_2)$. This line is rational because it is the *gcd* of two rational polynomials. Now we can divide each conic by their common line l, and receive for each conic an other rational line. In this case the problem reduces to the intersection of rational lines.

2. **case: $\xi \in \mathbb{R}$ is represented by an algebraic number**

In this case ξ necessarily is a simple root of r. We also know that $\xi \in \mathbb{D}_{Q_1} \cap \mathbb{D}_{Q_2}$, because a simple real root is necessarily caused by a real intersection point[3]. In general Q_1 consists of two arcs. Therefore, we have to find the arc the intersection point lies on. The basic idea is to evaluate both arcs of Q_1 at ξ and decide which one of the two resulting points also lies on Q_2. Let $I = [\underline{\xi}, \overline{\xi}]$, where $\underline{\xi}, \overline{\xi} \in \mathbb{Q}$, be the isolating interval of ξ. Further on we may refine I until $I \subset \mathbb{D}_{Q_1}$. Now we can use fast interval arithmetic, based on floating point arithmetic (see section **??**), to evaluate both arcs at I. Then we test whether one of the resulting boxes does not intersect Q_2 because then we know for sure that the other box corresponds to the intersection point we are looking for.

If this step would fail, we could refine the intervals until we get a result. But this may cause several iterations and increases the length of the representation. Instead we use the fact, that Q_1 and Q_2 intersect transversally at the intersection point (see figure **??**).

[3]A real root caused by a complex intersection point $p = (\xi, p_y)$, is at least a double root, because there is also a complex conjugated intersection point $\overline{p} = (\xi, \overline{p_y})$

Let

$$Y_{Q_1^+}(x) = \frac{-b_1(x) + \sqrt{b_1(x)^2 - 4\alpha_2 c_1(x)}}{2\alpha_2}$$

be the parameterization of the upper arc of Q_1. Now we evaluate the upper arc of Q_1 at $\underline{\xi}$ and $\overline{\xi}$ and receive two points

$$\underline{p} = (\underline{\xi}, Y_{Q_1^+}(\underline{\xi})),$$
$$\overline{p} = (\overline{\xi}, Y_{Q_1^+}(\overline{\xi})).$$

Because p is a transversal intersection point there will be a sign change between $Q_2(\underline{p})$ and $Q_2(\overline{p})$, if p lies on the upper arc of Q_1. So if

$$sign(Q_2(\underline{p})) \neq sign(Q_2(\overline{p})),$$

we know that the intersection point lies on the upper arc and otherwise on the lower arc. This evaluation is robust and should also be quite fast because each sign test involves only rational numbers and one square root.

In the case of a simple root we are able to represent the intersection point p unambiguously. The x-coordinate of p is given by an algebraic number. The y-coordinate is implicitly given by the arc the point lies on.

3. **case: ξ is an one-root-expression.**

   First, we check that $\xi \cap \mathbb{D}_{Q_1} \cap \mathbb{D}_{Q_2}$. ξ is an one-root-expression and we can represent it by `LEDA reals` . In order to determine the intersection points, we evaluate both arcs of Q_1 by `LEDA reals` .

$$\begin{aligned} p_\xi^+ &= (\xi, \frac{-b_1(\xi) + \sqrt{b_1(\xi)^2 - 4\alpha_2 c_1(\xi)}}{2\alpha_2}) \\ p_\xi^- &= (\xi, \frac{-b_1(\xi) - \sqrt{b_1(\xi)^2 - 4\alpha_2 c_1(\xi)}}{2\alpha_2}) \end{aligned}$$

   Then we check whether the points lie on Q_2. So if $Q_2(p_\xi^+) = 0$, then p_ξ^+ is an intersection point, and of course the same holds for p_ξ^-. $Q_2(p_\xi^+) = 0$ is a rational expression with up to two nested roots, which should be no problem for `LEDA reals` .

We also have to determine the multiplicity of the intersection points. Note that the multiplicity of ξ equals the sum of the multiplicities of the intersection points.

1. If ξ has multiplicity 1, the intersection point also has multiplicity 1.
2. If ξ has multiplicity 2, the situation is clear. If there is only one intersection point, then it has multiplicity 2. If there are two intersection points, each has multiplicity 1.
3. If ξ has multiplicity 3, we have to do some more. If there is only one intersection point, then it has multiplicity 3. But if there are two intersection points, we have to find the tangential one. We compute the normal vectors of Q_1 and Q_2 at p^+, $n = Q_1 p^+$ and $m = Q_2 p^+$. If

$$det \begin{pmatrix} n_1 & m_1 \\ n_2 & m_2 \end{pmatrix} = 0,$$

   then n, m are not linear independent and p^+ is the tangential intersection point.
4. If ξ has multiplicity 4, the situation is quite similar to multiplicity 3. In this case we have to test both points. If both points are tangential, then both have multiplicity 2. If only one point is tangential, then this one has multiplicity 3.

3.3 Predicates

In this section we want to discuss the predicates we need for the line sweep algorithm, namely x-compare, has-on, xy-compare and tangent-slope-compare.

3.3.1 x-compare

The comparison of two x-coordinates, is in the worst case a comparison of two algebraic numbers. So first of all, let us clarify how to compare two algebraic numbers. Let

$$\begin{array}{rcl} \xi & = & (f, [\alpha_1, \alpha_2]\ ) \quad \text{with} \quad f \in \mathbb{Q}[x] \quad \text{and} \quad \alpha_1 < \alpha_2 \in \mathbb{Q} \\ \zeta & = & (g, [\beta_1, \beta_2]\ ) \quad \text{with} \quad g \in \mathbb{Q}[x] \quad \text{and} \quad \beta_1 < \beta_2 \in \mathbb{Q} \end{array}$$

be two algebraic numbers. At first, we compare the intervals, because if the intervals do not overlap, we can already make our decision:

$$\begin{array}{rcl} \alpha_2 < \beta_1 & \Rightarrow & \xi < \zeta \\ \beta_2 < \alpha_1 & \Rightarrow & \zeta < \xi \end{array}$$

But if the intervals overlap, we can not be sure whether the two numbers are equal or not. But if $\xi = \zeta$, then it must be a common root of f and g. So if $\gcd(f, g)$ has a root in $[\alpha_1, \alpha_2] \cap [\beta_1, \beta_2]$, then $\xi = \zeta$ is this root. If not, we know that we can refine both intervals until they do not overlap anymore. Then we make our decision by comparing the intervals again. Note that we are able to replace f and g by a polynomial of lower degree if $\deg(\gcd(f, g)) > 0$.
But we also have to compare an algebraic number

$$\xi \quad = \quad (f, [\alpha_1, \alpha_2]\ ) \quad \text{with} \quad f \in \mathbb{Q}[x] \quad \text{and} \quad \alpha_1 < \alpha_2 \in \mathbb{Q},$$

with a number

$$\sigma \in \mathbb{Q}[\sqrt{\delta}];\ \delta \in \mathbb{Q}.$$

represented by `LEDA reals`. The comparison goes as follows. First of all we compare σ with the interval $[\alpha_1, \alpha_2]$:

$$\begin{array}{rcl} \sigma < \alpha_1 & \Rightarrow & \sigma < \xi \\ \alpha_2 < \sigma & \Rightarrow & \xi < \sigma \end{array}$$

In the case $\sigma \in [\alpha_1, \alpha_2]$ we test whether $\xi = \sigma$ by testing $f(\sigma) = 0$. If $f(\sigma) \neq 0$ we again refine the interval until $\sigma \notin [\alpha_1, \alpha_2]$.

3.3.2 has-on

The has-on predicate decides whether an intersection point p , defined by the intersection of two arcs f and g, lies on a other arc h. The x-coordinate of a point p is defined by a root ξ of the resultant res_{fg} of Q_f and Q_g with respect to y. The y-coordinate is implicitly given by one of the arc p lies on, let's say f.
Let us assume that we already tested that p is in the x-range of h. This can be done by a few x-compares. We also tested that h does not overlap f or g, which is a simple equality test of the underlying basecurves. Now we really have to decide whether p lies on h. But before I sketch a possible implementation of the predicate, let me first show you two cases, in which we are able to test whether p lies on h or not.

1. case: ξ is a one-root-expression.

   In this case, we can represent ξ by `LEDA reals`. Thus we can compare the y-coordinates of both arcs at ξ by using `LEDA reals`. So if

   $$Y_f(\xi) = Y_h(\xi),$$

   then p lies on h.

2. case: ξ is a simple root of the resultant res_{fh}.

   Of course ξ is only given by an algebraic number with the isolating interval $[\underline{\xi}, \overline{\xi}]$, because otherwise we would be in the 1.case. We know that Q_f and Q_h cross at ξ, but we still have to check that this intersection really lies on the two arcs f and h. So we know, that the order of the y-coordinates of f and h must interchange while passing through this point if and only if p lies on h.

   So if

   $$sign(Y_f(\underline{\xi}) - Y_h(\underline{\xi})) \neq sign(Y_f(\overline{\xi}) - Y_h(\overline{\xi})),$$

   then p lies on h.

Implementation:
If we are lucky, we already have a representation of ξ in `LEDA reals` and we use case 1. But if not, we reduce the question to one of the two cases described above. So let ξ be the algebraic number:

$$\xi \;=\; (\text{res}_{fg}, [\underline{\xi}, \overline{\xi}]\ ) \quad \text{with} \quad \text{res}_{fg} \in \mathbb{Q}[x] \quad \text{and} \quad \underline{\xi}, \overline{\xi} \in \mathbb{Q}$$

In a first step we use fast interval arithmetic, based on floating point arithmetic (see section **??**), to evaluate f at ξ using $[\underline{\xi}, \overline{\xi}]$. Then we test, whether Q_h cuts the resulting box $b_p = ([\underline{\xi}, \overline{\xi}], Y_f([\underline{\xi}, \overline{\xi}]))^T$. So if the resulting interval $I = b_p^T Q_h b_p$ contains zero, then it is possible that p lies on h.

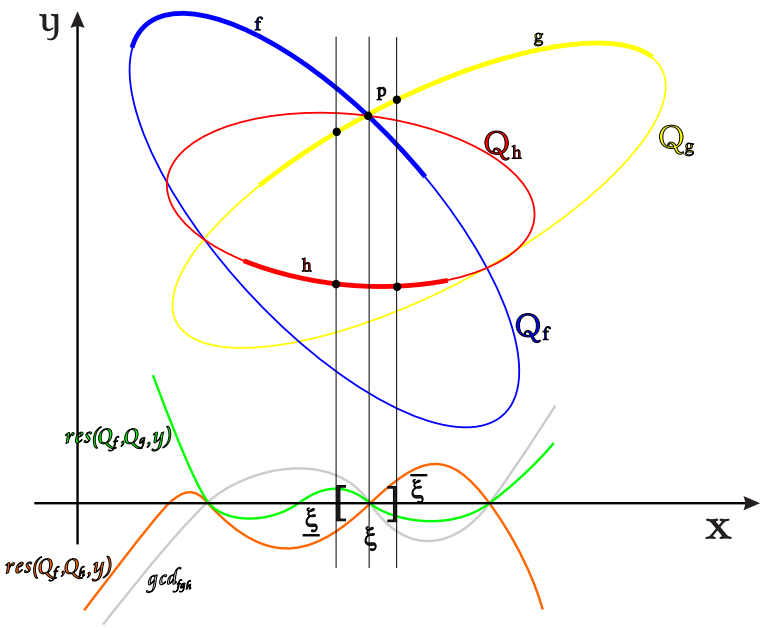

Figure 3.2: Computation of the has-on predicate, as described in the 2. case.

Of course we want to solve the question by using case 1, because it is obviously the cheaper way to decide whether p lies on h or not. Therefore, we compute $\gcd_{fgh} = \gcd(\mathrm{res}_{fg}, \mathrm{res}_{fh})$. If p lies on h, ξ must also be a root of res_{fh} and therefore a root of $\gcd_{fgh}$. It depends on $\deg(\gcd_{fgh})$ what we do next.

a. $\deg(\gcd_{fgh}) = 0$: Then we know that Q_f, Q_g and Q_h do not have a common point at all and return false.

b. $\deg(\gcd_{fgh}) = 1$: If ξ is the root of $\gcd_{fgh}$, then we are able to represent $\xi \in \mathbb{Q}$ by `LEDA reals` and switch to case 1, otherwise return false.

c. $\deg(\gcd_{fgh}) = 2$: ξ is a one-root-expression and we can represent it by `LEDA reals`. If ξ is a root of $\gcd_{fgh}$ we switch to case 1, otherwise return false.

d. $\deg(\gcd_{fgh}) = 3$: First we check whether ξ is a root of $\gcd_{fgh}$. Then we compute $\gcd(\mathrm{res}_{fh}, \mathrm{res}'_{fh})$ to check the multiplicity of ξ in res_{fh}. If ξ is a simple root of res_{fh}, we switch to case 2. And if ξ is a double root of res_{fh}, we can represent it by `LEDA reals` and switch to case 1.

e. If $\deg(\gcd_{fgh}) = 4$, we know that $\mathrm{res}_{fg} \sim \mathrm{res}_{fh}$, and it follows that ξ is a simple root of res_{fh} because we already know, that it is a simple root of res_{fg}. Therefore we can switch directly to case 2.

3.3.3 xy-compare

This predicate compares two intersection points p and q lexicographically. At first it compares the x-coordinates of both points. To do so, we can use **x-compare** and we only have to compare the y-coordinates if $p_x = q_x$.
So let f be the arc p lies on and h the arc q lies on. We already know that $p_x = q_x = \xi$. If we can represent ξ by `LEDA reals` we can compare the y-coordinate of both points with the help of `LEDA reals`. So the question is, what do we do if ξ is an algebraic number?
In order to compare the y-coordinate, we should also have a bounding interval for the y-coordinate of an intersection point p. The first idea might be to do a resultant computation for the y-coordinates as we already did for the x-coordinate. But this is very expensive and basically doubles the costs for an intersection point. But remember that we only compare the y-coordinates, if and only if the x-coordinates are equal. Let

$$\xi \quad = \quad (r, [\underline{\xi}, \overline{\xi}]\ ) \quad \text{with} \quad r \in \mathbb{Q}[x] \quad \text{and} \quad \underline{\xi}, \overline{\xi} \in \mathbb{Q}\ .$$

the x-coordinate of p and

$$Y_f(x) = -\frac{b_1 x + b_0}{2\alpha_2} + \frac{\sqrt{h_2 x + h_1 x + h_0}}{2\alpha_2}$$

be the parameterization of f. The basic idea is to evaluate f at $\underline{\xi}$ and $\overline{\xi}$ to receive a bounding y-interval of p_y. But beware of one trap: We can only guarantee that p_y is in this y-interval, if and only if f is y-monotone in $[\underline{\xi}, \overline{\xi}]$. But there only exist up to two extreme points, which are easy to compute and more over are in $\mathbb{Q}[\sqrt{\delta}]$, with $\delta \in \mathbb{Q}$. So if $[\underline{\xi}, \overline{\xi}]$ contains such a point, we refine the interval until the arc is y-monotone and then compute the bounding interval of y.
Note that this work can be done earlier in the algorithm, when creating the point for example. We do not need to do this in every xy-compare. An other possibility is to compute the bounding interval for p_y only if necessary, but then store it with the point p.
Now we compare the two bounding intervals of p_y and q_y. In most of the cases the two intervals will not overlap and we are done. But if the two intervals overlap we have to check whether p_y equals q_y. We already know that $p_x = q_x = \xi$, so if p lies on the arc of q we know that $p = q$. This can be done by the **has-on** test described above. If $p \neq q$ we can refine the isolating interval $[\underline{\xi}, \overline{\xi}]$ of $\xi = p_x = q_x$, until the resulting y-intervals do not overlap anymore.

3.3.4 y-order-right-of-common-point

This predicate determines the y-order of two arcs f and g just right of a common point p.

If p_x is the algebraic number

$$\xi \;=\; (\mathrm{res}_{fg}, [\underline{\xi}, \overline{\xi}]\,) \quad \text{with} \quad r \in \mathbb{Q}[x] \quad \text{and} \quad \underline{\xi}, \overline{\xi} \in \mathbb{Q}\ ,$$

we simply compare the two arcs at $\overline{\xi}$, and return

$$Y_f(\overline{\xi}) < Y_g(\overline{\xi}).$$

Note that $[\underline{\xi}, \overline{\xi}]$ has to be an isolating interval for res_{fg}. This is not necessarily true if ξ is defined by another polynomial, for example the resultant of two other arcs which may also go through this point.
In the case, where ξ is represented by `LEDA reals` one could compare the two arcs by comparing the first derivative, this involves three square roots and we may also have to go up to the third derivative to get an unambiguous answer. Note that the comparison $Y_f(\overline{\xi}) < Y_g(\overline{\xi})$ only involves two square roots, and it seems that this is the cheapest way to compare the two arcs at all. Therefore it is probably better to compute $\overline{\xi} \in \mathbb{Q}$, which lies between ξ and the next root of res_{fg} and then compare the two arcs by $Y_f(\overline{\xi}) < Y_g(\overline{\xi})$.
Obviously we can use this predicate to sort all outgoing arcs according to their y-order just right of p, but this process obviously is in $O(n \log(n))$, where n is the number of outgoing arcs. The original sweep-line-algorithm for line-segments does not need to sort the segments, because it uses the fact that crossing segments simply reverse their order, which only causes a linear running time for this process. So, if we want to keep up the good running time of the sweep-line-algorithm, we need something that takes advantage of the known y-order left of p. Thus we use `y-order-right-of-common-point` only to insert new arcs into the y-structure, which start at p.
Therefore, let us first consider the arcs C_1 to C_k which pass through p. We assume that the curves are numbered according to their y-order just left of p. Let s_i be the multiplicity of intersection of the curves C_i and C_{i+1} at p; see [**?**, Chapters I and IV] for a formal definition. Intuitively, the multiplicity is one if the curves meet at p and have different slopes, the multiplicity is two if the curves have identical tangent but different radii of curvature, the multiplicity is three if the curves have same tangent and identical radii of curvature but different ... . For example, the multiplicity of intersection between $y = x^i, i > 0$ and $y = 0$ at the origin is equal to i. Two curves meeting at p cross at p if the multiplicity of the intersection is odd, and they touch, but do not cross, if the multiplicity is even. The multiplicity of intersection between C_i and C_j for $i < j$ is $\min\{s_i, \ldots, s_{j-1}\}$ because the multiplicity of intersection is the number of identical initial coefficients in the local Taylor series expansion. For distinct conics, the multiplicity of intersection at any point is at most 4. For example, the multiplicity of intersection at the origin between $y(1-x) = x^2$ and $y = x^2 + y^2$ is three.

The following algorithm determines the y-order of our curves C_1 to C_k just to the right of p in time $O(k)$. Make four passes over the sequence of curves passing through p. In the j-th pass, $4 \geq j \geq 1$, form maximal subsequences of curves, where two curves belong to the same subsequence if they are not separated by a multiplicity less than j, and reverse the order of each subsequence.

Lemma 3.3: The algorithm above correctly computes the y-order of the segment passing through a common point p immediately to the right of p from the order immediately to the left of p.

Proof. Consider two arbitrary curves C_h and C_i with $h < i$. Their y-order right of p differs from their y-order left of p iff $s = \min\{s_h, \ldots, s_{i-1}\}$ is odd. Next observe that s is also exactly the number of times C_h and C_i belong to the same subsequence, i.e., the number of times their order is reversed. We conclude the order of C_i and C_h is reversed iff C_i and C_h cross at p. □

Thus we have a process, which is linear in the number of arcs passing through p.

3.4 Summary

We will end up our considerations of planar quadric arrangements here. We showed how to classify all types of conics. We gave an exact representation of the x-monotone arcs of a conic. And we can provide all necessary predicates needed for the a sweep-line algorithm for conic segments. The algorithm will be complete, exact and fast. Complete in the sense that it can handle all inputs including arbitrary degeneracies. Exact in that it always delivers the mathematical correct result. And fast in the sense that it stays in the time complexity of Bentley-Ottmann.
The actual efficient implementation of the predicates required in our algorithms will be non-trivial, since they involve costly resultant and GCD computations and algebraic numbers. Thus we will have to make an intensive use of filtering techniques like interval arithmetic. Further one it will be quite tricky to figure out a good order of the decisions we need to make within a predicate.
Our goal is to finally provide a planar map class for CGAL[**?**] as it is already implemented for arrangements of circular arcs and line segments. For a more detailed discussion also dealing with this aspect see [**?**].
As a further work we should think of a sweep-line algorithms for algebraic curves of higher degree. Thus our approach using x-monotone parameterizations and the handling of multiple intersections as described in section **??** generalizes for higher degree curves. But the situation obviously gets more complex as we now have to consider self intersections for instance.

Chapter 4

Three dimensional Quadrics

4.1 Definition

Every quadric is defined as the zero-set in $\mathbb{R}^3$ of a quadratic implicit equation $\mathbf{Q}$, which we represent by homogeneous coordinates

$$X^T \mathbf{Q} X = 0, \tag{4.1}$$

where $X = (x_1, x_2, x_3, 1)$. $\mathbf{Q}$ is the symmetric 4×4 matrix

$$\mathbf{Q} = \begin{pmatrix} \alpha_1 & \alpha_4 & \alpha_5 & \alpha_7 \\ \alpha_4 & \alpha_2 & \alpha_6 & \alpha_8 \\ \alpha_5 & \alpha_6 & \alpha_3 & \alpha_9 \\ \alpha_7 & \alpha_8 & \alpha_9 & \alpha_{10} \end{pmatrix},$$

with $\alpha_i \in \mathbb{R}$ and some $\alpha_i \neq 0$ for $i \leq 6$. It is also common to write equation (**??**) as

$$x \mathbf{Q_u} x + 2\mathbf{q}^T x + \alpha_{10} = 0, \tag{4.2}$$

where $\mathbf{q} = (\alpha_7, \alpha_8, \alpha_9)$ and $\mathbf{Q_u}$ denotes the upper 3×3 matrix of $\mathbf{Q}$. $\mathbf{Q_u}$ is called the principal submatrix of $\mathbf{Q}$. We will use the same notation $\mathbf{Q}$ for a quadratic implicit equation and its associated matrix.

4.2 Classification

4.2.1 Sylvesters Inertia Law

Theorem 4.1: For each diagonal form received by a real nonsingular transformation of a real quadric form A, beside the rank r, the number p_A of positive diagonal elements (inertia index), and consequently also the number $q_A = r - p_A$ of negative diagonal elements is invariant.

Proof. We have to show, that for two different diagonal forms of the symmetric matrix $A = A^T$ the number of positive elements is invariant. Let $Q = x^T Ax$ be the equation in the original coordinate frame and $x = By$ with $B^{-1} = (\beta_{ij})$ be the nonsingular transformation, which sends A into the first diagonal form $D_b = \text{diag}(b_1, \ldots, b_n)$.

$$\sum_{i=1}^{r} b_i y_i^2 = y^T D_b y = x^T B^{-1T} D_b B^{-1} x = \sum_{i=1}^{r} b_i(\beta_{i1}x_1 + \cdots + \beta_{in}x_n)^2 = x^T Ax$$

Now let $x = Cz$ with $C^{-1} = (\gamma_{ij})$ be the nonsingular transformation, which sends A into a second diagonal form $D_c = \text{diag}(c_1, \ldots, c_n) \neq D_b$.

$$\sum_{i=1}^{r} c_i z_i^2 = z^T D_c z = x^T C^{-1T} D_c C^{-1} x = \sum_{i=1}^{r} c_i(\gamma_{i1}x_1 + \cdots + \gamma_{in}x_n)^2 = x^T Ax$$

Obviously we can state

$$\sum_{i=1}^{r} b_i(\beta_{i1}x_1 + \cdots + \beta_{in}x_n)^2 = \sum_{i=1}^{r} c_i(\gamma_{i1}x_1 + \cdots + \gamma_{in}x_n)^2, \ \forall x \in \mathbb{R}^n. \qquad (4.3)$$

Let p_1 be the number of positive diagonal elements of D_b and let p_2 be the index of D_c. Now we assume that $p_1 > p_2$. But we can choose $x = (x_1, \ldots, x_n) \neq 0$, such that the y_i belonging to the negative b_i, and also the z_i belonging to the positive c_i vanish. This is possible because we only have to solve $r - p_1 + p_2 < r \leq n$ equations in n variables. For this $x \neq 0$ the left side of equation (**??**) is ≥ 0, while the right side is ≤ 0. Therefore, we have to conclude $y = z = 0$ and also $x = 0$. This is contradictory to $x \neq 0$ and consequently our premise $p_1 > p_2$ must be false. □

Definition 4.2: The inertia of a symmetric matrix A is represented by the signature

$$I_A = (p_A, n_A, z_A),$$

where p_A denotes the number of positive eigenvalues, n_A the number of negative eigenvalues and z_A the number of vanishing eigenvalues of $\mathbf{A}$.

4.2.2 Classification by the inertia of the quadric

For $\mathbb{PR}^3$ it is enough to classify a quadric by the inertia of $\mathbf{Q}$, but in order to distinguish the affine real types of the quadric, we also have to take the inertia of $\mathbf{Q_u}$ into consideration.

inertia of $\mathbf{Q}$	inertia of $\mathbf{Q_u}$	affine real type
(4,0,0)	(3,0,0)	$\emptyset$(imaginary ellipsoid)
(3,1,0)	(3,0,0)	ellipsoid
	(2,1,0)	hyperboloid of two sheets
	(2,0,1)	elliptic paraboloid
(2,2,0)	(2,1,0)	hyperboloid of one sheeet
	(1,1,1)	hyperbolic paraboloid
(3,0,1)	(3,0,0)	singular point
	(2,0,1)	$\emptyset$(imaginary elliptic cylinder)
(2,1,1)	(2,1,0)	cone
	(2,0,1)	elliptic cylinder
	(1,1,1)	hyperbolic cylinder
	(1,0,2)	parabolic cylinder
(2,0,2)	(2,0,1)	line
	(1,0,2)	$\emptyset$(imaginary parallel planes)
(1,1,2)	(1,1,1)	intersecting planes
	(1,0,2)	parallel planes
	(0,0,3)	simple plane
(1,0,3)	(1,0,2)	double plane
	(0,0,3)	$\emptyset$

Chapter 5

Classification of quadric intersection

Before we think about an algorithm that deals with several quadrics, we need to clarify what happens if we intersect two of them. In this section we will give a general method to distinguish the different cases of quadric intersection in any dimension, see section **??**. Of course we are especially interested in the three dimensional case. Thus we will show all appearing types of intersection curves in section **??**.

5.1 Elementarteiler or invariant-factors

In this section we want to evolve a theory to classify the intersection of two quadrics $\mathbf{A}$ and $\mathbf{B}$. Of course our main interest is in the three dimensional case (n=4), but the theory works for all dimensions.
Let $\mathbf{A}$, $\mathbf{B} \in \mathbb{Q}^{n\times n}$ be two quadric forms and let p be a point in $\mathbf{A} \cap \mathbf{B}$, then $\mathbf{p}$ satisfies both equations.

$$\begin{aligned} \mathbf{x}^T\mathbf{A}\mathbf{x} &= 0 \\ \mathbf{x}^T\mathbf{B}\mathbf{x} &= 0 \end{aligned}$$

But then, $\mathbf{p}$ also satisfies any linear combination of both equations.

$$\begin{aligned} \mathbf{p}^T(\mu\mathbf{A} + \nu\mathbf{B})\mathbf{p} &= \mathbf{p}^T(\mu\mathbf{A}\mathbf{p} + \nu\mathbf{B}\mathbf{p}) \\ &= \mu\mathbf{p}^T\mathbf{A}\mathbf{p} + \nu\mathbf{p}^T\mathbf{B}\mathbf{p} \\ &= 0 + 0 \end{aligned}$$

Thus the intersection curve of two different quadrics, obtained by a linear combination of $\mathbf{A}$ and $\mathbf{B}$, equals the intersection curve of $\mathbf{A}$ and $\mathbf{B}$.

Definition 5.1: We call the set of quadric forms

$$\mathbf{P} = \{\mu\mathbf{A} + \nu\mathbf{B} | \mu, \nu \in \mathbb{R} \text{ and } \mathbf{A}, \mathbf{B} \text{ are quadric forms}\},$$

the quadric pencil of $\mathbf{A}$ and $\mathbf{B}$. For a linear combination $\mu\mathbf{A} + \nu\mathbf{B}$ we can also write $\mathbf{A} + \frac{\nu}{\mu}\mathbf{B}$. And indeed

$$\mathbf{P}(\lambda) = \mathbf{A} - \lambda\mathbf{B}, \text{where } \lambda \in \mathbb{R} \cup \{\pm\infty\}$$

is the most common way to express the quadric pencil of $\mathbf{A}$ and $\mathbf{B}$. Note that $\lambda = \pm\infty$ is the only way to obtain $\mathbf{P}(\lambda) = \mathbf{B}$.

The quadric pencil plays an important role for the characterization of the intersection of $\mathbf{A}$ and $\mathbf{B}$. Because the type of the intersection curve strongly depends on the singular quadrics appearing in the pencil. If the pencil contains a double plane and a cone for instance, then the intersection curve has to be a conic counted twice. Namely the intersection of the plane with the cone.
The singular quadrics of the pencil are characterized by the roots of the polynomial $\det(P(\lambda))$, but if $\mathbf{B}$ is a singular quadric, we will not get a root for $\mathbf{B}$, because $\mathbf{B}$ corresponds to $\lambda = \pm\infty$. In general the pencil of $\mathbf{A}$ and $\mathbf{B}$ will be non-singular[1]. This means that it contains at least one non-singular quadric. In this case we can assume that $\mathbf{B}$ is regular, because instead of $\mathbf{B}$ we can also choose any other quadric out of the pencil different from $\mathbf{A}$ without changing the intersection curve. In this case the determinant of the quadric pencil $det(\mathbf{P}(\lambda))$ is a polynomial in λ of degree n. Therefore we can have up to n singular quadrics in a non-singular pencil. We will also take the complex quadrics into consideration, so we can identify each singular quadric by a factor $(\lambda - a)$, where $a \in \mathbb{C}$, of the determinant of the pencil. As the type of the singular quadrics, these factors are invariant under any linear transformation.

$$det(T^T(\mathbf{A} - \lambda\mathbf{B})T) = det(T)^2 det(\mathbf{A} - \lambda\mathbf{B})$$

Therefore, we can identify each singular quadric in the pencil by its factor in $det(\mathbf{P}(\lambda))$. If all these factors are distinct, they will form a complete set of invariant factors. This means, if $\mathbf{A}'$ and $\mathbf{B}'$ are an other pair of forms such that $det(\mathbf{A}' - \lambda\mathbf{B}')$ has the same factors as $det(\mathbf{A} - \lambda\mathbf{B})$, then $\mathbf{A}$, $\mathbf{B}$ can be transformed into $\mathbf{A}'$, $\mathbf{B}'$ respectively by the same linear substitution. But, if some of the factors are repeated, it does not follow, as a rule, that the set of invariants is complete without further information. As an example consider the case, that the determinant has a squared factor $(\lambda - a)$. The singular quadric belonging to this factor can have rank $n - 2$, but there are also cases in which the quadric still has rank $n - 1$.
Suppose, that the factor $(\lambda - c)$ appears raised to the power l_0 in the determinant. Let l_1 be the index of that power of $(\lambda - c)$ in the *gcd* of all first minors. In the

[1] In the singular case $det(\mathbf{P}(\lambda) \equiv 0)$, all quadrics are singular.

same way let l_2, l_3 ... be the indices of $(\lambda - c)$ found from all the second, third ... minors. Let l_r be the first index in the sequence which is zero, so that the rth minors are not all divisible by $(\lambda - c)$, although all preceding minors so are divisible. The aggregate of all factors such as $(\lambda - c)$, together with the corresponding indices l_0, l_1, l_2,..., l_r, give a complete set of invariants for the pencil of quadrics. But, it is more common to use, instead of the indices l_0, l_1, l_2,..., l_{r-1}, their successive differences $e_1 = l_0 - l_1$, $e_2 = l_1 - l_2$,...,$e_r = l_{r-1} - l_r = l_{r-1}$. The powers $(\lambda - c)^{e_1}$,$(\lambda - c)^{e_2}$,...,$(\lambda - c)^{e_r}$ of $(\lambda - c)$ are called **invariant factors** or **Elementarteiler**[2] to the base $(\lambda - c)$ of the determinant of the pencil. Note that the sum of the indices of the Elementarteiler to the base $(\lambda - c)$ is l_0 and consequently the sum of all indices of the Elementarteiler of the pencil is n which is the degree of $\det(\mathbf{A} + \lambda\mathbf{B})$. Further the first derivative of the determinant $det(\mathbf{A} - \lambda\mathbf{B})$ can be expressed as a sum of first minors, so that $(l_0 - 1) \geq l_1$. It directly follows that $e_1 \geq 1$. Similarly each of the indices e_2, e_3,..., e_r is not less than 1.

K. Weierstrass [**?**] was the first who showed that any non-singular pencil can be reduced to a canonical type, each Elementarteiler giving a group of terms in the reduced forms. But he had neither a general method of reduction nor for finding the canonical type, which was then introduced by T. J. Bromwich [**?**]. This canonical type only depends on the Elementarteiler. If two quadric pencils have the same Elementarteiler, we can transform one into the other by means of a linear substitution. We have thus a complete theory for classifying a quadric pencil.

In the list of reduced forms we will use a symbol called the characteristic by Segre, which will be best explained by an example. Supposing that we had a set of Elementarteiler

$$(\lambda - a)^2,\ (\lambda - a)\ ,(\lambda - b)^3\ ,(\lambda - c),$$

they would be denoted by [(21)31]. Thus the indices in round brackets $(e_1 e_2 \ldots e_r)$ belong to the same factor while the square brackets only surround all the indices.

5.2 Classification of quadric intersection in 3D

Now we want to take a closer look on the appearing cases in 3D. Each case is identified by the Segre characteristic. Note that this characteristic only distinguishes the cases in $\mathbb{PC}^3$. This means that an intersection point for instance might be complex or is actually at infinity.

[2]Introduced in this form and with this name by K. Weierstrass [**?**].

Segre characteristic	Reduced form	Geometric form of the intersection curve of $\mathbf{A}-\lambda B$
[1111]	$\mathbf{A}$=$k_1x_1^2 + k_2x_2^2 + k_3x_3^2 + k_4x_4^2$ $\mathbf{B}$=$k_1c_1x_1^2 + k_2c_2x_2^2 + k_3c_3x_3^2 + k_4c_4x_4^2$	A general twisted quadric of the first species
[(11)11]	As in the first case, with $c_2 = c_1$	Two conics intersecting in two distinct points P, Q (not necessarily real points): the quadrics touch at P, Q
[(11)(11)]	As in the first case, with $c_2 = c_1$,$c_3 = c_4$	Four generators intersecting in four points, at which the quadrics touch
[(111)1]	As in the first case, with $c_3 = c_2 = c_1$	A single conic counted twice: the quadrics touch all along the conic (ring-contact)
[(1111)]	As in the first case, with $c_4 = c_3 = c_2 = c_1$	No proper intersection: all the quadrics coincide
[211]	$\mathbf{A}$=$2x_1x_2 + k_3x_3^2 + k_4x_4^2$ $\mathbf{B}$=$2c_1x_1x_2 + k_1x_1^2 + k_3c_3x_3^2 + k_4c_4x_4^2$	A nodal quadric: all the quadrics touch at the node

Segre characteristic	Reduced form	Geometric form of the intersection curve of $\mathbf{A}-\lambda B$
[(21)1]	As in the sixth case, with $c_3 = c_1$	Two conics which touch: all the quadrics have stationary contact at the point of contact
[2(11)]	As in the sixht case, with $c_4 = c_3$	A conic and two generators, intersecting in three points, at which the quadrics touch
[(211)]	As in the sixth case, with $c_4 = c_3 = c_1$	Two generators counted twice: the quadrics touch all along the generators
[22]	$\mathbf{A}$=$2x_1x_2 + 2x_3x_4$ $\mathbf{B}$=$2c_1x_1x_2 + k_1x_1^2 + c_2x_3x_4 + k_3x_3^2$	A generator and a twisted cubic: the generator cuts the cubic in two distinct points, at which the quadrics touch
[(22)]	As in the previous case, with $c_3 = c_1$	Three generators one counted twice: this generator intersects each of the other two
[31]	$\mathbf{A}$=$2x_1x_2 + k_3x_3^2 + k_4x_4^2$ $\mathbf{B}$=$c_1(2x_1x_2+k_3x_3^2)+ c_4k_4x_4^2 + 2x_2x_3$	A cuspidal quadric, the quadrics have stationary contact at the cusp
[(31)]	As in the previous case, with $c_4 = c_1$	Two generators and a conic which touches the plane of the generators at their intersection
[4]	$\mathbf{A}$=$2x_1x_2 + 2x_3x_4$ $\mathbf{B}$=$2c_1(x_1x_2+x_3x_4)+ 2x_2x_3 + k_4x_4^2$	A generator and a twisted cubic, the generator touches the cubic
[{3}1]	$\mathbf{A}$=$2x_1x_2 + k_4x_4^2$ $\mathbf{B}$=$2x_1x_2 + k_4c_4x_4^2$	This is the singular case. A conic and a generator counted twice. The conic touches the generator.

Now we give an example for each appearing case. Each example shows two

quadrics and the real part of their intersection curve. We decided to intersect the infinite examples by an invisible sphere. This may cause some confusion, but in most cases this was the only way to see something at all. But you can consider the boarders of the infinite quadrics also as examples for case [1111].

Figure 5.1: [1111] A general twisted quadric of the first species

Case [1111] can be regarded as the general case. The pencil contains four different cones. The intersection curve may also decompose into two connected components. Figure **??** shows two ellipsoids and their intersection curve. The curve only consists of one connected component.

Figure 5.2: [211] A nodal quadric: all the quadrics touch at the node.

As we already have said, case [1111] has to be subdivided into the case of one

connected component and two connected components of the intersection curve. Case [211] can be regarded as the case in between these two. The intuition is, that the two connected components touch. In figure **??** a hyperbolic paraboloid intersects the ellipsoid while touching in the singular point of the curve. Case [211] also includes that two quadrics only touch in the singular point while the rest of the intersection curve may be complex.

Figure 5.3: [31] A cuspidal quadric, the quadrics have stationary contact at the cusp.

Figure **??** shows a cusp, the ellipsoid and the two sheet hyperboloid have stationary contact at the cusp. Intuitively one loop of the eight in case [211], is degenerated to the singular point that forms the cusp.

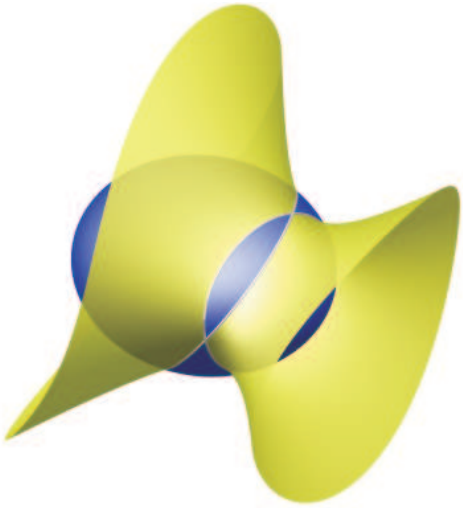

Figure 5.4: [(11)11] Two conics intersecting in two distinct points P, Q (not necessarily real points): the quadrics touch at P, Q

Figure 5.5: [(111)1] A single conic counted twice: the quadrics touch all along the conic (ring-contact).

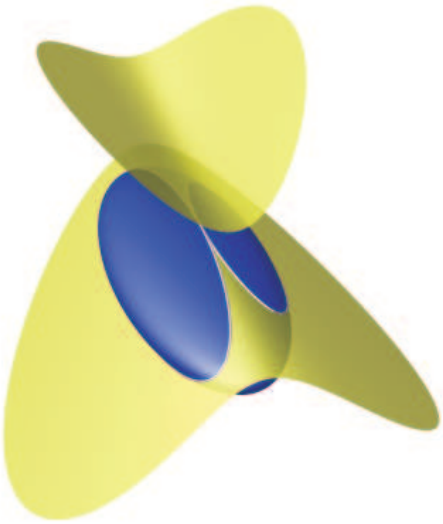

Figure 5.6: [(21)1] Two conics which touch: all the quadrics have stationary contact at the point of contact .

Figure 5.7: [2(11)] A conic and two generators, intersecting in three points, at which the quadrics touch.

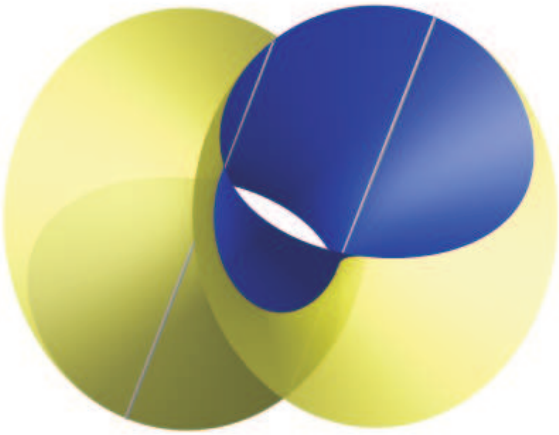

Figure 5.8: [(211)] Two generators counted twice: the quadrics touch all along the generators.

Figure 5.9: [(31)] Two generators and a conic which touches the plane of the generators at their intersection.

Figure 5.10: [(11)(11)] Four generators intersecting in four points, at which the quadrics touch

Figure 5.11: [(22)] Three generators one counted twice: this generator intersects each of the other two.

Figure 5.12: [22] A generator and a twisted cubic: the generator cuts the cubic in two distinct points, at which the quadrics touch.

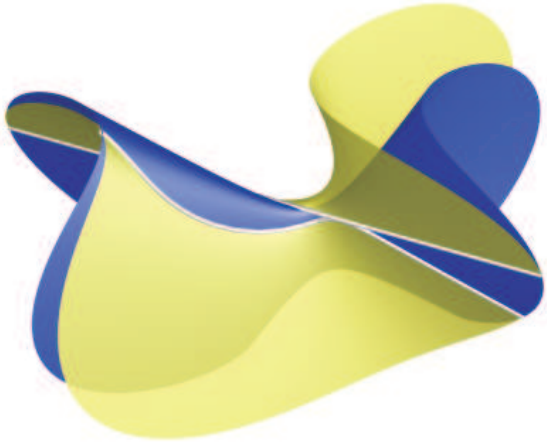

Figure 5.13: [4] A generator and a twisted cubic, the generator touches the cubic .

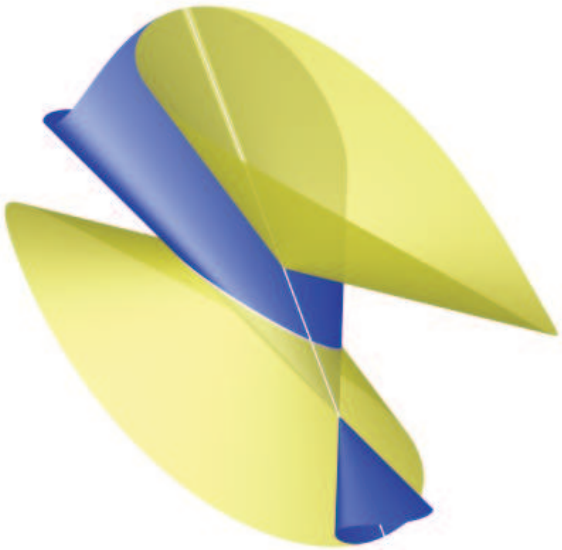

Figure 5.14: [{3}1] This is the singular case. A conic and a generator counted twice.

Figure **??** shows an example of the singular case. All except one quadric in the pencil are cones. Choosing a λ for $P(\lambda)$ corresponds to choosing a singular point of the cone along the generator. The cone could also be defined by the conic and the singular point. The quadric degenerates to two intersecting planes, if the point equals the touching point of the conic and the generator. One plane is the tangential plane along the generator and the other is the plane where the conic is embedded in. For a more detailed discussion of the singular case see [**?**]

Chapter 6

Spatial quadric arrangements

In this chapter we will describe our algorithm to calculate the intersection of a set of oriented quadric surfaces. We will give an unambiguous representation of the intersection curves and points. And finally we will discuss the predicates needed during the algorithm.

6.1 Algorithm for computing a cell within an arrangement of quadrics

We use an incremental algorithm to compute the boundary representation of the intersection body. This body may be composed of disjoint lumps each having several shells. The whole boundary consists of faces, edges and vertices which are embedded in quadric surfaces, intersection curves and intersection points, respectively. First of all we have to point out how to adequately represent the intersection curves (section**??**) and their intersection points (section **??**). Then we explain the data structure (section **??**) and the necessary topological operations to intersect a new quadric with an intermediate intersection body (section**??**).

6.1.1 Representation of an intersection curve

In order to deal with degenerated cases like tangential intersection, singular points and others, it is necessary to have an exact parameterization of the intersection curve. Unfortunately, it is not possible to get a rational parameterization in all cases, but we are able to give an exact parameterization involving only up to two square roots. For more details see [**?**].
In the case [1111], which is the general case, and also in the cases [211] and [31], the parameterization of the intersection curve is of the form

$$\mathbf{s}(t) = \mathbf{U} \cdot \begin{pmatrix} t \cdot b(t) \\ a(t) \pm \sqrt{c(t)} \\ t \cdot (a(t) \pm \sqrt{c(t)}) \\ b(t) \end{pmatrix},$$

where $\mathbf{U} \in \mathbb{Q}^{4\times4}$ and $a(t)$, $b(t)$ and $c(t) \in \mathbb{Q}[\sqrt{\delta}, \sqrt{\delta'}][t]$. $a(t)$ and $b(t)$ are polynomials of degree 2 and $c(t)$ is a polynomial of degree 4. We call

$$\mathbb{D}_s = \{t \ \in \mathbb{R} |\ c(t) \geq 0\}$$

the domain of s. As c is a polynomial of degree 4, $\mathbb{D}_s$ can be composed of up to three intervals. The methods, how to obtain this parameterization, will be discussed in section **??**.
Note that we have to deal with a positive and a negative arc like in the case of conics. But in general there is no linear dependency between a coordinate and the parameter t. In particular it is not a monotone parameterization of the intersection curve.

6.1.2 Representation of an intersection point

Let **A**, **B** and **C** denote the three quadrics we want to intersect. We are interested in all common points on their boundary. Let $\mathbf{s}_{AB}(t_{AB})$ be the parameterization of the intersection curve of the quadrics **A** and **B**. If we intersect this curve with the third quadric **C** we receive a polynomial $\mathbf{res}_{ABC}$ of degree 8 in the parameter t_{AB}. Each root of this polynomial corresponds to a triple intersection point of the three quadrics **A**, **B** and **C**. For a more detailed description see section **??**. Like in the case of conics, we can represent an intersection point unambiguously by the algebraic number for its parameter value and the exact parameterization of the arc it lies on. But note, that every intersection point lies at least on the three curves $\mathbf{s}_{AB}$, $\mathbf{s}_{AC}$ and $\mathbf{s}_{BC}$ and for each curve we receive a different parameter value. Therefore, we need a procedure that determines the `corresponding-intersection-points` on the other intersection curves, see section **??**.

6.1.3 The data structure

We consider all input quadrics as solid bodies. The result of a sequence of intersection operations on these bodies can again be regarded as a solid body. We represent such a solid body by describing its boundary, which may consist of several connected components. The topological entities of the body are its lumps, shells, faces, loops, edges and vertices. The geometric entities are the quadric surfaces, the intersection curves between two surfaces and the common intersection points of three or more surfaces. The two-dimensional boundary of the

body consists of a set of faces. Each face is embedded in a quadric surface. The one-dimensional boundary of a face is characterized by its loops. Each loop is composed of a set of oriented edges. The geometric information associated with an edge is given by the space curve the edge lies on. These space curves arise when two quadrics intersect. The zero-dimensional boundary of an edge consists of two vertices whose Cartesian coordinates correspond to multiple intersection points between the input quadrics.
In order to facilitate an efficient handling of the entities of the boundary representation we store some additional information about the neighborhood of the topological entities. It is sufficient to arrange the lumps of the body, the shells of a lump, the faces of a shell, and the loops of a face in simply linked lists. Every entity also has a pointer to its superior entity. A loop is represented as a doubly linked list of its oriented edges and it knows about the face it belongs to. In a two-manifold each edge is adjacent to exactly two faces. We store edges as two opposite oriented edges with a mutual reference. In addition each edge has a pointer to the unique loop it occurs in and pointers to its two vertices.
Naturally the intersection curve of two quadrics is not necessarily finite. But it would be much easier, if we could avoid a special handling of unbounded objects during the actual algorithm. Therefore, we compute a bounding sphere of all appearing intersection points in the set. This guarantees that the topology does not change outside the bounding sphere. We then choose this bounding sphere as the initial quadric. From there we can guarantee that all appearing edges and loops can be treated as finite objects. Of course we mark the entities on the sphere appropriately, in order to recognize them as infinite objects.

6.1.4 Topological operations

We want to intersect a face $\mathbf{f}$, which is embedded in a quadric $\mathbf{Q}_1$, with a second quadric $\mathbf{Q}_2$. See figure **??**! The boundary of the face $\mathbf{f}$ may consist of several loops (broken lines). For the sake of lucidity the different edges and vertices of the loops are not shown. The orientation of the edges within a loop is chosen such that the interior of $\mathbf{f}$ always lies to the left when moving in forward direction of the edges and looking opposite to the surface normal. This means that the vector $\mathbf{n}(\mathbf{p}) \times \frac{d\mathbf{r}}{dt}(\mathbf{p})$ points into the interior of $\mathbf{f}$ for each point $\mathbf{p} = \mathbf{r}(t)$ of the edge. Here $\mathbf{n}(\mathbf{p})$ is the normal vector of $\mathbf{Q}_1$ and $\frac{d\mathbf{r}}{dt}(\mathbf{p})$ is the tangential vector of $\mathbf{r}(t)$ at the point $\mathbf{p}$.
The first step in the computation of $\mathbf{f} \cap \mathbf{Q}_2$ is the determination of the parametric intersection curve $\mathbf{s}(t)$ (solid lines) between $\mathbf{Q}_1$ and $\mathbf{Q}_2$. This curve may consist of up to three loops. Each loop corresponds to an interval of $\mathbb{D}_s$ and consists of the positive and the negative arc. Next we calculate all intersection points (in our example $\mathbf{p}_1, \ldots, \mathbf{p}_{16}$) between $\mathbf{s}(t)$ and all edges of the loops of $\mathbf{f}$. This calculation has not only to provide the Cartesian coordinates of each intersection point, but also its parameter values with respect to $\mathbf{s}(t)$ and with respect to the

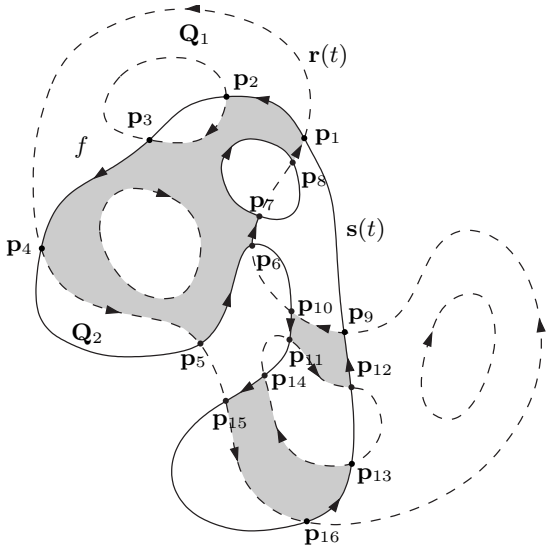

Figure 6.1: The intersection of a face $\mathbf{f}$, which is embedded in $\mathbf{Q}_1$, with a second quadric $\mathbf{Q}_2$.

curves $\mathbf{r}(t)$ of the intersected edges. These parameter values enable us to sort the intersection points along the curves they belong to.
In the next step we identify the new loops of $\mathbf{f} \cap \mathbf{Q}_2$. We start at an arbitrary intersection point $\mathbf{p}$ and follow the leftmost curve until we arrive at the next intersection point. We continue in this fashion until we return to our starting point. The decision which curve to follow at an intersection point $\mathbf{p}$ will be taken by the predicate `get-leftmost-edge` for an incoming edge at the point $\mathbf{p}$. This procedure produces an ordered sequence of edges of a new loop alternating between the original boundary curves of $\mathbf{f}$ and the newly created intersection curve. For these loops we can be sure that they belong to the final result. But as you see, figure **??**, there are also loops containing no new intersection point. We call these loops virgin loops, because they have not been touched yet. Up to now, we do not know whether they belong to the final result or not. Therefore, we have to test them separately, see section **??**.
As figure **??** illustrates $\mathbf{f} \cap \mathbf{Q}_2$ may decomposes into several connected components (the gray shaded regions). In order to compute all new faces, we have to determine the nesting of all loops embedded in $\mathbf{Q}_2$. We say that a loop l_1 includes an other loop l_2, if l_2 lies inside the area defined by l_1. Otherwise we say that l_1 excludes l_2. This relationship of two loops is determined by the predicate loop-in-loop, see **??**. As figure **??** illustrates, the nesting of the loops can be quite complex. Therefore, we store this relationship in two graphs $G1$ and $G2$, figure **??**, the vertices are the loops on the quadric. $G1$ is an undirected graph, an edge between two loops l_1 and l_2 means that they include each other. $G2$ is a directed graph, an edge from l_1 to l_2 means that l_1 includes l_2, but l_2 excludes l_1.
Now, we have to find the new faces. Obviously the loops of a face will form a

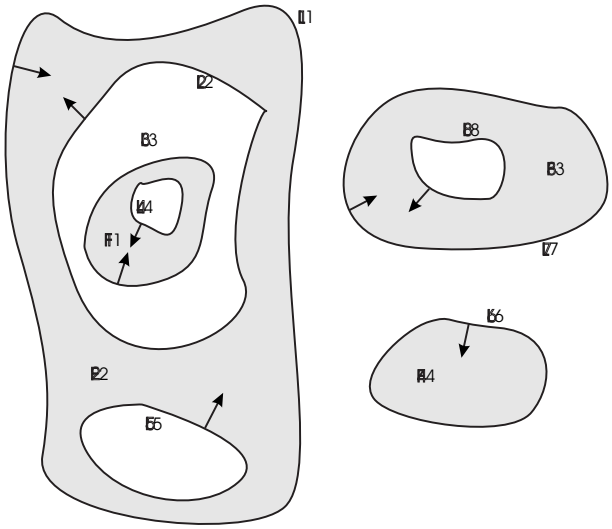

Figure 6.2: Example for the nesting of loops, showing the final faces

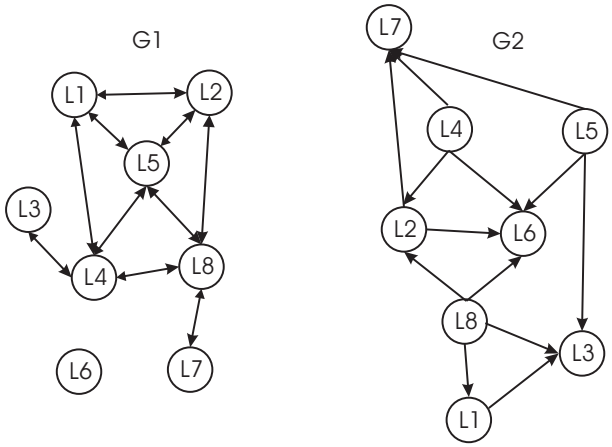

Figure 6.3: The graphs expressing the loop relationship in figure **??**

clique in $G1$. In the example, see figure **??**, $L1$, $L2$ and $L5$ form a clique while defining $F2$, but also $L1$, $L5$ and $L4$ form a clique. That's why we need $G2$. We start at some loop in $G2$ and follow the edges until we find a sink. $G2$ is obviously not cyclic and this guarantees us the existence of at least one sink. This is a so called outermost loop of a face and we can identify the face by this loop. In the example, see figure **??**, these loops are $L3$, $L6$, and $L7$. After we have found such a loop, we create a new face defined by this loop together with its direct neighbors in $G1$. Then we remove the loops we used from both graphs, and restart this process until the graphs are empty.
The operations for calculating $\mathbf{f} \cap \mathbf{Q}_2$ have to be performed for every face of the body. As a result, we obtain all new faces except those faces which lie on the surface of $\mathbf{Q}_2$. The boundary of these faces automatically ensues from sewing up all edges with their opposite directed counterparts. The connected components of the face set form the shells and the nesting of these shells yields the lumps of the intersection body. The nesting of the shells can be determined by ray-shooting.

6.2 Intersection of quadrics

In this section we want to show how to compute the intersection of quadrics. In particular we will give an exact parameterization of the intersection of two quadrics and we will see how to determine an exact representation of an intersection point of three quadrics.

6.2.1 Intersection of two quadrics

Our goal is to give an exact parameterization of the intersection curve of two quadrics **A** and **B**, in general position (case [1111]).

6.2.2 Intersection with a ruled quadric

A ruled quadric is a quadric that can be swept by a line. Therefore it can be parameterized by a parameterization $X(u, v)$ that is linear in at least one parameter.

$$X(u, v) = a(u) + v \cdot r(u),$$

where $a(u)$ is the starting point and $r(u)$ the direction vector of the line.

Let **A** be a ruled quadric and $X(u, v) = a(u) + v \cdot r(u)$ be the parameterization of **A**. As it is linear in v, we receive a polynomial which is quadratic in v, if we plug $X(u, v)$ into the implicit equation of **B**.

$$X(u, v)^T \mathbf{B} X(u, v) = b_2(u)v^2 + b_1(u)v + b_0(u) = 0$$

Now we can give an explicit solution for the parameter v

$$v(u) = \frac{-b_1 \pm \sqrt{b_1^2 - 4b_0b_2}}{2b_2},$$

which directly gives us a parameterization of $A \cap B$.

6.2.3 Intersection with a arbitrary quadrics

Let **A** and **B** be two non-ruled quadrics. In this case we have to search for a ruled quadric within the pencil

$$\mathbf{P}(\lambda) = \mathbf{A} + \lambda \mathbf{B}$$

of **A** and **B**. In 1976 J.Levin [**?**] already showed:

inertia of $\mathbf{Q}$	inertia of $\mathbf{Q_u}$	affine real type
(4,0,0)	(3,0,0)	$\emptyset$(imaginary ellipsoid)
(3,1,0)	(3,0,0)	ellipsoid
	(2,1,0)	hyperboloid of two sheets
	(2,0,1)	elliptic paraboloid
(2,2,0)	(2,1,0)	*hyperboloid of one sheeet*
	(1,1,1)	*hyperbolic paraboloid*
(3,0,1)	(3,0,0)	singular point
	(2,0,1)	$\emptyset$(imaginary elliptic cylinder)
(2,1,1)	(2,1,0)	cone
	(2,0,1)	elliptic cylinder
	(1,1,1)	*hyperbolic cylinder*
	(1,0,2)	*parabolic cylinder*
(2,0,2)	(2,0,1)	*line*
	(1,0,2)	$\emptyset$(imaginary parallel planes)
(1,1,2)	(1,1,1)	*intersecting planes*
	(1,0,2)	*parallel planes*
	(0,0,3)	*simple plane*
(1,0,3)	(1,0,2)	*double plane*
	(0,0,3)	$\emptyset$

Table 6.1: Euclidean type of a quadric $\mathbf{Q}$ in terms of the inertias of Q and Q_u. The ruled quadrics are emphasized.

Theorem 6.1: The intersection of two quadric surfaces lies in a plane, a pair of planes, a hyperbolic or parabolic cylinder, or a hyperbolic paraboloid.

In a first step he searches for a quadric with a singular submatrix. If neither $\mathbf{A}_u$ nor $\mathbf{B}_u$ are singular, he computes $\det(\mathbf{P}_u(\lambda)) = \det(\mathbf{B}_u)\lambda^3+K_2\lambda^2+K_1\lambda+\det(\mathbf{A}_u)$. Because of its degree, this polynomial has at least one real root. This guarantees, that the pencil has at least one quadric with a singular submatrix. All these quadrics, see table **??**, except the elliptic paraboloid and the elliptic cylinder, are ruled once. And, if the found quadric is an elliptic one, he is able to construct a parabolic hyperboloid by a linear combination of the elliptic quadric and $\mathbf{A}$. Thus the pencil always contains a ruled quadric.
But the main disadvantage of this method is, that in general the root of the degree three polynomial is irrational, and also can not be represented by an one-root-expression. Thus, we can not use Levin's method to derive an exact parameterization of the intersection curve.
Therefore, we follow an approach proposed by S.Lazard et al. [**?**]. If we can not directly give a suitable parameterization of $\mathbf{A}$ or $\mathbf{B}$, we determine

$$\Delta(\lambda) = det(\mathbf{A} + \lambda\mathbf{B}).$$

This is a polynomial of degree 4. By Theorem **??** there exists at least one ruled quadric in the pencil. For these quadrics, see table **??**, $\Delta(\lambda) \geq 0$. Thus in general we can choose a rational λ_0, such that $det(\mathbf{P}(\lambda_0)) > 0$. It follows that $\mathbf{P}$ has either inertia (4,0,0) or (2,2,0). If $\mathbf{P}$ has inertia (4,0,0), then $\mathbf{A} \cap \mathbf{B} = \emptyset$. If $\mathbf{P}$ has inertia (2,2,0), then $\mathbf{P}$ is either a *hyperboloid of one sheet* or a *hyperbolic paraboloid* which are both ruled quadrics.
If $\Delta(\lambda) \leq 0 \; \forall \lambda \in \mathbb{R}$, then $\Delta(\lambda)$ has at least one double root. If it has one double root, this root is rational. And by Levin this quadric has to be a ruled one. In the case of two double roots, which would cause irrational coefficients, S. Lazard et al. were able to show, that in this case the intersection of $\mathbf{A}$ and $\mathbf{B}$ only consists of two points, which we do not need to parameterize. Thus in general we have to deal with hyperboloids of one sheet or hyperbolic paraboloids.
The next important step to avoid roots is to use a Gram-Schmidt-orthogonalization, see **??**, to compute a transformation matrix $U \in \mathbb{Q}^{4\times4}$, such that

$$U^T P U = D = \begin{pmatrix} d_1 & 0 & 0 & 0 \\ 0 & -d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & -d_4 \end{pmatrix},$$

where $d_i > 0 \forall i$. Of course we can already give a parameterization of the quadric P, but this parameterization involves four square roots $\sqrt{d_1}$, $\sqrt{d_2}$, $\sqrt{d_3}$ and $\sqrt{d_4}$. To reduce the number of square roots we perform an other transformation

$$V = \begin{pmatrix} \frac{1}{d_1} & 1 & 0 & 0 \\ \frac{1}{\sqrt{d_1 d_2}} & -\frac{d_1}{\sqrt{d_1 d_2}} & 0 & 0 \\ 0 & 0 & -\frac{1}{d_3} & 1 \\ 0 & 0 & -\frac{1}{\sqrt{d_3 d_4}} & -\frac{d_3}{\sqrt{d_3 d_4}} \end{pmatrix}.$$

This transformation only uses the square roots $\sqrt{d_1 d_2}$ and $\sqrt{d_3 d_4}$. We receive

$$P' = V^T U^T P U V = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{pmatrix}.$$

The implicit equation of P' is

$$x_1 x_2 - x_3 x_4 = 0.$$

Obviously the parameterization of P'

$$X(u, v) = \begin{pmatrix} u \\ v \\ uv \\ 1 \end{pmatrix}$$

is linear in even both parameters. We plug this parameterization into the implicit equation of the quadric $B' = V^T U^T BUV$. This gives us a polynomial b of total degree 4 but only quadratic in both parameters. We rewrite this polynomial as a polynomial in v

$$X(u,v)^T B' X(u,v) = b_2(u)v^2 + b_1(u)v + b_0(u) = 0.$$

Therefore, we can give an explicit solution for the parameter v

$$v(u) = \frac{-b_1 \pm \sqrt{b_1^2 - 4b_0 b_2}}{2b_2}$$

and receive the parameterization of $\mathbf{A} \cap \mathbf{B}$:

$$X(u) = UV \begin{pmatrix} u \cdot 2b_2(u) \\ -b_1(u) \pm \sqrt{b_1(u)^2 - 4b_0(u)b_2(u)} \\ u \cdot (-b_1(u) \pm \sqrt{b_1(u)^2 - 4b_0(u)b_2(u)}) \\ 2b_2(u) \end{pmatrix}$$

with coefficients in $\mathbb{Q}[\sqrt{d_1 d_2}, \sqrt{d_3 d_4}]$

6.2.4 Intersection with a third quadric

If we want to determine the intersection with a third quadric $\mathbf{C}$, we can plug the parameterization $UVX(u,v)$ of $\mathbf{P}$ into $\mathbf{C}$. We again receive a polynomial c of total degree 4, but only quadratic in both parameters and coefficients in $\mathbb{Q}[\sqrt{d_1 d_2}, \sqrt{d_3 d_4}]$.

$$c(u,v) = c_2(u)v^2 + c_1(u)v + c_0(u)$$

We already determined $\mathbf{P} \cap \mathbf{B}$, see **??**, and received the polynomial

$$b(u,v) = b_2(u)v^2 + b_1(u)v + b_0(u)$$

The common roots of these two polynomials correspond to the intersection of $\mathbf{A} \cap \mathbf{B} \cap \mathbf{C}$. Therefore we compute the resultant res_{ABC} of b and c with respect to v. By theorem **??** res_{ABC} is a polynomial of degree at most 8. Each root of this polynomial corresponds to an intersection point of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. But as in the case of conics we still have to decide, whether the points lies on the positive or negative arc of the intersection curve.
Let ξ be a root of r_{ABC} in the domain of $X(u)$. We know that the multiplicity of ξ equals the sum of the multiplicities of the intersection points at ξ, see **??**. Therefore, we have to do a case by case study, depending on the multiplicity of ξ.

1. case: ξ is a simple root

   Let $\xi \in \mathbb{R}$ be a simple root of r_{ABC}. In this case we know, that there is only one transversal intersection point $\mathbf{p}$ at the position ξ. If ξ is a one-root-expression, see **??**, we simply evaluate the point by `LEDA reals` . But, in general ξ will be represented as an algebraic number.

   $$\xi \;=\; (r_{ABC}, [\underline{\xi}, \overline{\xi}]\ ) \quad \text{with} \quad r_{ABC} \in \mathbb{Q}[\sqrt{\delta_1}, \sqrt{\delta_2}][x] \quad \text{and} \quad \underline{\xi}, \overline{\xi} \in \mathbb{Q}\ .$$

   Instead of ξ, we can take the rational endpoints of the isolating interval $\underline{\xi}$ and $\overline{\xi}$. Now we evaluate the positive arc of $X(u)$ at $\underline{\xi}$ and $\overline{\xi}$ and receive two points

   $$\begin{aligned} \underline{p} &= X^+(\underline{\xi}) \\ \overline{p} &= X^+(\overline{\xi})). \end{aligned}$$

   Because $\mathbf{p}$ is a transversal intersection point and $[\underline{\xi}, \overline{\xi}]$ is an isolating interval of r_{ABC}, there is a sign change between $\mathbf{C}(\underline{p})$ and $\mathbf{C}(\overline{p})$, iff $\mathbf{p}$ lies on the positive arc of $X(u)$. So if

   $$sign(\mathbf{C}(\underline{p})) \neq sign(\mathbf{C}(\overline{p})),$$

   we know that the intersection point lies on the positive arc and otherwise on the negative arc.

2. case: ξ is a double root

   In this case we can have two transversal intersection points or a double intersection point at the position ξ. Unfortunately r_{ABC} can have up to four double roots. Thus in general, we still have to represent ξ as an algebraic number of degree four.

   $$\xi \;=\; (g, [\underline{\xi}, \overline{\xi}]\ ) \quad \text{with} \quad g = gcd(r_{ABC}, r'_{ABC}) \quad \text{and} \quad \underline{\xi}, \overline{\xi} \in \mathbb{Q}\ .$$

   In the case of two transversal intersection points we can detect both by the sign change argument as in the case of a simple root. If this test detects no sign change, we know that there is one double intersection point. In this case we use interval arithmetic to decide whether the double point $\mathbf{p}$ lies on the positive or on the negative arc: We evaluate both arcs with interval arithmetic and plug them into the implicit equation of $\mathbf{C}$. We receive an interval for the positive and one for the negative arc.

   $$I^+_\xi = C(X^+([\underline{\xi}, \overline{\xi}]))$$

$$I_\xi^- = C(X^-([\underline{\xi}, \overline{\xi}]))$$

We know, that if the intersection point lies on an arc, then the corresponding interval contains zero. So if only one interval contains zero, we know that $\mathbf{p}$ lies on the arc corresponding to this interval. If both intervals contain zero we can refine the isolating interval of ξ, until only one interval contains zero.

3. case: ξ has multiplicity ≥ 3

   In this case ξ is an one-root-expression. Therefore we can evaluate both arcs at ξ exactly. If

   $$C(X^+(\xi)) = 0,$$

   then one intersection point lies on the positive arc. Of course we can do the same with the negative arc.

6.3 Elementary Procedures

In this section we want to discuss the elementary procedures we need for the algorithm.

6.3.1 Find corresponding intersection points

In section **??** we have seen, that we are able to represent an intersection point $\mathbf{p}$ unambiguously by the exact parameterization of the arc it lies on and its parameter value. This parameter value might be represented exactly using `LEDA reals` or by an algebraic number. In general the three quadrics $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ can have up to eight common points. Each point lies on the three curves $\mathbf{s}_{AB}$, $\mathbf{s}_{AC}$ and $\mathbf{s}_{BC}$. But each curve lies in a different parameter space, therefore each point has a different parameter value for each curve. This means that for each point on $\mathbf{s}_{AB}$ we still have to find the corresponding intersection point on $\mathbf{s}_{AC}$.
Two points are identical iff the Cartesian coordinates are identical, but in general the parameter value of a point will be an algebraic number. Therefore, we use interval arithmetic and receive a bounding box for each point on $\mathbf{s}_{AB}$ and for each point on $\mathbf{s}_{AC}$. We can guarantee that each box contains at least the point it belongs to. Therefore, two boxes representing the same point will necessarily overlap. In the best case we will receive pairs of overlapping boxes and can make our decision. But it can happen that more than two boxes overlap. If for instance two points are too close, then we receive four overlapping boxes. But we already know, that the four boxes actually represent two distinct points. Therefore we know, that we can refine the isolating interval until we reach a dissection into box pairs.

6.3.2 Sorting intersection points along the intersection curve

Let $\mathbf{s}_{AB}$ be the intersection curve of the quadrics $\mathbf{A}$ and $\mathbf{B}$. The intersection of this curve with the two quadrics $\mathbf{C}$ and $\mathbf{D}$ gives us two polynomials res_{ABC} and res_{ABD}. The roots of these polynomials correspond to the parameter values of the intersection points on $\mathbf{s}_{AB}$. Of course we already know for each intersection point the arc it lies on (see section **??**). Therefore sorting the points along the curve is just comparing two algebraic numbers. But note that this predicate will be used very frequently and we should spend some effort into the efficiency of this predicate.

6.3.3 Get leftmost edge

This predicate arises during the intersection of a face which is embedded in a quadric $\mathbf{Q}_1$ with a second quadric $\mathbf{Q}_2$. For a given intersection point $\mathbf{p}$ and an

incoming edge the predicate has to decide which edge to follow after this point. The orientation of the edges within a loop is chosen such that the interior of $\mathbf{f}$ always lies to the left when moving in forward direction of the edges and looking opposite to the surface normal. The same holds for the arcs of the intersection curve of $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Therefore we have to follow the *left most edge*.
In general, where we think of curves without self-intersection/singular points, we only need to consider two candidates, namely the outgoing edge of the new curve and the outgoing edge of the old face. The idea is to take a point on the edge and test, whether this point lies in the interior of the intersection. We already determined an isolating interval for the intersection point on both curves. Therefore, we can evaluate the edge at the rational border of the isolating interval in the right direction and receive a suitable test point $\mathbf{p}$. If $\mathbf{p}$ lies in the intersection of all quadrics we know that we have to follow this edge. But to insure this we only have to test this point with one quadric. The point on the old edge is tested by Q_2, and the point on the new intersection curve is tested by the quadric corresponding to the outgoing edge of the face.
This test only fails if the intersection point is a singular point of an intersection curve. In this case we have to take more than two edges into consideration and will receive at least two possible edges by the test described above. In fact the edges will be part of the result, but we can not simply choose one of them, because this would cause degenerated loops. The situation at a singular point is very complex, because of the huge variety of the cases in which singular points can occur. Thus this will be part of further work.

6.3.4 Loop in loop

This predicate determines whether a loop l_1 includes or excludes a loop l_2.
Of course l_1 can only include l_2 if both loops lie on the same sheet of their quadric $\mathbf{Q_1}$. For this test we take an uncritical point p_1 of l_1. Uncritical means that we choose a point which lies on an edge and is defined by parameter value $t_1 \in \mathbb{Q}$. Then we determine the sheet of the quadric the point lies on using `LEDA reals` .
In the same way we choose a point p_2 on l_2.
If both loops lie on the same sheet we really have to test whether l_1 includes l_2. The idea is to construct a path from l_1 to l_2 on $\mathbf{Q_1}$. Then we determine whether the way locally leaves l_1 to the inside or outside. To construct this path we take the points p_1, p_2 and a third point p_3. These points define a plane, and this plane defines a conic on $\mathbf{Q_1}$. This conic necessarily intersects both loops in p_1 and p_2, respectively. But in order to define a way between l_1 and l_2 the conic has to stay finite between p_1 and p_2. Therefore we have to choose p_3 depending on the type of $\mathbf{Q_1}$. For example for an hyperbolic paraboloid we choose its origin. Then we compute all intersection points of both loops with the plane and order them along the conic. After that we choose two neighbored intersection points p_1' of l_1 and p_2' of l_2 and determine whether the conic leaves l_1 to the inside or outside at

p_1'. This is a sign test of the determinant of the tangential vectors of the conic and the edge and the normal vector of Q_1 at p_1'.
The whole computation is done by interval arithmetic, see section **??**. Therefore it may happen that we can not order the points on the conic, or that the sign of the determinant is ambiguous. This will not happen very often because we have chosen the points p_1 and p_2 at uncritical positions. Nevertheless, if the computation fails we restart the process with higher precision and with new randomly chosen points p_1 and p_2. This randomized method stays in contrast to the other constructions use due to the algorithm, thus we should think of an deterministic and maybe exact method in further work.
Note that this construction is symmetric in l_1 and l_2, thus we also can determine whether l_2 includes l_1 or not.

6.3.5 Handle virgin loops

The problem appears in $\mathbf{f} \cap \mathbf{Q}_2$, where $\mathbf{f}$ is embedded $\mathbf{Q}_2$. The loops which have been created by an intersection point are, by construction, part of the result. But old loops of the old faces or new loops of the intersection curve of $\mathbf{Q_1}$ and $\mathbf{Q_2}$, which have no new intersection point are not necessarily part of the result. I call these loops virgin loops, because they haven't been touched yet. For a old loop we have to check whether it lies inside $\mathbf{Q_2}$. Because it does not intersect the intersection curve of $\mathbf{Q_1}$ and $\mathbf{Q_2}$, we know that it either lies completely inside or outside of $\mathbf{Q_2}$. Therefore we only have to test one point p_1 on the loop.

$$p_1^T \mathbf{Q_2} p_1 < 0$$

For a virgin loop of the intersection curve between $\mathbf{Q_1}$ and $\mathbf{Q_2}$ we have to check, whether it lies inside of all other quadrics except $\mathbf{Q_1}$ and $\mathbf{Q_2}$. We evaluate a point p_2 on the loop with a rational parameter value t_2. Then we test whether p_2 lies inside all other quadrics.

$$p_2^T \mathbf{Q_i} p_2 < 0 \ \forall i \notin 1, 2$$

Testing the point is enough, because we know that the loop does not intersect any loop on $\mathbf{Q_1}$.
But for the virgin loops received from the new intersection curve, up to now we do not know in which of the old faces it lies. It lies in an old face if it lies inside all loops of this face, which we determine by the predicate `loop-in-loop`. Thus we have to use such an loop within the graphs G1 and G2 for this face, as described in section **??**

6.4 Summary

Chapter 7

Appendix

7.1 Interval arithmetic

In this section, we will give a description of the main tool to accelerate our calculations, the interval arithmetic. The main idea is to use fast floating point arithmetic and keep track of the rounding errors at the same time.
Sometimes, it is not possible to express a number explicitly. For example, we have to use algebraic numbers to express the root of a polynomial of higher degree. Then, the root is expressed as an isolating interval with rational endpoints, it lies in, and the polynomial. This leads to interval arithmetic without rounding.

7.1.1 Interval arithmetic without rounding

First let us define the arithmetic operations on intervals.

Definition 7.1: $I(\mathbb{Q})$ = set of all closed real intervals with rational endpoints. Let $\circ \in \{+, -, \cdot, /\}$ and A, $B \in I(\mathbb{Q})$, then

$$A \circ B := \{x = a \circ b \mid a \in A, b \in B\}$$

Let $A = [a_{min}, a_{max}]$ and $B = [b_{min}, b_{max}]$, then

$$\begin{aligned}
A + B &:= [a_{min} + b_{min}, a_{max} + b_{max}] \\
A - B &:= [a_{min} - b_{max}, a_{max} - b_{min}] \\
A \cdot B &:= [\min\{a_{max}b_{max}, a_{max}b_{min}, a_{min}b_{max}, a_{min}b_{min}\}, \\
&\qquad \max\{a_{max}b_{max}, a_{max}b_{min}, a_{min}b_{max}, a_{min}b_{min}\}] \\
A/B &:= [a_{min}, a_{max}] \cdot [\frac{1}{b_{max}}, \frac{1}{b_{min}}] \text{ for } 0 \notin B
\end{aligned}$$

7.1.1.1 Properties of interval arithmetic without rounding

commutativity:	$A + B = B + A$	$A \cdot B = B \cdot A$
associativity:	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
neutral elements:	$[0, 0]$ $0 \in A - A$	$[1, 1]$ $1 \in A/A$
subdistributivity:	$A \cdot (B + C) \subseteq A \cdot B + A \cdot C$	
	$a \cdot (B + C) = a \cdot B + a \cdot C$	$a \in \mathbb{Q}$

inclusion monotonicity:
$A_1 \subseteq A_2$ and $B_1 \subseteq B_2$
$\Rightarrow A_1 \circ B_1 \subseteq A_2 \circ B_2$ for$\circ \in \{+, -, \cdot, /\}$

7.1.2 Interval arithmetic with rounding

The next idea is to use interval arithmetic as a filtering technique. But a filter should be very fast, therefore we use intervals bounded by floating points instead of rational numbers. The problem is, that we have to keep track of the rounding errors caused by floating point arithmetic. Therefore we have to round such that we still can keep up the **inclusion property:**

$$\text{if } a \in A \text{ and } b \in B \Rightarrow a \circ b \in A \circ B.$$

The idea is to use rounding to minus infinity $\downarrow$ for lower bounds and to use rounding to plus infinity $\uparrow$ for upper bounds.
Let $\circ \in \{+, -, \cdot, /\}$ and $A = [a_{inf}, a_{sup}]$ and $B = [b_{inf}, b_{sup}]$, then

$$A \circ B \supseteq \{x = a \circ b \mid a \in A, b \in B\}.$$

$$\begin{aligned}
A + B &:= [(a_{inf} + b_{inf})\downarrow, (a_{sup} + b_{sup})\uparrow] \\
A - B &:= [(a_{inf} - b_{sup})\downarrow, (a_{sup} - b_{inf})\uparrow] \\
A \cdot B &:= [(\min\{a_{sup}b_{sup}, a_{sup}b_{inf}, a_{inf}b_{sup}, a_{inf}b_{inf}\})\downarrow, \\
&\qquad (\max\{a_{sup}b_{sup}, a_{sup}b_{inf}, a_{inf}b_{sup}, a_{inf}b_{inf}\})\uparrow] \\
A/B &:= [a_{inf}, a_{sup}] \cdot [(\frac{1}{b_{sup}})\downarrow, (\frac{1}{b_{inf}})\uparrow] \text{ for } 0 \notin B
\end{aligned}$$

But note that switching the rounding modes spends too much time. Therefore a naive implementation that switches the rounding mode for each arithmetic operation is too costly. It would be much better to use only one rounding mode for all arithmetic operations. Therefore we always use round to plus infinity

and represent the lower bound of an interval by its negative value. Therefore, a rounding to plus infinity for this value is actually a rounding to minus infinity.
In our implementation we used `LEDA bigfloats` [**?**]. As they provide the different rounding modes, and a flexible precision.
In most cases interval arithmetic is good enough to make topological statements. For example, we can determine the sign of an expression, if zero is not included in the resulting interval, and if the resulting interval contains zero we still have some possibilities. So, if we know for some reason that the expression is not zero, it is possible to refine the interval until it has an unambiguous sign. In particular we can increase the precision of the arithmetic, or refine the initial intervals of the algebraic numbers.
But in some cases we can even use interval arithmetic to figure out whether an expression is zero. Suppose there are more than one expression, but we know that only one of them is zero. Than we can test, whether only one interval contains zero. In this case we know that the corresponding expression actually is zero.

7.2 Uspenskys Algorithm

An important aspect for our algorithms is real root isolation. Thus we need an exact routine, that is able to determine isolating intervals of all roots for any given polynomial. There are several methods such as Kronecker's Algorithm [**?**], Sturm sequences [**?**], or zero isolation by differentiation [**?**] to achieve root isolation, but turned out be pretty slow. Therefore we desided to use a variant of Uspensky's algorithm, proposed by G.E. Collins et. al. [**?**]
Uspenskys method is based in part on a special case of **Descartes' rule of signs**.

Theorem 7.2: For a polynomial $P(x) = \sum_{0 \leq i \leq n} a_i x^i \in \mathbb{R}[x]$, let $V(P)$ be the number of sign variations in its coefficient sequence, i.e., number of pairs i, j with $0 \leq i < j \leq n = \deg(P)$, $a_i a_j < 0$ and $a_{i+1} = \cdots = a_{j-1} = 0$. Then $V(P) - pos(P)$ is an even non-negative integer, where $pos(P)$ is the number of positive real roots of P, counted with multiplicity. In particular, if $V(P) = 0$, then P has no positive roots, and if $V(P) = 1$, then P has exactly one simple positive real root.

Proof. A proof of Descartes' Rule for polynomials of arbitrary degree can be carried out by induction. The base case for degree 1 polynomials is easy to verify! So assume that P is a polynomial with positive leading coefficient. The final coefficient of P is given by a_0.
If $a_0 > 0$, then the number of sign changes must be even, since the first and last coefficient of P are both positive. Moreover, the number of roots must also be even, since P is also positive for very large x, so the graph of P can only cross the x-axis an even number of times. Similar arguments show that if $a_0 < 0$, then the number of sign changes is odd and the number of positive roots is odd. Thus the number of sign changes and number of roots have the same parity.
If P had more roots than sign changes then it must have at least 2 more roots. But P' is a polynomial with zeroes between each of the roots of P, so P' has at least 1 more root than sign changes of P. This yields a contradiction because P' has no more sign changes than P does, and the inductive hypothesis then implies that P' has no more roots than sign changes of P. □

By transforming a polynomial A to a polynomial A' whose positive roots correspond to the roots of A in a given interval, the two special cases of Descartes' rule can be applied to conclude in favorable cases that an interval has no real roots or is an isolating interval for A.
The algorithm starts with an initial interval $(0, b)$ obtained from a root bound b. Then the polynomial A is transformed such that all positive root are now within the interval $(0, 1)$. Then the subalgorithm applies Descartes' rule to this interval to determine if this interval is an isolating interval. If more that one variation is obtained for the transformed polynomial, the interval is bisected and the subalgorithm is applied recursively to the polynomials $A/(2x - 1)$ and

$2^n A((x-1)/2$ (where $n = \deg(A)$), whose roots correspond to the roots in the two subintervals.
The difficulty is that Descartes' rule actually counts all roots with positive real parts. Thus we may subdivide an interval that only contains complex roots. Therefore a termination of the algorithm is not obvious. A termination proof can be given by the theorem of Vincent [**?**], showing that all complex roots after a finite number of bisections finally have negative real part.

Uspenskys algorithm:

main algorithm:

input: A square free polynomial A, with $deg(A) = n > 0$.
output: Return a list L of the isolating intervals for all positive real roots of A.

(1) determine a root bound of the form 2^k
(2) **if** $(k \geq 0)$ **then** $B(x) := A(2^k x)$
else $B(x) := 2^{-kn} A(2^k x)$
(3) $L' :=$ subalgorithm(B)
(4) **for each** (a'_i, b'_i) **in** L' $\{L.\text{add}((2^k a'_i, 2^k b'_i))\ \}$
(5) **return** L

subalgorithm:

input: A square free polynomial A, with $deg(A) = n > 0$.
output: Return a list L of the isolating intervals for all positive real roots of A in (0,1).

(1) $A^* := (x+1)^n A(1/(x+1))$
$r := V(A^*)$
if $(r == 0)$ **then return** $L := ()$
if $(r == 1)$ **then return** $L := ((0,1))$
(2) **if** $(A(1/2) == 0)$
then $L.\text{add}((1/2, 1/2))$; $A := A/(2x-1)$
(3) $A' := 2^n A(x/2)$;
$L' :=$ subalgorithm(A')
foreach (a'_i, b'_i) **in** L' $\{L.\text{add}((a'_i/2, b'_i/2))\}$
(4) $A'' := A'(x+1)$;
$L'' :=$ subalgorithm(A'')
foreach (a''_i, b''_i) **in** L'' $\{L.\text{add}(((a''_i+1)/2, (b''_i+1)/2))\}$
(5) **return** L

7.3 Root isolation by floating point arithmetic

The main disadvantage of Uspenskys Algorithm is, that it is an exact algorithm, meaning that it is still very slow compared to algorithms using floating point arithmetic. The disadvantage of these methods is, that we can not rely on the computed roots, without further analysis.
The idea is to use a fast algorithm for floating point arithmetic to compute an approximation of all roots of a polynomial. And in a second step, to compute an error bound for each root. Then we use this error bound to check whether we achieved root isolation.
To compute this error bound we use a numerical analysis due to Smith [**?**], which is actually base on the theorem of Gerschgorin-circles. We only state the theorem for square free polynomials.

Theorem 7.3: Let x_i, $1 \leq i \leq n$ be estimates for the roots of an n-th degree square free polynomial $f(x)$ and let $g(x) = \prod_{1 \leq i \leq n}(x - x_i)$ (ideally $f(x)/g(x) = a_n$). Let

$$\sigma_i = \frac{f(x_i)}{a_n g'(x_i)} = \frac{f(x_i)}{a_n \prod_{j \neq i}(x_i - x_j)} \quad \text{for } 1 \leq i \leq n.$$

Let Γ_i be the disk centered at x_i with radius $n|\sigma_i|$. Then the union of the disks contains all roots of f. Moreover, a connected component consisting of k disks contains exactly k roots of f. Observe that $g'(x) = \sum_i \prod_{j \neq i}(x - x_j)$.

Proof. Consider matrix $A = diag(x_1, \ldots, x_n) - \Sigma$ where the i-th row of Σ has all its entries equal to σ_i. The characteristic polynomial of A, i.e., $\det(\lambda I - A)$, is equal to $f(x)/a_n$. The equality is shown by showing that the polynomials agree on the points x_i, $1 \leq i \leq n$. Observe that both polynomials are monic (leading coefficient one) and hence it suffices to show that they agree on n distinct points. Indeed,

$$\det(x_i I - A) = \begin{vmatrix} x_i - x_1 + \sigma_1 & \sigma_1 & \ldots & \sigma_1 & \ldots & \sigma_1 \\ \sigma_2 & x_i - x_2 + \sigma_2 & \ldots & \sigma_2 & \ldots & \sigma_2 \\ \ldots & & & & & \\ \sigma_1 & \sigma_i & \ldots & \sigma_i & \ldots & \sigma_i \\ \ldots & & & & & \\ \sigma_n & \sigma_n & \ldots & \sigma_n & \ldots & x_i - x_n + \sigma_n \end{vmatrix}$$

Subtracting the i-th column from all other columns shows that the determinant is equal to $\sigma_i \prod_{j \neq i}(x_i - x_j) = f(x)/a_n$.
The eigenvalues of A are therefore equal to the roots of f and the claim follows from Gerschgorin's theorem. □

Gerschgorin's theorem defines a family of disks containing the eigenvalues of a matrix.

Lemma 7.4 (Gerschgorin): Let A be a matrix. Then all eigenvalues of A are contained in the union of the circles Γ_i. Γ_i is centered at a_{ii} and has radius equal to the sum of the absolute values of the off-diagonal elements.

Proof. Let λ be an eigenvalue of A and let w be the corresponding eigenvector. Then for all i

$$\sum_j a_{ij} w_j = \lambda w_i$$

and hence

$$\sum_{j \neq i} a_{ij} w_j = (\lambda - a_{ii}) w_i$$

Let i be such that w_i has the largest absolute value among the w_j. Then

$$|\lambda - a_{ii}| = |\sum_{j \neq i} a_{ij} w_j| / |w_i| \leq \sum_{j \neq i} |a_{ij}|.$$

□

For the implementation we use an iterated eigenvalue algorithm for approximating the roots of univariate polynomials, as described by S. Fortune [**?**]. Then we use interval arithmetic to determine the error bound for each root. Now we represent each root using interval arithmetic, thus every root corresponds to an box in $\mathbb{C}$. If a box does not intersect any other box, we achieved root isolation for the corresponding root. If such a box intersects the real axis, we check for a real root by a sign variation argument.
In a filtering step, it is also possible to uses this method for polynomials with coefficients represented by interval arithmetic. First we approximate the roots of the polynomial, defined by the midpoints of the intervals of each coefficient. Then we determine the error bounds for each root by Smith's method, using the original intervals of the coefficients.
Note that for two roots, which are nearly degenerated, especially the error bound for these two roots is very bad. Therefore, the discs for these two roots will still overlap, while all other root already achieved root isolation. Thus, if we recompute all roots with higher precision, the time we spend to recompute the already isolated roots is obviously wasted. But, there are algorithms, for example introduced by D.Manocha et. al. [**?**] that take care of this fact. Thus we should think of such an algorithm for further implementations.

7.4 Gram-Schmidt-orthogonalization

Let $A = A^T \in \mathbb{Q}^{n\times n}$. Our goal is to give a rational transformation matrix $U \in \mathbb{Q}^{n\times n}$ such that

$$U^T AU = D = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & d_{n-1} & 0 \\ 0 & \dots & 0 & d_n \end{pmatrix}$$

Let $U = [u_1, \dots, u_n]$ be such a transformation matrix, then these vectors fulfill the following condition:

$$u_i^T A u_j = d_i \delta_{ij} \ \forall i, j \in \{1, \dots, n\},$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Let a and b two arbitrary vectors in $\mathbb{Q}^n$. If $a^T Aa \neq 0$ we can define $b' := b - \frac{b^T Aa}{a^T Aa} a$, such that

$$\begin{aligned} a^T Ab' &= a^T A(b - \frac{b^T Aa}{a^T Aa} a) \\ &= a^T Ab - \frac{(a^T Aa)(b^T Aa)}{a^T Aa} \\ &= a^T Ab - b^T Aa \text{ by symmetry of } A \text{ follows} \\ &= 0 \end{aligned}$$

This observation leads to the **Gram-Schmidt-orthogonalization**:
Let $A = A^T \in \mathbb{R}^{n\times n}$. We start with a set of n linear independent vectors $V = \{a_i \mid i = 1 \dots n\}$.
In step k we select a vector in V, without loss of generality let this vector be a_k, such that $a_k^T Aa_k \neq 0$. Then we define $u_k := a_k$ and redefine the remaining vectors in V by

$$a_j := a_j - \frac{a_j^T Aa_k}{a_k^T Aa_k} \ \forall j > k.$$

If there is no such vector a_k with $a_k^T Aa_k \neq 0$, we search for a pair of vectors in V such that $a_i^T Aa_j \neq 0$. We can conclude

$$(a_i + a_j)^T A(a_i + a_j) = 2a_i^T Aa_j \neq 0.$$

Therefore we replace a_j by $a_j + a_i$ and proceed as above.
If there is neither a vector nor such a pair of vectors in V we have finished up the diagonalization and fill up the matrix U by the remaining vectors in V.

7.5 Resultants

7.5.1 Resultants of univariate polynomials

Let us first consider univariate polynomials. In the following we will state all theorems over the rational numbers because this is the field we are working over. Nevertheless everything also holds over the complex numbers $\mathbb{C}$. We will denote the *degree* of a polynomial f by $\deg(f)$ and the *leading coefficient* by $\mathrm{ldcf}(f)$. The degree of the zero polynomial is equal to -1: $\deg(0) = -1$. We call a polynomial $h \in \mathbb{Q}[x]$ a *factor* of $f \in \mathbb{Q}[x]$ if there exists another polynomial $f_1 \in \mathbb{Q}[x]$ with $f = h \cdot f_1$. Especially if f is the zero polynomial then every $h \in \mathbb{Q}[x]$ is a factor of f.
When do two univariate polynomials f and g of positive degree have a nonconstant common factor?

Theorem 7.5: Let $f, g \in \mathbb{Q}[x]$ be two rational polynomials of degrees $\deg(f) = n > 0$ and $\deg(g) = m > 0$. Then f and g have a nonconstant common factor if and only if there are rational polynomials A and B with $\deg(A) < m$ and $\deg(B) < n$ which are not both zero such that $Af + Bg = 0$.

Proof. First assume that f and g have a nonconstant common factor h. Then we can write $f = hf_1$ and $g = hg_1$, where $f_1, g_1 \in \mathbb{Q}[x]$. It is easy to check that $A = g_1$ and $B = -f_1$ fulfill the required properties. Conversely assume that there are complex polynomials A and B of degree at most $m-1$ and $n-1$ which are not both zero such that $Af = -Bg$. Then all irreducible factors of $A(x)f(x)$ divide $B(x)g(x)$. But B has degree at most $n-1$ and therefore cannot contain all factors of f with their multiplicities. □

Now in order to determine the existence of a common factor of

$$\begin{aligned} f(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0, \quad a_n \neq 0, \quad \text{and} \\ g(x) &= b_m x^m + b_{m-1}x^{m-1} + \cdots + b_0, \quad b_m \neq 0, \end{aligned}$$

we have to decide whether two polynomials A and B with the required properties can be found. This questions can be answered with the help of linear algebra: A and B are polynomials of degree at most $m-1$ and $n-1$ and that is why there are all in all $m+n$ unknown coefficients $c_{m-1}, \ldots, c_0, d_{n-1}, \ldots, d_0$ of A and B:

$$\begin{aligned} A(x) &= c_{m-1}x^{m-1} + \cdots + c_1 x + c_0 \\ B(x) &= d_{n-1}x^{n-1} + \cdots + d_1 x + d_0. \end{aligned}$$

The polynomial $A(x)f(x) + B(x)g(x)$ has degree at most $n+m-1$ in x. Each of its coefficients has to be zero in order to achieve $Af + Bg = 0$:

$$\begin{array}{ccccccc}
a_n c_{m-1} & + & b_m d_{n-1} & & = & 0 & \text{coefficient of } x^{m+n-1} \\
a_{n-1}c_{m-1} + a_n c_{m-2} & + & b_{m-1}d_{n-1} + b_m d_{n-2} & & = & 0 & \text{coefficient of } x^{m+n-2} \\
\ddots & & \ddots & & \vdots & & \\
 & a_0 c_0 & + & b_0 d_0 & = & 0 & \text{coefficient of } x^0.
\end{array}$$

So we get $n+m$ linear equations in the unknowns c_i, d_i with coefficients in a_i, b_i. Written in matrix style this system of linear equations has the form:

$$(c_{m-1}, \ldots, c_0, d_{n-1}, \ldots, d_0) \cdot \begin{pmatrix}
a_n & a_{n-1} & \ldots & a_0 & & & \\
 & a_n & a_{n-1} & \ldots & a_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & a_n & a_{n-1} & \ldots & a_0 \\
b_m & b_{m-1} & \ldots & b_0 & & & \\
 & b_m & b_{m-1} & \ldots & b_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & b_m & b_{m-1} & \ldots & b_0
\end{pmatrix} = (0, \ldots, 0)^T$$

where the empty places are filled with zeros. We know from linear algebra that this system of linear equations has a nonzero solution if and only if the determinant of the coefficient matrix is equal to zero. This leads to the following definition:

Definition 7.6: The $(m+n)\times(m+n)$ coefficient matrix with m rows of a–entries and n rows of b–entries

$$\textsc{Syl}(f,g) := \begin{pmatrix}
a_n & a_{n-1} & \ldots & a_0 & & & \\
 & a_n & a_{n-1} & \ldots & a_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & a_n & a_{n-1} & \ldots & a_0 \\
b_m & b_{m-1} & \ldots & b_0 & & & \\
 & b_m & b_{m-1} & \ldots & b_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & b_m & b_{m-1} & \ldots & b_0
\end{pmatrix}$$

where the empty spaces are filled by zeros is called the *Sylvester matrix* of f and g. The determinant of the matrix is called the *resultant* of f and g: $\operatorname{res}(f,g) := \det(\textsc{Syl}(f,g))$.

From the above observations we immediately obtain a criterion for testing whether two polynomials f and g have a common factor.

Proposition 7.7: Given $f, g \in \mathbb{Q}[x]$ of positive degree, the resultant $\operatorname{res}(f,g) \in \mathbb{Q}$ is equal to zero if and only if f and g have a nonconstant common factor. If

we work over the complex numbers then $\text{res}(f,g) = 0$ if and only if f and g have a common complex root.

7.5.2 Properties of resultants

There are some useful properties of resultants:

Lemma 7.8: Let $f, g \in \mathbb{Q}[x]$ and $\alpha \in \mathbb{C}$.

1. For $\deg(f) > 0$ and $\deg(g) > 0$ it is $\text{res}(\alpha \cdot f, g) = \alpha^m \cdot \text{res}(f,g)$.
2. If $\deg(g) > 0$ it holds $\text{res}((x-\alpha)\cdot f, g) \;=\; g(\alpha)\cdot \text{res}(f,g)$.

Proof. 1. Let $\deg(f) = n$ and $\deg(g) = m$. Then

$$\alpha \cdot f(x) = \alpha a_n x^n + \alpha a_{n-1} x^{n-1} + \cdots + \alpha a_0.$$

It is clear from the definition of the resultant of f and g that one can extract from each of the first m rows of the Sylvester matrix a factor α leading to the stated result.

2. Again let $f(x) = \sum_{i=0}^{n} a_i x^i$ and $g(x) = \sum_{i=0}^{m} b_i x^i$. Then the coefficients of $(x-\alpha)f$ in order of decreasing powers of x are

$$(a_n \;,\; a_{n-1} - \alpha a_n \;,\; a_{n-2} - \alpha a_{n-1} \;,\; \ldots \;,\; a_0 - \alpha a_1 \;,\; -\alpha a_0).$$

We do some transformations with the Sylvester matrix of $(x-\alpha)f$ and g. First we add α times column i to column $(i+1)$ for all $i = 1, \ldots, m+n$. This puts the rows with the f–entries into the right form:

$$\begin{array}{ccccccccc}
a_n & a_{n-1} & \ldots & a_0 & 0 & & & & \\
 & a_n & a_{n-1} & \ldots & a_0 & 0 & & & \\
 & & \ddots & \ddots & & \ddots & & & \cdot \\
 & & & a_n & a_{n-1} & \ldots & a_0 & 0 &
\end{array}$$

Each non zero entry of the last $(n+1)$ rows is now equal to $g(\alpha)/\alpha^i$ for a suitable i. By $g(\alpha)/\alpha^i$ we mean the substitution of α for x into the integral part of $g(x)$ divided by x^i. For example the former entry b_i is replaced by $g(\alpha)/\alpha^i = b_m\alpha^{m-i} + \cdots + b_{i+1}\alpha + b_i$ for all $1 \le i \le m$. To simplify the matrix we subtract α times row j from row $j-1$ for $j = m+2, \ldots, n+m+1$. This leads to the following matrix which has the same determinant as the Sylvester matrix of $(x-\alpha)f$ and g:

$$\begin{pmatrix}
a_n & a_{n-1} & \dots & a_0 & 0 & & & & \\
 & a_n & a_{n-1} & \dots & a_0 & 0 & & & \\
 & & \ddots & \ddots & & \ddots & & & \\
 & & & a_n & a_{n-1} & \dots & a_0 & 0 & \\
b_m & b_{m-1} & \dots & b_0 & & & & & \\
 & b_m & b_{m-1} & \dots & b_0 & & & & \\
 & & \ddots & \ddots & & \ddots & & & \\
 & & & b_m & b_{m-1} & \dots & b_0 & 0 & \\
 & & & & g(\alpha)/\alpha^m & \dots & g(\alpha)/\alpha^1 & g(\alpha) &
\end{pmatrix}.$$

The last column has only one non zero entry $g(\alpha)$ and the cofactor of this entry is equal to the resultant of f and g. That proofs our claim.

□

The Lemma leads to the following important characterization of resultants:

Theorem 7.9: Let $f, g \in \mathbb{Q}[x]$, $a = \text{ldcf}(f)$, $b = \text{ldcf}(g)$, $\deg(f) = n > 0$, $\deg(g) = m > 0$ with roots

$$\alpha_1, \dots, \alpha_n, \beta_1 \dots, \beta_m \in \mathbb{C}.$$

For the resultant of f and g the following holds:

$$\text{res}(f, g) \;=\; a^m b^n \prod_{i=1}^{n} \prod_{j=1}^{m} (\alpha_i - \beta_j).$$

Proof. Writing $f(x) = a \cdot \prod_{i=0}^{n}(x - \alpha_i)$ and $g(x) = b \cdot \prod_{j=0}^{m}(x - \beta_j)$ we get from Lemma **??**

$$\begin{aligned}
\text{res}(f, g) &= a^m \text{res}\left(\prod_{i=1}^{n}(x - \alpha_i), g\right) \\
&= a^m g(\alpha_1) \text{res}\left(\prod_{i=2}^{n}(x - \alpha_i), g\right) \\
&= \dots \\
&= a^m g(\alpha_1) \dots g(\alpha_n) \\
&= a^m b^n \prod_{i=1}^{n} \prod_{j=1}^{m} (\alpha_i - \beta_j).
\end{aligned}$$

□

An immediate surprising consequence is that the resultant of two polynomials is invariant under translation.

Corollary 7.10: Let $f, g \in \mathbb{Q}[x]$ with $\deg(f) > 0$ and $\deg(g) > 0$. For a rational number $a \in \mathbb{Q}$ we define $\tilde{f}(x) := f(x + a)$ and $\tilde{g}(x) := g(x + a)$. Then it is $\mathrm{res}(f, g) = \mathrm{res}(\tilde{f}, \tilde{g})$.

7.5.3 Computation of resultants

The computation of resultants can be performed more efficiently than using the obvious determinant computation. This is based on the following idea: let $f(x) = q(x)g(x) + r(x)$ where $\deg(f) = n$, $\deg(g) = m$, $\deg(r) = l$, and $n \geq m > l$. By extending the definition of resultants to situations where f or g or both are constant, one can show that

$$\begin{aligned} \mathrm{res}(f, g) &= (-1)^{m(n-l)} b^{n-l} \mathrm{res}(r, g) \qquad (b = \mathrm{ldcf}(g)) \qquad (7.1) \\ &= (-1)^{nm} b^{n-r} \mathrm{res}(g, r). \qquad (7.2) \end{aligned}$$

Thus, the resultant of f and g can be expressed in the terms of the resultant of g and r. Since r is the remainder of f divided by g, we can apply a Euclidean–like algorithm: given f and g, we construct the Euclidean remainder sequence

$$f_0 = f \ , \ f_1 = g \ , \ \ldots \ , \ f_h$$

where $f_{i+1} = (f_{i-1} \mod f_i)$ and $f_{h+1} = 0$. If f_h is nonconstant, the $\mathrm{res}(f, g) = 0$. Otherwise, we can repeatedly apply the formulas **??** until the basic case given by $\mathrm{res}(f_{h-1}, f_h) = f_h^{\deg(f_{h-1})}$.

7.5.4 Resultants of multivariate polynomials

Suppose we are given $f, g \in \mathbb{Q}[x_1, \ldots, x_d]$ with positive degree in x_d. We write

$$\begin{aligned} f &= a_n x_d^n + \cdots + a_0 x_d^0, \quad a_n \neq 0 \\ g &= b_m x_d^m + \cdots + b_0 x_d^0, \quad b_n \neq 0. \end{aligned}$$

Thus we regard f and g as polynomials in x_d with coefficients a_i and b_j that are polynomials in $\mathbb{Q}[x_1, \ldots, x_{d-1}]$. We define the *resultant* of f and g with respect to x_d as in Definition **??**:

$$\mathrm{res}(f,g,x_d) \;=\; \det \begin{pmatrix} a_n & a_{n-1} & \dots & a_0 & & & \\ & a_n & a_{n-1} & \dots & a_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & a_n & a_{n-1} & \dots & a_0 \\ b_m & b_{m-1} & \dots & b_0 & & & \\ & b_m & b_{m-1} & \dots & b_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & b_m & b_{m-1} & \dots & b_0 \end{pmatrix}.$$

The resultant $\mathrm{res}(f,g,x_d)$ is a polynomial in $\mathbb{Q}[x_1,\dots,x_{d-1}]$. By $\deg(f)$ we denote the *total degree* of a multivariate polynomial f.

Proposition 7.11: Let $f,g \in \mathbb{Q}[x_1,\dots,x_d]$ with

$$\begin{aligned} f &= a_n x_d^n + \dots + a_0 x_d^0, \quad a_n \neq 0 \\ g &= b_m x_d^m + \dots + b_0 x_d^0, \quad b_n \neq 0. \end{aligned}$$

Further let $l = \max\{\deg(a_i) \mid i = 0 \dots n\}$ and let $k = \max\{\deg(b_i) \mid i = 0 \dots m\}$ Then $\deg(\mathrm{res}(f,g,x_d)) \leq nk + ml$.

Proof. The Sylvester matrix consists of n rows with coefficients of g that have maximal degree k and also of m rows with coefficients of f that have maximal degree l. Thus by definition of the determinant the total degree of $\deg(\mathrm{res}(f,g,x_d)) \leq nk + ml$. □

Also the total degree of $\mathrm{res}(f,g,x_d)$ can be bounded from above by the total degrees of f and g.

Proposition 7.12: Assume that f and g are polynomials with $\deg(f) = n$ and $\deg(g) = m$. Then $\mathrm{res}(f,g,x_d)$ either is the zero polynomial or a polynomial of total degree at most $n \cdot m$.

Proof. We first homogenize the polynomials f and g by introducing a new variable z. Then each monomial of the resulting homogeneous polynomials F and G has degree n or m, respectively. We will show that $\mathrm{res}(F,G,x_d) \in \mathbb{Q}[x_1,\dots,x_{d-1},z]$ either is zero or it is homogeneous of degree mn. From that we can conclude by substituting $z = 1$ that $\mathrm{res}(f,g,x_d) \neq 0$ is a polynomial of total degree at most $n \cdot m$.
It is easy to see that a nonzero polynomial $H(x_1,\dots,x_d)$ is homogeneous of degree n if and only if $H(tx_1,\dots,tx_d) \;=\; t^n H(x_1,\dots,x_d)$.
Now look at the resultant of the two homogeneous polynomials $F,G \in \mathbb{Q}[x_1,\dots,x_d,z]$:

$$\begin{aligned}
&\mathrm{res}(F,G,x_d)(tx_1,\ldots,tx_{d-1},tz)\\
&= \det\begin{pmatrix}
A_n & tA_{n-1} & \ldots & t^nA_0 & & & \\
 & A_n & tA_{n-1} & \ldots & t^nA_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & A_n & tA_{n-1} & \ldots & t^nA_0 \\
B_m & tB_{m-1} & \ldots & t^mB_0 & & & \\
 & B_m & tB_{m-1} & \ldots & t^mB_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & B_m & tB_{m-1} & \ldots & t^mB_0
\end{pmatrix}.
\end{aligned}$$

Each A_i is a homogeneous polynomial of degree $n-i$ and each B_j is homogeneous of degree $m-j$. Multiply the ith row of A's by t^i and the jth row of B's by t^j. We obtain

$$\begin{aligned}
&t^p \cdot \mathrm{res}(F,G,x_d)(tx_1,\ldots,tx_{d-1},tz)\\
&= \det\begin{pmatrix}
tA_n & t^2A_{n-1} & \ldots & t^{n+1}A_0 & & & \\
 & t^2A_n & t^3A_{n-1} & \ldots & t^{n+2}A_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & t^mA_n & t^{m+1}A_{n-1} & \ldots & t^{n+m}A_0 \\
tB_m & t^2B_{m-1} & \ldots & t^{m+1}B_0 & & & \\
 & t^2B_m & t^3B_{m-1} & \ldots & t^{m+2}B_0 & & \\
 & & \ddots & \ddots & & \ddots & \\
 & & & t^nB_m & t^{n+1}B_{m-1} & \ldots & t^{n+m}B_0
\end{pmatrix}\\
&= t^q \cdot \mathrm{res}(F,G,x_d)(x_1,\ldots,x_{d-1},z)
\end{aligned}$$

where $p = m(m+1)/2 + n(n+1)/2$, and $q = (m+n)(m+n+1)/2$. Hence

$$\mathrm{res}(F,G,x_d)(tx_1,\ldots,tx_{d-1},tz) \;=\; t^{nm}\mathrm{res}(F,G,x_d)(x_1,\ldots,x_{d-1},z).$$

□

Proposition 7.13: Given $f,g \in \mathbb{C}[x_1,\ldots,x_d]$ regarded as polynomials in x_d with coefficients in $\mathbb{C}[x_1,\ldots,x_{d-1}]$ and leading coefficients a and b not equal to zero. If $\mathrm{res}(f,g,x_d)$ vanishes at $(c_1,\ldots,c_{d-1}) \in \mathbb{C}^{d-1}$, then either

1. a or b vanish at $(c_1,\ldots,c_{d-1})$, or
2. there is $c_d \in \mathbb{C}$ such that f and g vanish at $(c1,\ldots,c_d) \in \mathbb{C}^d$.

Proof. It suffices to show that $f(c_1,\ldots,c_{d-1},x_d)$ and $g(c_1,\ldots,c_{d-1},x_d)$ have a common root when $a(c_1,\ldots,c_{d-1})$ and $b(c_1,\ldots,c_{d-1})$ are both nonzero. This can be easily shown with the help of Proposition **??**. □

Bibliography

[1] J.L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transaction on Computers C 28*, pages 643–647, 1979.

[2] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, and E. Schömer. A computational basis for conic arcs and boolean operations on conic polygons. *submitted to ESA 2002*, 2002.

[3] R. Bix. *Conics and Cubics: A Concrete Introduction to Algebraic Curves.* Springer Verlag, 1998.

[4] T. J. Bromwich. *Quadric forms and their classification by means of invariant factors.* Cambrige Tracs in Mathematics and Mathematical Physics, 1906.

[5] G. E. Collins and R. Loos. Polynomial real root isolation by differentiation. *SYMSAC*, pages 15–25, 1976.

[6] G. E. Collins and R. Loos. Real zeros of polynomials. *Computer Algebra*, pages 4:83–94, 1982.

[7] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. A new algorithm for the robust intersection of two general quadrics. *submitted to Solid Modeling and Applications*, 2002.

[8] A. Fabri, G. J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. On the design of CGAL, the computational geometry algorithms library. *Software – Practice and Experience*, pages 30:1167–1202, 2000.

[9] S. Fortune. An iterated eigenvalue algorithm for approximating the roots of univariate polynomials. *ISSAC*, pages 121–128, 2001.

[10] D. Halperin. Arrangements. *Handbook of Discrete and Computational Geometry*, pages 389–412, 1997.

[11] C. Hoffmann. *Geometric and Solid Modeling*. Moregan-Kaufmann AC, 1989.

[12] J. Keyser, T. Culver, M. Foskey, S. Krishnan, and D. Manocha. Esolid - a system for exact boundary evaluation. In *ACM*, 2002.

[13] J. Keyser, T. Culver, D. Manocha, and S. Krishnan. Mapc: A library for efficient and exact manipulation of algebraic points and curves. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1999.

[14] D. E. Knuth. *The Art of Computer Programming*, volume 2. Reading Mass: Addison Wesley, 1 edition, 1969.

[15] S. Krishnan, M. Foskey, T. Culver, J. Keyser, and D. Manocha. PRECISE: Efficient multiprecision evaluation of algebraic roots and predicates for reliable geometric computation. *Symposium on Computational Geometry*, pages 274–283, 2001.

[16] L. Kronecker. Uber den zahlbegriff. *Crelle J. reine und angew. Mathematik 101*, pages 337–395, 1887.

[17] J. Levin. A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. *Commun. ACM*, 19(10):555–563, 1976.

[18] Kurt Mehlhorn and Stefan Näher. *The LEDA Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[19] B.T. Smith. Error bounds for the zeroes of a polynomial based upon gershgorin's theorem. *JACM*, pages 17:661–674, 1970.

[20] M. Vincent. Sur la resolution des equations numereiques. *Jour. des Mathematique Pures et Appliques*, pages 1:341–372, 1836.

[21] K. Weierstrass. Zur theorie der bilinearen und quadratischen formen. *Berliner Monatsberichte*, page 310, 1968.

[22] R. Wein. High-level filtering for arrangements of conic arcs. *submitted to ESA 2002*, 2002.