Intersecting Quadrics: An Efficient and Exact Implementation

Sylvain Lazard
LORIA-INRIA Lorraine
Campus scientifique, B.P. 239
54506 Vandœuvre-lès-Nancy
cedex, France
Sylvain.Lazard@loria.fr

Luis Mariano Peñaranda*
Fac. de Ciencias Exactas,
Ingeniería y Agrimensura
Univers. Nacional de Rosario
Pellegrini 250
2000 Rosario, Argentina
luis@fceia.unr.edu.ar

Sylvain Petitjean
LORIA-CNRS
Campus scientifique, B.P. 239
54506 Vandœuvre-lès-Nancy
cedex, France
Sylvain.Petitjean@loria.fr

ABSTRACT

We present the first complete, exact and efficient C++ implementation of a method for parameterizing the intersection of two implicit quadrics with integer coefficients of arbitrary size. It is based on the near-optimal algorithm recently introduced by Dupont et al. [2].

Unlike existing implementations, it correctly identifies and parameterizes all the connected components of the intersection in all cases, returning parameterizations with rational functions whenever such parameterizations exist. In addition, the coefficient fields of the parameterizations are either minimal or involve one possibly unneeded square root.

We prove upper bounds on the size of the coefficients of the output parameterization and compare these bounds to observed values. We give other experimental results and present some examples.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms

Algorithms, Experimentation.

Keywords

Intersection of quadrics, C++ implementation, experimental analysis.

1. INTRODUCTION

Computing an explicit representation of the intersection of two general quadrics (i.e., quadratic surfaces) is a fundamental problem in areas such as solid modeling, computational geometry and computer graphics. The range of applications covers well-known problems like computing arrangements [10, 13], boundary evaluation [12] and convex hull computation [5].

Past work. Until recently, the only known general method for computing a parametric representation of the intersection between two arbitrary quadrics was that of J. Levin [7]. This method is based on an analysis of the pencil generated by the two quadrics, i.e. the set of linear combinations of the two quadrics.

Though useful for curve tracing, Levin's method has serious limitations. When the intersection is singular or reducible, a parameterization by rational functions is known to exist, but Levin's pencil method fails to find it and generates a parameterization that involves the square root of some polynomial. In addition, since it introduces algebraic numbers of very high degree (for instance in the computation of eigenvalues and eigenvectors), a correct implementation using exact arithmetic is essentially out of reach. In addition, when a floating point representation of numbers is used, the method may output results that are wrong (geometrically and topologically) and it may even fail to produce any parameterization at all and crash.

Over the years, Levin's seminal work has been extended and refined in several different directions. Wilf and Manor [19] use a classification of quadric intersections by the Segre characteristic (see [1]) to drive the parameterization of the intersection by the pencil method. Recently, Wang, Goldman and Tu [17] further improved the method making it capable of computing structural information on the intersection and its various connected components and able to produce a parameterization by rational functions when such a parameterization exists. Whether the refined algorithm is numerically robust is open to question.

Another method of algebraic flavor was introduced by Farouki, Neff and O'Connor [3] for parameterizing the intersection in degenerate situations. In such cases, using a combination of classical concepts (Segre characteristic) and algebraic tools (factorization of multivariate polynomials), the authors show that explicit information on the morphological type of the intersection curve can be reliably obtained. A notable feature of this method is that it can output an exact parameterization of the intersection in simple cases, when the input quadrics have rational coefficients. No im-

*Work done while this author was visiting LORIA (supported by the International Relations Delegation of INRIA).

SCG'04, June 8–11, 2004, Brooklyn, New York, USA.

plementation is reported however.

Rather than restricting the type of the intersection, others have sought to restrict the type of the input quadrics, taking advantage of the fact that geometric insights can then help compute the intersection curve [9, 14]. Specialized routines are devised to compute the intersection curve in each particular case. Even though such geometric approaches are numerically more stable than the algebraic ones, they are essentially limited to the class of so-called natural quadrics, i.e., the planes, right cones, circular cylinders and spheres.

Apart from [2], perhaps the most interesting of the known algorithms for computing an explicit representation of the intersection of two arbitrary quadrics is the method of Wang, Joe and Goldman [18]. This algebraic method is based on a birational mapping between the intersection curve and a plane cubic curve. The cubic curve is obtained by projection from a point lying on the intersection. Then the classification and parameterization of the intersection are obtained by invoking classical results on plane cubics. The authors claim that their algorithm is the first to produce a complete topological classification of the intersection (singularities, number and types of connected components, ...). Numerical robustness issues have however not been studied and the intersection may not be correctly classified. Also, the center of projection is currently computed using Levin's (enhanced) method: with floating point arithmetic, the center of projection will in general not exactly lie on the curve, which is another source of numerical instability.

Contributions. In this paper, we present the first exact, robust, efficient and usable implementation of an algorithm for parameterizing the intersection of two arbitrary quadrics, given in implicit form, with integer coefficients. (Note that quadrics with rational or finite floating-point coefficients can be trivially converted to integer form.) This implementation is based on the parameterization method described in [2].

Precisely, our implementation has the following features:

- it computes an exact parameterization of the intersection of two quadrics with integer coefficients of arbitrary size;
- it correctly identifies, separates and parameterizes all the connected components of the intersection and gives all the relevant topological information;
- it places no restriction of any kind on the type of the intersection or the type of the input quadrics;
- the parameterization is rational when one exists; otherwise the intersection is a smooth quartic and the parameterization involves the square root of a polynomial;
- the parameterization is either optimal in the degree of the extension of $\mathbb{Z}$ on which its coefficients are defined or, in a small number of well-identified cases, involves one extra possibly unnecessary square root;
- the implementation is carefully designed so that the size of the coefficients is kept small;
- it is fast and efficient and can routinely compute parameterizations of the intersection of quadrics with input coefficients having ten digits in less than 50 milliseconds on a mainstream PC.

Our C++ implementation can be queried via a web interface at `http://www.loria.fr/isa/qi`. The code will be released to educational and research audiences shortly.

The paper is organized as follows. In Section 2, we recall the main ideas of the parameterization algorithm we introduced in [2] and describe its implementation. In Section 3, we focus on two cases covering the two main parameterization philosophies used in our implementation, prove theoretical bounds on the size of the output coefficients and compare those bounds to observed values. We give more experimental results and performance evaluation in Section 4. Finally, we show the output produced by our implementation for some examples in Section 5, before concluding.

2. ALGORITHM AND IMPLEMENTATION

In this section, we give a brief presentation of the basic ideas underpinning our near-optimal parameterization method [2]. We then move on to a description of the main design choices we made to implement it.

2.1 Preliminaries

In what follows, all the matrices considered are 4×4 real matrices, unless otherwise specified. We call *quadric* associated with a symmetric matrix S the set

$$Q_S = \{\mathbf{x} \in \mathbb{P}^3 \mid \mathbf{x}^T S \mathbf{x} = 0\},$$

where $\mathbb{P}^3 = \mathbb{P}^3(\mathbb{R})$ denotes the real projective space of dimension 3 ($\mathbf{x}^T S \mathbf{x}$ is quadratic and homogeneous in the coordinates of $\mathbf{x}$). In the rest of this paper, points and parameterizations are assumed to live in projective space. Recall that a point of $\mathbb{P}^3$ has four coordinates.

We define the *inertia* of S and Q_S as the pair

$$\sigma_S = (\max(\sigma^+, \sigma^-), \min(\sigma^+, \sigma^-)),$$

where σ^+ (resp. σ^-) is the number of positive (resp. negative) eigenvalues of S. The *rank* of S is the sum $\sigma^+ + \sigma^-$. Recall that Sylvester's Inertia Law asserts that the inertia of S (and thus the rank) is invariant by a real projective transformation [6].

We call *projective cones* (or simply *cones*) the quadrics of rank 3. For the benefit of the reader, we recall that, in affine real space, quadrics of inertia $(4, 0)$ are empty, quadrics of inertia $(3, 1)$ are ellipsoids, hyperboloids of two sheets or elliptic paraboloids, and quadrics of inertia $(2, 2)$ are hyperboloids of one sheet or hyperbolic paraboloids (see [2] for a complete characterization of affine quadrics). Also, quadrics of inertia $(2, 1)$ are cones or cylinders. All the quadric *surfaces* except those of inertia $(3, 1)$ are ruled surfaces.

Given two matrices S and T, let $R(\lambda, \mu) = \lambda S + \mu T$. The set $\{R(\lambda, \mu) \mid (\lambda, \mu) \in \mathbb{P}^1\}$ is called the *pencil* of matrices generated by S and T. For the sake of simplicity, we sometimes write a member of the pencil $R(\lambda) = \lambda S - T, \lambda \in \mathbb{R} \cup \{\infty\}$. Associated to a pencil of matrices is a pencil of quadrics $\{Q_{R(\lambda,\mu)} \mid (\lambda, \mu) \in \mathbb{P}^1\}$. Recall the classical result that the intersection of two distinct quadrics of a pencil is independent of the choice of the two quadrics.

The binary quartic form $\det R(\lambda, \mu)$ is called the *determinantal equation* of the pencil. The quadrics of the pencil of rank less than or equal to 3 are exactly the quadrics $Q_{R(\lambda,\mu)}$ such that $\det R(\lambda, \mu) = 0$.

2.2 Near-optimal parameterization algorithm

2.2.1 *Main ideas*

Let $\{Q_{R(\lambda,\mu)} \mid (\lambda, \mu) \in \mathbb{P}^1\}$, with $R(\lambda, \mu) = \lambda S + \mu T$, be a pencil of quadrics. The main idea of existing methods for parameterizing the intersection of two quadrics based on

an analysis of their pencil (Levin's and derivatives) is as follows: find a quadric Q_R of some particularly simple form in the pencil generated by Q_S and Q_T (assume $Q_R \neq Q_S$), parameterize this quadric, plug the parameterization $\mathbf{X}$ in the equation of Q_S, solve the resulting equation $\mathbf{X}^T S\mathbf{X} = 0$, and plug the result in $\mathbf{X}$, finally giving the parameterization of the intersection.

The key to making this procedure work in practice is to find a quadric Q_R that is ruled and thus admits a parameterization that is linear in one of its parameters so that the equation $\mathbf{X}^T S\mathbf{X} = 0$ has degree 2. Levin's main result was to prove that a pencil of quadrics always contains at least one *"simple" ruled quadric* [7]. However, since such quadrics are found by first finding the zeros of the determinant of the upper left 3×3 submatrix of $R(\lambda, \mu)$, a cubic equation, and since cubic equations have generically no rational root (by Hilbert's Irreducibility Theorem), Levin's algorithm introduces non-rational numbers at an early stage and, in practice, floating-point arithmetic has to be used, resulting in numerical robustness problems.

The principal contribution of [2] was to show that, by a careful choice of the intermediate quadric Q_R, the appearance of algebraic numbers can be kept to a minimum. One major result is encapsulated in Theorem 3 of [2]: except when the intersection is reduced to two real points, the pencil contains at least one ruled quadric whose coefficients are rational and such a quadric can be easily computed. In addition, thanks to new worst-case optimal (in the number of square roots) parameterizations of ruled projective quadrics, we can always find such a rational ruled quadric Q_R with a parameterization involving only one square root.

Some of the basic ingredients used in our algorithm or to infer information about the intersection are the Segre classification of pencils and its refinement over the reals (the Canonical Form Theorem for pairs of real symmetric matrices – see [16]), a projective setting, ad hoc projective transformations to compute the canonical form of a projective quadric, and Sylvester's Inertia Law [6].

The basic principles underlying the design of our implementation are as follows:

- compute structural information on the intersection and its various real components as early as possible;
- use the structural information gathered to drive the parameterization process and make the right choices so that the output is optimal or near-optimal from the point of view of the degree of the extension of $\mathbb{Z}$ on which its coefficients are defined.

In our implementation we were interested not just in optimizing the number of square roots in the output but also in minimizing the size of the output coefficients. For this reason, the basic philosophy is to use as intermediate ruled quadric Q_R a quadric with rational coefficients of the smallest rank that we can easily find, the rationale being, for instance, that the parameterization of a cone involves coefficients of smaller asymptotic size than the coefficients of the parameterization of a quadric of inertia $(2, 2)$. There are essentially two cases: (i) Q_R has rank 4; (ii) Q_R has rank 3 or less.

2.2.2 Q_R has rank 4

The main case where Q_R has rank 4 is when the intersection is a smooth quartic (we leave the other cases aside due to lack of space). In this situation, the quartic determinantal equation $\det R(\lambda)$ has no multiple root. It could well be that at least one of its simple roots is rational and that a Q_R with rank less than 4 could have been used, but checking this via the Rational Root Theorem can be very time consuming[1]. Since generically a degree-four equation has no rational root, we prefer instead to isolate the real zeros of the determinantal equation using an implementation of Uspensky's algorithm [11]. We then take (at most two) rational test points λ_i outside the isolating intervals in the areas where $\det R(\lambda) > 0$. If one of the quadrics $R(\lambda_i)$ has inertia $(4, 0)$, the intersection is empty (it is a complex smooth quartic), a consequence of Finsler's Theorem (see [2]). Otherwise, we proceed.

We now have a quadric $R_0 = R(\lambda_0)$ of inertia $(2, 2)$ and a range of values $I = [a, b]$ such that $\lambda_0 \in I$ and $\det R(\lambda) > 0$ for all $\lambda \in I$. In the worst case, the parameterization of Q_R involves two square roots [2]. We can improve this situation as follows. First, compute a point $\mathbf{p}_0$ on Q_{R_0}. Approximate this point by a point $\mathbf{p}$ with integer coordinates (recall that $\mathbf{p}$ is a projective point). Find the quadric $Q_R = Q_{R(\lambda_1)}$ through $\mathbf{p}$. If $\mathbf{p}$ is close enough to $\mathbf{p}_0$, then $\lambda_1 \in I$ and $\det R > 0$. We thus have a quadric of inertia $(2, 2)$ containing a point in $\mathbb{P}^3(\mathbb{Z})$: such a quadric can be parameterized with at most one square root [2].

Plugging the parameterization $\mathbf{X}((u, v), (s, t))$ of Q_R, with $(u, v), (s, t) \in \mathbb{P}^1$, in the equation of any other quadric of the pencil gives a bihomogeneous equation that has degree two in (u, v) and two in (s, t). Solving this equation for (s, t) in terms of (u, v) and replugging in the parameterization of Q_R gives a parameterization of the smooth quartic:

$$\mathbf{X}(u, v) = \mathbf{X}_1(u, v) \pm \mathbf{X}_2(u, v)\sqrt{\Delta(u, v)},$$

where $\mathbf{X}_1(u, v)$ (resp. $\mathbf{X}_2(u, v)$) is a vector of homogeneous polynomials of degree 3 (resp. 1) and $\Delta(u, v)$ is a homogeneous polynomial of degree 4.

If $\delta = \det R$ is a square, then all of these polynomials have rational coefficients and the parameterization is *optimal* in terms of the degree of the extension of $\mathbb{Z}$ on which it is defined. If δ is not a square, then we can only conclude that the parameterization is *near-optimal*: it might well be that there exists another quadric $Q_{R'}$ of inertia $(2, 2)$ in the pencil such that $\det R'$ is a square, implying that $\sqrt{\delta}$ could have been avoided in the output (see Section 5.2 for an example). Finding such a quadric however implies, in general, finding a rational point on a hyperelliptic curve (see [2]), a problem known to be very hard.

2.2.3 Q_R has rank < 4

Though not generic, the situation where Q_R has rank < 4 happens quite often in practice since it covers in particular all the types of intersection corresponding, in the Segre characterization, to the determinantal equation having a single multiple root λ_0. Indeed, in that case, the multiple root is both real (otherwise its complex conjugate would also be a multiple root of $\det R(\lambda)$) and rational (otherwise its algebraic conjugate would also be a multiple root of $\det R(\lambda)$). So the associated quadric $Q_R = Q_{R(\lambda_0)}$ has rational coefficients and has rank 3 or less.

[1] If however one of the initial quadrics has rank 3, then it should be used to parameterize the intersection. Doing so results in a parameterization having the same algebraic complexity in the worst case, but of smaller coefficient size.

The general philosophy for parameterizing the intersection is to parameterize Q_R, plug the parameterization in any other quadric of the pencil, and solve the resulting equation in the parameters. There are however many situations in which this procedure can be simplified by the fact that we can find a rational point on Q_R outside its singular locus and thus parameterize Q_R rationally, and that we know enough information on the intersection to greatly simplify the solving and factorization of the equation in the parameters.

Let us illustrate this on the example of an intersection consisting of a cubic and a line that are tangent. The determinantal equation in this case has a quadruple root corresponding to a cone Q_R of inertia $(2, 1)$. By the above argument, Q_R has rational coefficients. So the vertex $\mathbf{c}$ of Q_R has rational coordinates. $\mathbf{c}$ is the point of tangency of the cubic and the line of the intersection. Assume $Q_R \neq Q_S$. The line of the intersection is necessarily rational (otherwise its conjugate would be in the intersection). This line can be found by intersecting the cone Q_R with the plane tangent to Q_S at $\mathbf{c}$. Picking any point with rational coordinates on this line other than $\mathbf{c}$ gives a non-singular rational point on the cone. A projective cone having a rational point other than its singular locus can be rationally parameterized. Plugging this parameterization in Q_S gives an equation in the parameters of the cone which factors into two terms of total degree 1 and 3. Each factor can then be solved rationally for one parameter in terms of the other. The linear factor yields the line of the intersection and the cubic factor yields the cubic.

2.3 Implementation

Our implementation builds upon the *LiDIA* [8] and *GMP* [4] C/C++ libraries. *LiDIA* was originally developed for computational number theory purposes, but includes many types of simple parameterized and template classes that are useful for our application. Apart from simple linear algebra routines and algebraic operations on univariate polynomials, we use *LiDIA*'s number theory package and its ability to manipulate vectors of polynomials, polynomials having other polynomials as coefficients, etc. On top of it, we have added our own data structures. We have compiled *LiDIA* so that it uses *GMP* multiprecision integer arithmetic. From now on, we refer to the multiprecision integers as `bigint`s, following the terminology of *LiDIA*.

Our implementation consists of more than 17,000 lines of source code, which is essentially divided into the following chapters:

- *data structures* (1,500 lines): structures for intersections of quadrics, for components of the intersection, for homogeneous polynomials with `bigint` coefficients (coordinates of components), for homogeneous polynomials with `bigint` polynomials as coefficients, and basic operations on these structures...
- *elementary operations* (2,000 lines): computing the inertia of a quadric of `bigint`s, the coefficients of the determinantal equation, the gcd of the derivatives of the determinantal equation, the adjoint of a matrix, the singular space of a quadric, the intersection between two linear spaces, applying Descartes's Sign Rule, the Gauss decomposition of a quadratic form into a sum of squares, isolating the roots of a univariate polynomial using Uspensky's method, ...
- *number theory and optimizations* (1,500 lines): gcd optimizations of the `bigint` coefficients of a polynomial, a vector or a matrix, optimizations of the coefficients of pairs and triples of vectors, reparameterization of lines so that its representative points have small height, ...
- *quadric parameterizations* (2,000 lines): parameterization of a quadric of inertia $(2, 2)$ with `bigint` coefficients going through a rational point, of a cone (resp. conic), of a cone (resp. conic) with a rational point, of a pair of planes, ...
- *intersection parameterizations* (9,000 lines): dedicated procedures for parameterizing the components of the intersection in all possible cases, i.e. when the determinantal equation has no multiple root (1,500 lines), one multiple root (3,000 lines), two multiple roots (1,500 lines) or when it vanishes identically (3,000 lines).
- *printing and debugging* (1,000 lines): turning on debugging information with the `DEBUG` preprocessor directive, checking whether the computed parameterizations are correct, pretty printing the parameterizations, ...

3. HEIGHT OF OUTPUT COEFFICIENTS

In this section, we prove theoretical bounds on the height of the coefficients of the parameterizations computed by our intersection software. We do this for the two types of intersection already outlined in the previous section: smooth quartic and cubic and tangent line.

The smooth quartic case is important because it is the generic intersection situation (given two random quadrics, the intersection is a smooth quartic with probability 1) and because it is also the worst case from the point of view of the height of the coefficients involved. The cubic and tangent line case is used to validate a key design choice we made, which is to take the quadric with rational coefficients of lowest possible rank to parameterize the intersection. We compare what happens for this type of intersection when we use for Q_R a quadric of inertia $(2, 2)$ or a quadric of inertia $(2, 1)$.

In what follows, the *size* of an integer e is $\log_{10}|e| + 1$ if e is not zero and 1 otherwise. The *size* of an algebraic number $e_1 + \sqrt{\delta}\, e_2$, where e_1, e_2, δ are integers, is the maximum of the sizes of e_1, e_2 and δ. The *size* of a matrix, a polynomial or a parameterization is the maximum of the sizes of its coefficients. The *height* of a quantity E (a point, polynomial, matrix or parameterization) in terms of the coefficients of E' is the ratio of the size of E and the size of E'. When E depends on the coefficients of the input matrices S and T, we sometimes drop the reference to S and T and simply talk about the *height* of E.

In the following subsections, we give asymptotic upper bounds on the heights of parameterizations. For the sake of simplicity, we refer to these bounds as the heights of the parameterizations.

3.1 Smooth quartic

We consider here the case where the intersection is a smooth quartic. Let Q_R be the quadric of inertia $(2, 2)$ used to parameterize the intersection and $\mathbf{p}$ a point of $\mathbb{P}^3(\mathbb{Z})$ on Q_R, as described in Section 2.2.2. We assume that the size of $\mathbf{p}$ is constant. This assumption is very realistic in practice; experimentally, we have observed that the coordinates of $\mathbf{p}$ are integers between -2 and 2 most of the time. We prove the following result.

PROPOSITION 3.1. *The parameterization of a smooth quartic*

$$\mathbf{X}(u, v) = \mathbf{X}_1(u, v) \pm \mathbf{X}_2(u, v)\sqrt{\Delta(u, v)}$$

is such that the height in terms of S and T is asymptotically at most 27 for the polynomials of $\mathbf{X}_1$, 8 for the polynomials of $\mathbf{X}_2$ and 38 for $\Delta(u,v)$.

PROOF. We first show how the parameterization of Q_R is computed and bound the height of its coefficients.

Let M_0 be the projective transformation sending the point $\mathbf{p}_0 = (1,0,0,0)^T$ to the point $\mathbf{p}$. Let R_0 denote the quadric obtained from R through the projective transformation M_0: $R_0 = M_0^T\, R\, M_0$. It follows from Sylvester's Inertia Law [6] that R_0 has the same inertia as R, i.e. $(2,2)$. Moreover, the point $\mathbf{p}_0$ belongs to Q_{R_0} since $M_0\,\mathbf{p}_0 = \mathbf{p}$.

Note that the size of M_0 is the size of $\mathbf{p}$, i.e. 1, and thus the height of R_0 in terms of S and T is the same as the height of R. For clarity, we rename R_0 and $\mathbf{p}_0$ by R and $\mathbf{p}$, respectively.

Let $\mathbf{x}$ denote the vector $(x,y,z,w)^T$. Let L be $1/2$ times the differential of quadric Q_R at $\mathbf{p}$ (one can trivially show that the matrix of L is the first row of R) and let i be such that $R[1,i] \neq 0$ (such an i necessarily exists). We compute the polynomial division of $Q_R = \mathbf{x}^T R\mathbf{x}$ by $L\mathbf{x}$ with respect to the variable $\mathbf{x}[i]$. The result of the division is

$$R[1,i]^2\,(\mathbf{x}^T R\mathbf{x}) = (L\mathbf{x})\,(L'\mathbf{x}) + A, \tag{1}$$

where $L'[\xi] = -R[i,i]\,R[1,\xi] + 2\,R[1,i]\,R[i,\xi]$ for $\xi = 1,\ldots,4$ and

$$A = c[j]\,\mathbf{x}[j]^2 + c[k]\,\mathbf{x}[k]^2 + 2\,c_{jk}\,\mathbf{x}[j]\,\mathbf{x}[k]$$

where j and k are equal to the two values in $\{2,3,4\}$ distinct from i, and

$$\begin{aligned} c[\xi] &= R[\xi,\xi]\,R[i,1]^2 + R[i,i]\,R[\xi,1]^2 \\ &\qquad - 2\,R[\xi,1]\,R[i,1]\,R[i,\xi], \quad \xi \in \{j,k\}, \\ c_{jk} &= R[j,k]\,R[i,1]^2 + R[j,1]\,R[k,1]\,R[i,i] \\ &\qquad - (R[j,1]\,R[k,i] + R[k,1]\,R[j,i])\,R[i,1]. \end{aligned}$$

We assume in the following that $c[j] \neq 0$ (if $c[j] = 0$ but $c[k] \neq 0$, we exchange the roles of j and k; otherwise the analysis is different but similar and we omit it here). For clarity we denote in the following $c = c[j]$ and $r = R[1,i]$.

We consider the projective transformation M such that, in the new projective frame, the quadric Q_R has equation (up to a factor)

$$\mathbf{x}'^T M^T R M \mathbf{x}' = 4x'y' + z'^2 - cw'^2.$$

In accordance with Equation (1) we choose $x' = L\mathbf{x}$, $y' = L'\mathbf{x}$. We apply Gauss' decomposition of quadratic forms into sum of squares to A and set $z' = c[j]\,\mathbf{x}[j] + c_{jk}\,\mathbf{x}[k]$ and $w' = \mathbf{x}[k]$. Precisely, we define M such that its adjoint has its first row equal to L, its second row equal to L', and the last two rows equal to zero except for the entry $[3,j]$ equal to $c[j]$, the entry $[3,k]$ equal to c_{jk}, and the entry $[4,k]$ equal to 1.

Straightforward computations show that the four columns of M can be simplified by the factors rc, r, $2r$, and $2r^2$, respectively. We then get

$$\mathbf{x}^T M^T R M \mathbf{x} = r^2 c\,(4xy + z^2 - \det(R)\,w^2). \tag{2}$$

If i,j,k are equal to $2,3,4$ respectively, M is equal to

$$M = \begin{pmatrix} R[2,2] & -c & R[2,2]\,R[1,3] - r\,R[2,3] & M[1,4] \\ -2\,r & 0 & -r\,R[1,3] & M[2,4] \\ 0 & 0 & r^2 & M[3,4] \\ 0 & 0 & 0 & r\,c \end{pmatrix},$$

$$\begin{aligned} M[1,4] &= r\,(R[1,4]\,(R[2,2]\,R[3,3] - R[2,3]^2) \\ &\qquad + R[3,4](r\,R[2,3] - R[2,2]\,R[1,3]) \\ &\qquad\qquad + R[2,4](R[1,3]\,R[2,3] - r\,R[3,3])), \\ M[2,4] &= r\,(R[1,4]\,(R[1,3]\,R[2,3] - r\,R[3,3]) \\ &\qquad + R[1,3]\,(r\,R[3,4] - R[1,3]\,R[2,4])), \\ M[3,4] &= r\,(-r^2\,R[3,4] - R[2,2]\,R[1,3]R[1,4] \\ &\qquad + r\,(R[1,3]\,R[2,4] + R[1,4]\,R[2,3])). \end{aligned}$$

We can easily parameterize the quadric of Equation (2) and the parameterization of the original Q_R is, with $\delta = \det(R)$ and (u,v) and (s,t) in $\mathbb{P}^1(\mathbb{R})$,

$$M_0 M \left(ut\sqrt{\delta},\quad sv\sqrt{\delta},\quad (us - tv)\sqrt{\delta},\quad us + tv\right)^T. \tag{3}$$

To evaluate the height of this parameterization in terms of S and T, first note that the matrix R is the matrix $\lambda S + \mu T$ of the pencil such that $(\lambda,\mu) \in \mathbb{P}^1$ is solution of

$$\mathbf{p}^T(\lambda S + \mu T)\mathbf{p} = 0.$$

So $(\lambda,\mu) = (-\mathbf{p}^T T\mathbf{p}, \mathbf{p}^T S\mathbf{p})$ has height 1 and $R = \lambda S + \mu T$ has height 2 in terms of S and T, since the size of $\mathbf{p}$ is 1. Also, $\delta = \det R$ has height 4 in terms of R and thus height 8 in terms of S and T. When δ is a square, $\sqrt{\delta}$ has height 4 in terms of S and T.

The heights of the four columns of M are 1, 3, 2 and 4 respectively in terms of R, and so 2, 6, 4 and 8 respectively in terms of S and T. The worst case for the height of the coefficients of the parameterization of Q_R happens when $\sqrt{\delta}$ is a square, which we assume for the rest of the proof. It then follows from (3) that the coordinates of the parameterization of Q_R are polynomials of the form

$$\rho_1\,ut + \rho_2\,sv + \rho_3\,us + \rho_4\,tv \tag{4}$$

where the heights of $\rho_1,\ldots,\rho_4$ are 6, 10, 8 and 8, respectively, in terms of S and T. Let $h_s = 10$ and $h_t = 8$ denote the maximum heights of the coefficients of s and t in (4), respectively.

When substituting the parameterization of Q_R into the equation of one of the initial quadrics (say S), we obtain an equation which can be written as

$$a\,s^2 + b\,st + c\,t^2 = 0, \tag{5}$$

where a, b and c depend on (u,v). It follows from (4) that the heights of a,b,c are $h_a = 21$ $(= 2h_s + 1)$, $h_b = 19$ $(= h_s + h_t + 1)$ and $h_c = 17$ $(= 2h_t + 1)$, respectively. When substituting the solution $(s = 2c,\ t = -b \pm \sqrt{b^2 - 4ac})$ into the parameterization (3) we obtain a parameterization of the smooth quartic in which each coordinate has the form

$$\alpha(u,v) \pm \beta(u,v)\sqrt{b^2 - 4ac}.$$

The height of the coefficients of α is 27 $(= h_s + h_c = h_t + h_b)$, the height of the coefficients of β is 8 $(= h_t)$ and the height of the coefficients of Δ is 38 $(= 2h_b = h_a + h_c)$. □

Figure 1 shows how the observed height of the coefficients of $\Delta(u,v)$ evolves as a function of the input size s for three different versions of our implementation (see Section 4 for the details). For each value of s in a set of samples between 0 and 60, we have generated random quadrics with coefficients in the range $[-10^s, 10^s]$, computed the height of the coefficients of the parameterization of the smooth quartic and

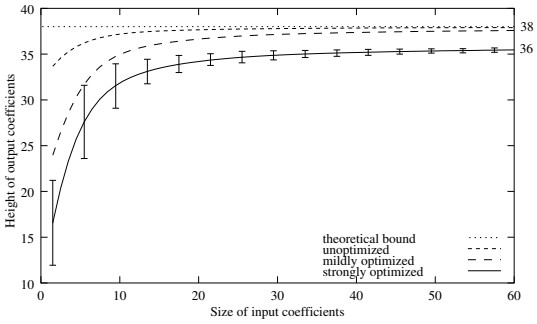


Figure 1: Evolution of the height of $\Delta(u, v)$ (smooth quartic case) as a function of the size of the input, with the standard deviation displayed on the optimized plot.

averaged the results. The plots of Figure 1 show that the observed height of the coefficients converges to 38 (i.e., the theoretical height) when no gcd computation is performed for simplifying the output parameterization, and converges to 36 when gcd computation is performed. We ran experiments with inputs of size up to 10,000 and observed the same limit of 36 on the height of the coefficients when gcd computation is performed. We do not have any satisfactory explanation as of why the theoretical height is not reached in that case.

3.2 Cubic and tangent line

We now consider the case of an intersection consisting of a cubic and a tangent line. In this case, we can parameterize the intersection using an intermediate rational quadric Q_R of inertia either $(2, 2)$ or $(2, 1)$: the pencil contains an instance of both types of quadrics.

We prove the following theoretical bounds on the height of the coefficients of the parameterizations of the cubic and the line.

PROPOSITION 3.2. *When a quadric Q_R of inertia $(2, 2)$ is used to parameterize the intersection, the parameterizations of the cubic and the line have asymptotic height at most 27 in terms of S and T.*

PROOF. In this situation, the determinant of the intermediate quadric Q_R is necessarily a square because Q_R contains a rational line (see [2]). So the bounds found in the proof of Proposition 3.1 apply. In particular the heights h_s, h_t, h_a, h_b, and h_c of the coefficients of Equations (4) and (5) apply. Equation (5) factors into two terms, one of degree 0 and the other of degree 2 in, say, (u, v), and both linear in, say, (s, t); (5) can be written as

$$(\alpha s + \beta t)(\alpha' s + \beta' t) = a\, s^2 + b\, st + c\, t^2 = 0,$$

where α, β are constants and α', β' are polynomials in (u, v). Since $\alpha\beta' + \beta\alpha' = b$, α and the coefficients of α' have height at most h_b. Similarly, $\beta\beta' = c$ thus β and the coefficients of β' have height at most h_c. Substituting the solutions $(s = \beta, t = -\alpha)$ and $(s = \beta', t = -\alpha')$ into the parameterization (3), we get parameterizations of the cubic and the line whose coefficients have height at most $h_s + h_c = h_t + h_b = 27$. □

PROPOSITION 3.3. *When a quadric Q_R of inertia $(2, 1)$ is used to parameterize the intersection, then asymptotically the parameterization of the line has height at most 11 and the parameterization of the cubic has height at most 20 in terms of S and T.*

PROOF. We follow the algorithm outline given in Section 2.2.3 to determine the height of the output.

Let λ_0 be the quadruple real root of the determinantal equation. The determinantal equation can be written as

$$\det R(\lambda) = \gamma(\lambda - \lambda_0)^4.$$

Since the coefficients of $\det R(\lambda)$ have height 4 in terms of the coefficients of S and T (they are 4×4 determinants), λ_0 has height 1. So the coefficients of the cone $Q_R = Q_{\lambda_0 S - T}$ have height 2.

Finding the singular point of Q_R amounts to finding a point $\mathbf{c} \in \mathbb{P}^3(\mathbb{Z})$ in the kernel of R, i.e. such that $R\mathbf{c} = 0$. If we decompose R such that R_u is the upper left 3×3 matrix of R and $\mathbf{r}_4$ is the first three coordinates of the last column of R, $\mathbf{c}$ such that $\mathbf{c}_u$ is the first three coordinates and c_4 is the last and we assume $\det R_u \neq 0$ (R has rank 3 so one of its 3×3 minors is non-zero), then $\mathbf{c}$ is found by solving

$$R_u \mathbf{c}_u = -c_4 \mathbf{r}_4.$$

A solution is thus $\mathbf{c} = (-R_u^* \mathbf{r}_4, \det R_u)$, where R_u^* is the adjoint of R_u. The coefficients of R_u^* and $\mathbf{r}_4$ have height respectively 2 and 1 in terms of the coefficients of R. So the coordinates of $\mathbf{c}$ have height 3 in terms of the coefficients of R, i.e. 6 in terms of the coefficients of S and T.

Since the line of the intersection is the (double) intersection of Q_R and the tangent plane to Q_S at $\mathbf{c}$, a point $\mathbf{p}$ on this line is any point satisfying

$$R\mathbf{p} = S\mathbf{c}. \tag{6}$$

(Observe that if $\mathbf{p}$ is a solution, any $a_1\mathbf{p} + a_2\mathbf{c}$ is also solution.) The right-hand side of (6) has height $6 + 1 = 7$ in the coefficients of S and T. As above, one can assume that $\det R_u \neq 0$ and there is a unique point $\mathbf{p}$ having zero as last coordinate. This point satisfies $\mathbf{p}_u = R_u^*(S\mathbf{c})_u$ and the height of its coordinates is $4 + 7 = 11$. Overall, the coefficients of the line $(\mathbf{c}, \mathbf{p})$ have height 11.

Now, we need to parameterize a cone (Q_R) containing a rational point $(\mathbf{p})$. First, we apply to R a projective transformation P sending the point $(0, 0, 0, 1)^T$ to $\mathbf{c}$ and the point $(0, 0, 1, 0)^T$ to $\mathbf{p}$. We are left with the problem of parameterizing a cone going through the origin. Such a cone has equation

$$a_1 x^2 + a_2 xy + a_3 y^2 + a_4 yz + a_5 xz = 0 \tag{7}$$

and a parameterization is given by

$$\begin{pmatrix} a_5 & 0 & a_4 & 0 \\ 0 & a_4 & a_5 & 0 \\ -a_1 & -a_3 & -a_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u^2 \\ v^2 \\ uv \\ s \end{pmatrix},$$

where (u, v, s) lives in a real quasi-projective space (see [2]) and where a_1, a_2, a_3 have height 2 and a_4, a_5 have height $11 + 2 = 13$. Lifting the parameterization to the original space by multiplying by matrix P, we find a parameterization $\mathbf{X}(u, v, s)$ of Q_R such that the coefficients of u^2, v^2, uv have height 13 and the coefficients of s have height 6. Plugging $\mathbf{X}(u, v, s)$ in the equation of any other quadric of the pencil gives an equation in the parameters of the form:

$$as^2 + b(u, v)s + c(u, v) = 0, \tag{8}$$

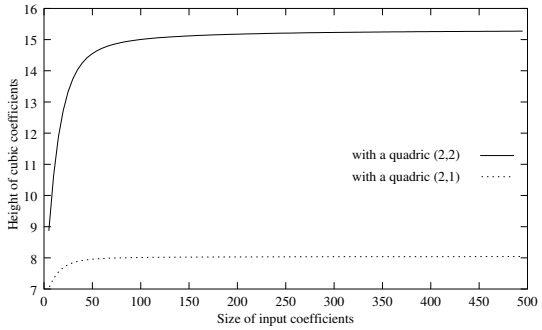


Figure 2: Observed height of the parameterization of the cubic in the cubic and tangent line case.

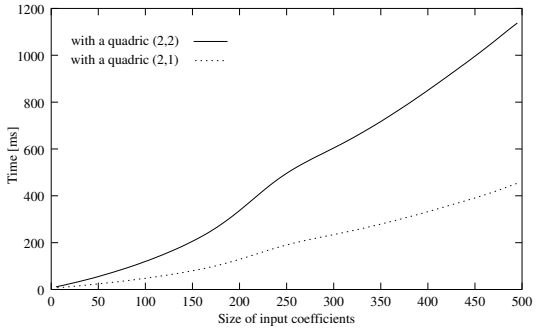


Figure 3: Computation time for the cubic and tangent line case.

where $b(u,v)$ is a polynomial of degree 2 and height $1+6+13=20$ and $c(u,v)$ a polynomial of degree 4 and height $2\times13+1=27$. Observe first that $a=0$ since the singularity of the cone, which is a point of the intersection, is reached at $(u,v)=(0,0)$ and at this point $s\neq 0$ necessarily (because $\mathbf{X}(u,v,s)$ is a faithful parameterization of the cone). We also know that (8) has a linear factor corresponding to the line of the intersection and it is not too difficult to realize that this factor is $a_5\,u+a_4\,v$, where a_4 and a_5 are as in (7). So, after factoring out the linear term, (8) can be rewritten

$$b'(u,v)s+c'(u,v)=0,$$

where $b'(u,v)$ is a polynomial of degree 1 and height $20-13=7$ and $c'(u,v)$ is a polynomial of degree 3 and height $27-13=14$. We can solve this equation rationally for s and, multiplying to clear the denominators, we get a parameterization of the cubic of height $14+6=20$. □

The difference in the heights of the parameterizations underscored in the above two propositions is vindicated by some experiments we made. In the smooth quartic case, random examples could be generated by taking input quadrics with random coefficients. But we have to proceed differently here. We start with a canonical pair of quadrics intersecting in a cubic and a tangent line and apply to this pair a random transformation. More precisely, given a canonical pair S, T, four random `bigints` r_1, r_2, r_3, r_4 and a random real projective transformation P, we take as new input quadrics the quadrics of matrix:

$$S'=P^T(r_1S+r_2T)P,\quad T'=P^T(r_3S+r_4T)P.$$

If we take all the random coefficients in the range $[-\lceil\sqrt[3]{10^s}\rceil,\lceil\sqrt[3]{10^s}\rceil]$, then the quadrics S' and T' have size s (the size of the canonical pair S,T can be neglected).

Figure 2 shows the height of the coefficients of the parameterization of the cubic when a quadric Q_R of inertia $(2,2)$ or $(2,1)$ is used (with mild optimizations turned on). The plots clearly show that the coefficients of the cubic are smaller when a cone is used to parameterize the intersection. The fact that the observed heights are, in the limit, so different from the theoretical bounds (8 instead of 22 when a cone is used) is possibly a consequence of the way S' and T' are generated: it certainly does not reflect a truly random distribution in the space of quadrics with integer coefficients of size s intersecting in a cubic and a tangent line.

Figure 3 further reinforces our choice of using a cone: the parameterizations have not just smaller coefficients, they are also faster to compute.

4. EXPERIMENTAL RESULTS

We now report on some experimental results and findings from our implementation.

The experiments were made on a Dell Precision 360 with a 2.60 GHz Intel Pentium CPU. *LiDIA*, *GMP* and our own code were compiled with g++ 3.2.2.

4.1 Implementation versions

In what follows, we compare three versions of our implementation:

- *unoptimized*: nothing is done to simplify the coefficients either during the computations or in the parameterizations computed;
- *mildly optimized*: some gcds are performed at an early stage (optimization of the coefficients and of the roots of the determinantal equation, optimization of the coordinates of singular and rational points, ...) to avoid hampering later calculations with unnecessarily big numbers;
- *strongly optimized*: mildly optimized, plus extraction of the square factors of some `bigints` (like in the smooth quartic case, where $\sqrt{\delta}=\sqrt{\det R}$ can be replaced by $b\sqrt{a}$ if $\det R=a\,b^2$) and gcd simplifications of the coefficients of the final parameterizations.

For the extraction of the square factors of an integer n, the strongly optimized version finds all the prime factors of n up to $\min(\lceil\sqrt[3]{n}\rceil,$ `MAXFACTOR`$)$, where `MAXFACTOR` is a predefined global variable.

Let us finally mention that we tried a fourth version of our implementation where the extraction of the square factors is done by fully factoring the numbers (using the Elliptic Curve Method and the Quadratic Sieve implemented in *LiDIA* [8]). But this version is almost of no interest: for small input coefficients, the strongly optimized version already finds all the necessary factors, and for medium to large input coefficients, integer factoring becomes extremely time consuming.

4.2 Performance evaluation

Let us first discuss the impact of the `MAXFACTOR` variable on the output. Figure 4 shows that a value of 10^5 has a dramatic impact on computation time while all values less than 10^4 are acceptable. We have determined that the best compromise between efficiency and complexity of the output is obtained by setting `MAXFACTOR` to 10^3, which we assume

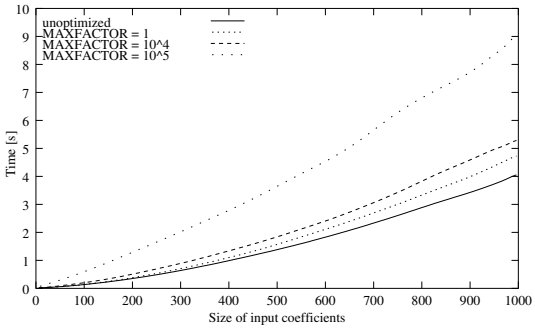


Figure 4: Evolution of execution time in the smooth quartic case as a function of the size of the input for very large input sizes.

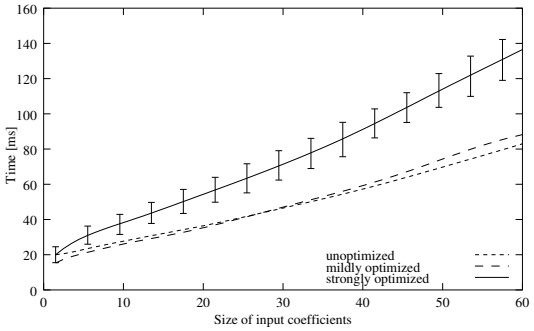


Figure 5: Evolution of execution time in the smooth quartic case as a function of the input size, with the standard deviation shown on the optimized plot.

now.

Figure 5 shows the evolution of the aggregate computation time in the smooth quartic case, which is the most computationally demanding case, with the three versions outlined above. What we infer from these plots is first that the computation times for the unoptimized and mildly optimized versions are very similar, while we observe (see Figure 1) a dramatic improvement in the height of the output coefficients with the mildly optimized version for reasonably small inputs. This explains our choice of putting the mild optimizations in the form of a preprocessor directive, not a binary argument: they might as well have been called *mandatory optimizations*.

A second lesson to be learned from Figures 1 and 5 is that for a size of input ranging from roughly 5 to 60, the computation time is roughly 30% larger for the strongly optimized version than for the mildly optimized. At the same time, the height of the output is between 20% (input size of 5) and 5% (input size of 60) smaller. For large values of the input size (Figure 4), the difference in computation time between the mildly optimized and the strongly optimized versions drops to less than 10%, but not much is gained in terms of height of the output.

Another interesting piece of information inferred from Figure 1 is that the standard deviation of the height of the output coefficients is large for small input size in the strongly optimized version. This means that in the good cases the height of the output is dramatically smaller than the height in the mildly optimized case, and in the bad cases is similar

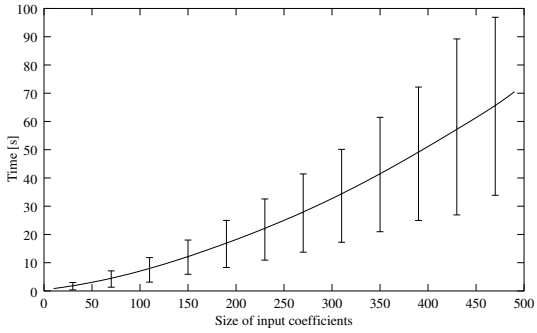


Figure 6: Computation time for 120 pairs of quadrics covering all intersection cases, with standard deviation.

to it.

Deciding to spend time on optimization essentially depends on the application. For most real-world applications, where the size of the input quadrics is small by construction, we believe optimizing is important: it should be kept in mind that the computed parameterizations are often the input to a later processing step (like in boundary evaluation) and limiting the growth of the coefficients at an early stage makes good sense.

A last comment that can be made looking at Figure 4 concerns the efficiency of our implementation. Indeed, those plots show that we can compute the parameterization of the intersection of two quadrics with coefficients having 400 digits in 1 second and 1,000 digits in 5 seconds (on average).

Efficiency can be measured in a different way. In Figure 6, we have plotted the total computation time, with the strongly optimized version, for a file containing 120 pairs of quadrics covering all intersection situations over the reals. The "random" quadrics were generated as in Section 3.2. For an input size $s = 500$, the total computation time is roughly 72 seconds, on average, for the 120 pairs of quadrics, i.e. 0.6 second per intersection. This should be compared to the 1.7 seconds on average needed to compute the intersection in the smooth quartic case for the same size of input (Figure 4). This difference is simply explained by the fact that very degenerate intersections (like when the determinantal equation vanishes identically, which represents 36 of the 120 quadrics in the file) are usually much faster to compute.

Our last word will be on memory consumption. Our implementation eats up very little memory. In the smooth quartic case, the total memory chunks allocated sum up to less than 64 kilobytes for input sizes up to 20. It takes input sizes of more than 700 digits to get to the 1 MB range of used memory.

5. EXAMPLES

We now give three examples of parameterizations computed by our algorithm. Other examples can be tested by querying our parameterization server.

Comparing our results with the parameterizations computed with other methods does not make much sense since our implementation is the first to output exact parameterizations in all cases. However, for the sake of illustration, our first two examples are taken from the paper describing the plane cubic curve method of Wang, Joe and Goldman [18].

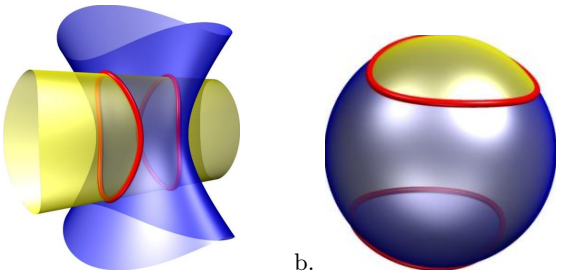
QI output 1 Execution trace for Example 2.

```
>> quadric 1: 19*x^2 + 22*y^2 + 21*z^2 - 20*w^2
>> quadric 2: x^2 + y^2 + z^2 - w^2

>> launching intersection
>> determinantal equation: - 175560*l^4 - 34358*l^3*m - 2519*l^2*m^2 - 82*l*m^3 - m^4
>> gcd of derivatives of determinantal equation: 1
>> number of real roots: 4
>> intervals: ]-14/2^8, -13/2^8[, ]-26/2^9, -25/2^9[, ]-25/2^9, -24/2^9[, ]-3/2^6, -2/2^6[
>> picked test point 1 at [ -13 256 ], sign > 0 -- inertia [ 2 2 ] found
>> picked test point 2 at [ -3 64 ], sign > 0 -- inertia [ 2 2 ] found
>> quadric (2,2) found: - 16*x^2 + 5*y^2 - 2*z^2 + 9*w^2
>> decomposition of its determinant [a,b] (det = a^2*b): [ 12 10 ]
>> a point on the quadric: [ 3 0 0 4 ]
>> param of quadric (2,2): [0, - 24*s*u - 24*t*v, 0, 0] + sqrt(10)*[3*t*u + 6*s*v, 0, 12*s*u - 12*t*v, - 4*t*u + 8*s*v]
>> status of smooth quartic param: near-optimal
>> end of intersection

>> complex intersection: smooth quartic
>> real intersection: smooth quartic, two real bounded components
>> parameterization of smooth quartic, branch 1:
  [(72*u^3 + 4*u*v^2)*sqrt(10) + 3*v*sqrt(10)*sqrt(Delta), - 340*u^2*v + 10*v^3 - 24*u*sqrt(Delta), (- 118*u^2*v + 5*v^3)*sqrt(10)
   + 12*u*sqrt(10)*sqrt(Delta), (96*u^3 - 12*u*v^2)*sqrt(10) - 4*v*sqrt(10)*sqrt(Delta)]
>> parameterization of smooth quartic, branch 2:
  [(72*u^3 + 4*u*v^2)*sqrt(10) - 3*v*sqrt(10)*sqrt(Delta), - 340*u^2*v + 10*v^3 + 24*u*sqrt(Delta), (- 118*u^2*v + 5*v^3)*sqrt(10)
   - 12*u*sqrt(10)*sqrt(Delta), (96*u^3 - 12*u*v^2)*sqrt(10) + 4*v*sqrt(10)*sqrt(Delta)]
>> Delta = 20*u^4 - 140*u^2*v^2 + 5*v^4

>> time spent: < 10 ms
```



a. b.

Figure 7: Two examples of pairs of quadrics intersecting in a smooth quartic.

5.1 Example 1

Our first example is Example 4 from [18]. The two quadrics are a quadric of inertia $(2,1)$ (elliptic cylinder) and a quadric of inertia $(2,2)$ (hyperboloid of one sheet). The curve of intersection C has implicit equation

$$C : 4\,x^2 + z^2 - w^2 = x^2 + 4\,y^2 - z^2 - w^2 = 0.$$

A rendering of the intersection is given in Figure 7.a (made with *Surf* [15]).

In [18], the authors find the following parameterization for C:

$$\mathbf{X}(u,v) = \mathbf{X}_1(u,v) \pm \mathbf{X}_2(u,v)\sqrt{\Delta(u,v)}, \qquad (9)$$

with

$$\mathbf{X}_1(u,v) = \begin{pmatrix} 0.0 \\ 1131.3708\,u^3 - 5760.0\,u^2v + 10861.1602\,uv^2 - 8192.0\,v^3 \\ -1600.0\,u^3 + 10861.1602\,u^2v - 21504.0\,uv^2 + 11585.2375\,v^3 \\ 1600.0\,u^3 + 3620.2867\,u^2v + 5120.0\,uv^2 + 11585.2375\,v^3 \end{pmatrix},$$

$$\mathbf{X}_2(u,v) = \begin{pmatrix} -80.0\,u + 1181.0193\,v \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix},$$

and $\Delta(u,v) = 905.0967\,u^3v - 3328.0\,u^2v^2 + 2896.3094\,uv^3$. The authors report a computation error on this example (measured as the maximum distance from a sequence of sample points on the curve to the input quadrics) of order $O(10^{-7})$.

Our implementation outputs the following exact and simple result in less than 10 ms:

$$\mathbf{X}(u,v) = \begin{pmatrix} 2\,u^3 - 6\,uv^2 \\ 7\,u^2v + 3\,v^3 \\ 10\,u^2v - 6\,v^3 \\ 2\,u^3 + 18\,uv^2 \end{pmatrix} \pm \begin{pmatrix} -2\,v \\ u \\ 2\,u \\ 2\,v \end{pmatrix} \sqrt{-3\,u^4 + 26\,u^2v^2 - 3\,v^4}.$$

The polynomials involved in the parameterization are defined in $\mathbb{Q}[u,v]$, which means we are in the lucky case where the intermediate quadric of inertia $(2,2)$ found to parameterize the intersection has a square as determinant. So the parameterization obtained is optimal in the extension of $\mathbb{Z}$ on which its coefficients are defined.

5.2 Example 2

Our second example is Example 5 from [18]. It is the intersection of a sphere and an ellipsoid that are very similar (see Figure 7.b):

$$C : 19\,x^2 + 22\,y^2 + 21\,z^2 - 20\,w^2 = x^2 + y^2 + z^2 - w^2 = 0.$$

In [18], the authors compute the parameterization (9) with

$$\mathbf{X}_1(u,v) = \begin{pmatrix} -0.72\,u^3 - 0.72\,u^2v + 0.08\,uv^2 + 0.08\,v^3 \\ 0.0 \\ 0.72\,u^3 - 1.2\,u^2v - 0.72\,uv^2 - 0.08\,v^3 \\ 1.0182\,u^3 + 0.3394\,u^2v + 0.3394\,uv^2 + 0.1131\,v^3 \end{pmatrix},$$

$$\mathbf{X}_2(u,v) = \begin{pmatrix} 0.0 \\ 1.697\,u + 0.5656\,v \\ 0.0 \\ 0.0 \end{pmatrix},$$

and $\Delta(u,v) = 0.48\,u^3v - 0.32\,u^2v^2 - 0.16\,uv^3$.

Our implementation gives the result displayed in Output 1. Since the polynomials of $\mathbf{X}(u,v)$ involve a square root $\sqrt{10}$,

QI output 2 Execution trace for Example 3.

```
>> quadric 1: - 4*x^2 - 56*x*y - 24*x*z - 79*y^2 - 116*y*z
+ 70*y*w - 85*z^2 - 20*z*w + 9*w^2
>> quadric 2: 6*x^2 + 84*x*y + 36*x*z + 45*y^2 + 160*y*z
- 210*y*w + 131*z^2 + 30*z*w - 45*w^2

>> complex intersection: two tangent conics
>> real intersection: two tangent conics
>> parameterization of conic:
  [- 39*u^2 + 443*u*v - 7254*v^2, 3*u^2 - 66*u*v + 1388*v^2,
   6*u^2 - 132*u*v + 701*v^2, - 5*u^2 + 110*u*v - 3005*v^2]
>> parameterization of conic:
  [- 39*u^2 + 443*u*v - 4004*v^2, 3*u^2 - 66*u*v + 1138*v^2,
   6*u^2 - 132*u*v + 201*v^2, - 5*u^2 + 110*u*v - 1205*v^2]

>> time spent: 10 ms
```

the quadric Q_R of inertia $(2, 2)$ used to parameterize the intersection is such that its determinant is not a square. As explained in Section 2.2.2, the parameterization is thus only near-optimal in the sense that it is possible, though not necessary, that the square root can be avoided in the coefficients.

It turns out that in this particular example it can be avoided. Consider the cone Q_R corresponding to the rational root $(-1, 21)$ of the determinantal equation:

$$Q_R : -Q_S + 21\,Q_T = 2x^2 - y^2 - w^2.$$

Q_R contains the obvious rational point $(1, 1, 0, 1)$, which is not its singular point. This implies that it can be rationally parameterized. Plugging this parameterization in the equation of Q_S or Q_T gives a simple parameterization for C:

$$\mathbf{X}(u, v) = \begin{pmatrix} u^2 + 2\,v^2 \\ 2\,uv \\ u^2 - 2\,v^2 \\ 0 \end{pmatrix} \pm \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \sqrt{2\,u^4 + 4\,u^2 v^2 + 8\,v^4}.$$

5.3 Example 3

Our last example illustrates the fact that our implementation is complete in the sense that it computes parameterizations in all possible cases. It concerns two quadrics intersecting in two tangent conics. The execution trace (with debugging information turned off) is given in Output 2.

6. CONCLUSION

We have presented a C++ implementation of an algorithm for parameterizing intersections of quadrics. The implementation is exact, efficient and covers all the possible cases of intersection. This implementation is based on the *LiDIA* library and uses the multiprecision integer arithmetic of *GMP*.

Future work will be devoted to understanding the gaps between predicted and observed values for the height of the coefficients of the parameterizations, to working out predicates and filters for making the code robust with floating point data (many classes and data structures have already been templated for a future use with floating point coefficients) and to porting our code to the *CGAL* geometry algorithms library.

7. ACKNOWLEDGMENTS

The authors wish to acknowledge Laurent Dupont for a preliminary implementation of the parameterization algorithm in *MuPAD*, Guillaume Hanrot for his C implementation of Uspensky's algorithm, Daniel Lazard for his help in designing the parameterization algorithm, and Etienne Petitjean for his NetTask socket management tool which makes possible the querying of our parameterization software via a web interface.

8. REFERENCES

[1] T. Bromwich. *Quadratic Forms and Their Classification by Means of Invariant Factors*. Cambridge Tracts in Mathematics and Mathematical Physics, 1906.

[2] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics. In *Proc. of SoCG (ACM Symposium on Computational Geometry), San Diego*, pages 246–255, 2003.

[3] R. Farouki, C. Neff, and M. O'Connor. Automatic parsing of degenerate quadric-surface intersections. *ACM Transactions on Graphics*, 8(3):174–203, 1989.

[4] *GMP*: The GNU MP Bignum Library. The Free Software Foundation. `http://www.swox.com/gmp`.

[5] C.-K. Hung and D. Ierardi. Constructing convex hulls of quadratic surface patches. In *Proceedings of 7th CCCG (Canadian Conference on Computational Geometry), Quebec, Canada*, pages 255–260, 1995.

[6] T. Lam. *The Algebraic Theory of Quadratic Forms*. W.A. Benjamin, Reading, MA, 1973.

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

[8] *LiDIA*: A C++ Library for Computational Number Theory. Darmstadt University of Technology. `http://www.informatik.tu-darmstadt.de/TI/LiDIA`.

[9] J. Miller and R. Goldman. Geometric algorithms for detecting and calculating all conic sections in the intersection of any two natural quadric surfaces. *Graphical Models and Image Processing*, 57(1):55–66, 1995.

[10] B. Mourrain, J.-P. Técourt, and M. Teillaud. Predicates for the sweeping of an arrangement of quadrics in 3D. Technical Report ECG-TR-242205-01, Effective Computational Geometry for Curves and Surfaces (European project), 2003.

[11] F. Rouillier and P. Zimmermann. Efficient isolation of a polynomial real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2004.

[12] R. Sarraga. Algebraic methods for intersections of quadric surfaces in GMSOLID. *Computer Vision, Graphics, and Image Processing*, 22:222–238, 1983.

[13] E. Schömer and N. Wolpert. An exact and efficient approach for computing a cell in an arrangement of quadrics. *Computational Geometry: Theory and Applications*, 2003. Special Issue on Robust Geometric Algorithms and their Implementations, submitted.

[14] C.-K. Shene and J. Johnstone. On the lower degree intersections of two natural quadrics. *ACM Transactions on Graphics*, 13(4):400–424, 1994.

[15] *Surf*: A Tool To Visualize Algebraic Curves and Surfaces. Stephan Endrass, Hans Huelf, Ruediger Oertel, Ralf Schmitt, Kai Schneider and Johannes Beigel. `http://surf.sourceforge.net`.

[16] F. Uhlig. Simultaneous block diagonalization of two real symmetric matrices. *Linear Algebra and Its Applications*, 7:281–289, 1973.

[17] W. Wang, R. Goldman, and C. Tu. Enhancing Levin's method for computing quadric-surface intersections. *Computer-Aided Geometric Design*, 20(7):401–422, 2003.

[18] W. Wang, B. Joe, and R. Goldman. Computing quadric surface intersections based on an analysis of plane cubic curves. *Graphical Models*, 64(6):335–367, 2002.

[19] I. Wilf and Y. Manor. Quadric-surface intersection curves: shape and structure. *Computer-Aided Design*, 25(10):633–643, 1993.