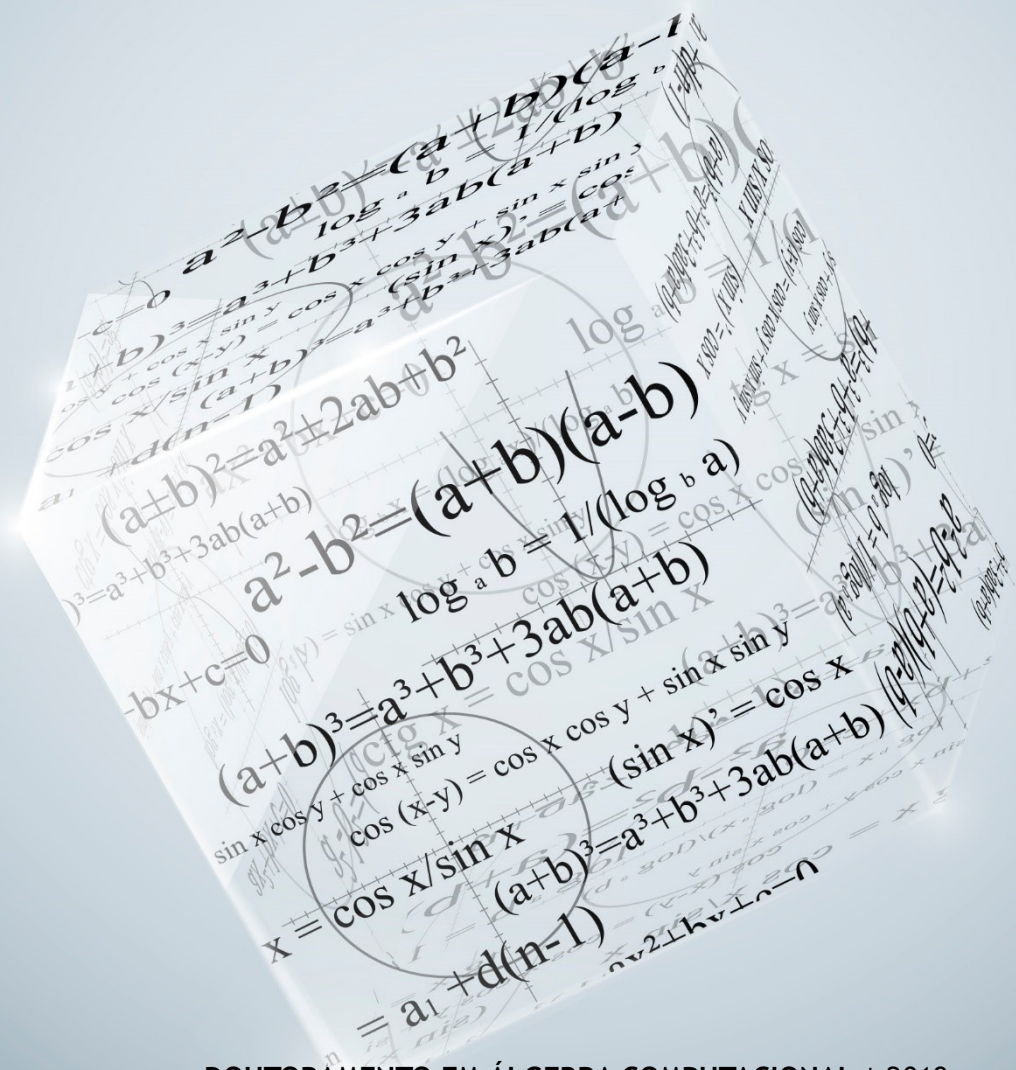
UNIVERSIDADE ABERTA
www.uab.pt

Computing the intersection of two quadrics through projection and lifting



DOUTORAMENTO EM ÁLGEBRA COMPUTACIONAL | 2019

Alexandre Emanuel Batista da Silva Trocado

Computing the intersection of two quadrics through projection and lifting.

Alexandre Emanuel Batista da Silva Trocado

Dissertation submitted in partial fulfillment for the degree of Doctor in Computational Algebra

Supervisors:

Prof. Dr. Laureano Gonzalez-Vega

Prof. Dr. João Jorge Ribeiro Soares Gonçalves de Araújo

at

Universidade Aberta

Departamento de Ciências e Tecnologia

November 2019

Declaration of Authorship

I, *Alexandre Emanuel Batista da Silva Trocado* declare that this Dissertation, entitled - *Computing the intersection of two quadrics through projection and lifting* - and the work presented in it, are my own. I confirm that:

- This work was entirely or mainly done while applying in candidature for a research degree at this University.
- Whenever any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- When I have consulted the published work of others, this was always clearly attributed.
- In cases of quotation from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- All main sources of help have been acknowledged.
- Whenever the dissertation involved joint work with others, it has been made clear what was done by others and what was contributed by myself.
- According to the "regulamento do doutoramento em álgebra computacional" this dissertation includes the papers that have been already been published or accepted for publication.

Signed:

_____

Date:

_____

"Your time is limited, so don't waste it living someone else's life;

Don't be trapped by dogma, which is living with the results of other people's thinking;

Don't let the noise of others' opinions drown out your own inner voice;

And most important, have the courage to follow your heart and intuition".

Steve Jobs

Abstract

The aim of this dissertation is to study the intersection of two quadrics, improving from the theoretical point of view through the demonstration of new results and from the practical point of view by implementing an algorithm in two softwares, GeoGebra and Maple.

The method used to determine the intersection curve of two quadrics was the projection onto a plane of the intersection curve, analysis of this curve and lifting. This work includes three articles, one accepted (see Chapter 2) and two other published (see Chapters 3 and 4).

The first paper (see Chapter 2) is devoted to the description of the method by defining a theoretical framework also used in the following two papers. In this first paper new results are presented: to determine the analytic expression of the cutcurve (projection curve onto a plane of the quadrics' intersection curve) which must be defined in a plane region bounded by the silhouette curves (projection curves of boundaries of both quadrics); to determine singular points of the cutcurve and common points between the cutcurve and the silhouette curves. The algebraic method was defined using resultants and subresultants and its performance was tested by an implementation on Maple software.

Chapter 3 is devoted to adapting the developed algorithm to the GeoGebra software and its aim is to contribute to the creation of a tool that allows this software to compute the intersection curve of any two quadrics.

In Chapter 4, the algorithm defined in Chapter 2 is developed and implemented on Maple software. The aim of this work is to contribute to the accuracy of Maple `intersectplot` command.

Finally, some conclusions taken from the work developed so far are presented; besides, other considerations about the potential research in this area will be given.

Keywords: Quadrics, Intersection curve, Cutcurve, Silhouette curve, Resultants, Subresultants, GeoGebra, Maple.

Resumo

O objetivo desta dissertação é o estudo da interseção de duas quádricas, do ponto de vista teórico através da demonstração de novos resultados e, do ponto de vista prático, implementando um algoritmo em dois *softwares*, o GeoGebra e o Maple.

As quádricas são as mais simples superfícies curvas e determinar a sua interseção tem sido um problema de interessante resolução nas últimas décadas. Com o aumento do uso de computadores, a sua determinação e descrição ganhou uma maior relevância.

Muitos problemas surgem frequentemente em aplicações de engenharia, modelação geométrica, projeto assistido por computador e robótica [2]. Um grande número de algoritmos foi proposto em [6], [23] e [2], baseados em técnicas aritméticas de ponto flutuante sensíveis a erros de arredondamento e com baixos tempos de execução em detrimento da precisão. Por outro lado, métodos simbólicos baseados na aritmética exata garantem a exatidão dos resultados, mas com tempos de execução altos. Portanto, a técnica a ser usada ao lidar com problemas de interseção deve ser cuidadosamente escolhida para obter tempos de execução aceitáveis.

Durante vários anos deve-se a Levin ([27] e [28]) o único método geral conhecido para a determinação e representação da interseção de duas quádricas. Este método basea-se na análise do conjunto definido pela combinação linear das duas quádricas. No entanto, este método, por vezes, falha quando a curva de interseção é singular ou quando é usado um método de representação que recorre a ponto flutuante.

Ao longo do tempo, este método proposto por Levin foi alvo de diferentes melhorias e recentemente Dupont e outros ([12], [13], [14]) propuseram um algoritmo que determina, recorrendo à análise de várias dezenas de casos no espaço projetivo, a interseção de duas quádricas. A performance deste algoritmo foi analisada em [26].

Nesta dissertação, o método utilizado recorre à projeção num plano da curva de interseção, à análise da topologia dessa curva e ao seu levantamento. Este trabalho inclui três artigos, um submetido para publicação (ver Capítulo 2) e dois outros publicados (ver Capítulos 3 e 4).

Porquê o recurso ao GeoGebra e ao Maple?
O GeoGebra é um *software* de geometria dinâmica que pode ser usado em todos os níveis de ensino que combina funcionalidades de Geometria 2D, 3D, álgebra, cálculo algébrico simbólico

(CAS), representação gráfica, cálculo e estatística. A este *software* está associada uma larga comunidade de utilizadores espalhada pelo mundo, fazendo com que este seja considerado um dos *softwares* mais utilizados na educação ao nível do ensino básico e secundário.

Por outro lado, o Maple é um poderosa ferramenta de cálculo algébrico simbólico. Esta ferramenta permite que estudantes, educadores e matemáticos consigam efetuar cálculo simbólico, numérico e construção de algoritmos, programáveis através de uma linguagem e comandos próprios, combinando assim, o poder dos algoritmos com as funcionalidades do CAS.

De uma forma geral, o GeoGebra tem a vantagem de ser um *software* livre, de código aberto e de utilização simples que permite a interação com os objetos matemáticos, em diferentes perspetivas, de uma forma bastante intuitiva. No entanto, algumas das suas funcionalidades ainda possuem algumas limitações, em particular, a interseção de objetos 3D. Por outro lado, o Maple tem a desvantagem de ser um software comercial, por dificultar a utilização universal, e a interação com os objetos matemáticos não é tão intuitiva como o do GeoGebra, mas os seus *packages* permitem lidar com conteúdos matemáticos muito mais exigentes do que o GeoGebra.

O primeiro artigo é dedicado à descrição do método, definindo uma estrutura teórica também usada nos dois artigos seguintes. Neste primeiro artigo, novos resultados são apresentados: a forma para determinar a expressão analítica da curva de projeção (*cutcurve*) que deve ser definida numa região plana delimitada pelas curvas silhueta (curvas de projeção dos limites de ambas quadricas); um método para determinar pontos singulares da curva de projeção e pontos comuns entre a *cutcurve* e as curvas de silhueta. O método algébrico foi definido recorrendo a resultantes e subresultantes, sendo o seu desempenho testado através de uma implementação no software Maple.
Em alguns casos, foi possível determinar a exata parametrização da curva de interseção (envolvendo radicais se necessário), noutros o resultado (topologicamente correto) foi apresentado como o levantamento da discretização dos ramos da curva de projeção, uma vez que os seus pontos singulares estejam todos determinados.

O principal objetivo do segundo artigo é criar uma ferramenta para determinar a curva de interseção de duas quadráticas. Algo que o GeoGebra permite apenas para casos muito simples. Neste artigo, é apresentada uma implementação do algoritmo descrito no primeiro artigo, no GeoGebra. Essa implementação é feita usando apenas comandos do GeoGebra que

interagem com as janelas Álgebra, CAS, 2D e 3D. Para testar a eficiência e aplicabilidade desse algoritmo (e a sua implementação no GeoGebra), foram gerados aleatoriamente e testados 50 exemplos. Os casos analisados foram aqueles em que duas quadráticas são definidas por dois polinómios de grau 2, na variável z, o caso de uma quádrica é definida a partir de polinómio de grau 2 e de um polinómio de grau 1 e, finalmente, o caso em que as duas quadráticas são definidas por dois polinómios do grau 1 na variável z.

No terceiro artigo, é apresentada a implementação no *software* Maple do método descrito e analisado no primeiro artigo. O comando `intersectplot` do Maple representa a curva de interseção, num espaço tridimensional, entre duas de superfícies bidimensionais. Neste artigo, mostra-se como essa implementação no Maple melhora os resultados produzidos pelo comando `intersectplot` quando se determina a curva de interseção entre duas quádricas em 3D.

Esta abordagem não pretende classificar a curva de interseção entre as duas quadráticas consideradas. O principal objetivo é produzir de maneira muito direta uma descrição da curva de interseção que seja topologicamente correta. Esta é a razão pela qual permitimos que o levantamento da curva de projeção, quando possível, recorra ao uso de radicais ou contamos com a discretização dos ramos da curva de projeção (determinados exclusivamente pelos pontos determinados nessa curva).

A implementação no GeoGebra mostrou-se funcional, mas demonstrou ser lenta nos casos em que exitiram muitos pontos singulares da curva da curva de projeção. Usando esse processo, pontos singulares que vêm da interseção tangencial não foram determinados e esses pontos podem ser produzidos pela função `locus` GeoGebra. Alguns problemas técnicos foram detectados pela falta de precisão do comando do `locus` do GeoGebra durante a representação da curva de projeção, o que não invalidou o funcionamento correto do algoritmo. Em alguns casos, o GeoGebra poderá produzir melhores resultados se for possível aumentar a precisão da funcionalidade `locus`.

Em relação à implementação do algoritmo no Maple, seu desempenho foi analisado apenas nos casos mais complicados, nos quais as duas quadráticas são definidas por polinómios de segundo grau em z. Em alguns desses casos, foi possível obter a parametrização exata da curva de interseção mesmo com radicais. É possível destacar que alguns pontos não determinados pelo comando `intersectplot` do Maple foram calculados, usando o método descrito nesta dissertação.

Num futuro próximo, o desempenho do algoritmo poderá ser otimizado e a sua implementação no Maple deverá considerar não apenas o caso geral, ou seja, considerar as duas quadrádicas definidas por polinómios de grau menor que 2 na variável z.

Acknowledgements

I leave here an eternal thanks to Prof. Dr. Laureano Gonzalez-Vega, who besides being the supervisor of this dissertation, listened to me, appeased my anguish, and clarified my ideas.

I would like to thank the co-supervisor of the dissertation, Prof. Dr. João Jorge Ribeiro Soares Gonçalves de Araújo, for all the support provided from the beginning of the doctoral course to its end, which culminates in this work.

To all the professors who accompanied me during the doctoral course, for what they taught me and for the suggestions given to improve my academic work, I leave my thanks and my appreciation.

Thank you everyone!

Contents

To my parents, my wife and my son I dedicate this work to them and leave my eternal gratitude.

Chapter 1

Introduction

1.1 Some historical notes

The history of surfaces' intersection begins with the history of curves dating back to ancient Greece. With the rising of computers usage, in the recent decades, its determination and description has become more relevant. Many problems arise often in engineering applications so, geometric modeling, computer aided design/computer aided manufacturing and robotics [2] and a large number of algorithms have been proposed in [6], [23] and [2]. These algorithms are based on floating point arithmetic techniques that are sensitive to rounding errors and with low running times to the detriment of the precision. On the other hand, symbolic methods based on exact arithmetic guarantee the correctness of the results but with high running times. Therefore, the technique to be used when dealing with intersection problems must be carefully chosen in order to achieve acceptable running times ([4, 37]).

This work will focus on the intersection of two quadrics that are the simplest curved surfaces (algebraic surfaces of degree two) and on determining the exact parameterization of its intersection curve. Due to Levin ([27] and [28]), for many years the only known general method for computing a parametric representation of the intersection curve between two quadrics is based on the analysis of the set of linear combinations of both quadrics (pencil generated by them). However, this method fails when the intersection curve is singular. Also, if a floating point representation of numbers is used, the method sometimes outputs topologically wrong results [12].

For several years Levin's method was, in many ways, improved and expanded, namely by Sarraga [35] that implemented in GMSOLID (a system developed by General Motors Research Laboratories). Wilf *et al* ([44]) extended it by improving the parameterization scheme using the Segre characteristic and Wang *et al* [43] enhanced Levin's method by allowing it to process the intersection curve of two general quadrics within its own self-contained framework, making it capable of computing geometric and structural information - irreducibility, singularity and the number of connected componentes. Recently Dupont *et al.* ([12], [13], [14]) presented a near-optimal algorithm for computing the explicit representation of the intersection curve of two arbitrary quadrics whose coefficients are rational numbers, in the projective space, by using the reduction of quadratic forms and producing new results, characterising the intersection curve of two quadrics. Also, the performance of this algorithm was analysed in [26].

Farouki *et al.* [15] made a complete study of the degenerated cases, presenting a procedure that avoids the case-by-case situation, using factorization of multivariate polynomials and Segre characteristic. When the input quadrics have rational coefficients, this method outputs the exact parameterization for simple cases.

Chionh *et al.* ([8]) used multivariate resultants to compute the intersection of three implicit quadric surfaces. This method computes intersections in bounded portions of these surfaces and is very computationally demanding when 6 resultants, the greatest common divisor of 3 polynomials of degree up to 16 must be computed. Wolpert ([45]) studied the arrangements of a set of quadrics while Mourrain *et al.* ([32]) proposed an algorithm that reduced the problem to a dynamic two-dimensional problem. Geismann *et al.* ([16]) proposed an alternative method based in the projection and analysis onto one plane. The idea is to determine the projection of intersection curves, analyse them and make the lifting of these curves. An implementation of this method was made in [4] and theoretical aspects can be found in [36].

Taking the advantage that geometric approaches are numerically more stable then algebraic ones ([12]), instead of analyzing the type of intersection, others focused on the type of quadrics to intersect ([30], [17], [31], [24] and [38]). However these approaches are limited to planar intersections and natural quadrics (plane, cone, cylinder and sphere).

In this work a new method is proposed to determine the intersection curve of two quadrics. This method uses the projection of the intersection curve onto the plane, analyses this plane curve and makes its lifting.

1.2 Mathematical tools

The main mathematical tools used in this work are presented in what follows. Concerning our problem of computing the intersection curve between two quadrics we will use the resultant to compute the implicit equation of a plane curve containing the projection of the intersection curve and we will use subresultants to analyse the topology of such a plane curve. One more subresultant will be used to perform the lifting from the projection to the intersection curve we are computing.

1.2.1 Resultants and subresultants

Let

$$P(T) = \sum_{i=0}^{m} a_{m-i} T^i \qquad \text{and} \qquad Q(T) = \sum_{i=0}^{n} b_{n-i} T^i$$

be two polynomials with coefficients in a field ($\mathbb{Q}$ or $\mathbb{R}$ in our case). We define the j–th subresultant polynomial of P and Q with respect to T in the following way (as in [29]):

$$\mathbf{Sres}_j(P,Q;T) = \begin{vmatrix} a_0 & a_1 & a_2 & \dots & \dots & a_m & & & \\ & \ddots & \ddots & \ddots & & & \ddots & & \\ & & a_0 & a_1 & a_2 & \dots & \dots & a_m & \\ & & & & & 1 & -T & & \\ & & & & & & \ddots & \ddots & \\ & & & & & & & 1 & -T \\ b_0 & b_1 & b_2 & \dots & \dots & \dots & b_n & & \\ & \ddots & \ddots & \ddots & & & & \ddots & \\ & & b_0 & b_1 & b_2 & \dots & \dots & \dots & b_n \end{vmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} n-j \\ \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} j \\ \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} m-j \end{matrix}$$

and we define the j–th subresultant coefficient of P and Q with respect to T, $\mathbf{sres}_j(P,Q;T)$, as the coefficient of T^j in $\mathbf{Sres}_j(P,Q;T)$. The resultant of P and Q with respect to T is:

$$\mathbf{Resultant}(P,Q;T) = \mathbf{Sres}_0(P,Q;T) = \mathbf{sres}_0(P,Q;T) \ .$$

As defined in [18] the subresultant polynomial on index j of two multivariate polynomials f and g can be defined, in order to one variable, considering the other variables as parameters.

There are many different ways of defining and computing subresultants: for proofs, more details and other references for resultants and subresultants see for example [46], [9], [7], [41] and [20].

Since the resultant of P and Q is equal to the polynomial $\mathbf{sres}_0(P,Q;T)$, $\mathbf{sres}_0(P,Q;T)=0$ if and only if there exists T_0 such that $P(T_0)=0$ and $Q(T_0)=0$.

Example 1.1. Let f and g be the polynomials that implicitly define two quadrics

$$f(x,y,z)=z^2+(y-2x-1)z-x^2-y^2-xy+x-y+1$$

$$g(x,y,z)=z^2+(x-y)z+x^2+y^2-xy-2x+y-5$$

The resultant polynomial will allow to define, under some conditions, the portion of the curve that is projection onto (x,y)-plane of the intersection curve of the two quadrics (see 1.4).

The resultant of f and g with respect to z is:

$$\mathbf{Resultant}(f,g;z)=\begin{vmatrix} 1 & y-2x-1 & -x^2-y^2-xy+x-y+1 & 0 \\ 0 & 1 & y-2x-1 & -x^2-y^2-xy+x-y+1 \\ 1 & x-y & x^2+y^2-xy-2x+y-5 & 0 \\ 0 & 1 & x-y & x^2+y^2-xy-2x+y-5 \end{vmatrix} =$$

$$=7x^4-11x^3y-17x^3+23x^2y^2+12x^2y-50x^2-6xy^3-8xy^2+26xy+10x+4y^4+6y^3-29y^2-9y+31$$

Subresultants allow to easily characterise the degree of the greatest common divisor of two univariate polynomials whose coefficients depend on one or several parameters. More generally, the determinants $\mathbf{sres}_j(P,Q;T)$, which are the formal leading coefficients of the polynomials in the subresultant sequence for P and Q, can be used to compute the greatest common divisor of P and Q due to the following equivalence:

$$\mathbf{Sres}_i(P,Q;T)=\gcd(P,Q)\iff\begin{cases}\mathbf{sres}_0(P,Q;T)=\ldots=\mathbf{sres}_{i-1}(P,Q;T)=0\\ \mathbf{sres}_i(P,Q;T)\neq 0\end{cases}$$

Example 1.2. The first subresultant of f and g (defined in example 1.1) with respect to z is determined by:

$$\begin{aligned}\mathbf{Sres}_1(f,g;z) &= \begin{vmatrix} 1 & y-2x-1 & -x^2-y^2-xy+x-y+1 \\ 0 & 1 & -z \\ 1 & x-y & x^2+y^2-xy-2x+y-5 \end{vmatrix} \\ &= (-3\,x+2\,y-1)\,z-2\,x^2-2\,y^2+3\,x-2\,y+6 \\ &= f(x,y,z)-g(x,y,z)\end{aligned}$$

Thus, if (x,y) is a point in the projection of the intersection curve then

$$\mathbf{Sres}_1(f,g;z)=0 \Longleftrightarrow f(x,y,z)=g(x,y,z) \Longleftrightarrow z=-\frac{2\,x^2+2\,y^2-3\,x+2\,y-6}{3\,x-2\,y+1}$$

and we get the lifting of (x,y) (assuming that the denominator does not vanish). This equation will allow us to determine the z coordinate of the regular points of the projection curve on the (x,y)-plane (we will prove that these points never vanish the previous denominator).

1.2.2 Computing the topology of a real algebraic plane curve defined implicitly

If a real algebraic plane curve $\mathcal{C}$ is defined implicitly by a polynomial $f(x,y)\in\mathbb{R}[x,y]$

$$\mathcal{C}=\{(\alpha,\beta)\in\mathbb{R}^2\colon f(\alpha,\beta)=0\}$$

then a point $(\alpha,\beta)\in\mathbb{C}^2$ is called:

- a critical point of $\mathcal{C}(f)$ if

$$f(\alpha,\beta)=0, \qquad \frac{\partial f}{\partial y}(\alpha,\beta)=0; \qquad \frac{\partial f}{\partial x}(\alpha,\beta)\neq 0; \tag{1.1}$$

- a singular point of $\mathcal{C}(f)$ if

$$f(\alpha,\beta)=0, \qquad \frac{\partial f}{\partial y}(\alpha,\beta)=0 \qquad \frac{\partial f}{\partial x}(\alpha,\beta)=0; \tag{1.2}$$

- a regular point of $\mathcal{C}(f)$ if

$$f(\alpha, \beta) = 0, \text{and it is not critical and is not singular.} \tag{1.3}$$

The usual strategy used to determine the topology of $\mathcal{C}$ uses what it is called a cylindrical algebraic decomposition of $f(x, y)$ (see [3]). This is done usually by proceeding in the following way ([19]):

- Computing the discriminant of f with respect to y

$$\Delta_f(x) = \textbf{Resultant}(f(x, y), f_y(x, y); y)$$

  and its real roots: $\alpha_1 < ... < \alpha_r$.

- For every α_i, computing the real roots $f(\alpha_i, y)$: $\beta_{i,1} < ... < \beta_{i,s_i}$.

- For every point $(\alpha_i, \beta_{i,j})$, computing the number of half-branches of $\mathcal{C}$ to the right and to the left of this point.

The real roots of $\Delta_f(x)$ give us the x-coordinates of the singular points of the curve (for example self-crossing, cusps, isolated points, ...) and of the critical points. With this information and the analysis of what happens on the vertical lines defined by these points it is possible to characterise the topology of the curve $\mathcal{C}$. For more details and several references see [5] and [10].

Example 1.3. Let $\mathcal{C}$ be a curve defined implicitly by $f(x, y)$ in Example 1.1:

$$f(x, y) = 7x^4 - 11x^3y - 17x^3 + 23x^2y^2 + 12x^2y - 50x^2 - 6xy^3 - 8xy^2 + 26xy + 10x + 4y^4$$

$$+6y^3 - 29y^2 - 9y + 31$$

The discriminant of f with respect to y is equal to:

$$\Delta_f(x) = 138750352\,x^{12} - 243208992\,x^{11} - 1790904848\,x^{10} - 40872352\,x^9$$

$$+4936965104\,x^8 + 326579840\,x^7 - 9059351904\,x^6 - 2063951232\,x^5 + 9072756720\,x^4 + 2765801376\,x^3$$

$$-4552994448\,x^2 - 1019503584\,x + 983706768$$

and its real roots, in increasing order, are:

$$\begin{array}{lll} \alpha_1 \approx -1.631140481 & \alpha_2 \approx -1.103061710 & \alpha_3 \approx -1.094228628 \\ \alpha_4 \approx 0.632690166 & \alpha_5 \approx 0.8463106867 & \alpha_6 \approx 4.342039172 \end{array}$$

For every value of α_i we need to determine the real roots of $f(\alpha_i, y)$. In this way we get the coordinates of the critical, singular and regular points of $\mathcal{C}$ on the vertical lines $x = \alpha_i$.

By using the subresultants of $f(x, y)$ and $f_y(x, y)$ with respect to y, we determine the critical points of $\mathcal{C}$:

$$\begin{array}{ll} \{(-1.631140481, -0.3772686115)\,, & (-1.103061710, -1.424763044) \\ (0.8463106867, 0.4343856851)\,, & (4.342039172, 0.8131041693)\} \end{array}$$

By using the subresultants of $f(x, y)$ and $f_y(x, y)$ with respect to y, we determine the singular points of $\mathcal{C}$:

$$\{(-1.094228628, -1.141342928)\,, (0.6326901663, 1.449035249)\}$$

The regular points of $\mathcal{C}$ on the vertical lines $x = \alpha_i$ are:

$$\begin{array}{ll} \{(-1.103061710, -1.137471335)\,, & (-1.103061710, 0.8324048576)\,, \\ (-1.094228628, -1.703648024)\,, & (-1.094228628, 0.8449909373)\,, \\ (0.6326901663, -2.861087178)\,, & (0.6326901663, -0.5879480692)\,, \\ (0.8463106867, -2.637712804)\,, & (0.8463106867, 1.538407464)\} \end{array}$$

Let $\delta_j = \dfrac{\alpha_i + \alpha_{i+1}}{2}$ with $i, j \in \{1, \ldots, 5\}$. By using the subresultant coefficients of $f(x, y)$ and $f_y(x, y)$ with respect to y, we determine the number of real roots of $f\,(\delta_j, y)$ and we get the number of branches between the vertical lines $x = \alpha_i$ and $x = \alpha_{i+1}$.

In Figure 1.1 the red lines are determined by the critical points of $\mathcal{C}$ while the blue ones are determined by its singular points. In Figure 1.2 is possible to observe, in more detail, what is happening with $\mathcal{C}$ between the vertical lines $x = \alpha_2$ and $x = \alpha_3$.

In this example we have 2, 4, 4, 4 and 2 branches of $\mathcal{C}$ and the connections between branches and points are easy to fix since, in this particular case, we have, at most, one critical point or one singular point in each vertical line. In Figure 1.3 we can find the representation of the curve branches like segments, the critical points of $\mathcal{C}$ (in red), the singular points of $\mathcal{C}$ (in blue) and the regular points of $\mathcal{C}$ on the vertical lines $x = \alpha_i$ (in yellow).

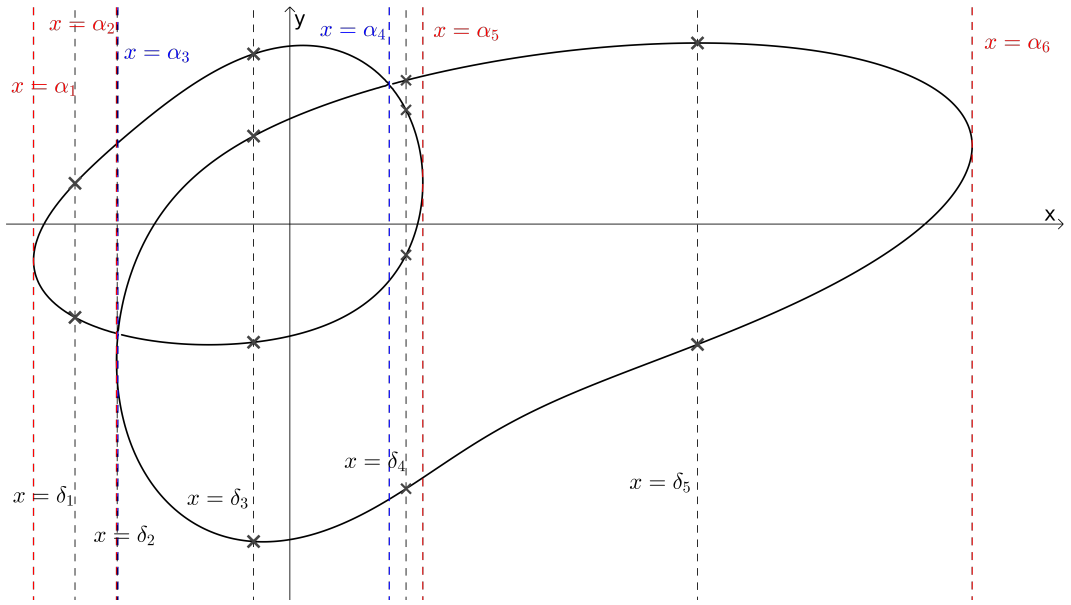


Figure 1.1: The analysis of the topology of the curve $\mathcal{C}$.

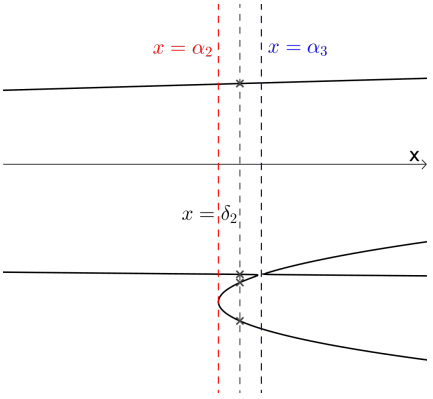


Figure 1.2: Branches between $x = \alpha_2$ and $x = \alpha_3$.

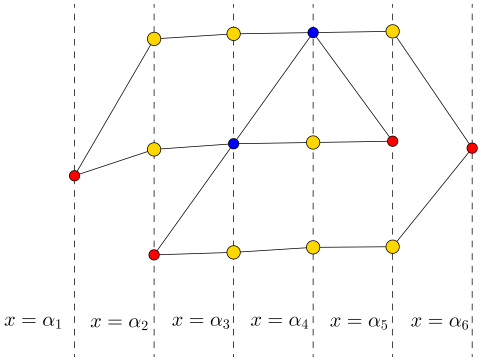


Figure 1.3: Connecting branches and points.

The curves whose topology we need to compute are quartic curves and, in most cases, they satisty the condition about the number of critical and singular points on each vertical line. When this does not happen, we will make a linear change of coordinates or we will apply an ad–hoc analysis for these particular quartic curves whose geometry is especially simple (as we have proven).

1.3 Why to use GeoGebra and Maple

GeoGebra is a dynamic mathematics software for all educational levels that combines Geometry in 2D and 3D, algebra, computer algebra system (CAS), graphing, calculus and statistics. This software has a very large community of users all over the world and became the leading software provider for science, technology, engineering and mathematics (STEAM) education, widely used in primary and secondary education because it is a free software. In addition, it is available on a wide variety of platforms (computers, tablet, smartphones), its great advantage is the multiple representation of a mathematical object by combining the different windows. If an object changes in one of its windows, the other representations are immediately updated. This software can have a positive effect preparing K-12 teachers of mathematics and instructing middle grades/secondary mathematics students ([22], [1]) or even at college level [11]. Also, Saha *et al.* ([33]) showed that computer assisted instruction, with GeoGebra, as a supplement to traditional classroom instruction was more effective than traditional instruction alone.

On the other hand, Maple is a powerful Computer Algebra System, a software that facilitates symbolic and numerical computing. It is used by students, educators, mathematicians, statisticians and engineers for exact and numeric computation. In education, Maple has been used more in college education with very positive contributions as it can be seen in [34] and [25]. But Maple is not just a CAS, with this software it is possible to code in a programming language defined by its own commands and syntax and thus combining the power of CAS with algorithms.

Overall, GeoGebra has the advantage of being a free and open source software, which allows it to spread faster and users interact with mathematical objects in a very intuitive way. However some of its capabilities are still limited, such as the intersection of 3D objects, particularly quadrics, and the production of algorithms in an easy way. On the other hand, Maple has the disadvantage of being a commercial software, which hinders the adoption and interaction with mathematical objects not as intuitive as GeoGebra. However, with Maple and its packages the user can work with more demanding mathematical content and learn to code in a very simple way.

We will use GeoGebra to implement an algorithm determining the intersection of two quadrics through projection and lifting and Maple as a tool to analyse its efficiency.

1.4 The problem to solve

Our main goal is to compute a representation of the intersection curve of two quadrics, implicitly defined by two polynomials, through projection and lifting.

Let f and g be the two polynomials in $\mathbb{R}[x, y, z]$

$$f(x, y, z) = z^2 + p_1(x, y)z + p_0(x, y) \qquad g(x, y, z) = z^2 + q_1(x, y)z + q_0(x, y)$$

with $\deg(p_1(x, y)) \leq 1$, $\deg(p_0(x, y)) \leq 2$, $\deg(q_1(x, y)) \leq 1$ and $\deg(q_0(x, y)) \leq 2$. Computing the intersection of the two quadrics defined by f and g is equivalent to solving in $\mathbb{R}^3$ the system of equations

$$f(x, y, z) = 0, \qquad g(x, y, z) = 0.$$

Let $\Delta_{\mathcal{E}_1}(x, y) = p_1(x, y)^2 - 4p_0(x, y)$ and $\Delta_{\mathcal{E}_2}(x, y) = q_1(x, y)^2 - 4q_0(x, y)$ be the discriminants of $f(x, y, z)$ and $g(x, y, z)$ (respectively) with respect to z and $\mathbf{S}_0(x, y)$ the resultant of f and g, with respect to z. Then:

$$\mathbf{S}_0(x, y) = 0 \Longleftrightarrow (p_0(x, y) - q_0(x, y))^2 - (p_1(x, y) - q_1(x, y)) \begin{vmatrix} p_0(x, y) & p_1(x, y) \\ q_0(x, y) & q_1(x, y) \end{vmatrix} = 0.$$

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two quadrics in $\mathbb{R}^3$ defined by $f(x, y, z) = 0$ and $g(x, y, z) = 0$, respectively. The curve defined by $\Delta_{\mathcal{E}_i} = 0$ will be called the silhouette of $\mathcal{E}_i$ (the outer boundary of every quadric projected onto the (x, y)-plane). The cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ is the set

$$\left\{(x, y) \in \mathbb{R}^2 : \mathbf{S}_0(x, y) = 0, \Delta_{\mathcal{E}_1}(x, y) \geq 0, \Delta_{\mathcal{E}_2}(x, y) \geq 0\right\}. \tag{1.4}$$

This curve is the projection of the intersection curve on the plane $z = 0$ and it is delimited by the silhouette curves (which are two conics). First, the projection step consists in determining the cutcurve and its analysis including computing its singular and critical points and its relation with the silhouette curves. Singular points of the cutcurve are classified in two types. Those belonging to the line $p_1(x, y) = q_1(x, y)$ are easy to compute and difficult to lift. On the other hand, singular points not belonging to this line are more complicated to compute but very easy to lift (corresponding to tangential intersection points between $\mathcal{E}_1$ and $\mathcal{E}_2$.

Regular points of the cutcurve will be lifted after solving the degree one equation $f(x,y,z) - g(x,y,z) = 0$ with respect to z, which is in this case the first subresultant of index 1 of f and g. The lifting of the singular points on the line $p_1(x,y) = q_1(x,y)$ requires to solve $f(x,y,z) = 0$ or $g(x,y,z) = 0$ with respect do z.

Simpler cases of quadrics are easier to deal with. For example, if f and g are the two polynomials in $\mathbb{R}[x,y,z]$

$$f(x,y,z) = z^2 + p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = q_1(x,y)z + q_0(x,y)$$

with $\deg(p_1(x,y)) \leq 1$, $\deg(p_0(x,y)) \leq 2$, $\deg(q_1(x,y)) \leq 1$ and $\deg(q_0(x,y)) \leq 2$ and $\mathcal{E}_1$ and $\mathcal{E}_2$ are the two quadrics defined by $f(x,y,z) = 0$ and $g(x,y,z) = 0$ then, in this case, the cutcurve is:

$$\left\{(x,y) \in \mathbb{R}^2 : \mathbf{S}_0(x,y) = 0, \Delta_{\mathcal{E}_1}(x,y) \geq 0\right\}.$$

If (α,β) is a point of the cutcurve then its lifing is given by solving the equation $g(x,y,z) = 0$ with respect to z. Nevertheless, if $q_1(\alpha,\beta) = 0$ and then $q_0(\alpha,\beta) = 0$ the lifting is given by solving the degree 2 equation $f(\alpha,\beta,z) = 0$ with respect to z.

If we consider f and g two polynomials in $\mathbb{R}[x,y,z]$ of degree 1 with respect to z

$$f(x,y,z) = p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = q_1(x,y)z + q_0(x,y)$$

with $\deg(p_1(x,y)) \leq 1$, $\deg(p_0(x,y)) \leq 2$, $\deg(q_1(x,y)) \leq 1$, $\deg(q_0(x,y)) \leq 2$ and $\mathcal{E}_1$, $\mathcal{E}_2$ are the two quadrics defined by $f(x,y,z) = 0$ and $g(x,y,z) = 0$ then, in this case, the cutcurve is:

$$\left\{(x,y) \in \mathbb{R}^2 : \mathbf{S}_0(x,y) = 0\right\} - \left\{(x,y) \in \mathbb{R}^2 : p_1(x,y) = q_1(x,y) = 0, (p_0(x,y) \neq 0 \text{ or } q_0(x,y) \neq 0)\right\}.$$

If $(\alpha,\beta,\gamma) \in \mathbb{R}^3$ satisfies $f(\alpha,\beta,\gamma) = g(\alpha,\beta,\gamma) = 0$ then (α,β) is a point of the cutcurve and its lifting is given by solving the degree 1 equation $f(\alpha,\beta,z) = 0$ or $g(\alpha,\beta,z) = 0$ with respect to z. If $p_1(\alpha,\beta) = p_0(\alpha,\beta) = 0$ and $q_1(\alpha,\beta) = q_0(\alpha,\beta) = 0$ then the vertical line $\{(\alpha,\beta,z) : z \in \mathbb{R}^3\}$ is in both quadrics.

In our approach we present an algorithm that produces in a very direct way a description of the intersection curve by performing in a very efficient way the cylindrical algebraic decomposition for the particular case of two quadrics. If it is possible to parameterize the cutcurve then our output will provide the exact parameterization of the intersection curve eventually by using

radicals if needed. On the other hand, the discretization of the branches of the cutcurve will be performed whose lifting will provide the intersection curve between the two considered quadrics.

Chapter 2

Computing the intersection of two quadrics through projection and lifting (see [39])

The aim of this paper is to present a new method to determine the intersection curve of two quadrics through projection onto a plane and lifting. In some cases, it will be possible to determine the exact parameterization of the intersection curve (involving radicals if needed) and, in others, the output (topologically correct) will be presented as the lifting of the discretisation of the branches of the projection curve once its singular points have been fully determined. The way the lifting will be made is the main criteria used to decide how to analyze the projection of the intersection curve to be computed.

This is a revised and enlarged version of the article ([39]) where, in order to simplify the description of the algorithm and to avoid a long case-by-case analysis, we have dealt only with quadrics whose defining equations have the shape $z^2 + p_1(x, y)z + p_0(x, y)$ and $z^2 + q_1(x, y)z + q_0(x, y)$. This case-by-case analysis is described here.

In sections 2 and 3 some mathematical tools are presented while section 4 analyses the projection of the intersection curve of two quadrics onto a plane called the curcurve in what follows.

Section 5 is devoted to study the cutcurve determining its singular points and its intersection with the silhouette curves (the boundary of the projection of the two quadrics). This section

introduces closed formulae for the singular and critical points of the cutcurve, characterising those coming tangential intersection points of the two considered quadrics and showing how to lift the regular and the singular points of the curcurve to the intersection curve of the two quadrics.

In section 6, the algorithm is presented together with several examples. This section shows also the results of its implementation in Maple when applied to 50 examples taken https://gamble.loria.fr/qi/server/. Finally, the last section introduces some conclusions.

Computing the intersection of two quadrics through projection and lifting

Alexandre Trocado[a], Laureano Gonzalez-Vega[1b]

[a] *Universidade Aberta, Portugal*
[b] *Universidad de Cantabria, Spain*

Abstract

This paper is devoted to presenting a new approach to determine the intersection of two quadrics based on the detailed analysis of its projection in the plane (the so called cutcurve) allowing to perform the corresponding lifting correctly. This approach is based on a new computational characterisation of the singular points of the cutcurve and on how this curve is located with respect to the projection of the considered quadrics (whose boundaries are the so called silhouette curves).

Keywords: Quadrics, Intersection curve, Cutcurve, Subresultants, Lifting.

1. Introduction

Quadrics are the simplest curved surfaces used in many areas and computing their intersection is a relevant problem. Algorithms dealing with this problem based on floating point arithmetic techniques are sensitive to rounding errors achieving a low running time to the detriment of their correctness. On the other hand, using symbolic methods guarantees the correctness of the results because they are based on exact arithmetic (if the considered quadrics are defined in exact terms) but their performance is typically and significantly lower than using methods based on numerically techniques ([2, 19]).

Levin ([13],[12]) developed a method to parameterize the intersection curve of two quadrics based on the analysis of the pencil generated by them. Wilf *et al.* ([23]) improved Levin's ruled-surface parameterization scheme. However, Levin's method often fails to find the intersection curve when it is singular and generates a parameterization that involves the square root of some polynomial ([3]). Also, when working with floating point numbers, sometimes Levin's method outputs results that are topologically wrong and even fail to produce any parameterization ([3]). Wang *et al.* ([22]) reduced the computation of the intersection curve to the analysis of plane cubic curves. Farouki *et al.* ([6]) made a complete

[1]Partially supported by the Spanish Ministerio de Ciencia, Innovacción y Universidades under the Project MMTM2017-88796-P.

Email addresses: mail@alexandretrocado.com (Alexandre Trocado), laureano.gonzalez@unican.es (Laureano Gonzalez-Vega)

study of the degenerated cases of quadric intersection by using factorization of multivariate polynomials and Segre characteristic. This method showed the exact parameterization of the intersection curve in many cases.

Later Wang *et al.* ([21]) improved Levin's method making it capable of computing geometric and structural information - irreducibility, singularity and the number of connected components. Dupont *et al.* ([3], [4], [5]) presented a near-optimal algorithm for computing the explicit representation of the intersection of two arbitrary quadrics whose coefficients are rational numbers in the projective space by using the reduction of quadratic forms and producing new results characterising the intersection curve of two quadrics. The performance of this algorithm was analysed in [11].

Others have restricted the kind of quadrics to be considered and defined specific routines to each case ([15], [8], [16], [10], [20]) taking advantage of the fact that a geometric approach is typically more stable than the algebraic ones ([3]). However these approaches are limited to planar intersections and natural quadrics. Mourrain et al. ([17]) proposed an algorithm that reduces the intersection of two quadrics to a dynamic two–dimensional problem.

An alternative way to compute the intersection of two quadric surfaces in the three-dimensional space is based on analysing its projection onto one plane ([7]). The idea of this method is to reduce the three-dimensional problem to computing the arrangement of three plane algebraic curves defined implicitely. After determining and analysing the projection of the intersection curve onto a plane, the intersection curve can be recovered by determining the lifting of the projection curve. Implementation and theoretical aspects of this approach are also described in [2] and [18], respectively.

In this paper, a new method is presented to determine the intersection curve of two quadrics through projection onto a plane and lifting. In some cases, it will be possible to determine the exact parameterization of the intersection curve (involving radicals if needed) and, in others, the output (topologically correct) will be presented as the lifting of the discretisation of the branches of the projection curve once its singular points have been fully determined. The way the lifting will be made is the main criteria followed to analyse the cutcurve.

This paper is organized as follows. In Section 2, we briefly review some preliminaries on conics and quadrics. In Section 3 some mathematical tools as resultants and subresultants are briefly presented for sake of completeness. Resultants are used in Section 4 to characterise the projection of the intersection curve (called, in what follows, the cutcurve) by using a bivariate polynomial of degree four, at most. Our approach is based on the analysis of the arrangement of the cutcurve and the silhouette of both quadrics, as in Figure 1, following [2] and [18]. Section 5 is devoted to introduce simpler methods to characterise the singular points of the cutcurve as well as its lifting by using the subresultants. Some examples are given in Section 6 where the results of the implementation in Maple are reported and the conclusions are presented in Section 7.

In order to simplify the description of the algorithm and to avoid a long case-by-case analysis we will deal only with quadrics whose defining equations have the shape $z^2 + p_1(x,y)z + p_0(x,y)$ and $z^2 + q_1(x,y)z + q_0(x,y)$. The general case is very easy to derive from what we describe here.

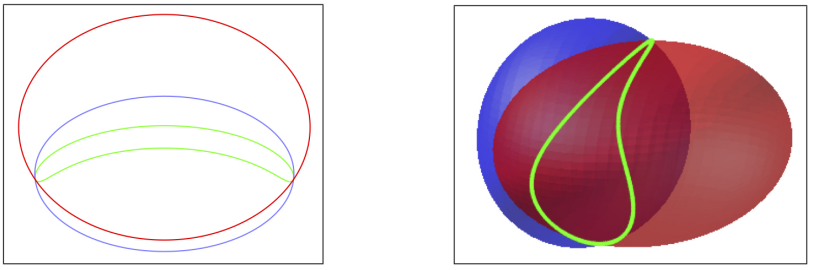


Figure 1: Left: Silhouette curves of two quadrics (in red and blue) and the cutcurve (in green) in the plane. Right: the intersection curve and the two quadrics

2. Representing quadrics (and conics)

This section is devoted to introduce how quadrics will be represented when computing their intersection curve. Since we will project the considered quadrics onto the xy plane and the boundary of this region will be a finite number of conic arcs we introduce here how these regions will be represented and manipulated.

2.1. Quadrics and conics

Quadrics are the one of the simplest surfaces defined by degree two polynomials in x, y and z. The equation of any quadric $\mathcal{A}$ in $\mathbb{R}^3$ can be written as

$$a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + 2a_{12}xy + 2a_{13}xz + 2a_{23}yz + 2a_{14}x + 2a_{24}y + 2a_{34}z + a_{44} = 0$$

or in matricial form $\begin{pmatrix} x & y & z & 1 \end{pmatrix} A \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T = 0$ where A is the symmetric matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{pmatrix}.$$

For conics the treatment is analogous (and standard). They will be used later to define the boundary of regions in the plane, typically the region where the projection of the intersection curve of the two considered curves live. In what follows we assume that it is easy to determine the intersection points of two conics and to manipulate the regions of the plane defined by two inequalities involving degree two or one polynomials.

3. Mathematical tools

In this section we will make a brief introduction to resultants and subresultants and how they will be applied later to compute the intersection curve between two quadrics.

3.1. Resultants

Resultants and subresultants will be the algebraic tool to use to determine, both, the projection of the intersection curve between the two considered quadrics and its lifting from the plane to the space since they allow a very easy and compact way to characterise the greatest common divisor of two polynomials ($f(x, y, z)$ and $g(x, y, z)$ in our case) when they involve parameters (x and y in our case, since we are going to eliminate z).

Definition 3.1. *Let*

$$P(T) = \sum_{i=0}^{m} a_{m-i} T^i \qquad \text{and} \qquad Q(T) = \sum_{i=0}^{n} b_{n-i} T^i$$

be two polynomials with coefficients in a field ($\mathbb{Q}$ or $\mathbb{R}$ in our case). We define the j–th subresultant polynomial of P and Q with respect to T in the following way (as in [14]):

$$\mathbf{Sres}_j(P, Q; T) = \begin{vmatrix} a_0 & a_1 & a_2 & \dots & \dots & a_m & & & & \\ & \ddots & \ddots & \ddots & & & \ddots & & & \\ & & a_0 & a_1 & a_2 & \dots & \dots & a_m & & \\ & & & & & 1 & -T & & & \\ & & & & & & \ddots & \ddots & & \\ & & & & & & & 1 & -T & \\ b_0 & b_1 & b_2 & \dots & \dots & \dots & b_n & & & \\ & \ddots & \ddots & \ddots & & & & \ddots & & \\ & & b_0 & b_1 & b_2 & \dots & \dots & \dots & b_n & \end{vmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} n-j \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} j \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} m-j$$

and we define the j–th subresultant coefficient of P and Q with respect to T, $\mathbf{sres}_j(P, Q; T)$, as the coefficient of T^j in $\mathbf{Sres}_j(P, Q; T)$. The resultant of P and Q with respect to T is:

$$\mathbf{Resultant}(P, Q; T) = \mathbf{Sres}_0(P, Q; T) = \mathbf{sres}_0(P, Q; T) \ .$$

There are many different ways of defining and computing subresultants: see [9] for a short introduction and for a pointer to several references.

Subresultants allow to characterise easily the degree of the greatest common divisor of two univariate polynomials whose coefficients depend on one or several parameters. Since the resultant of P and Q is equal to the polynomial $\mathbf{sres}_0(P, Q; T)$, $\mathbf{sres}_0(P, Q; T) = 0$ if and only if there exists T_0 such that $P(T_0) = 0$ and $Q(T_0) = 0$.

More generally, the determinants $\mathbf{sres}_j(P, Q; T)$, which are the formal leading coefficients of the subresultant sequence for P and Q, can be used to compute the greatest common divisor of P and Q thanks to the following equivalence:

$$\mathbf{Sres}_i(P, Q; T) = \gcd(P, Q) \Longleftrightarrow \begin{cases} \mathbf{sres}_0(P, Q; T) = \ldots = \mathbf{sres}_{i-1}(P, Q; T) = 0 \\ \mathbf{sres}_i(P, Q; T) \neq 0 \end{cases} \tag{1}$$

Let f and g be the two polynomials in $\mathbb{R}[x,y,z]$

$$f(x,y,z) = z^2 + p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = z^2 + q_1(x,y)z + q_0(x,y)$$

($\deg(p_1(x,y)) \leq 1$, $\deg(p_0(x,y)) \leq 2$, $\deg(q_1(x,y)) \leq 1$ and $\deg(q_0(x,y)) \leq 2$) defining the quadrics whose intersection curve is to be computed. Then the resultant of f and g, with respect to z, is equal to:

$$\mathbf{S}_0(x,y) \stackrel{\text{def}}{=} \mathbf{Resultant}(f,g;z) = \begin{vmatrix} 1 & p_1(x,y) & p_0(x,y) & 0 \\ 0 & 1 & p_1(x,y) & p_0(x,y) \\ 1 & q_1(x,y) & q_0(x,y) & 0 \\ 0 & 1 & q_1(x,y) & q_0(x,y) \end{vmatrix} =$$

$$= (p_0(x,y) - q_0(x,y))^2 - (p_1(x,y) - q_1(x,y)) \begin{vmatrix} p_0(x,y) & p_1(x,y) \\ q_0(x,y) & q_1(x,y) \end{vmatrix}.$$

The degree of $\mathbf{S}_0(x,y)$ is at most four. The first subresultant of f and g, with respect to z, is equal to:

$$\mathbf{S}_1(x,y;z) \stackrel{\text{def}}{=} \mathbf{Sres}_1(f,g;z) = (q_1(x,y) - p_1(x,y))z + (q_0(x,y) - p_0(x,y)) = g(x,y,z) - f(x,y,z).$$

Computing the intersection of the two quadrics defined by f and g is equivalent to solving in $\mathbb{R}^3$ the polynomial system of equations

$$f(x,y,z) = 0, \qquad g(x,y,z) = 0.$$

The solution set to be computed, when non empty, may include curves and isolated points. We will use that the above polynomial system of equations, under some conditions, is equivalent to

$$\mathbf{S}_0(x,y) = 0, \qquad (q_1(x,y) - p_1(x,y))z + (q_0(x,y) - p_0(x,y)) = 0.$$

Analyzing $\mathbf{S}_0(x,y) = 0$ in $\mathbb{R}^2$ will be called the projection step and moving the information obtained in $\mathbb{R}^2$ to $\mathbb{R}^3$ will be called the lifting step. We follow here the terminology used when computing the cylindrical algebraic decomposition of a finite set of multivariate polynomials (see [1], for example)

4. Projecting the intersection curve

In this section we will characterise the projection of the intersection curve of two quadrics onto the (x,y)–plane. The usual way of dealing with projections of algebraic sets involves tools coming from Elimination Theory. Since we project from $\mathbb{R}^3$ onto $\mathbb{R}^2$ such a description will involve polynomial inequalities.

Let f and g be the two polynomials in $\mathbb{R}[x,y,z]$

$$f(x,y,z) = z^2 + p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = z^2 + q_1(x,y)z + q_0(x,y) \qquad (2)$$

with $\deg(p_1) \leq 1$, $\deg(p_0) \leq 2$, $\deg(q_1) \leq 1$ and $\deg(q_0) \leq 2$.

Let $\Delta_{\mathcal{E}_1}(x,y) = p_1(x,y)^2 - 4p_0(x,y)$ and $\Delta_{\mathcal{E}_2}(x,y) = q_1(x,y)^2 - 4q_0(x,y)$ be the discriminants of $f(x,y,z)$ and $g(x,y,z)$ (respectively) with respect to z.

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be the corresponding quadrics:

$$\mathcal{E}_1 : \left\{(x,y,z) \in \mathbb{R}^3 : f(x,y,z) = 0\right\} \qquad \mathcal{E}_2 : \left\{(x,y,z) \in \mathbb{R}^3 : g(x,y,z) = 0\right\},$$

and Π be the projection :

$$\begin{array}{rccc} \Pi : & \mathbb{R}^3 & \to & \mathbb{R}^2 \\ & (x,y,z) & \mapsto & (x,y) \end{array}$$

Next (easy to prove) theorem characterises the set $\pi(\mathcal{E}_1 \cap \mathcal{E}_2)$, the projection of the intersection curve of $\mathcal{E}_1$ and $\mathcal{E}_2$.

Theorem 4.1.

$$\Pi(\mathcal{E}_1 \cap \mathcal{E}_2) = \left\{(x,y) \in \mathbb{R}^2 : \mathbf{S}_0(x,y) = 0, \Delta_{\mathcal{E}_1}(x,y) \geq 0, \Delta_{\mathcal{E}_2}(x,y) \geq 0\right\}.$$

Proof. If $(a,b) \in \Pi(\mathcal{E}_1 \cap \mathcal{E}_2)$ then there exists $c \in \mathbb{R}$ such that $(a,b,c) \in \mathcal{E}_1 \cap \mathcal{E}_2$. Thus $f(a,b,c) = 0$ and $g(a,b,c) = 0$, $f(a,b,z)$ and $g(a,b,z)$ have a common root ($c \in \mathbb{R}$) and we conclude that $\mathbf{S}_0(a,b) = 0$. As c is a real root of $f(a,b,z)$ and $g(a,b,z)$ we also have $\Delta_{\mathcal{E}_1}(a,b) \geq 0$ and $\Delta_{\mathcal{E}_2}(a,b) \geq 0$.

On the other hand, if $(a,b) \in \mathbb{R}^2$ verifies $\mathbf{S}_0(a,b) = 0$ then $f(a,b,z)$ and $g(a,b,z)$ have a common root $c \in \mathbb{C}$: $f(a,b,c) = 0$ and $g(a,b,c) = 0$ (according to (1)). However, if $\Delta_{\mathcal{E}_1}(a,b) \geq 0$ and $\Delta_{\mathcal{E}_2}(a,b) \geq 0$ then c must be a real solution of $f(a,b,z) = 0$ and $g(a,b,z) = 0$, concluding that $(a,b) \in \Pi(\mathcal{E}_1 \cap \mathcal{E}_2)$. □

Previous theorem gives the precise description of the projection of the intersection curve of two quadrics when their defining equations have the structure introduced in (2). It corresponds to the part of the curve

$$\{(x,y) \in \mathbb{R}^2 \colon \mathbf{S}_0(x,y) = 0\}$$

inside the region

$$\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2} \stackrel{\text{def}}{=} \{(x,y) \in \mathbb{R}^2 \colon \Delta_{\mathcal{E}_1}(x,y) \geq 0, \Delta_{\mathcal{E}_2}(x,y) \geq 0\}.$$

As expected this set is a semialgebraic set in $\mathbb{R}^2$ since, according to Tarski Principle (see [1]), the projection of any semialgebraic set is a semialgebraic set too. Thus, the region where the projection of the intersection curve of the two considered quadrics lives is bounded by a finite set of conic arcs since any $\Delta_{\mathcal{E}_i}(x,y)$ is a polynomial in $\mathbb{R}[x,y]$ of total degree equal to 2.

In [2], the curve in $\mathbb{R}^2$ defined by $\mathbf{S}_0(x,y) = 0$ is called the cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ and the curve in $\mathbb{R}^2$ defined by $\Delta_{\mathcal{E}_i}(x,y) = 0$ the silhouette of $\mathcal{E}_i$. We modify slightly the cutcurve definition to make it more suitable for our purposes.

Definition 4.2. *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two quadrics in $\mathbb{R}^3$ defined by $f(x,y,z)=0$ and $g(x,y,z)=0$ respectively. The cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ is the set*

$$\left\{(x,y)\in\mathbb{R}^2 : \mathbf{S}_0(x,y)=0, \Delta_{\mathcal{E}_1}(x,y)\geq 0, \Delta_{\mathcal{E}_2}(x,y)\geq 0,\right\}$$

According to Theorem 4.1 the cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ is equal to the projection of $\mathcal{E}_1\cap\mathcal{E}_2$ onto the xy plane, $\Pi(\mathcal{E}_1\cap\mathcal{E}_2)$. The cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ can be a curve, part of a curve (i.e. a semialgebraic set) or even a single point, but always a semialgebraic set.

Example 4.3. Let f and g be the polynomials

$$f(x,y,z)=z^2+x^2+y^2-7 \qquad g(x,y,z)=z^2-x^2+xy+2x-y^2$$

defining the two quadrics $\mathcal{G}_1$ and $\mathcal{G}_2$ whose intersection curve is to be computed. In this case we have:

$$\mathbf{S}_0(x,y)=(-2x^2+xy-2y^2+2x+7)^2,$$
$$\Delta_{\mathcal{G}_1}(x,y)=-4x^2-4y^2+28, \qquad \Delta_{\mathcal{G}_2}(x,y)=4x^2-4xy+4y^2-8x$$

and

$$\mathcal{A}_{\mathcal{G}_1,\mathcal{G}_2}=\{(x,y)\in\mathbb{R}^2\colon\ -x^2-y^2+7\geq 0,\ x^2-xy+y^2-2x\geq 0\}.$$

In this case the curve in $\mathbb{R}^2$ defined by $\mathbf{S}_0(x,y)=0$ is not contained completely in $\mathcal{A}_{\mathcal{G}_1,\mathcal{G}_2}$: the projection of $\mathcal{G}_1\cap\mathcal{G}_2$ is equal to the portion of the ellipse $-2x^2+xy-2y^2+2x+7=0$ inside the circle $x^2+y^2\leq 7$ (see Figure 2).

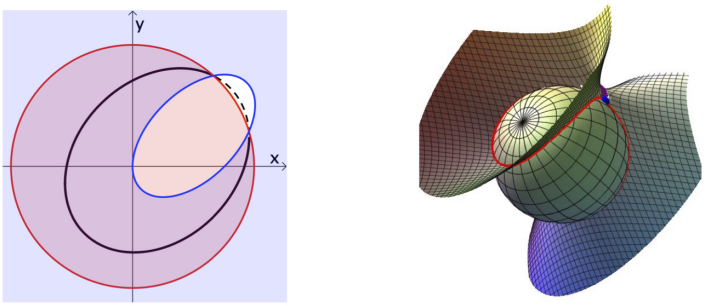


Figure 2: Example where the cutcurve (black) is not the whole curve $\mathbf{S}_0(x,y)=0$.

We will pay also special attention to the intersection points between the cutcurve and the silhouette curves. These points will include those points where the cutcurve "stops" and where the cutcurve "starts" again (unless both curves are tangent). Figure 2 shows one concrete example of this situation.

For quadrics whose defining equations have a different structure than the shown in (2), similar formulae can be derived for the cutcurve, i.e. for the projection of their intersection. For example, if

$$f(x,y,z)=z^2+p_1(x,y)z+p_0(x,y) \qquad g(x,y,z)=q_1(x,y)z+q_0(x,y) \tag{3}$$

then $\Pi(\mathcal{E}_1 \cap \mathcal{E}_2)$ is given by

$$\left\{(x,y) \in \mathbb{R}^2 : \begin{array}{l} \mathbf{S}_0(x,y) = 0 \\ p_1(x,y)^2 - 4p_0(x,y) \geq 0 \\ q_1(x,y) \neq 0 \end{array}\right\} \bigcup \left\{(x,y) \in \mathbb{R}^2 : \begin{array}{l} \mathbf{S}_0(x,y) = 0 \\ p_1(x,y)^2 - 4p_0(x,y) \geq 0 \\ q_1(x,y) = 0,\ q_0(x,y) = 0 \end{array}\right\}.$$

5. Lifting to $\mathbb{R}^3$ the cutcurve in $\mathbb{R}^2$

In this section we study the lifting to $\mathbb{R}^3$ of the cutcurve of the two quadrics whose intersection is to be computed. We will pay special attention to the singular points of the cutcurve since they are the points where more complicated situations we must deal with when lifting the cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ to $\mathcal{E}_1 \cap \mathcal{E}_2$. This means that, first, we must be able of isolating them in order to achieve its lifting in the easiest and most efficient possible way.

5.1. Determining the singular points of the cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$

Let f and g be two polynomials in $\mathbb{R}[x,y,z]$ defined by:

$$f(x,y,z) = z^2 + p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = z^2 + q_1(x,y)z + q_0(x,y)$$

with $\deg(p_1) \leq 1$, $\deg(p_0) \leq 2$, $\deg(q_1) \leq 1$, $\deg(q_0) \leq 2$. We restrict our attention to this case because this is the most complicated situation we must deal with: for those quadrics whose equations have a different (and simpler) structure, the singular points of the cutcurve are easier to compute since their lifting will be given automatically by one of the two equations (being of degree in z smaller than or equal to 1).

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be the corresponding quadrics defined by f and g and $\mathbf{S}_0(x,y) = 0$ the implicit equation of its cutcurve. Next two theorems will help to determine easily the singular points of the cutcurve. The first one shows that those points in $\mathbf{S}_0(x_0,y_0) = 0$ and in the line $p_1(x,y) = q_1(x,y)$ are always singular points of the cutcurve.

In what follows we will use very often the following (easy to prove) lemma. We thank the reviewer for pointing out the importance of this fact that we were using without isolating it properly.

Lemma 5.1. *Let (x_0,y_0) be such that $\mathbf{S}_0(x_0,y_0) = 0$, z_0 such that:*

$$f(x_0,y_0,z_0) = z_0^2 + p_1(x_0,y_0)z_0 + p_0(x_0,y_0) = 0 \quad g(x_0,y_0,z_0) = z_0^2 + q_1(x_0,y_0)z_0 + q_0(x_0,y_0) = 0$$

and $p_1(x_0,y_0) = q_1(x_0,y_0)$. Then $p_0(x_0,y_0) = q_0(x_0,y_0)$.

Next lemma gives a very convenient description of the partial derivatives of $\mathbf{S_0}(x,y)$.

Lemma 5.2. *If $\mathbf{S_0}(x_0,y_0) = 0$ then there exists z_0 such that $f(x_0,y_0,z_0) = 0$, $g(x_0,y_0,z_0) = 0$*

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0) = \begin{vmatrix} 1 & p_1(x_0,y_0) & p_0(x_0,y_0) & z_0 f_x(x_0,y_0,z_0) \\ 0 & 1 & p_1(x_0,y_0) & f_x(x_0,y_0,z_0) \\ 1 & q_1(x_0,y_0) & q_0(x_0,y_0) & z_0 g_x(x_0,y_0,z_0) \\ 0 & 1 & q_1(x_0,y_0) & g_x(x_0,y_0,z_0) \end{vmatrix},$$

$$\frac{\partial \mathbf{S}_0}{\partial y}(x_0,y_0)=\begin{vmatrix} 1 & p_1(x_0,y_0) & p_0(x_0,y_0) & z_0 f_y(x_0,y_0,z_0) \\ 0 & 1 & p_1(x_0,y_0) & f_y(x_0,y_0,z_0) \\ 1 & q_1(x_0,y_0) & q_0(x_0,y_0) & z_0 g_y(x_0,y_0,z_0) \\ 0 & 1 & q_1(x_0,y_0) & g_y(x_0,y_0,z_0) \end{vmatrix}.$$

Proof. The existence of z_0 comes from the fact that $\mathbf{S}_0(x,y)$ is the resultant of $f(x,y,z)$ and $g(x,y,z)$ with respect to z. By performing elementaty operations on the columns of the matrix giving $\mathbf{S}_0(x,y)$ we obtain:

$$\mathbf{S}_0(x,y)=\mathbf{Resultant}(f,g;z)=\begin{vmatrix} 1 & p_1 & p_0 & 0 \\ 0 & 1 & p_1 & p_0 \\ 1 & q_1 & q_0 & 0 \\ 0 & 1 & q_1 & q_0 \end{vmatrix}=\begin{vmatrix} 1 & p_1 & p_0 & zf \\ 0 & 1 & p_1 & f \\ 1 & q_1 & q_0 & zg \\ 0 & 1 & q_1 & g \end{vmatrix}.$$

And applying the rule for calculating the derivative of a determinant we have:

$$\frac{\partial \mathbf{S}_0}{\partial x}=\begin{vmatrix} 0 & p_1 & p_0 & zf \\ 0 & 1 & p_1 & f \\ 0 & q_1 & q_0 & zg \\ 0 & 1 & q_1 & g \end{vmatrix}+\begin{vmatrix} 1 & p_{1x} & p_0 & zf \\ 0 & 0 & p_1 & f \\ 1 & q_{1x} & q_0 & zg \\ 0 & 0 & q_1 & g \end{vmatrix}+\begin{vmatrix} 1 & p_1 & p_{0x} & zf \\ 0 & 1 & p_{1x} & f \\ 1 & q_1 & q_{0x} & zg \\ 0 & 1 & q_{1x} & g \end{vmatrix}+\begin{vmatrix} 1 & p_1 & p_0 & zf_x \\ 0 & 1 & p_1 & f_x \\ 1 & q_1 & q_0 & zg_x \\ 0 & 1 & q_1 & g_x \end{vmatrix}.$$

Replacing (x,y,z) by (x_0,y_0,z_0) in this equation produces:

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0)=\begin{vmatrix} 1 & p_1(x_0,y_0) & p_0(x_0,y_0) & z_0 f_x(x_0,y_0,z_0) \\ 0 & 1 & p_1(x_0,y_0) & f_x(x_0,y_0,z_0) \\ 1 & q_1(x_0,y_0) & q_0(x_0,y_0) & z_0 g_x(x_0,y_0,z_0) \\ 0 & 1 & q_1(x_0,y_0) & g_x(x_0,y_0,z_0) \end{vmatrix}.$$

Replacing x by y in the previous argument we prove also the remaining equality. □

Theorem 5.3. *If* $\mathbf{S}_0(x_0,y_0)=0$ *and* $p_1(x_0,y_0)=q_1(x_0,y_0)$ *then*

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0)=0 \qquad \frac{\partial \mathbf{S}_0}{\partial y}(x_0,y_0)=0.$$

Proof. Since $p_1(x_0,y_0)=q_1(x_0,y_0)$, by using Lemma 5.1 and Lemma 5.2, we get

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0)=\begin{vmatrix} 1 & p_1(x_0,y_0) & p_0(x_0,y_0) & z_0 f_x(x_0,y_0,z_0) \\ 0 & 1 & p_1(x_0,y_0) & f_x(x_0,y_0,z_0) \\ 1 & p_1(x_0,y_0) & p_0(x_0,y_0) & z_0 g_x(x_0,y_0,z_0) \\ 0 & 1 & p_1(x_0,y_0) & g_x(x_0,y_0,z_0) \end{vmatrix}=$$

$$=\begin{vmatrix} 1 & p_1(x_0,y_0) & p_0(x_0,y_0) & z_0 f_x(x_0,y_0,z_0) \\ 0 & 1 & p_1(x_0,y_0) & f_x(x_0,y_0,z_0) \\ 0 & 0 & 0 & z_0\left(f_x(x_0,y_0,z_0)-g_x(x_0,y_0,z_0)\right) \\ 0 & 0 & 0 & f_x(x_0,y_0,z_0)-g_x(x_0,y_0,z_0) \end{vmatrix}=0.$$

Replacing x by y in the previous argument we prove also that $\dfrac{\partial \mathbf{S}_0}{\partial y}(x_0,y_0)=0$. □

Theorem 5.3 and Lemma 5.1 imply that the singular points of the cutcurve in the line $p_1(x,y) = q_1(x,y)$ are also in the conic $p_0(x,y) = q_0(x,y)$.

Next theorem and corollary, together with what we have just proven, allows to conclude that the singular points of the cutcurve come from two different sources:

- either they are in the line $p_1(x,y) = q_1(x,y)$ (and in the conic $p_0(x,y) = q_0(x,y)$), or
- they are not in the line $p_1(x,y) = q_1(x,y)$ and they are the projection of a tangential intersection point between $\mathcal{E}_1$ and $\mathcal{E}_2$.

Note that not all tangential intersection points between $\mathcal{E}_1$ and $\mathcal{E}_2$ come from the second option.

Theorem 5.4. *If (x_0, y_0) is a singular point of the cutcurve and $p_1(x_0, y_0) \neq q_1(x_0, y_0)$ then*

$$\frac{\partial f}{\partial x}(x_0,y_0,z_0)\frac{\partial g}{\partial z}(x_0,y_0,z_0) - \frac{\partial f}{\partial z}(x_0,y_0,z_0)\frac{\partial g}{\partial x}(x_0,y_0,z_0) = 0$$

$$\frac{\partial f}{\partial y}(x_0,y_0,z_0)\frac{\partial g}{\partial z}(x_0,y_0,z_0) - \frac{\partial f}{\partial z}(x_0,y_0,z_0)\frac{\partial g}{\partial y}(x_0,y_0,z_0) = 0$$

where

$$z_0 = -\frac{q_0(x_0,y_0) - p_0(x_0,y_0)}{q_1(x_0,y_0) - p_1(x_0,y_0)}.$$

Proof. It is enough to check that:

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0) = (q_1(x_0,y_0) - p_1(x_0,y_0)) \cdot \begin{vmatrix} f_x(x_0,y_0,z_0) & f_z(x_0,y_0,z_0) \\ g_x(x_0,y_0,z_0) & g_z(x_0,y_0,z_0) \end{vmatrix}$$

and

$$\frac{\partial \mathbf{S}_0}{\partial y}(x_0,y_0) = (q_1(x_0,y_0) - p_1(x_0,y_0)) \cdot \begin{vmatrix} f_y(x_0,y_0,z_0) & f_z(x_0,y_0,z_0) \\ g_y(x_0,y_0,z_0) & g_z(x_0,y_0,z_0) \end{vmatrix}.$$

We only prove the first equality. The proof of the second one is the same but replacing x by y.

Let (x_0, y_0) be such that $\mathbf{S}_0(x_0, y_0) = 0$ and z_0 such that:

$$f(x_0,y_0,z_0) = z_0^2 + p_1(x_0,y_0)z_0 + p_0(x_0,y_0) \qquad g(x_0,y_0,z_0) = z_0^2 + q_1(x_0,y_0)z_0 + q_0(x_0,y_0) = 0$$

By using Lemma 5.2 we have:

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0) = \begin{vmatrix} 1 & p_1(x_0,y_0) & p_0(x_0,y_0) & z_0 f_x(x_0,y_0,z_0) \\ 0 & 1 & p_1(x_0,y_0) & f_x(x_0,y_0,z_0) \\ 1 & q_1(x_0,y_0) & q_0(x_0,y_0) & z_0 g_x(x_0,y_0,z_0) \\ 0 & 1 & q_1(x_0,y_0) & g_x(x_0,y_0,z_0) \end{vmatrix} = \begin{vmatrix} 1 & p_1 & p_0 & z_0 f_x \\ 0 & 1 & p_1 & f_x \\ 1 & q_1 & q_0 & z_0 g_x \\ 0 & 1 & p_1 & g_x \end{vmatrix}.$$

Then

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0)=\begin{vmatrix} 1 & p_1 & p_0 & z_0 f_x \\ 0 & 1 & p_1 & f_x \\ 0 & q_1-p_1 & q_0-p_0 & z_0(g_x-f_x) \\ 0 & 0 & q_1-p_1 & g_x-f_x \end{vmatrix}=\begin{vmatrix} 1 & p_1 & f_x \\ q_1-p_1 & q_0-p_0 & z_0(g_x-f_x) \\ 0 & q_1-p_1 & g_x-f_x \end{vmatrix}=$$

$$=\begin{vmatrix} q_0-p_0 & z_0(g_x-f_x) \\ q_1-p_1 & g_x-f_x \end{vmatrix}-(q_1-p_1)\begin{vmatrix} p_1 & f_x \\ q_1-p_1 & g_x-f_x \end{vmatrix}.$$

Since $p_1(x_0,y_0)\neq q_1(x_0,y_0)$ then

$$-z_0(q_1(x_0,y_0)-p_1(x_0,y_0))=q_0(x_0,y_0)-p_0(x_0,y_0)$$

and

$$\begin{aligned}\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0) &= \begin{vmatrix} -z_0(q_1-p_1) & z_0(g_x-f_x) \\ q_1-p_1 & g_x-f_x \end{vmatrix}-(q_1-p_1)\begin{vmatrix} p_1 & f_x \\ q_1 & g_x \end{vmatrix}= \\ &= \begin{vmatrix} -z_0 & z_0 \\ 1 & 1 \end{vmatrix}(q_1-p_1)(g_x-f_x)-(q_1-p_1)\begin{vmatrix} p_1 & f_x \\ q_1 & g_x \end{vmatrix}= \\ &= (q_1-p_1)\left(-2z_0(g_x-f_x)-\begin{vmatrix} p_1 & f_x \\ q_1 & g_x \end{vmatrix}\right).\end{aligned}$$

Since $f_z(x_0,y_0,z_0)=2z_0+p_1(x_0,y_0)$ and $g_z(x_0,y_0,z_0)=2z_0+q_1(x_0,y_0)$ we have:

$$p_1=f_z-2z_0 \qquad q_1=g_z-2z_0$$

and

$$\begin{aligned}\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0) &= (q_1-p_1)\left(-2z_0(g_x-f_x)-\begin{vmatrix} f_z-2z_0 & f_x \\ g_z-2z_0 & g_x \end{vmatrix}\right)= \\ &= (q_1-p_1)(-2z_0g_x+2z_0f_x-g_x(f_z-2z_0)+f_x(g_z-2z_0))= \\ &= (q_1-p_1)(f_xg_z-f_zg_x).\end{aligned}$$

Finally we have

$$\frac{\partial \mathbf{S}_0}{\partial x}(x_0,y_0)=(q_1(x_0,y_0)-p_1(x_0,y_0))\cdot\begin{vmatrix} f_x(x_0,y_0,z_0) & f_z(x_0,y_0,z_0) \\ g_x(x_0,y_0,z_0) & g_z(x_0,y_0,z_0) \end{vmatrix}.$$

Since (x_0,y_0) is a singular point of the curve and $p_1(x_0,y_0)\neq q_1(x_0,y_0)$ we conclude that

$$\begin{vmatrix} f_x(x_0,y_0,z_0) & f_z(x_0,y_0,z_0) \\ g_x(x_0,y_0,z_0) & g_z(x_0,y_0,z_0) \end{vmatrix}=0$$

as desired. □

Corollary 5.5. *If (x_0, y_0) is a singular point of the cutcurve, $p_1(x_0, y_0) \neq q_1(x_0, y_0)$ and*

$$z_0 = -\frac{q_0(x_0, y_0) - p_0(x_0, y_0)}{q_1(x_0, y_0) - p_1(x_0, y_0)}$$

then $f(x_0, y_0, z_0) = 0$, $g(x_0, y_0, z_0) = 0$ and the quadrics defined by f and g have the same tangent plane at (x_0, y_0, z_0).

Proof. The tangent planes to $f = 0$ and $g = 0$ at (x_0, y_0, z_0) are given by:

$$\frac{\partial f}{\partial x}(x_0, y_0, z_0)(x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0, z_0)(y - y_0) + \frac{\partial f}{\partial z}(x_0, y_0, z_0)(z - z_0) = 0$$

$$\frac{\partial g}{\partial x}(x_0, y_0, z_0)(x - x_0) + \frac{\partial g}{\partial y}(x_0, y_0, z_0)(y - y_0) + \frac{\partial g}{\partial z}(x_0, y_0, z_0)(z - z_0) = 0$$

Previous theorem implies that the quadrics $f = 0$ and $g = 0$ have the same tangent plane at (x_0, y_0, z_0). □

Next we show how to compute easily the singular points of the cutcurve when they are in the line $p_1(x, y) = q_1(x, y)$. Next lemma will connect $\mathbf{S}_0(x, y)$ with the line $p_1(x, y) - q_1(x, y) = 0$ and with $\Delta_{\mathcal{E}_1}(x, y)$ and $\Delta_{\mathcal{E}_2}(x, y)$ (the discriminants of $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively).

Lemma 5.6.

$$\mathbf{S}_0(x, y) = \frac{1}{16}\Big[(p_1 - q_1)^4 + (\Delta_{\mathcal{E}_1} - \Delta_{\mathcal{E}_2})^2 - 2(p_1 - q_1)^2(\Delta_{\mathcal{E}_1} + \Delta_{\mathcal{E}_2})\Big].$$

Proof. Since $\Delta_{\mathcal{E}_1} = p_1^2 - 4p_0$ and $\Delta_{\mathcal{E}_2} = q_1^2 - 4q_0$, we have

$$\frac{1}{16}\Big[(p_1 - q_1)^4 + (\Delta_{\mathcal{E}_1} - \Delta_{\mathcal{E}_2})^2 - 2(p_1 - q_1)^2(\Delta_{\mathcal{E}_1} + \Delta_{\mathcal{E}_2}))\Big] =$$

$= {p_0}^2 - 2p_0q_0 + {q_0}^2 + {p_1}^2q_0 + {q_1}^2p_0 - p_1q_0q_1 - p_0p_1q_1 = (p_0 - q_0)^2 - (p_1 - q_1)(p_0q_1 - q_0p_1) = \mathbf{S}_0(x, y)$
as desired. □

As a consequence of the equality in the previous lemma, we conclude that we can always compute the intersection points between the cutcurve and the line $p_1(x, y) - q_1(x, y) = 0$ by just solving a degree two equation.

Corollary 5.7. *The intersection points between the cutcurve and the line $p_1(x, y) - q_1(x, y) = 0$ are the same than the intersection points between the curve*

$$\Delta_{\mathcal{E}_1}(x, y) - \Delta_{\mathcal{E}_2}(x, y) = 0$$

and the line $p_1(x, y) - q_1(x, y) = 0$.

In some cases the line $p_1(x,y) - q_1(x,y) = 0$ (or part of it) is contained completely in the cutcurve: this can be checked directly by performing the corresponding substitution in $\mathbf{S}_0(x,y)$ or by using the previous corollary. In the particular case when $p_1(x,y) \equiv q_1(x,y)$ we have that

$$\mathbf{S}_0(x,y) = (p_0(x,y) - q_0(x,y))^2$$

and, as expected, all points in the cutcurve are singular. But this implies that the cutcurve is a conic ($p_0(x,y) - q_0(x,y) = 0$) and all further computations (including the lifting) are greatly simplified.

Next proposition shows that the "vertices" of the region $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$ belonging to $\mathbf{S}_0(x,y) = 0$ are always singular points of this curve.

Proposition 5.8. *If (α,β) is a point such that*

$$\mathbf{S}_0(\alpha,\beta) = 0, \Delta_{\mathcal{E}_1}(\alpha,\beta) = 0, \Delta_{\mathcal{E}_2}(\alpha,\beta) = 0$$

then $p_1(\alpha,\beta) = q_1(\alpha,\beta)$ and (α,β) is a singular point of the cutcurve.

Proof. From Lemma 5.6 we have

$$\mathbf{S}_0(\alpha,\beta) = 0, \Delta_{\mathcal{E}_1}(\alpha,\beta) = 0, \Delta_{\mathcal{E}_2}(\alpha,\beta) = 0 \Longleftrightarrow \left(\frac{(p_1 - q_1)^2}{4}\right)^2 = 0 \Longleftrightarrow p_1 = q_1$$

as desired. □

Finally, we show how to determine those singular points of the cutcurve not belonging to the line $p_1(x,y) = q_1(x,y)$. These points, according to Corollary 5.5, come from tangential intersection points of the two considered quadrics and they are very easily lifted but more difficult to determine than those in the line $p_1(x,y) = q_1(x,y)$. To determine these points we have to solve the system of equations

$$\mathbf{S}_0(x,y) = 0, \quad \frac{\partial \mathbf{S}_0}{\partial x}(x,y) = 0 \quad \frac{\partial \mathbf{S}_0}{\partial y}(x,y) = 0, \quad p_1(x,y) \neq q_1(x,y)$$

inside $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$. In order to solve this system of equations we need the polynomial containing the x-coordinates of the singular points of $\mathbf{S}_0(x,y)$ in the line $p_1(x,y) - q_1(x,y) = 0$.

Definition 5.9. *When $\deg(p_1(x,y) - q_1(x,y), y) = 1$, the squarefree part of the polynomial obtained after replacing y in $\Delta_{\mathcal{E}_1}(x,y) - \Delta_{\mathcal{E}_2}(x,y)$ by the value obtained by solving $p_1(x,y) - q_1(x,y) = 0$ in terms of y will be denoted by $\tau_{\mathcal{E}_1,\mathcal{E}_2}(x)$. Otherwise $\tau_{\mathcal{E}_1,\mathcal{E}_2}(x) = p_1(x,y) - q_1(x,y)$.*

Next theorem shows how to use subresultants to determine the tangential intersection points of the two considered quadrics whose projection is outside the line $p_1(x_0,y_0) - q_1(x_0,y_0) = 0$.

Theorem 5.10. *Let (x_0,y_0) be a singular point of the cutcurve such that $p_1(x_0,y_0) \neq q_1(x_0,y_0)$. If*

- $U_0(x)$ *is the squarefree part of* $\mathbf{Sres}_0\left(\mathbf{S}_0, \dfrac{\partial \mathbf{S}_0}{\partial x}; y\right)$.
- $U_1(x,y) = \mathbf{Sres}_1\left(\mathbf{S}_0, \dfrac{\partial \mathbf{S}_0}{\partial x}; y\right) = U_{11}(x)y + U_{10}(x)$.
- $V_0(x)$ *is the squarefree part of* $\mathbf{Sres}_0\left(\mathbf{S}_0, \dfrac{\partial \mathbf{S}_0}{\partial y}; y\right)$.
- $V_1(x,y) = \mathbf{Sres}_1\left(\mathbf{S}_0, \dfrac{\partial \mathbf{S}_0}{\partial y}; y\right) = V_{11}(x)y + V_{10}(x)$.
- $W(x)$ *is the squarefree part of* $U_{10}(x)V_{11}(x) - V_{10}(x)U_{11}(x)$.

then x_0 *is a real root of the polynomial*

$$\Omega_{\mathcal{E}_1,\mathcal{E}_2}(x) = \frac{\gcd(W(x), U_0(x), V_0(x))}{\gcd(W(x), \tau_{\mathcal{E}_1,\mathcal{E}_2}(x))}$$

and

$$y_0 = -U_{10}(x_0)/U_{11}(x_0) \quad \textit{and/or} \quad y_0 = -V_{10}(x_0)/V_{11}(x_0).$$

Proof. Since (x_0, y_0) is a singular point of the cutcurve, we have the following equalities:

$$U_0(x_0) = 0, U_1(x_0, y_0) = 0, V_0(x_0) = 0, V_1(x_0, y_0) = 0.$$

By using $U_1(x_0, y_0) = 0$ and $V_1(x_0, y_0) = 0$ we have

$$U_{10}(x_0)V_{11}(x_0) - V_{10}(x_0)U_{11}(x_0) = 0$$

and that x_0 is a real root of $\gcd(U_{10}(x)V_{11}(x) - V_{10}(x)U_{11}(x), U_0(x), V_0(x))$ and a real root of $\gcd(W(x), U_0(x), V_0(x))$. Since $p_1(x_0, y_0) \neq q_1(x_0, y_0)$, according to Corollary 5.7 and Definition 5.9, we have $\tau_{\mathcal{E}_1,\mathcal{E}_2}(x_0) \neq 0$ and this allows to conclude that $\Omega_{\mathcal{E}_1,\mathcal{E}_2}(x_0) = 0$ as desired. Note that, under these conditions, $U_{11}(x_0) \neq 0$ or $V_{11}(x_0) \neq 0$. If this is not the case then $(y - y_0)^2$ is a common factor of $\mathbf{S}_0(x_0, y)$, $\dfrac{\partial \mathbf{S}_0}{\partial x}(x_0, y)$ and $\dfrac{\partial \mathbf{S}_0}{\partial y}(x_0, y)$ but this is not possible since the degree in y of $\mathbf{S}_0(x_0, y)$ is bounded by 4. □

5.2. Determining the intersection points of the cutcurve with the silhouette curves

Next propositions provide the way to determine in a simpler way the points in the cutcurve which belong to each silhouette curve.

Proposition 5.11. *If* $\mathbf{S}_0(\alpha, \beta) = 0$ *then*

$$\Delta_{\mathcal{E}_1}(\alpha,\beta) = 0 \overset{(I)}{\Leftrightarrow} \Delta_{\mathcal{E}_2}(\alpha,\beta) = (p_1(\alpha,\beta) - q_1(\alpha,\beta))^2 \overset{(II)}{\Leftrightarrow} 2\left(q_0(\alpha,\beta) + p_0(\alpha,\beta)\right) = p_1(\alpha,\beta)q_1(\alpha,\beta)$$

$$\Delta_{\mathcal{E}_2}(\alpha,\beta) = 0 \overset{(I)}{\Leftrightarrow} \Delta_{\mathcal{E}_1}(\alpha,\beta) = (p_1(\alpha,\beta) - q_1(\alpha,\beta))^2 \overset{(II)}{\Leftrightarrow} 2\left(q_0(\alpha,\beta) + p_0(\alpha,\beta)\right) = p_1(\alpha,\beta)q_1(\alpha,\beta)$$

Proof. Using Lemma 5.6, we have

$$\Delta_{\mathcal{E}_2}^2 - 2(p_1 - q_1)^2\Delta_{\mathcal{E}_2} + (p_1 - q_1)^4 = 0 \quad \Longleftrightarrow \quad \Delta_{\mathcal{E}_2} = (p_1 - q_1)^2$$

and we get the first equivalence. Using this one and $\Delta_{\mathcal{E}_2} = {q_1}^2 - 4q_0$, we conclude $p_1q_1 = 2(p_0 + q_0)$. The second one is similar. □

As a consequence we have that solving each system

$$\mathbf{S}_0(x, y) = 0, \quad \Delta_{\mathcal{E}_i}(x, y) = 0$$

is the same than solving the simpler system

$$2(p_0(x, y) + q_0(x, y)) = p_1(x, y)q_1(x, y), \quad \Delta_{\mathcal{E}_i}(x, y) = 0.$$

An easy consequence of the previous proposition is the fact that the cutcurve and the silhouette curves have no common points outside the line $p_1(x, y) = q_1(x, y)$.

Corollary 5.12. *The system*

$$\mathbf{S}_0(x, y) = 0,\ p_1(x, y) \neq q_1(x, y),\ \Delta_{\mathcal{E}_1}(x, y) = 0,\ \Delta_{\mathcal{E}_2}(x, y) = 0$$

has no real solutions.

5.3. Lifting the points of the cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$

Next it is shown how to perform the lifting of the points of the cutcurve.

Theorem 5.13. *If (α, β) is a point in the cutcurve such that $q_1(\alpha, \beta) \neq p_1(\alpha, \beta)$ then the z-coordinate of the point in the intersection curve is given by:*

$$z = \frac{p_0(\alpha, \beta) - q_0(\alpha, \beta)}{q_1(\alpha, \beta) - p_1(\alpha, \beta)}.$$

If (α, β) is a point of the cutcurve such that $q_1(\alpha, \beta) = p_1(\alpha, \beta)$ then the lifting of this singular point can be made by using $g(\alpha, \beta, z) = 0$ or $f(\alpha, \beta, z) = 0$.

Proof. As we have seen

$$\mathbf{S}_1(x, y; z) = \mathbf{Sres}_1(f, g; z) = (q_1(x, y) - p_1(x, y))z + (q_0(x, y) - p_0(x, y)).$$

If $(\alpha, \beta) \in \Pi(\mathcal{E}_1 \cap \mathcal{E}_1)$ then

$$\mathbf{S}_1(\alpha, \beta; z) = 0 \Leftrightarrow z = \frac{p_0(\alpha, \beta) - q_0(\alpha, \beta)}{q_1(\alpha, \beta) - p_1(\alpha, \beta)}$$

as desired. □

6. Experimentation

This section will present the experimentation performed together with some examples showing how to compute the intersection curve of two quadrics by using the results presented in the previous sections. In all examples two polynomials f and g define two quadrics $\mathcal{E}_1$ and $\mathcal{E}_2$ respectively. In these examples

$$\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2} = \{(x,y) \in \mathbb{R}^2 \colon \Delta_{\mathcal{E}_1}(x,y) \geq 0,\ \Delta_{\mathcal{E}_2}(x,y) \geq 0\}$$

defines the region where the cutcurve, defined by $\mathbf{S}_0(x,y) = 0$, lives. Typically the lifting of the cutcurve will be made by using $\mathbf{S}_1(x,y;z)$. When one of the singular points of the cutcurve can not be lifted by using $\mathbf{S}_1(x,y;z)$, we will use $g(x,y,z)$ or $f(x,y,z)$ for that purpose (for those singular points outside the line $p_1(x,y) = q_1(x,y)$ we can use $\mathbf{S}_1(x,y;z)$ for performing the lifting too).

The way of proceeding will be always the following one:

1. Compute the implicit equation for the cutcurve $\mathbf{S}_0(x,y)$.
2. Compute the description of the region $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$ where the cutcurve lives. These two polynomial inequalities will be used to determine which points on the projection we are going to consider.
3. Compute the singular points of the cutcurve $\mathbf{S}_0(x,y)$ in the line $p_1(x,y) = q_1(x,y)$: just solving a degree two univariate equation by Corollary 5.7. Their lifting is made by using f or g. This step requires to solve three univariate degree two equations
4. Compute the singular points of the cutcurve $\mathbf{S}_0(x,y)$ outside the line $p_1(x,y) = q_1(x,y)$: points coming from the projection of a tangential intersection whose lifting is made by using $\mathbf{S}_1(x,y;z)$ (computations guided by Theorem 5.10).
5. Compute the regular points of the cutcurve which are in the silhouette curves by using the Proposition 5.11 and their lifting by using $\mathbf{S}_1(x,y;z)$. These points will lead the discretisation of the cutcurve or the computation of the intervals where the parameterisation determined can be evaluated (since it might involve radicals) since they contain the points where the cutcurve "starts" and "stops" (if any). This step requiere to intersect two couples of conics.
6. Compute the branches of the cutcurve (always inside $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$) by closed formulae involving radicals or discretising them (and their lifting by using $\mathbf{S}_1(x,y;z)$). Either solving degree two equations of computing numerically the solutions of the, at most degree four, univariate equation $\mathbf{S}_0(\alpha,y)$ for several values of α.

In the next four examples, the lifting of the cutcurve will be made after discretising the regular branches of the cutcurve (always inside $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$). All examples include computations made with Maple. In the (right) next figures the cutcurve is drawn in black, silhouette curves are drawn in blue (and in the last example in red and blue) and the line $p_1(x,y) = q_1(x,y)$ will be always dotted. The admissible region is always the darker region.

Example 6.1. Let f and g be the polynomials

$$f(x,y,z) = z^2 + (-6x - y - 1)z - 9x^2 - 3xy + 4y^2 + 9x - 9y - 2$$

$$g(x,y,z) = z^2 - 2z + x^2 - 3y^2 + 9x - 2y + 6$$

defining two ellipsoids $\mathcal{E}_1$ and $\mathcal{E}_2$, whose intersection curve is to be computed. In this case we have:

$$\mathbf{S}_0(x,y) = 136x^4 + 72x^3y - 238x^2y^2 - 78xy^3 + 46y^4 + 432x^3 + 230x^2y - 15xy^2 - 108y^3$$
$$+249x^2 + 204xy - 28y^2 + 33x + 100y + 54$$

$$\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2} = \left\{ (x,y) \in \mathbb{R}^2 : \begin{array}{l} 72x^2 + 24xy - 15y^2 - 24x + 38y + 9 \geq 0, \\ -4x^2 + 12y^2 - 36x + 8y - 20 \geq 0 \end{array} \right\}.$$

From Corollary 5.7, the singular points of the cutcurve in the line $p_1(x,y) = q_1(x,y)$ are determined by solving:

$$76x^2 + 24xy - 27y^2 + 12x + 30y + 29 = 0 \wedge 6x + y + 1 = 2.$$

In this way we get the singular points

$$A = \left(\frac{9}{104} + \frac{\sqrt{10345}}{520}, \frac{25}{52} - \frac{3\sqrt{10355}}{260} \right) \qquad B = \left(\frac{9}{104} - \frac{\sqrt{10345}}{520}, \frac{25}{52} + \frac{3\sqrt{10355}}{260} \right).$$

The first one, A, is an isolated point of the cutcurve but outside $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$ and will not be considered. The second one, B, will be lifted by using $f(x,y,z) = 0$ or $g(x,y,z) = 0$ providing two different points in the intersection curve.

Point $C = (-0.5989698028, -0.6502822952)$ is common to $\mathbf{S}_0(x,y) = 0$ and $\Delta_{\mathcal{E}_1}(x,y) = 0$ and, by using Proposition 5.11, was determined by solving:

$$-16x^2 - 6xy + 2y^2 + 24x - 24y + 6 = 0 \wedge -4x^2 + 12y^2 - 36x + 8y - 20 = 0.$$

Points $D = (-2.336955328, -6.163216205)$ and $E = (21.765280490, -32.199082657)$ are common to $\mathbf{S}_0(x,y) = 0$ and $\Delta_{\mathcal{E}_1}(x,y) = 0$ and, from Proposition 5.11, were determined by solving:

$$-16x^2 - 6xy + 2y^2 + 24x - 24y + 6 = 0 \wedge 72x^2 + 24xy - 15y^2 - 24x + 38y + 9 = 0$$

The lifting of $\Pi(\mathcal{E}_1 \cap \mathcal{E}_2)$, outside the singular points of the cutcurve, will be made by using $\mathbf{S}_1(x,y;z)$:

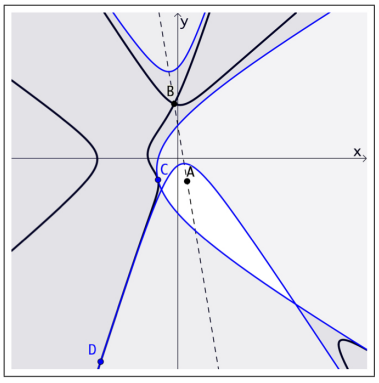
$$z = \frac{10x^2 + 3xy - 7y^2 + 7y + 8}{-6x - y + 1}.$$

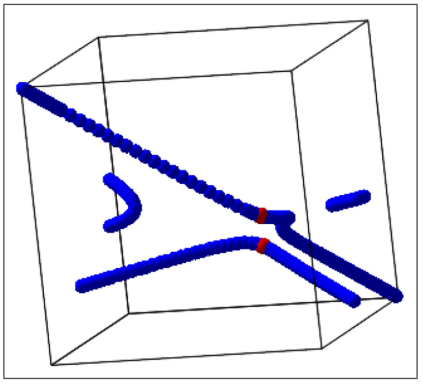
Example 6.2. Let f and g be the polynomials

$$f(x,y,z) = z^2 + xz + y \qquad g(x,y,z) = z^2 + yz + x$$

defining two hyperbolic paraboloids, $\mathcal{E}_1$ and $\mathcal{E}_2$, whose intersection curve is to be computed. In this case we have

$$\mathbf{S}_0(x,y) = (x-y)^2(x+y+1)$$





and

$$\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2} = \{(x,y) \in \mathbb{R}^2 \colon x^2 - 4y \geq 0,\ y^2 - 4x \geq 0\}.$$

All the points in the line $p_1(x,y) = q_1(x,y)$ belong to the cutcurve and they are singular: this can be checked by solving (according to Corollary 5.7):

$$x^2 - y^2 + 4x - 4y = 0 \wedge x - y = 0.$$

In this case, there is one special singular point, $D = (-1/2, -1/2)$, that has been already classified since it belongs to the line $p_1(x,y) = q_1(x,y)$.

Points $O = (0,0)$, $B = (1,-2)$ and $C = (4,4)$ are common to $\mathbf{S}_0(x,y) = 0$ and $\Delta_{\mathcal{E}_1}(x,y) = 0$ and, from Proposition 5.11, were determined by solving

$$-xy + 2x + 2y = 0 \wedge y^2 - 4x = 0.$$

On the other hand, points $A = (-2,1)$, $O = (0,0)$ and $C = (4,4)$ are common to $\mathbf{S}(x,y) = 0$ and $\Delta_{\mathcal{E}_2}(x,y) = 0$ and, from Proposition 5.11, were determined by solving

$$-xy + 2x + 2y = 0 \wedge x^2 - 4y = 0.$$

Note that points O and C are common to $\mathbf{S}_0 = 0$, $\Delta_{\mathcal{E}_1} = 0$, $\Delta_{\mathcal{E}_2} = 0$ and, as seen in Propostion 5.8, belong to the line $x - y = 0$.

The lifting of $\Pi(\mathcal{E}_1 \cap \mathcal{E}_2)$, outside the singular points of the cutcurve, will be made by using $\mathbf{S}_1(x,y;z)$:

$$z = \frac{x-y}{x-y} = 1.$$

Singular points of the cutcurve will be lifted by using $g(x,y,z)$:

$$g(x,y,z) = 0 \Leftrightarrow z = -\frac{y}{2} + \frac{\sqrt{y^2-4x}}{2} \vee z = -\frac{y}{2} - \frac{\sqrt{y^2-4x}}{2}.$$

To characterise the intersection curve of $\mathcal{E}_1$ and $\mathcal{E}_2$ we must determine the cutcurve $\mathbf{S}_0(x,y) = 0$ and its lifting. We will use the following functions:

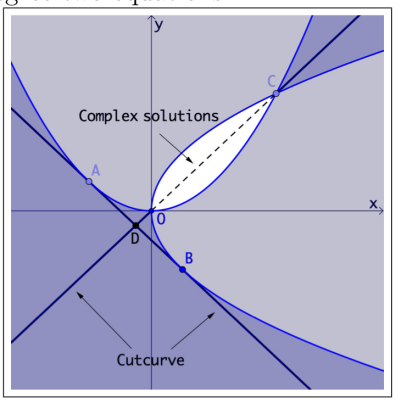
- For $x \in \mathbb{R}$, we define:
$$h_1(x) = x \qquad h_2(x) = -x - 1$$
- Let e_1 and e_2 be the functions defined by:
$$e_1(x, y) = -\frac{y}{2} + \frac{\sqrt{y^2 - 4x}}{2} \qquad e_2(x, y) = -\frac{y}{2} - \frac{\sqrt{y^2 - 4x}}{2}$$

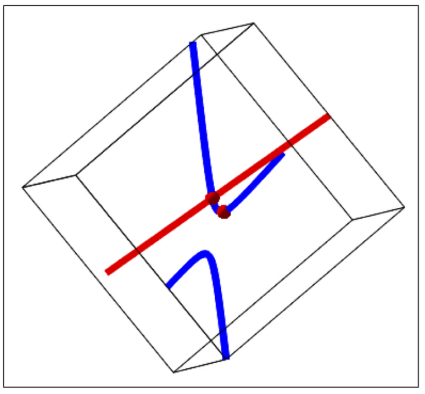
The parameterisation of the intersection curve is given by the following three components:

- For $x \in]-\infty, 0] \cup [4, +\infty[$: $(x, h_1(x), e_1(x, h_1(x))$.
- For $x \in]-\infty, 0] \cup [4, +\infty[$: $(x, h_1(x), e_2(x, h_1(x))$.
- For $x \in \mathbb{R}$: $(x, h_2(x), 1)$.

The intervals are determined by analysing the radical expressions and solving the corresponding (univariate) polynomial inequalities.

By using the QI online computation server (available at `https://gamble.loria.fr/qi/server/`), we get a parameterisation of the intersection curve without involving radicals. Instead here we show how to get a parameterisation of the intersection curve by solving several degree two equations.





Example 6.3. Let f and g be the polynomials

$$f(x, y, z) = z^2 + (y - 2x - 1)z - x^2 - y^2 - xy + x - y + 1$$

$$g(x, y, z) = z^2 + (x - y)z + x^2 + y^2 - xy - 2x + y - 5$$

defining one hyperboloid of one sheet and one ellipsoid, $\mathcal{E}_1$ and $\mathcal{E}_2$, whose intersection curve is to be computed. In this case we have

$$\mathbf{S}_0(x, y) = 7x^4 - 11x^3y + 23x^2y^2 - 6xy^3 + 4y^4 - 17x^3 + 12x^2y - 8xy^2 + 6y^3 - 50x^2 + 26xy$$
$$-29y^2 + 10x - 9y + 31$$

$\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2} = \{(x,y) \in \mathbb{R}^2 \colon 8x^2 + 5y^2 + 2y - 3 \geq 0,\ -3x^2 + 2xy - 3y^2 + 8x - 4y + 20 \geq 0\}$.

From Corollary 5.7, singular points of the cutcurve in the line $p_1(x,y) = q_1(x,y)$ were determined by solving:

$$11x^2 - 2xy - 8x + 8y^2 + 6y - 23 \wedge -3x + 2y - 1 = 0$$

They are

$$A = \left(-\frac{3}{13} - \frac{3\sqrt{14}}{13}, \frac{2}{13} - \frac{9\sqrt{14}}{26}\right) \qquad B = \left(-\frac{3}{13} + \frac{3\sqrt{14}}{13}, \frac{2}{13} + \frac{9\sqrt{14}}{26}\right)$$

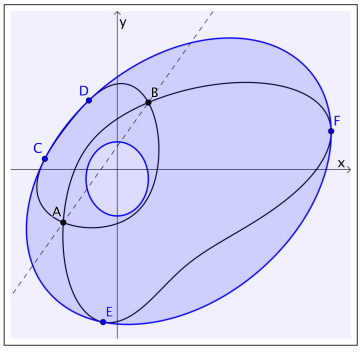
and can be lifted by using $g(x,y,z) = 0$ or $f(x,y,z) = 0$ producing four (singular) points in the intersection curve.

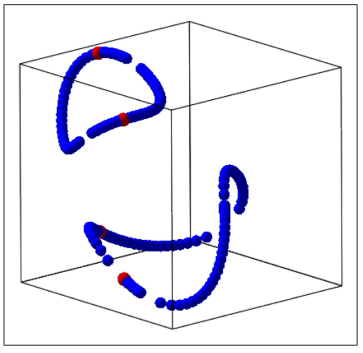
Common points to $\mathbf{S}_0(x,y) = 0$ and $\Delta_{\mathcal{E}_2}(x,y) = 0$ can be determined, according to Proposition 5.11, by solving

$$2x^2 - 7xy + y^2 - x - y - 8 = 0 \wedge -3x^2 + 2xy - 3y^2 + 8x - 4y + 20 = 0.$$

These points are $C = (-1.468654, 0.233082)$, $D = (-0.575622, 1.494633)$, $E = (-0.285566, -3.292475)$ and $F = (4.341889, 0.829820)$. The lifting of $\Pi(\mathcal{E}_1 \cap \mathcal{E}_2)$, outside the singular curves of the cutcurve, will be made by using $\mathbf{S}_1(x,y;z)$:

$$z = -\frac{2x^2 + 2y^2 - 3x + 2y - 6}{3x - 2y + 1}.$$





Last example shows a situation where the introduced technics are specially useful to determine the intersection curve of the two considered quadrics.

Example 6.4. Let f and g be the polynomials

$$f(x,y,z) = z^2 + \left(-\frac{2}{3}x + \frac{2}{3}y\right)z + \frac{x^2}{3} + \frac{y^2}{3} - \frac{1}{3}$$

$$g(x,y,z) = z^2 + \left(\frac{-2x+24y-2}{17}\right) z + \frac{x^2}{17} + \frac{2x}{17} - \frac{3}{17} + \frac{12y^2}{17}$$

defining two ellipsoids, $\mathcal{E}_1$ and $\mathcal{E}_2$, whose intersection curve is to be computed. In this case we have:

$$\mathbf{S}_0(x,y) = \frac{196}{2601}x^4 + \frac{616}{2601}x^3y + \frac{920}{2601}x^2y^2 + \frac{836}{2601}xy^3 + \frac{361}{2601}y^4 - \frac{112}{2601}x^3 - \frac{56}{867}x^2y - \frac{112}{2601}xy^2$$
$$-\frac{76}{2601}y^3 - \frac{104}{867}x^2 - \frac{632}{2601}xy - \frac{368}{2601}y^2 + \frac{176}{2601}x + \frac{184}{2601}y + \frac{52}{2601}$$

$$\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2} = \left\{(x,y) \in \mathbb{R}^2 \colon\ -2x^2 - 2xy - 2y^2 + 3 \geq 0, -4x^2 - 6xy - 15y^2 - 8x - 6y + 13 \geq 0\right\}$$

Singular points of the cutcurve in the line $p_1(x,y) = q_1(x,y)$ can be determined, as seen in Corollary 5.7, by solving

$$-434x^2 - 362xy - 38y^2 + 288x + 216y + 399 = 0 \wedge -14x - 19y + 3 = 0$$

They are

$$A = \left(\frac{3}{14} - \frac{11\sqrt{95}}{70}, \frac{11\sqrt{95}}{95}\right) \qquad C = \left(\frac{3}{14} + \frac{11\sqrt{95}}{70}, -\frac{11\sqrt{95}}{95}\right)$$

and are outside $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$: therefore they will not be lifted. Moreover, the cutcurve has a third singular (and isolated) point $B = (1,0)$ which is inside $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$ but not in the line $p_1(x,y) = q_1(x,y)$. This means that B is the projection of a tangential intersection point of $\mathcal{E}_1$ and $\mathcal{E}_2$ and it has been computed by using Theorem 5.10.

Points

$$D = (-1.310086292, 1.116297338) \qquad E = (-1.032926046, -0.320076179)$$

are common to $\mathbf{S}_0(x,y) = 0$ and $\Delta_{\mathcal{E}_1}(x,y) = 0$ and, from Proposition 5.11, were determined by solving

$$18x^2 + 26xy + 29y^2 + 4x + 2y - 26 = 0 \wedge -2x^2 - 2xy - 2y^2 + 3 = 0.$$

On the other hand, the points

$$F = (-1.310059433, 1.116308957) \qquad G = (0.4229961827, -1.105788551)$$

are common to $\mathbf{S}_0(x,y) = 0$ and $\Delta_{\mathcal{E}_2}(x,y) = 0$ and, from Proposition 5.11, were determined by solving

$$18x^2 + 26xy + 29y^2 + 4x + 2y - 26 = 0 \wedge -4x^2 - 6xy - 15y^2 - 8x - 6y + 13 = 0.$$

Figure 3 (left) shows the location of all these points with respect to the cutcurve and the silhouette curves. In order to determine what is the relative position of the points A, F and

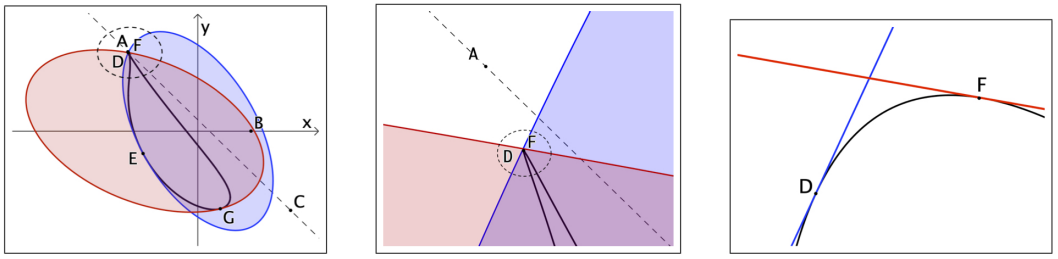


Figure 3: Cutcurve and silhouette curves showing where points A, F and D are (left). Location of points A, F and D with respect to the cutcurve and the silhouette curves (center and right).

D with respect to this three curves we need to use the results introduced in the previous section. Figure 3 (center and right) shows in detail what is happening in that area.

Point A is outside the region $\mathcal{A}_{\varepsilon_1,\varepsilon_2}$ and does not play any role when computing the intersection curve between the two considered quadrics. Points F and D does belong to the intersection between the cutcurve and one of the silhouette curves. Proposition 5.8 helps to conclude that the intersection between the cutcurve and the two silhouette curves is empty and not A as Figure 3 (left) could suggest.

The lifting of the regular points of the cutcurve and of the point B will be made by using $\mathbf{S}_1(x, y; z)$:

$$z = \frac{14x^2 - 19y^2 - 6x - 8}{28x + 38y - 6}.$$

Note that B is the projection of the point $(1, 0, 0)$ where the ellipsoids are tangent and this is the reason why it can be lifted by using $\mathbf{S}_1(x, y; z)$ (as seen in Theorem 5.4). The intersection curve of the two considered quadrics can be found in Figure 4.

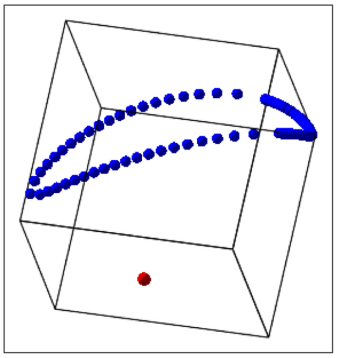


Figure 4: Two ellipsoids with a curve and an isolated point (tangent to both ellipsoids) in common.

The algorithm presented here has been fully implemented in Maple. Table 1 shows its behaviour when applied to a database of 50 examples most of them taken from the QI online

computation server available at `https://gamble.loria.fr/qi/server/`. In the annex the equations of each pair of quadrics used can be found.

Figure 5 shows the shape of the output of this implementation when applied to a particular case. In this concrete case the intersection curve is not discretised and it is presented by a parameterisation involving radicals but the output shows the intervals where x can be evaluated. For this concrete example the cutcurve has two singular points inside the admissible region and there are no tangential intersections whose projection is outside the line $p_1(x, y) = q_1(x, y)$.

```
par((-16*x^2+8*x*z+8*y^2-8*z^2+48*x-32*y-24*z)*(-1/8), (-12*x*z+4*y^2+2*z^2+8*x-16*y-4*z)*(1/2));
```

$$\left[\left[[x, 2+2\sqrt{x^2+1}, 2x], [x, 2-2\sqrt{x^2+1}, 2x], \left[x, 2+\frac{2\sqrt{4x^2-10x+4}}{3}, \frac{2x}{3}-\frac{4}{3}\right], \left[x, 2-\frac{2\sqrt{4x^2-10x+4}}{3}, \frac{2x}{3}-\frac{4}{3}\right], \left[\left[-\frac{1}{2}, \frac{1}{2}\right], [2, 3]\right]\right]\right]$$

```
sing((-16*x^2+8*x*z+8*y^2-8*z^2+48*x-32*y-24*z)*(-1/8), (-12*x*z+4*y^2+2*z^2+8*x-16*y-4*z)*(1/2));
```

$$[[\,], [[-1, 2-2\sqrt{2}, -2], [-1, 2+2\sqrt{2}, -2]], [\,]]$$

Figure 5: Using Maple to compute the intersection curve of two quadrics.

The meaning of the columns in Table 1 is the following:

- Third column reflects if the intersection curve has been discretised or not.
- Forth column shows the number of points computed when the intersection curve has been discretised.
- Fifth column shows if there are tangential intersection points whose projection is outside the line $p_1(x, y) = q_1(x, y)$

Example	Time	Discretisation	Number of Points	Tangential intersections
Example 1	0.395	yes	40	no
Example 2	1.295	yes	242	no
Example 3	1.380	yes	244	no
Example 4	0.313	no		no
Example 5	1.586	yes	476	no
Example 6	1.502	yes	472	no
Example 7	1.338	yes	362	no
Example 8	1.290	yes	202	yes
Example 9	0.822	yes	80	no
Example 10	3.791	yes	840	no
Example 11	0.193	no		no
Example 12	3.387	yes	762	no
Example 13	0.236	no		no
Example 14	0.353	no		no
Example 15	0.357	no		no
Example 16	0.292	no		no
Example 17	0.330	no		no
Example 18	0.294	yes	40	no
Example 19	0.224	no		no
Example 20	0.372	no		no
Example 21	3.114	yes	1114	no
Example 22	0.215	no		no
Example 23	0.247	no		no
Example 24	0.616	no		no
Example 25	0.265	no		no
Example 26	0.334	no		no
Example 27	3.541	yes	558	no
Example 28	0.296	no		no
Example 29	1.664	yes	472	no
Example 30	0.181	no		no
Example 31	0.143	no		no
Example 32	0.200	no		no
Example 33	0.290	no		no
Example 34	0.234	no		no
Example 35	0.273	no		no
Example 36	0.329	no		no
Example 37	0.351	no		no
Example 38	0.237	no		no
Example 39	0.242	no		no
Example 40	0.234	no		yes
Example 41	0.272	no		yes
Example 42	0.200	no		yes
Example 43	0.332	no		no
Example 44	0.899	yes	160	no
Example 45	0.310	no		no
Example 46	0.304	no		yes
Example 47	1.165	no		yes
Example 48	5.835	yes	1574	yes
Example 49	6.542	yes	1894	yes
Example 50	1.195	yes	120	yes

Table 1: Results from applying the Maple implementation to the quadrics in the Annex.

Its practical behaviour is quite good and admits several improvements specially the step where the intersection curve is discretised. At this moment this is made by solving as many times as needed the equation $S_0(\alpha, y) = 0$ for α belonging to the different intervals provided by the singular points and critical points of the curve $S_0(x, y) = 0$. It must be noted here that the equation $S_0(\alpha, y) = 0$ is never solved when α is the projection of a singular point

of $S_0(x, y) = 0$ since these points are previously computed by using the results in subsection 5.1.

7. Conclusions

We have introduced a new approach to deal with the computation of the intersection curve of two quadrics. The main ingredients of this approach are a detailed analysis of the cutcurve, its singular points and of its relation with the silhouette curves together with the using of an uniform way to perform the lifting of the cutcurve to the intersection curve of the two considered quadrics.

Concerning the analysis of the cutcurve we classify its singular points in two different types depending on how they will be lifted. Those belonging to the line $p_1(x, y) = q_1(x, y)$ are easy to compute and difficult to lift (but just solving a degree two equation) and those not in that line which are more complicated to be determined but easier to lift.

This approach is not intended to classify the intersection curve between the two considered quadrics. Its main goal is to produce in a very direct way a description of the intersection curve which is topologically correct. This is the reason why we allow in the lifting of the cutcurve, when possible, the use of radicals or we rely on the discretisation of the branches of the cutcurve (uniquely determined by the points computed in that curve).

The algorithm has been fully implemented in Maple showing a very good practical behaviour.

References

[1] Basu, S., Pollack, R., Roy, M.-F (2006). Algorithms in Real Algebraic Geometry. Springer–Verlag.

[2] Berberich, E., Hemmer, M., Kettner, L., Schmer, E., & Wolpert, N. (2005). An exact, complete and efficient implementation for computing planar maps of quadric intersection curves. In Proceedings of the Twenty-first32 Annual Symposium on Computational Geometry, SCG 05, ACM, New York, NY, USA, pp. 99-106.

[3] Dupont, L., Lazard, D., Lazard, S., & Petitjean, S. (2008). Near-optimal parameterization of the intersection of quadrics: I. The generic algorithm. Journal of Symbolic Computation 43 (3), 168-191.

[4] Dupont, L., Lazard, D., Lazard, S., & Petitjean, S. (2008). Near-optimal parameterization of the intersection of quadrics: II. A classification of pencils. Journal of Symbolic Computation 43 (3), 192–215.

[5] Dupont, L., Lazard, D., Lazard, S., & Petitjean, S. (2008). Near-optimal parameterization of the intersection of quadrics: III. Parameterizing singular intersections. Journal of Symbolic Computation 43 (3), 216–232.

[6] Farouki, R. T., Neff, C., & OConner, M. A. (1989). Automatic parsing of degenerate quadric-surface intersections. ACM Transactions on Graphics 8 (3), 174–203.

[7] Geismann, N., Hemmer, M., Schömer, E. (2001). Computing a 3-dimensional cell in an arrangement of quadrics: Exactly and actually!, in: Proceedings of the Seventeenth Annual Symposium on ComputationalGeometry, SCG 01, ACM, New York, NY, USA, pp. 264-273.

[8] Goldman, R. N., & Miller, J. R. (1991). Combining algebraic rigor with geometric robustness for the detection and calculation of conic sections in the intersection of two natural quadric surfaces, in: Proceedings of the First ACM Symposium on Solid Modeling Foundations and CAD/CAM Application, SMA '91, ACM, New York, NY, USA, pp. 221–231.

[9] Gonzalez-Vega, L., Rúa, I. (2009), Solving the implicitization, inversion and reparametrization problems for rational curves through subresultants, Computer Aided Geometric Design 26 (9) 941-961.
[10] Johnstone, J. K., & Shene, C. K. (1992). Computing the intersection of a plane and a natural quadric. Computers and Graphics 16 (2), 179–186.
[11] Lazard, S., Pearanda, L. M.,& Petitjean, S. (2006). Intersecting quadrics: An efficient and exact implementation. In Computational Geometry 35 (1), 74–99.
[12] Levin, J. Z. (1979). Mathematical models for determining the intersections of quadric surfaces. Computer Graphics and Image Processing 11 (1), 73–87.
[13] Levin, J. Z. (1976). A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. Communications of the ACM 19 (10), 555-563.
[14] Li, Y. Bin. (2006). A new approach for constructing subresultants. Applied Mathematics and Computation 183 (1), 471–476.
[15] Miller, J. R. (1987). Geometric approaches to nonplanar quadric surface intersection curves. ACM Transactions on Graphics 6 (4), 274–307.
[16] Miller, J. R., & Goldman, R. N. (1995). Geometric Algorithms for Detecting and Calculating All Conic Sections in the Intersection of Any 2 Natural Quadric Surfaces. Graphical Models and Image Processing 57 (1), 55-66.
[17] Mourrain, B., Técourt, J. P., & Teillaud, M. (2005). On the computation of an arrangement of quadrics in 3D. In Computational Geometry: Theory and Applications 30 (2), 145–164.
[18] Schomer, E., & Wolpert, N. (2006). An exact and efficient approach for computing a cell in an arrangement of quadrics. Computational Geometry: Theory and Applications 33 (1–2), 65–97.
[19] Sendra, J. R., Winkler, F. (1999). Algorithms for rational real algebraic curves. Fundamenta Informaticae 39 (1–2), 211–228.
[20] Shene, C.-K., & Johnstone, J. K. (1994). On the lower degree intersections of two natural quadrics. ACM Transactions on Graphics 13 (4), 400–424.
[21] Wang, W., Goldman, R., & Tu, C. (2003). Enhancing Levin's method for computing quadric–surface intersections. Computer Aided Geometric Design 20 (7), 401–422.
[22] Wang, W., Joe, B., & Goldman, R. (2002). Computing quadric surface intersections based on an analysis of plane cubic curves. Graphical Models 64 (6), 335–367.
[23] Wilf, I., & Manor, Y. (1993). Quadric-surface intersection curves: shape and structure. Computer-Aided Design 25 (10), 633–643.

8. Annex

Example	Quadrics
Example 1	$z^2+(2\,x-y+4)\,z+12\,x^2-24\,xy+11\,y^2+20\,x-18\,y+12$ $z^2+\left(-\frac{26\,x}{9}+5/9\,y+\frac{20}{9}\right)z+4/3\,x^2-1/9\,y^2-\frac{20\,x}{9}+2/9\,y+4/3$
Example 2	$z^2+(-x+3)\,z+x^2-3/2\,xy+1/2\,y^2-x-y/2+2$ $z^2-8\,x^2+18\,xy-4\,xz-7\,y^2-12\,x+10\,y-4$
Example 3	$z^2+\left(-8/5\,x+\frac{12}{5}\right)z-\frac{12\,x^2}{5}+\frac{28\,xy}{5}-\frac{13\,y^2}{5}-\frac{32\,x}{5}+\frac{24\,y}{5}-4/5$ $z^2+\left(-\frac{16\,x}{25}+\frac{84}{25}\right)z-\frac{28\,x^2}{25}+\frac{12\,xy}{5}-\frac{27\,y^2}{25}-\frac{112\,x}{25}+\frac{48\,y}{25}+\frac{52}{25}$
Example 4	$z^2+\left(-\frac{88\,x}{43}+\frac{84}{43}\right)z+\frac{60\,x^2}{43}-\frac{28\,xy}{43}+\frac{13\,y^2}{43}-\frac{64\,x}{43}-\frac{24\,y}{43}+\frac{52}{43}$ $z^2+\left(-\frac{80\,x}{23}+\frac{12}{23}\right)z+\frac{76\,x^2}{23}-\frac{60\,xy}{23}+\frac{27\,y^2}{23}+\frac{16\,x}{23}-\frac{48\,y}{23}-\frac{4}{23}$
Example 5	$z^2+(-2\,x+2)\,z+3/2\,xy-1/2\,y^2-3\,x+y/2+1$ $z^2+\left(-\frac{16\,x}{7}+\frac{12}{7}\right)z+\frac{10\,xy}{7}-3/7\,y^2-\frac{20\,x}{7}+2/7\,y+4/7$

Example	Quadrics
Example 6	$z^2+(-2\,x+2)\,z+3/2\,xy-1/2\,y^2-3\,x+y/2+1$ $z^2+\left(-\frac{32\,x}{15}+\frac{28}{15}\right)z+\frac{8\,x^2}{15}+2/3\,xy-1/5\,y^2-\frac{12\,x}{5}+2/15\,y+4/5$
Example 7	$z^2+(-2\,x+2)\,z+3/2\,xy-1/2\,y^2-3\,x+y/2+1$ $z^2+\left(-\frac{16\,x}{9}+\frac{20}{9}\right)z+\frac{10\,xy}{9}-1/3\,y^2-\frac{20\,x}{9}+2/9\,y+4/3$
Example 8	$z^2+(-3\,x+2\,y+1)\,z+4\,x^2-4\,xy+1/2\,y^2-2\,x+2\,y$ $z^2+\left(-\frac{30\,x}{13}+\frac{4\,y}{13}+\frac{22}{13}\right)z+\frac{16\,x^2}{13}-\frac{4\,xy}{13}-\frac{28\,x}{13}+\frac{4\,y}{13}+\frac{8}{13}$
Example 9	$z^2+(-8/3\,x+4/3)\,z+8/3\,x^2-2\,xy+2/3\,y^2-4/3\,x-2/3\,y$ $z^2+\left(-\frac{12\,x}{7}+\frac{16}{7}\right)z+\frac{16\,x^2}{7}-2\,xy+4/7\,y^2-\frac{12\,x}{7}-2/7\,y+\frac{8}{7}$
Example 10	$z^2+(-x+3)\,z-2\,x^2+3\,xy-y^2-4\,x+y+2$ $z^2+(-2\,x+2)\,z+\frac{4\,x^2}{13}+\frac{10\,xy}{13}-3/13\,y^2-\frac{32\,x}{13}+2/13\,y+\frac{12}{13}$
Example 11	$z^2+(-2\,x+2)\,z+3/2\,xy-1/2\,y^2-3\,x+y/2+1$ $z^2+4\,z+8\,x^2-10\,xy+3\,y^2+4\,x-2\,y+4$
Example 12	$z^2+(-4\,x+2\,y+2)\,z+2\,x^2-xy-6\,x+3\,y$ $z^2+(-2\,x-y+6)\,z-2\,x^2+3\,xy-10\,x+y+4$
Example 13	$z^2+(-2\,x+2)\,z+3\,x^2-3\,xy+5/4\,y^2-2\,y+2$ $z^2+\left(-\frac{12\,x}{5}+8/5\right)z+2\,x^2-6/5\,xy+1/2\,y^2-8/5\,x-4/5\,y+4/5$
Example 14	$z^2+(-2\,x+2)\,z+x^2-xy+3/4\,y^2-2\,y+2$ $z^2+(-4/3\,x+8/3)\,z+2/3\,x^2+2/3\,xy-1/2\,y^2-8/3\,x+4/3\,y+4/3$
Example 15	$z^2+(-2\,x+2)\,z+1/2\,y^2-2\,y+2$ $z^2+(-4/3\,x+8/3)\,z+4/3\,x^2-1/3\,y^2-8/3\,x+4/3\,y+4/3$
Example 16	$z^2+(-8/3\,x+4/3)\,z-4/3\,x^2+4\,xy-5/3\,y^2-16/3\,x+8/3\,y-4/3$ $z^2+\left(-\frac{16\,x}{11}+\frac{28}{11}\right)z-4/11\,x^2+\frac{12\,xy}{11}-\frac{5\,y^2}{11}-\frac{32\,x}{11}+\frac{8\,y}{11}+\frac{12}{11}$
Example 17	$z^2+\left(-\frac{32\,x}{15}+\frac{28}{15}\right)z+\frac{8\,x^2}{15}+4/5\,xy-1/3\,y^2-8/3\,x+\frac{8\,y}{15}+\frac{8}{15}$ $z^2+\left(-\frac{16\,x}{13}-\frac{32\,y}{39}+\frac{36}{13}\right)z+\frac{8\,x^2}{39}+\frac{28\,xy}{39}+1/13\,y^2-\frac{24\,x}{13}-\frac{40\,y}{39}+\frac{24}{13}$
Example 18	$z^2+(-8/3\,x+4/3)\,z-4/3\,x^2+4\,xy-5/3\,y^2-16/3\,x+8/3\,y-4/3$ $z^2+\left(-\frac{24\,x}{7}+\frac{32\,y}{21}+4/7\right)z+\frac{52\,x^2}{21}-4/3\,xy-1/7\,y^2-\frac{16\,x}{7}+\frac{40\,y}{21}-4/7$
Example 19	$z^2+(-8/3\,x+4/3)\,z+4/3\,xy-1/3\,y^2-8/3\,x$ $z^2+\left(-8/5\,x+\frac{12}{5}\right)z+\frac{12\,xy}{5}-7/5\,y^2-\frac{24\,x}{5}+\frac{16\,y}{5}$
Example 20	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{8\,x^2}{11}+4/11\,xy-1/11\,y^2-\frac{24\,x}{11}+\frac{8}{11}$ $z^2+(-8/3\,x+4/3)\,z+8/3\,x^2-4\,xy+7/3\,y^2+8/3\,x-16/3\,y+8/3$
Example 21	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{8\,x^2}{11}+4/11\,xy-1/11\,y^2-\frac{24\,x}{11}+\frac{8}{11}$ $z^2+\left(-\frac{504\,x}{253}+\frac{508}{253}\right)z+\frac{248\,x^2}{253}+\frac{12\,xy}{253}-\frac{7\,y^2}{253}-\frac{520\,x}{253}+\frac{16\,y}{253}+\frac{248}{253}$
Example 22	$z^2-4\,xz+2\,xy-1/2\,y^2-4\,x-2$ $z^2-4\,xz-2\,xy+5/2\,y^2+4\,x-8\,y+2$
Example 23	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{8\,x^2}{11}+4/11\,xy-1/11\,y^2-\frac{24\,x}{11}+\frac{8}{11}$ $z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{8\,x^2}{11}-4/11\,xy+\frac{5\,y^2}{11}-\frac{8\,x}{11}-\frac{16\,y}{11}+\frac{16}{11}$
Example 24	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{8\,x^2}{11}+4/11\,xy-1/11\,y^2-\frac{24\,x}{11}+\frac{8}{11}$ $z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z-\frac{248\,x^2}{11}+\frac{252\,xy}{11}-\frac{59\,y^2}{11}-\frac{8\,x}{11}-\frac{16\,y}{11}+\frac{16}{11}$
Example 25	$z^2+(-x+3)\,z+2\,x^2-y^2-6\,x+4\,y$ $z^2+(-6\,x-2)\,z+2\,y^2+4\,x-8\,y$
Example 26	$z^2+\left(-\frac{88\,x}{43}+\frac{84}{43}\right)z+\frac{52\,x^2}{43}-\frac{4\,xy}{43}-1/43\,y^2-\frac{96\,x}{43}+\frac{8\,y}{43}+\frac{36}{43}$

Example	Quadrics
	$z^2+\left(-\frac{32\,x}{17}+\frac{36}{17}\right)z+\frac{52\,x^2}{51}-\frac{4\,xy}{51}-\frac{y^2}{51}-\frac{112\,x}{51}+\frac{8\,y}{51}+\frac{52}{51}$
Example 27	$z^2+\left(-\frac{88\,x}{43}+\frac{84}{43}\right)z+\frac{52\,x^2}{43}-\frac{4\,xy}{43}-1/43\,y^2-\frac{96\,x}{43}+\frac{8\,y}{43}+\frac{36}{43}$ $z^2+\left(-\frac{24\,x}{5}-4/5\right)z+\frac{28\,x^2}{5}-4/5\,xy-1/5\,y^2-\frac{16\,x}{5}+8/5\,y-\frac{12}{5}$
Example 28	$z^2+4\,z-4\,x^2+6\,xy-2\,y^2-4\,x+2\,y+4$ $z^2+\left(-8/5\,x+\frac{12}{5}\right)z-4/5\,x^2+\frac{14\,xy}{5}-6/5\,y^2-4\,x+2\,y+4/5$
Example 29	$z^2+4\,z-4\,x^2+6\,xy-2\,y^2-4\,x+2\,y+4$ $z^2+\left(-\frac{16\,x}{9}+\frac{20}{9}\right)z+\frac{14\,xy}{9}-2/3\,y^2-\frac{28\,x}{9}+\frac{10\,y}{9}+\frac{8}{9}$
Example 30	$z^2+4\,z+8\,x^2-12\,xy+5\,y^2+8\,x-8\,y+8$ $z^2+\left(-\frac{16\,x}{7}+\frac{12}{7}\right)z+\frac{12\,xy}{7}-5/7\,y^2-\frac{24\,x}{7}+\frac{8\,y}{7}$
Example 31	$z^2+(-8/3\,x+4/3)\,z+4/3\,xy-1/3\,y^2-8/3\,x$ $z^2+(-8/3\,x+4/3)\,z+8/3\,x^2-4\,xy+7/3\,y^2+8/3\,x-16/3\,y+8/3$
Example 32	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{8\,x^2}{11}+4/11\,xy-1/11\,y^2-\frac{24\,x}{11}+\frac{8}{11}$ $z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{16\,x^2}{11}-\frac{12\,xy}{11}+\frac{7\,y^2}{11}-\frac{8\,x}{11}-\frac{16\,y}{11}+\frac{16}{11}$
Example 33	$z^2+(-8/3\,x+4/3)\,z+4/3\,xy-1/3\,y^2-8/3\,x$ $z^2+(-8\,x+16/3\,y-4)\,z+16/3\,x^2-4\,xy-1/3\,y^2-8/3\,x+16/3\,y-16/3$
Example 34	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{8\,x^2}{11}+4/11\,xy-1/11\,y^2-\frac{24\,x}{11}+\frac{8}{11}$ $z^2+\left(-\frac{40\,x}{11}+\frac{16\,y}{11}+4/11\right)z+\frac{24\,x^2}{11}-\frac{12\,xy}{11}-1/11\,y^2-\frac{24\,x}{11}+\frac{16\,y}{11}-\frac{8}{11}$
Example 35	$z^2+(-8/3\,x+4/3)\,z+4/3\,xy-1/3\,y^2-8/3\,x$ $z^2+(-8\,x+16/3\,y-4)\,z+16/3\,x^2-4\,xy-1/3\,y^2-8/3\,x+16/3\,y-16/3$
Example 36	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+4/11\,x^2+\frac{8\,xy}{11}-2/11\,y^2-\frac{24\,x}{11}+\frac{8}{11}$ $z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z-\frac{12\,x^2}{11}+\frac{32\,xy}{11}-\frac{10\,y^2}{11}-\frac{40\,x}{11}+\frac{8\,y}{11}+\frac{8}{11}$
Example 37	$z^2+(-5/2\,x+y/2+3/2)\,z+2\,x^2-xy+1/4\,y^2-3\,x+y$ $z^2+(-7/2\,x+y+3/2)\,z+2\,x^2-xy+1/4\,y^2-5\,x+2\,y$
Example 38	$z^2+(-3/2\,x-y/2+5/2)\,z+xy-1/4\,y^2-x-y+2$ $z^2+(-x/2-y+5/2)\,z+xy-1/4\,y^2+x-2\,y+2$
Example 39	$z^2+(-7/4\,x-y/4+9/4)\,z+1/2\,x^2+1/2\,xy-1/8\,y^2-3/2\,x-y/2+3/2$ $z^2+(-5/4\,x-y/2+9/4)\,z+1/2\,x^2+1/2\,xy-1/8\,y^2-x/2-y+3/2$
Example 40	$z^2+\left(-8/5\,x+\frac{12}{5}\right)z-4/5\,x^2+\frac{12\,xy}{5}-4/5\,y^2-\frac{16\,x}{5}+4/5\,y+8/5$ $z^2+\left(-\frac{16\,x}{13}+\frac{36}{13}\right)z-\frac{20\,x^2}{13}+\frac{36\,xy}{13}-\frac{14\,y^2}{13}-\frac{48\,x}{13}+\frac{20\,y}{13}+\frac{16}{13}$
Example 41	$z^2+\left(-\frac{32\,x}{17}+\frac{36}{17}\right)z+\frac{32\,xy}{17}-\frac{13\,y^2}{17}-\frac{64\,x}{17}+\frac{20\,y}{17}+\frac{16}{17}$ $z^2+(-8/3\,x+4/3)\,z+\frac{32\,x^2}{9}-\frac{8\,xy}{9}+1/9\,y^2-\frac{32\,x}{9}+4/9\,y+\frac{8}{9}$
Example 42	$z^2+\left(-\frac{24\,x}{13}+\frac{28}{13}\right)z+\frac{24\,xy}{13}-\frac{10\,y^2}{13}-\frac{48\,x}{13}+\frac{16\,y}{13}+\frac{12}{13}$ $z^2+\left(-\frac{16\,x}{5}+4/5\right)z+\frac{24\,x^2}{5}-8/5\,xy+2/5\,y^2-\frac{16\,x}{5}+4/5$
Example 43	$z^2+\left(-\frac{128\,x}{63}+\frac{124}{63}\right)z+\frac{176\,x^2}{63}-\frac{8\,xy}{21}-\frac{13\,y^2}{63}-\frac{304\,x}{63}+\frac{76\,y}{63}+\frac{8}{7}$ $z^2+\left(-\frac{136\,x}{71}+\frac{148}{71}\right)z+\frac{368\,x^2}{71}-\frac{208\,xy}{71}+\frac{29\,y^2}{71}-\frac{336\,x}{71}+\frac{92\,y}{71}+\frac{80}{71}$
Example 44	$z^2+\left(-\frac{16\,x}{7}+\frac{12}{7}\right)z+\frac{16\,xy}{7}-5/7\,y^2-\frac{32\,x}{7}+4/7\,y+\frac{8}{7}$ $z^2+\left(-8/5\,x+\frac{12}{5}\right)z+\frac{16\,x^2}{15}+\frac{8\,xy}{15}-\frac{7\,y^2}{15}-\frac{64\,x}{15}+4/3\,y+\frac{16}{15}$
Example 45	$z^2+\left(-\frac{24\,x}{11}+\frac{20}{11}\right)z+\frac{24\,xy}{11}-\frac{48\,x}{11}-\frac{8\,y^2}{11}+\frac{8\,y}{11}+\frac{12}{11}$

Example	**Quadrics**
	$z^2+\left(-\frac{32\,x}{19}+\frac{44}{19}\right)z+\frac{24\,x^2}{19}+\frac{8\,xy}{19}-\frac{80\,x}{19}-\frac{8\,y^2}{19}+\frac{24\,y}{19}+\frac{20}{19}$
Example 46	$z^2+\left(-\frac{128\,x}{65}+\frac{132}{65}\right)z-\frac{64\,x^2}{65}+\frac{112\,xy}{65}-\frac{29\,y^2}{65}-\frac{96\,x}{65}+\frac{4\,y}{65}+\frac{48}{65}$ $z^2+\left(-\frac{40\,x}{19}+\frac{36}{19}\right)z-\frac{128\,x^2}{57}+\frac{56\,xy}{19}-\frac{13\,y^2}{19}-\frac{64\,x}{57}-\frac{4\,y}{19}+\frac{40}{57}$
Example 47	$z^2+\left(-\frac{32\,x}{17}+\frac{36}{17}\right)z-\frac{16\,x^2}{17}+\frac{24\,xy}{17}-\frac{7\,y^2}{17}-\frac{16\,x}{17}+\frac{4\,y}{17}+\frac{8}{17}$ $z^2+(-8/3\,x+4/3)\,z-16/3\,x^2+16/3\,xy-y^2+\frac{16\,x}{9}-4/3\,y$
Example 48	$z^2+\left(-\frac{32\,x}{17}+\frac{36}{17}\right)z-\frac{16\,x^2}{17}+\frac{24\,xy}{17}-\frac{7\,y^2}{17}-\frac{16\,x}{17}+\frac{4\,y}{17}+\frac{8}{17}$ $z^2+(-8/3\,x+4/3)\,z-16/3\,x^2+16/3\,xy-y^2+\frac{16\,x}{9}-4/3\,y$
Example 49	$z^2+\left(-\frac{128\,x}{65}+\frac{132}{65}\right)z-\frac{56\,x^2}{13}+\frac{288\,xy}{65}-\frac{64\,y^2}{65}-\frac{16\,x}{65}-\frac{32\,y}{65}+\frac{44}{65}$ $z^2+\left(-\frac{40\,x}{17}+\frac{28}{17}\right)z-\frac{16\,x^2}{17}+\frac{32\,xy}{17}-\frac{7\,y^2}{17}-\frac{16\,x}{17}-\frac{4\,y}{17}+\frac{8}{17}$
Example 50	$z^2+(-2/3\,x+2/3\,y)\,z+1/3\,x^2+1/3\,y^2-1/3$ $z^2+\left(-2/17\,x+\frac{24\,y}{17}-2/17\right)z+1/17\,x^2+2/17\,x-\frac{3}{17}+\frac{12\,y^2}{17}$

Chapter 3

Intersecting two quadrics with GeoGebra (see [40])

The main goal of this paper, presented here, is to create a tool to determine the intersection curve of two quadrics. This is something that GeoGebra allows only for very simple cases.

In this chapter, an implementation of the algorithm presented in the previous chapter in GeoGebra is described and presented. This implementation is done using only GeoGebra commands that interact with Algebra, CAS, 2D, 3D windows. To test the efficiency and applicability of this algorithm (and of its implementation in GeoGebra), 50 randomly generated examples were tested. We analyze the case where two quadrics are defined by two polynomials of degree 2, in the variable z, the case of a quadric of a degree 2 and the other of a degree 1, and finally the case where two quadrics are defined by two polynomials of degree 1 in z.

Intersecting Two Quadrics with GeoGebra

Alexandre Trocado[1(✉)], Laureano Gonzalez-Vega[2], and José Manuel Dos Santos[1]

[1] Universidade Aberta, Lisbon, Portugal
mail@alexandretrocado.com, dossantosdossantos@gmail.com
[2] Universidad de Cantabria, Santander, Spain
laureano.gonzalez@unican.es

Abstract. This paper presents the first implementation in GeoGebra of an algorithm computing the intersection curve of two quadrics. This approach is based on computing the projection of the intersection curve, also known as cutcurve, determining its singularities and structure and lifting to 3D this plane curve. The considered problem can be used to show some of the difficulties arising when implementing in GeoGebra a geometric algorithm based on the algebraic analysis of the equations defining the considered objects.

Keywords: GeoGebra · Cutcurve · Intersection curve · Lifting

1 Introduction

Algorithms for computing quadrics intersection date back to the late seventies. Computing the representation of the curve defined as the intersection of two quadrics has been a relevant problem to solve over the last decades. Levin in 1976 and 1979 (see [6,7]) introduced a method failing when the intersection curve is singular and even generates results that are not topologically correct. Levin's method has been improved by Wang et al. (see [12]) making it capable of computing geometric and structural information. Besides, Dupont et al. (see [2]) succeeded in finding parameterizations that overcame the fact that Levin's method generated formulas that were not suited for further symbolic processing. On the other hand, Mourrain et al. (see [8]) studied a sweeping algorithm for computing the arrangement of a set of quadrics in $\mathbb{R}^3$ that reduces the intersection of two quadrics to a dynamic two-dimensional problem. Dupont et al. (see [3–5]) proposed algorithms that enable to compute in practice an exact form of the parameterization of the intersection curve of two quadrics with rational coefficients. These algorithms represent a substantial improvement of Levin's

Second author is partially supported by the Spanish Ministerio de Economia y Competitividad and by the European Regional Development Fund (ERDF), under the project MTM2017-88796-P.

M. Ćirić et al. (Eds.): CAI 2019, LNCS 11545, pp. 237–248, 2019.
https://doi.org/10.1007/978-3-030-21363-3_20

method and its subsequent refinements. Another approach is based on the using of the cutcurve and resultants (see [9,11]). This method can handle all kinds of inputs including all degenerate ones where intersection curves involve cutcurves with singularities. Here we propose an implementation of this method, in GeoGebra, adapting the algorithm developed by Trocado and Gonzalez-Vega [11] to the special characteristics of GeoGebra.

GeoGebra is a software system for doing dynamic geometry and algebra in the plane. Since 2001 GeoGebra has gone from a dynamic geometry software (DGS) to a powerful computational tool in several areas of mathematics. Powerful algebraic capabilities have been introduced in GeoGebra, such as an efficient spreadsheet that can deal with many kind of objects, an algebraic and symbolic system and several graphical views that extend the possibility of multidimensional representations. The recent 3D features allow more intuitive interaction with three-dimensional objects than most existing mathematical software. However, there are still missing capabilities in GeoGebra 3D, namely the determination of the intersection curve of two quadrics.

The aim of this paper is to present a new tool that allows to compute in GeoGebra a graphical representation of the intersection curve of two quadrics and, when possible, its parameterization. The implemented algorithm uses resultants to determine the projection of the intersection curve in the plane $z = 0$ (the so called cutcurve) and the lifting of its regular and singular points is made by using only one subresultant (the index one subresultant; see [11]). The implemented algorithm presented here does not need to compute any resultant or subresultant since they are provided fully precomputed (those formulae can be found in [11]). When the Computer Algebra capabilities of GeoGebra do not allow to compute a parameterization of the cutcurve (may be involving radicals) or when such a parameterization is very complicated to deal with, a discretization of this curve is determined. The lifting is independent of how the cutcurve is presented: we get either a discretization or a parameterization (involving in some cases radicals) of the intersection curve.

2 Mathematical Tools

Quadrics are the simplest non linear surfaces used in many areas and computing their intersection is a relevant problem.

Definition 1. *Quadrics are algebraic surfaces defined by the equation* ($a_{i,j} \in \mathbb{R}$)*:*

$$a_{11}x^2+a_{22}y^2+a_{33}z^2+2a_{12}xy+2a_{13}xz+2a_{23}yz+2a_{14}x+2a_{24}y+2a_{34}z+a_{44}=0.$$

In order to allow GeoGebra to compute the intersection curve of two quadrics by using the algorithm in [11] we only need to use resultants and subresultants. We will compute the intersection curve of two quadrics $\mathcal{E}_1$ and $\mathcal{E}_2$ presented by their implicit equations:

$$f(x,y,z) = z^2 + p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = z^2 + q_1(x,y)z + q_0(x,y)$$

with $\deg(p_1) \leq 1$, $\deg(p_0) \leq 2$, $\deg(q_1) \leq 1$ and $\deg(q_0) \leq 2$. Since p_1 and q_1 are two polynomials of degree one:

$$p_1 = a_1x + a_2y + a_3 \qquad q_1 = b_1x + b_2y + b_3.$$

The polynomials p_0 and q_0 have degree two:

$$p_0 = a_4x^2 + a_5xy + a_6y^2 + a_7x + a_8y + a_9 \quad q_0 = b_4x^2 + b_5xy + b_6y^2 + b_7x + b_8y + b_9.$$

Other cases (ie degree in z smaller than 2) can be considered too (details can be found in [11]).

As usual (see [11]), to determine the projection of the intersection curve on the plane $z = 0$, resultants will be used.

Definition 2. *Let f and g be the two polynomials in $\mathbb{R}[x, y, z]$*

$$f(x, y, z) = z^2 + p_1(x, y)z + p_0(x, y) \qquad g(x, y, z) = z^2 + q_1(x, y)z + q_0(x, y)$$

($\deg(p_1(x, y)) \leq 1$, $\deg(p_0(x, y)) \leq 2$, $\deg(q_1(x, y)) \leq 1$ and $\deg(q_0(x, y)) \leq 2$) defining the quadrics whose intersection curve is to be computed. Then the Sylvester resultant of f and g, with respect to z, is equal to:

$$\mathbf{S}_0(x, y) \stackrel{\text{def}}{=} \mathbf{Resultant}(f, g; z) = \begin{vmatrix} 1 & p_1(x,y) & p_0(x,y) & 0 \\ 0 & 1 & p_1(x,y) & p_0(x,y) \\ 1 & q_1(x,y) & q_0(x,y) & 0 \\ 0 & 1 & q_1(x,y) & q_0(x,y) \end{vmatrix}$$

$$= (p_0(x, y) - q_0(x, y))^2 - (p_1(x, y) - q_1(x, y)) \begin{vmatrix} p_0(x,y) & p_1(x,y) \\ q_0(x,y) & q_1(x,y) \end{vmatrix}.$$

The projection of the intersection curve is contained in the curve of $\mathbb{R}^2$ defined implicitly by $\mathbf{S}_0(x, y) = 0$.

Computing the intersection of the two quadrics defined by f and g is equivalent to solve in $\mathbb{R}$ the system of polynomial equations $f(x, y, z) = 0, g(x, y, z) = 0$ which is equivalent to solve

$$\mathbf{S}_0(x, y) = 0 \wedge (q_1(x, y) - p_1(x, y))\, z + (q_0(x, y) - p_0(x, y)) = 0. \qquad (1)$$

These two equations correspond to the subresultants of index 0 and 1 of f and g with respect to z (see [11]).

When f or g have degree 1 in z, $\mathbf{S}_0(x, y)$ is also the resultant of f and g with respect to z and the subresultant of index 1 is one of the quadrics of degree 1. The case of both equations with degree zero reduces to the intersection of two conics and, for sake of simplicity, will not be included here.

3 Implementation of the Algorithm in GeoGebra

3.1 3D Capabilities of GeoGebra for Intersecting Two Quadrics

GeoGebra allows us to represent and determine the intersection curve of some quadrics with a plane by using the command *IntersectPath*. We can also use the command *IntersectConic* to determine the intersection of two quadrics when this can be characterised as the intersection of a plane with a quadric [1]. Figure 1 provides some examples of these cases.

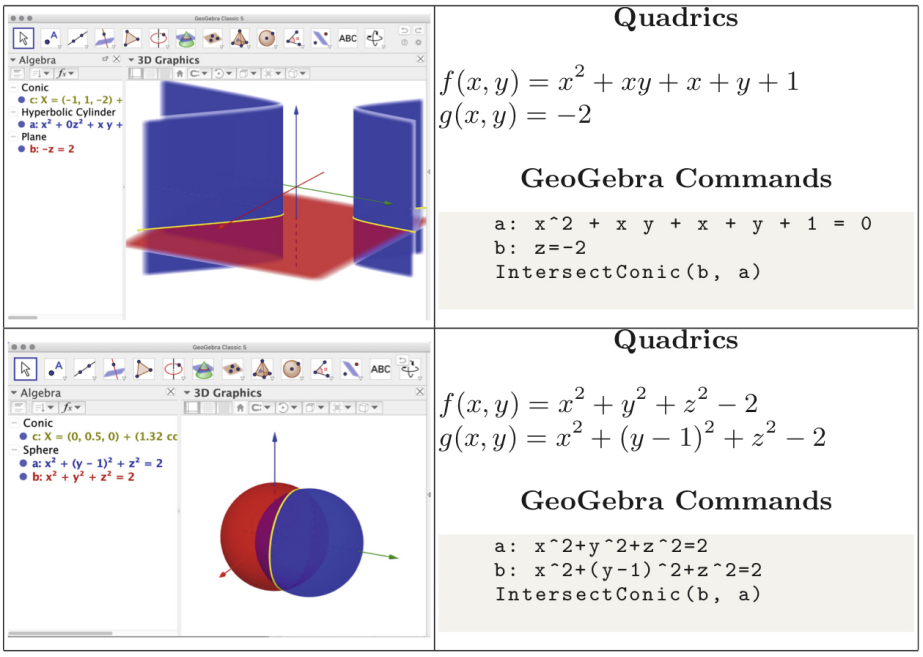


Fig. 1. Some cases when GeoGebra can determine the intersection of two quadrics.

3.2 The Algorithm

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two quadrics in $\mathbb{R}^3$ defined by $f(x,y,z) = 0$ and $g(x,y,z) = 0$ respectively. For computing the intersection curve of $\mathcal{E}_1$ and $\mathcal{E}_2$ in GeoGebra we will consider three cases depending on the degree in z of f and g: both of them with degree two, one with degree two and the other one with degree one and both of them with degree one. The cases of one equation (or both) with degree 0, for sake of simplicity, will not be included here but they follow the same strategy (see [11]). For the three cases considered here, we will show how the algorithm in [11] can be implemented in GeoGebra by adapting it to the special characteristics of this software in order to get a more efficient behavior of GeoGebra when computing the intersection curve of the two considered quadrics.

The method to be implemented in GeoGebra will be as follows (see [11]): starts by computing the projection of the intersection curve of $\mathcal{E}_1$ and $\mathcal{E}_2$ (the so-called cutcurve), continues with its analysis (paying special attention to its singular points) and ends with its lifting.

Two Quadrics of Degree 2 in z
Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two quadrics defined by:

$$f(x,y,z) = z^2 + p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = z^2 + q_1(x,y)z + q_0(x,y).$$

The cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ is the set

$$\left\{(x,y) \in \mathbb{R}^2 : \mathbf{S}_0(x,y) = 0, p_1(x,y)^2 - 4p_0(x,y) \geq 0, q_1(x,y)^2 - 4q_0(x,y) \geq 0\right\}.$$

The curves defined by $p_1(x,y)^2 - 4p_0(x,y) = 0$ and $q_1(x,y)^2 - 4q_0(x,y) = 0$ are called the silhouette curves of $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively.

The way of proceeding here will be the following one:

(a) Compute $\mathbf{S}_0(x,y)$.
(b) Compute the region

$$\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2} = \left\{(x,y) \in \mathbb{R}^2 : p_1(x,y)^2 - 4p_0(x,y) \geq 0, q_1(x,y)^2 - 4q_0(x,y) \geq 0\right\}$$

where the cutcurve lives: requires to compute the region defined by the two silhouette curves which are two conics.
(c) Compute the singular points of the cutcurve on the line $p_1 = q_1$.
(d) Compute the singular points of the cutcurve outside the line $p_1 = q_1$.
(e) Compute "enough" regular points of the cutcurve (either in closed form or through a discretization) and their lifting.
(f) Compute the lifting of the singular points of the cutcurve.

As seen in [11], we have:

$$\mathbf{S}_0(x,y) = \frac{1}{16}\left[(p_1 - q_1)^4 + (\Delta_{\mathcal{E}_1} - \Delta_{\mathcal{E}_2})^2 - 2(p_1 - q_1)^2(\Delta_{\mathcal{E}_1} + \Delta_{\mathcal{E}_2})\right] \quad (2)$$

where $\Delta_{\mathcal{E}_1} = p_1^2 - 4p_0$ and $\Delta_{\mathcal{E}_2} = q_1^2 - 4q_0$.

Fig. 2. Silhouette curves of two quadrics (in red and blue) and the cutcurve (in green)

The cutcurve and the region $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$ where the cutcurve lives can be defined in GeoGebra in closed form by using the commands[1].

```
*The validation region within a list *
l1:={((a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y +a6 y^2+a7 x+a8 y+a9))>=0,((b1 x+b2 y
    +b3)^2-4 (b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9))>=0}
*Definition of the curve defined by the resultant *
S0:=Implicitcurve(1 / 16 ((a1 x+a2 y+a3-(b1 x+b2 y+b3))^4+((a1 x+a2 y+a3)^2-4
     (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)-((b1 x+b2 y+b3)^2-4 (b4 x^2+ b5 x y+
    b6 y^2+b7 x+b8 y+b9)))^2-2(a1 x+a2 y+a3-(b1 x+b2 y+b3))^2 ((a1 x+a2 y+a3)
    ^2-4 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)+(b1 x+b2 y+b3)^2-4 (b4 x^2+b5 x
    y+b6 y^2+b7 x+b8 y+b9))))
*Definition of the cutcurve*
cond(x,y):=Product(l1) *CAS*
K1:=point(S0)
K2:=If(cond(x(K1),y(K1))>0,K1)
cutcurve:=Locus(K2, K1)
```

[1] In the GeoGebra commands presented here, we denote $x_\#$ by `x#`.

The lifting of the regular points of the cutcurve is determined by:

$$z = \frac{p_0(\alpha, \beta) - q_0(\alpha, \beta)}{q_1(\alpha, \beta) - p_1(\alpha, \beta)}. \tag{3}$$

All regular points of the cutcurve are outside the line $p_1(x, y) = q_1(x, y)$. The same formula is to be used for lifting the singular points of the cutcurve outside the line $p_1(x, y) = q_1(x, y)$.

Some cutcurves can not be parameterized easily in closed form even by using radicals: since the cutcurve has degree at most four, it can be parameterized by using radicals but this parameterization is quite complicated and does not take into account properly the real branches (see for example [10]). This is the reason why the lifting of the regular points of the cutcurve will be made on the discretization of the cutcurve branches (when such a parameterization is too involved). To do that in GeoGebra, we need to define a free point, A, on the cutcurve and compute its lifting by using the GeoGebra commands:

```
*Define a free point on the cutcurve*
A=Point(cutcurve)
*Lifting of point A*
P=(x(A),y(A),(a4 x(A)^2+a5 x(A) y(A)+a6 y(A)^2+a7 x(A)+a8 y(A)+a9-(b4 x(A)^2+
    b5 x(A) y(A)+b6 y(A)^2+b7 x(A)+b8 y(A)+b9))/ (b1 x(A)+b2 y(A)+b3-(a1 x(A)
    +a2 y(A)+a3)))
*Lifting of the cutcurve*
Locus(P,A)
```

In practice, if the cutcurve has singular points then `Locus`, close to some of these points, does not work very properly. This is the reason why the singular points will be determined separately and the lifting of the regular points close to them will be made in a different way.

Singular points of the cutcurve can be classified as follows: those lying on the intersection of the cutcurve and the line $p_1(x, y) = q_1(x, y)$ and those lying outside the line, $p_1(x, y) = q_1(x, y)$, always coming from tangential intersection points of $\mathcal{E}_1$ and $\mathcal{E}_2$. Singular points of the cutcurve on the line $p_1(x, y) = q_1(x, y)$ are determined by solving $\Delta_{\mathcal{E}_1}(x, y) - \Delta_{\mathcal{E}_2}(x, y) = 0 \wedge p_1(x, y) = q_1(x, y)$ which amounts to solve an univariate equation of degree 2. These singular points are stored in the list s. Only those singular points stored in s in $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$ will be stored in the list *s2*.

```
s:=Solutions({a1 x+a2 y+a3=b1 x+b2 y+b3, (a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6
     y^2+ a7 x+a8 y+a9)=(b1 x+b2 y+b3)^2-4 (b4 x^2+ b5 x y+b6 y^2+b7 x+b8 y+
    b9)},{x, y})  *CAS*
*Testing points of s that live in the validation region*
s2:=RemoveUndefined(Sequence(If(cond(Element(s, i, 1), Element(s, i, 2)) > 0,
     {Element(s, i, 1), Element(s, i, 2)}), i, 1, Length(s)))
```

The lifting of the points generated by the `Locus` GeoGebra function works properly on the regular points of the cutcurve and on its singular points outside the line $p_1(x, y) = q_1(x, y)$. Therefore we only need to deal with the lifting of the points in the list *s2* but also with those regular points of the cutcurve which are close to the points in the list *s2*. For that we remove from the cutcurve the circles with center the points in the list *s2* and radius $eps \in \mathbb{R}^+$ (defined by the user). To achieve this goal, we use the following GeoGebra command:

```
*Sequence of circles stored in a list l2*
l2:=Sequence((x-Element(s2,i,1))^2+(y-Element(s2,i,2))^2>=eps^2,i,1,Length(s2
    ))
```

The points on the cutcurve to be lifted by using (3) are characterized by the conditions defining the region $\mathcal{A}_{\mathcal{E}_1,\mathcal{E}_2}$ excluding the circles around the points on the line $p_1(x,y) = q_1(x,y)$ and on the cutcurve. In order to do this we define the list *l3* by using the conditions mentioned above, thus defining a new point, B existing only if the conditions in *l3* are verified. Making use of the point B, we define in GeoGebra a new curve, called *adaptcutcurve*, by using the following commands:

```
*Defining a list with both conditions*
l3:=Join(l1,l2)
*Function returning zero if and only if at least one condition is not
    verified*
d(x,y):=Product(l3) *CAS*
B:=If(d(x(A),y(A))>0,A)
*Representing the cutcurve excluding the circles around the singular points
    on the line p_1(x,y)=q_1(x,y)*
adaptcutcurve:=If(Length(s2)!=0, Locus(B, A), cutcurve)
```

Let C be a point in the curve *adaptcutcurve*. This point runs along the whole cutcurve except for several small circles around the singular points on the line $p_1(x,y) = q_1(x,y)$. The lifting of the curve *adaptcutcurve* is determined by the following GeoGebra commands:

```
*Defining the lifting of the adapted cutcurve*
C:=Point(adaptcutcurve)
P:=(x(C),y(C),(a4 (x(C))^2+a5 (x(C)) (y(C)) +a6 (y(C))^2+a7 (x(C))+a8 (y(C))+
    a9-(b4 (x(C))^2+b5 (x(C)) (y(C))+b6 (y(C))^2+b7 (x(C))+b8 (y(C))+b9))/(b1
     (x(C))+b2 (y(C)) +b3-(a1 (x(C))+a2 (y(C))+a3)))
adaptlift:=Locus(P,C)
```

The singular points of the cutcurve on the line $p_1(x,y) = q_1(x,y)$, stored in *s2*, will be lifted by using:

$$z = \frac{-p_1(\alpha,\beta) \pm \sqrt{p_1(\alpha,\beta)^2 - 4p_0(\alpha,\beta)}}{2} \tag{4}$$

or

$$z = \frac{-q_1(\alpha,\beta) \pm \sqrt{q_1(\alpha,\beta)^2 - 4q_0(\alpha,\beta)}}{2} \tag{5}$$

The GeoGebra commands performing the lifting of these points are:

```
*Definition of functions*
lift1:=(-(a1 x+a2 y+a3)+sqrt((a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6 y^2+a7 x+a8
    y+a9)))/2
lift2:=(-(a1 x+a2 y+a3)-sqrt((a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6 y^2+a7 x+a8
    y+a9)))/2
*Definition of the lift*
sing1:=Sequence((Element(s2,i,1),Element(s2,i,2),lift1(Element(s2,i,1),
    Element(s2,i,2))),i,1,Length(s2))
sing2:=Sequence((Element(s2,i,1),Element(s2,i,2),lift2(Element(s2,i,1),
    Element(s2,i,2))),i,1,Length(s2))
```

As mentioned before (see [11]) singular points of the cutcurve not belonging to the line $p_1(x,y) = q_1(x,y)$ come from tangential intersection points of $\mathcal{E}_1$ and $\mathcal{E}_2$. The determination of these points is quite complicated but they will

be very easy to lift: by continuity the lifting of the points produced by the `Locus` GeoGebra function around these singular points produces automatically their lifting to the intersection curve (and it is not necessary to compute them explicitly).

Regular points of the cutcurve in the silhouette curves (see Fig. 2) can be determined by solving (according to Eq. (2)):

$$2(p_0 + q_0) = p_1 q_1, \Delta_{\varepsilon_1} = 0 \qquad \text{and} \qquad 2(p_0 + q_0) = p_1 q_1, \Delta_{\varepsilon_2} = 0$$

which amounts, for each system, to intersect two conics. In order to determine the lifting of the regular points of the cutcurve in the silhouette curves we must use the following GeoGebra commands:

```
silh1:=Solutions({2*((a4 x^2+a5 x y+a6 y^2+ a7 x+a8 y+a9)+(b4 x^2+b5 x y+b6 y
     ^2+b7 x+b8 y+b9))=(a1 x+a2 y+a3)*(b1 x+b2 y+b3),((a1 x+a2 y+a3)^2-4 (a4 x
     ^2+a5 x y+a6 y^2+a7 x+a8 y+a9))},{x,y}) *CAS*
silh2:=Solutions({2*((a4 x^2+a5 x y+a6 y^2 +a7 x+a8 y+a9)+(b4 x^2+b5 x y+b6 y
     ^2+b7 x+ b8 y+b9))=(a1 x+a2 y+a3)*(b1 x+b2 y+ b3),((b1 x+b2 y+b3)^2-4 (b4
      x^2+b5 x y+b6 y^2+b7 x+b8 y+b9))},{x,y}) *CAS*
*Definition of the function that lifts regular points*
lift:=(a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9-(b4 x^2 +b5 x y+b6 y^2+b7 x+b8 y+b9)
     )/(b1 x+b2 y+b3-(a1 x+a2 y+a3))
ls1:=Sequence((Element(silh1,i,1), Element(silh1,i,2), lift(Element(silh1,i
     ,1), Element(silh1,i,2))),i,1,Length(silh1))
ls2:=Sequence((Element(silh2,i,1), Element(silh2,i,2), lift(Element(silh2,i
     ,1), Element(silh2,i,2))),i,1,Length(silh2))
```

At the neighbourhood of any singular point in $p_1(x, y) = q_1(x, y)$ we will compute n points for each branch to the left and to the right of the considered singular point. Note that near the silhouette curves, GeoGebra might not represent properly the cutcurve due to precision problems: this is specially important when the cutcurve is tangent to one of the silhouette curves. This discretization for each branch to the left and to the right of every singular point is the way we use to avoid the problems brought by GeoGebra into this situation.

```
*Defining x values near to singular points*
amp=eps/n
seq:=Sequence(Sequence(Element(s2,j,1) - i amp, i, -n, n),j,1,Length(s2))
*Definition of the curve inside the validation region and near to singular
     points*
e:=If((b1 x+b2 y+b3)^2-4 (b4 x^2+b5 x y+b6 y^2+b7 x +b8 y+b9)>=0 && (a1 x+a2
     y+ a3)^2-4 (a4 x^2+a5 x y+ a6 y^2+a7 x+a8 y+a9)>=0,1/16 ((a1 x+a2 y+a3-(
     b1 x+ b2 y+b3))^4+((a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)
     -((b1 x+b2 y+b3)^2-4 (b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9)))^2-2(a1 x+a2 y+
     a3-(b1 x+b2 y+b3))^2 ((a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6 y^2+ a7 x+a8 y
     +a9)+(b1 x + b2 y + b3)^2-4 (b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9))))
*Compute y coordinate of every point on the cutcurve from every x value
     stored in seq *
yseq:=Sequence(Sequence(NSolve(e(Element(seq,j,i),y),y),i,1,2*n+1),j,1,Length
     (s2)) *CAS*
```

For every x value stored in *seq* and y value in *yseq* we define a list, grouping x with y, and storing them in the list *pair* defined by using the following GeoGebra commands:

```
pair:=Sequence(Sequence(Sequence(If(Element(yseq,i,j)=={},(Element(seq,i,j),
     maxim),(Element(seq,i,j),RightSide(Element(Element(yseq,i,j),k)))),k,1,If
     (Element(yseq,i,j)=={},1,Length(Element(yseq,i,j)))),j,1,2*n+1),i,1,
     Length(s2))  *CAS*
```

The lifting of this set of points, the list *fill* below, will be constructed by using the function *lift*, (3), applied to any regular point whose distance to the singular one is between *eps* and *eps/10*. On the other hand, points whose distance to the singular point is less than *eps/10* will be lift by using the functions *lift1* (4) and *lift2* (5).

```
*Definition of the lifting*
fill:=Sequence(Sequence(Sequence(If(distance((x(Element(pair, k, j, i)), y(
     Element(pair, k, j, i))),(Element(s2,k,1),Element(s2,k,2)))<eps/10,(x(
     Element(pair, k, j, i)), y(Element(pair, k, j, i)), lift1(x(Element(pair,
      k, j, i)), y(Element(pair, k, j, i)))),If(eps/10<distance((x(Element(
     pair, k, j, i)), y(Element(pair, k, j, i))),(Element(s2,k,1),Element(s2,k
     ,2)))<eps,(x(Element(pair, k, j, i)), y(Element(pair, k, j, i))), lift(x(
     Element(pair, k, j, i)), y(Element(pair, k, j, i)))))), i, 1, Length(
     Element(pair,k,j))), j, 1, 2*n+1), k, 1, Length(s2))
```

Example 3. Let f and g be two ellipsoids defined by: $f(x,y,z) = z^2 + (-2/3x + 2/3y)\,z + 1/3x^2 + 2/3y^2 - 1/3$ $g(x,y,z) = z^2 + (-2/17x + 1/17y - 2/17)\,z + 1/4x^2 + 1/17y^2 + 2/17x - 3/17$.

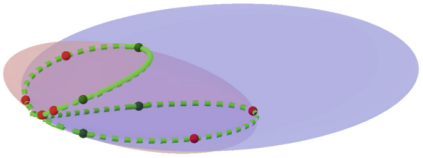


Fig. 3. Two ellipsoids, the intersection curve and relevant points.

In Fig. 3, green points result from the lifting of the singular points of the cutcurve, while the red ones result from the lifting of common points between the silhouette curves and the cutcurve.

In order to create a tool, in GeoGebra, it is necessary to use the menu *Tools* and then the option *Create New Tool*. All eighteen parameters, that define the two quadrics and the parameters n and *eps* must be selected as input. For the output it is necessary to select the lists *sing1, sing2, ls1, ls2, fill* and the locus *adaptlift*.

In the particular case when $p_1(x,y) \equiv q_1(x,y)$, the cutcurve verifies:

$$\mathbf{S}_0(x,y) = \left(\Delta_{\mathcal{E}_1} - \Delta_{\mathcal{E}_2}\right)^2/16 = (p_0 - q_0)^2.$$

The condition $\mathbf{S}_0(x,y) = 0 \wedge p_1(x,y) = q_1(x,y)$ is equivalent to $p_0(x,y) = q_0(x,y)$. Thus, all the points of the cutcurve are considered as singular and their lifting (by using (4) or (5)) is defined by:

```
part1:=If(a1 x + a2 y + a3 = b1 x + b2 y + b3,Sequence((Element(s,i,1),
     Element(s,i,2), lift1(Element(s,i,1), Element(s,i,2))),i,1,Length(s)))

part2:=If(a1 x + a2 y + a3 = b1 x + b2 y + b3,Sequence((Element(s,i,1),
     Element(s,i,2), lift2(Element(s,i,1), Element(s,i,2))),i,1,Length(s)))
```

It should be pointed out that, in this case, GeoGebra gives a parameterization of the intersection curve of the two considered quadrics.

Quadrics of Degree 2 and 1 in z

Let f and g be the polynomials in $\mathbb{R}[x,y,z]$ defined by:

$$f(x,y,z) = z^2 + p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = q_1(x,y)z + q_0(x,y)$$

In this case the cutcurve is defined by $\mathbf{S}_0(x,y) = p_0q_1^2 - p_1q_0q_1 + q_0^2$ in the region defined by $p_1^2 - 4p_0 \geq 0$. The lifting of any regular point (α,β) of the cutcurve will be determined by $z = -q_0(\alpha,\beta)/q_1(\alpha,\beta)$. Points of the cutcurve determined by $q_1(x,y) = 0$ can only be lifted by using (4) and, in this case, we only have one silhouette curve. The GeoGebra commands to use start defining the following elements:

```
l1:={(a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)>=0}
S0:=ImplicitCurve((b1 x+b2 y+b3)^2 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9) - (a1
     x+a2 y+a3) (b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9) (b1 x+b2 y+b3)+(b4 x^2 +b5
      x y+b6 y^2+b7 x+b8 y+b9)^2)
cond(x,y):=(a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)
lift:=(-(b4 x^2+b5 x y+b6 y^2+ b7 x + b8 y + b9)) / (b1 x + b2 y + b3)
P:=(x(C),y(C),(-(b4 (x(C))^2+b5 (x(C)) (y(C))+b6 (y(C))^2+b7 (x(C))+b8 (y(C))
      + b9))/(b1 (x(C))+b2 (y(C))+b3))
s:=Solutions({(b4 x^2 + b5 x y + b6 y^2 + b7 x + b8 y + b9)=0,(b1 x + b2 y +
     b3)=0},{x, y})
silh1:=Solutions({(b1 x+b2 y+b3)^2 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9) - (a1
     x + a2 y + a3) (b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9) (b1 x+b2 y+b3)+(b4 x
     ^2+b5 x y+b6 y^2+b7 x+b8 y+b9)^2=0,(a1 x+a2 y+a3)^2 - 4 (a4 x^2+a5 x y+
     a6 y^2+a7 x+a8 y+a9)=0},{x,y})
e:=If((a1 x+a2 y+a3)^2-4 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)>=0,(b1 x+b2 y +
     b3)^2 (a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)-(a1 x+a2 y+a3) (b4 x^2+b5 x y+
     b6 y^2+b7 x+b8 y+ b9) (b1 x+b2 y+b3)+(b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9)
     ^2)
```

The computation of the intersection curve in this case follows the same GeoGebra strategy shown previously.

Example 4. Let f be the hyperboloid of one sheet and g the hyperbolic paraboloid defined by:
$f(x,y,z) = z^2 + 3z - x^2 + y^2 - 3$
$g(x,y,z) = (x+y)z - 2x$
The intersection curve can be represented by GeoGebra: see Fig. 4.

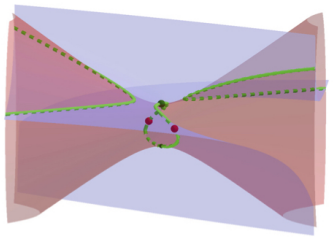


Fig. 4. Hyperboloid of one sheet and hyperbolic paraboloid intersection curve and relevant points in GeoGebra

Two Quadrics of Degree 1 in z

Let f and g be the polynomials in $\mathbb{R}[x,y,z]$ defined by:

$$f(x,y,z) = p_1(x,y)z + p_0(x,y) \qquad g(x,y,z) = q_1(x,y)z + q_0(x,y)$$

The cutcurve is defined by $\mathbf{S}_0(x,y) = p_1q_0 - p_0q_1$. Note that, in this case, $\mathbf{S}_0$ is a polynomial of degree three, at most. As before, in some cases GeoGebra gives us the exact parameterization of the cutcurve. However, in general, the way to proceed will be similar to the previous cases. As seen in [11], when $p_1(\alpha,\beta) \neq 0$ or $q_1(\alpha,\beta) \neq 0$, the lifting of (α,β) is given by:

$$z = -\frac{p_0(\alpha,\beta)}{p_1(\alpha,\beta)} \qquad or \qquad z = -\frac{q_0(\alpha,\beta)}{q_1(\alpha,\beta)},$$

respectively. If $p_1(\alpha,\beta) = p_0(\alpha,\beta) = 0$ then $p_0(\alpha,\beta) = q_0(\alpha,\beta) = 0$ and the line $\{(\alpha,\beta,z) : z \in \mathbb{R}\}$ is in the intersection curve.

In this case we use in GeoGebra the following commands:

```
e(x,y):=(a1 x+a2 y+a3)*(b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9)-(a4  x^2+a5 x y+a6
     y^2+a7 x+a8 y+a9)*(b1 x+b2 y+b3)
cutcurve:=ImplicitCurve(e(x,y))

*Determine points that p1=0 && p0=0 *
s2:=Solutions({a1 x+a2 y+a3=0,(a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+ a9)=0},{x, y})
    *CAS*
*Lifting points that p1!=0*
lift:=-(a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)/(a1 x+a2 y+a3)

*Define the region near to points that p1=0 && p0=0*
l3:=Sequence((x - Element(s2, i, 1))^2+(y - Element(s2, i, 2))^2 >=eps^2,i,1,
    Length(s2))

*Define adaptcutcurve - curve without region near to points that p1=0 & p0=0*
A=Point(cutcurve)
d(x,y):=Product(l3) *CAS*
B:=If(d(x(A),y(A))>0,A)
K1:=Point(cutcurve)
K2:=K1
adaptcutcurve:=If(Length(s2) != 0, Locus(B, A), Locus(K2,K1))
C:=Point(adaptcutcurve)
P:=(x(C),y(C),-(a4 (x(C))^2+a5 (x(C)) (y(C))+a6 (y(C))^2+a7 (x(C))+a8 (y(C))+
    a9)/(a1 (x(C))+a2 (y(C))+a3))

*Lifting of points that p1=0 && p0=0 *
lift1:=-((b4 x^2 + b5 x y + b6 y^2 + b7 x + b8 y + b9)/(b1 x + b2 y + b3))
sing1:=Sequence((Element(s2,i,1),Element(s2,i,2),lift1(Element(s2,i,1),
    Element(s2,i,2))),i,1,Length(s2))

*Determine points that p1=0 && p0=0 && q1=0 && q0=0 *
s3:=Solutions({a1 x + a2 y + a3 = 0,(a4 x^2+a5 x y+a6 y^2+a7 x+a8 y+a9)=0,(b1
     x + b2 y + b3)=0,b4 x^2+b5 x y+b6 y^2+b7 x+b8 y+b9=0},{x, y})

*Define vertical lines*
line:=Sequence(Line((Element(s3,i,1), Element(s3,i,2)),zAxis),i,1,Length(s3))
```

Example 5. Let f be the hyperbolic paraboloid and g the hyperboloid of one sheet defined by:
$f(x,y,z) = xz + x^2 + 2y - 1$
$g(x,y,z) = yz + x^2 + y^2 - 2x$
The parameterization of the intersection curve can be computed by GeoGebra by using the presented approach (Fig. 5).

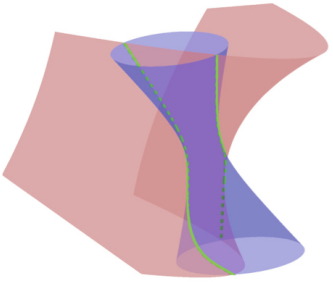


Fig. 5. Intersection between the hyperboloid of one sheet and the hyperbolic paraboloid.

4 Conclusion

For computing in GeoGebra the intersection curve of two quadrics, we have adapted to GeoGebra the algorithm in [11]. Applying this implementation to fifty

examples randomly generated, it produced the intersection curve in a few seconds for each case. Despite some limitations of the `Locus` function in GeoGebra the implementation worked efficiently in all cases. Sometimes the performance of the software was a little slow namely when there were many singular points. Another topic to be considered deals with the values of the parameters n and eps which need to be carefully chosen by the user. In conclusion, our goal was attained, producing an implementation in GeoGebra of the algorithm in [11].

References

1. Dos Santos, J.M.: Intersection of two surfaces in GeoGebra. Revista do Instituto GeoGebra de São Paulo **6**(2), 04–09 (2017)
2. Dupont, L., Lazard, S., Lazard, D., Petitjean, S.: Near-optimal parameterization of the intersection of quadrics. In: Proceedings of the Annual Symposium on Computational Geometry (2003). https://doi.org/10.1145/777829.777830
3. Dupont, L., Lazard, D., Lazard, S., Petitjean, S.: Near-optimal parameterization of the intersection of quadrics: II. A classification of pencils. J. Symbolic Comput. (2008). https://doi.org/10.1016/j.jsc.2007.10.012
4. Dupont, L., Lazard, D., Lazard, S., Petitjean, S.: Near-optimal parameterization of the intersection of quadrics: III. Parameterizing singular intersections. J. Symbolic Comput. (2008). https://doi.org/10.1016/j.jsc.2007.10.007
5. Dupont, L., Lazard, D., Lazard, S., Petitjean, S.: Near-optimal parameterization of the intersection of quadrics: I. The generic algorithm. J. Symbolic Comput. (2008). https://doi.org/10.1016/j.jsc.2007.10.006
6. Levin, J.: A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. Commun. ACM **10**, 555–563 (1976). https://doi.org/10.1145/360349.360355
7. Levin, J.Z.: Mathematical models for determining the intersections of quadric surfaces. Comput. Graph. Image Process. **11**(1), 73–87 (1979)
8. Mourrain, B., Tecourt, J.P., Teillaud, M.: On the computation of an arrangement of quadrics in 3D. Comput. Geom. Theory Appl. (2005). https://doi.org/10.1016/j.comgeo.2004.05.003
9. Schomer, E., Wolpert, N.: An exact and efficient approach for computing a cell in an arrangement of quadrics. Comput. Geom. Theory Appl. (2006). https://doi.org/10.1016/j.comgeo.2004.02.007
10. Sendra, J.R., Sevilla, D.: Radical parametrizations of algebraic curves by adjoint curves. J. Symbolic Comput. **46**(9), 1030–1038 (2011)
11. Trocado, A., Gonzalez-Vega, L.: On the intersection of two quadrics (2018, Submitted). http://arxiv.org/abs/1903.06983
12. Wang, W., Goldman, R., Tu, C.: Enhancing Levin's method for computing quadric-surface intersections. Comput. Aided Geom. Des. (2003). https://doi.org/10.1016/S0167-8396(03)00081-5

Chapter 4

Using Maple to compute the intersection curve of two quadrics: improving the `intersectplot` command (see [21])

We present here the implementation in Maple of a new method to determine the intersection curve of two quadrics through projection onto a plane and lifting. In some cases, it will be possible to determine the exact parameterisation of the intersection curve (involving radicals if needed) and, in others, the output (topologically correct) will be presented as the lifting of the discretisation of the branches of the projection curve once its singular points have been fully determined. The way the lifting will be made is the main criteria followed to analyse the cutcurve, the projection of the intersection curve to be computed.

The Maple `intersectplot` command plots the intersection curve in three-dimensional space between a pair of two-dimensional surfaces. We will show how this implementation in Maple improves the results produced by the `intersectplot` command when computing the intersection curve between two quadrics in 3D.

Using Maple to compute the intersection curve of two quadrics: improving the `intersectplot` command*

Laureano Gonzalez–Vega[1][0000−0002−3934−3890] and Alexandre Trocado[2][0000−0001−5589−8100]

[1] Universidad de Cantabria, Spain
laureano.gonzalez@unican.es
[2] Universidade Aberta, Portugal
mail@alexandretrocado.com

Abstract. The Maple `intersectplot` command plots the intersection curve in three-dimensional space between a pair of two-dimensional surfaces. We will present the implementation in Maple of a new algorithm computing the intersection curve between two quadrics in 3D that improves the results produced by the `intersectplot` command.

Keywords: Quadrics · intersection curve · resultant · Maple

1 Introduction

We present here the implementation in Maple of a new method to determine the intersection curve of two quadrics through projection onto a plane and lifting. In some cases, it will be possible to determine the exact parameterisation of the intersection curve (involving radicals if needed) and, in others, the output (topologically correct) will be presented as the lifting of the discretisation of the branches of the projection curve once its singular points have been fully determined. The way the lifting will be made is the main criteria followed to analyse the cutcurve, the projection of the intersection curve to be computed.

The introduction of algorithms for computing the intersection of two quadrics dates back to the late 1970s. The representation and the definition of quadrics' intersection has been a relevant problem to solve over the last decades. Levin in 1976 and 1979 ([7], [8]) introduced a method failing when the intersection curve is singular and even generates results that are not topologically correct. Levin's method was improved by Wang et al. ([13]) making it capable of computing geometric and structural information; besides, Dupont et al. (see [2]) succeeded in finding parameterizations that overcame the fact that Levin's method generated formulas that were not suited for further symbolic processing. On the

* First author is partially supported by the Spanish Ministerio de Economa y Competitividad and by the European Regional Development Fund (ERDF), under the project MTM2017-88796-P.

other hand, Mourrain et al. ([9]) studied a sweeping algorithm for computing the arrangement of a set of quadrics in $\mathbb{R}^3$ that reduces the intersection of two quadrics to a dynamic two-dimentional problem. L. Dupont et al. ([3], [4] and [5]) proposed algorithms that enable to compute in practice an exact form of the parameterization of two arbitrary quadrics with rational coefficients. These algorithms represented a substantial improvement over Levin's pencil method and its subsequent refinements. A different approach is based on the analysis of the projection of the intersection curve by using resultants ([10] and [11]). It is the algorithm in Trocado and Gonzalez-Vega ([11]) the one we have used to analyse the behaviour of the Maple `intersectplot` command plotting the intersection curve in three-dimensional space of a pair of two-dimensional surfaces when these two surfaces are quadrics.

This paper is organised as follows: section 2 describes briefly the algorithm implemented in Maple, section 3 compares this implementation with the Maple `intersectplot` command and the last section draws several conclusions.

2 The algorithm

We introduce here a brief presentation of the algorithm in [11] that we have implemented in Maple in order to analyse the efficiency and accuracy of the Maple `intersectplot` command plotting the intersections in three-dimensional space of a pair of two-dimensional surfaces when these two surfaces are quadrics. The algorithm in [11] performs a very efficient Cylindrical Algebraic Decomposition ([1]) for the particular case of two quadrics since, in this case, the projection of the intersection curve has several properties making its analysis easier than expected.

Let f and g be the two polynomials in $\mathbb{R}[x, y, z]$

$$f(x, y, z) = z^2 + p_1(x, y)z + p_0(x, y) \qquad g(x, y, z) = z^2 + q_1(x, y)z + q_0(x, y)$$

with $\deg(p_1) \leq 1$, $\deg(p_0) \leq 2$, $\deg(q_1) \leq 1$ and $\deg(q_0) \leq 2$.

Let $\Delta_{\mathcal{E}_1}(x, y) = p_1(x, y)^2 - 4p_0(x, y)$ and $\Delta_{\mathcal{E}_2}(x, y) = q_1(x, y)^2 - 4q_0(x, y)$ be the discriminants of $f(x, y, z)$ and $g(x, y, z)$ (respectively) with respect to z. Let $\mathbf{S}_0(x, y)$ the resultant of f and g, with respect to z.

Computing the intersection of the two quadrics defined by f and g is equivalent to solving in $\mathbb{R}^3$ the polynomial system of equations

$$f(x, y, z) = 0, \qquad g(x, y, z) = 0.$$

The solution set to be computed, when non empty, may include curves and isolated points. Analyzing $\mathbf{S}_0(x, y) = 0$ in $\mathbb{R}^2$ will be called the projection step and moving the information obtained in $\mathbb{R}^2$ to $\mathbb{R}^3$ will be called the lifting step.

The curve in $\mathbb{R}^2$ defined by $\mathbf{S}_0(x, y) = 0$ is called the cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ and the curve in $\mathbb{R}^2$ defined by $\Delta_{\mathcal{E}_i}(x, y) = 0$ the silhouette of $\mathcal{E}_i$.

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two quadrics in $\mathbb{R}^3$ defined by $f(x, y, z) = 0$ and $g(x, y, z) = 0$ respectively. The cutcurve of $\mathcal{E}_1$ and $\mathcal{E}_2$ is the set

$$\left\{(x, y) \in \mathbb{R}^2 : \mathbf{S}_0(x, y) = 0, \Delta_{\mathcal{E}_1}(x, y) \geq 0, \Delta_{\mathcal{E}_2}(x, y) \geq 0\right\}.$$

The main ingredients of this approach are a detailed analysis of the cutcurve, its singular points and of its relation with the silhouette curves together with the using of an uniform way to perform the lifting of the cutcurve to the intersection curve of the two considered quadrics.

Concerning the analysis of the cutcurve we classify its singular points in two different types depending on how they will be lifted. Those belonging to the line $p_1(x, y) = q_1(x, y)$ are easy to compute and difficult to lift (but just solving a degree two equation) and those not in that line are more complicated to be determined but easier to lift.

This approach is not intended to classify the intersection curve between the two considered quadrics. Its main goal is to produce in a very direct way a description of the intersection curve which is topologically correct. This is the reason why we allow in the lifting of the cutcurve, when possible, the use of radicals or we rely on the discretisation of the branches of the cutcurve (easily to determine by knowing the points computed in that curve). Next example shows one case where the intersection curve is presented by a parameterisation involving radicals.

Example 1. Let f and g be the polynomials

$$f(x, y, z) = z^2 + xz + y \qquad g(x, y, z) = z^2 + yz + x$$

defining two hyperbolic paraboloids, $\mathcal{E}_1$ and $\mathcal{E}_2$, whose intersection curve is to be computed. To characterise the intersection curve of $\mathcal{E}_1$ and $\mathcal{E}_2$ we must analyse the cutcurve

$$\mathbf{S}_0(x, y) = (x - y)^2(x + y + 1) = 0$$

and its lifting. Lifting the line $x+y+1 = 0$ is easy since these points are regular points of the cutcurve and it is enough to solve the equation $f - g = 0$ with respect to z (which, in this case, is the subresultant of index 1 of f and g with respect to z). Lifting the line $x - y = 0$, requires to solve $f = 0$ or $g = 0$ with respect to z.

By using the following functions:

- For $x \in \mathbb{R}$, we define:

$$h_1(x) = x \qquad h_2(x) = -x - 1$$

- Let e_1 and e_2 be the functions defined by:

$$e_1\,(x, y) = -\frac{y}{2} + \frac{\sqrt{y^2 - 4x}}{2} \qquad e_2\,(x, y) = -\frac{y}{2} - \frac{\sqrt{y^2 - 4x}}{2}$$

the parameterisation of the intersection curve is given by the following three components:

- For $x \in \,]-\infty, 0] \cup [4, +\infty[$: $(x, h_1(x), e_1(x, h_1(x)))$.
- For $x \in \,]-\infty, 0] \cup [4, +\infty[$: $(x, h_1(x), e_2(x, h_1(x)))$.
- For $x \in \mathbb{R}$: $(x, h_2(x), 1)$ (the lifting the line $x + y + 1 = 0$).

In Figure 1 it is shown, both, the cutcurve and the intersection curve of the two considered quadrics. It is worth to remark here that the cutcurve contains two half-lines and that the intersection curve shows two connected components and one self-intersection point.

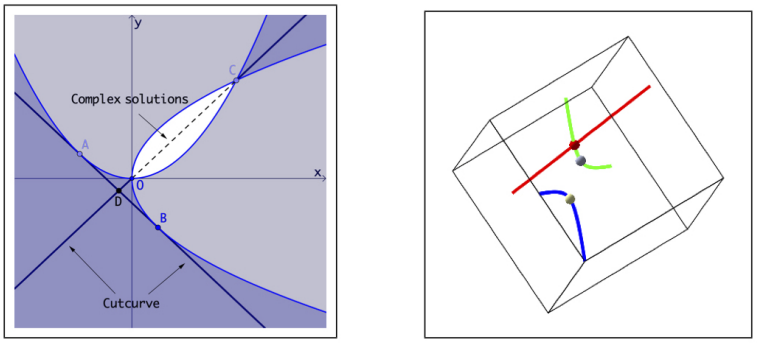


Fig. 1. (Left) The curcurve contains two half–lines. (Right) Intersection curve of $f = 0$ and $g = 0$: in red the lifting of the singular points of the cutcurve (including one complete line) and in gray those points where the parameterisation equations change.

3 The implementation and the comparison with the `intersectplot` command

The algorithm introduced in [11] and presented in the previous section has been implemented in Maple and its performance has been compared with the `intersectplot` command. We show next the advantages of our algorithm when compared with the `intersectplot` command.

First issue to mention is that `intersectplot` command works locally requiring to decide in advance the region where the intersection curve is to be computed. This means that some connected components of the intersection curve may be missing. Our algorithm does not miss any connected component of the intersection curve. Figure 2 shows how the `intersectplot` command needs to be applied several times in order to recover all the connected components of the intersection curve of the quadrics in Example 1 but with no guarantee that more connected components are missing (figure 1 shows the intersection curve as produced by the implementation of our algorithm).

Second issue to mention is that the `intersectplot` command may miss some points when the intersection curve is a finite set of isolated points. For example the quadrics defined by

$$4z^2 + 4x^2 - 4xy - 8xz + 3y^2 - 8y + 8z + 8 = 0$$

$$6z^2 + 4x^2 + 4xy - 8xz - 3y^2 - 16x + 8y + 16z + 8 = 0$$

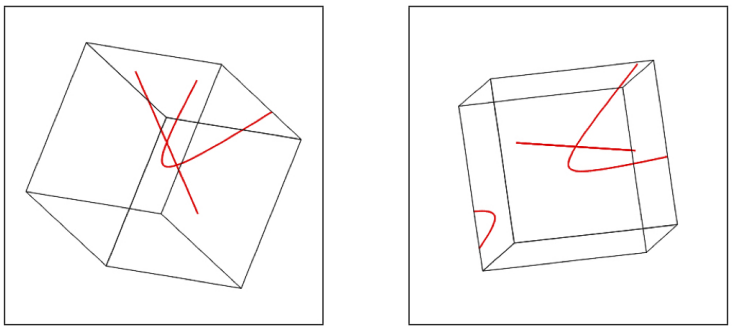


Fig. 2. Computing the intersection curve of $z^2+xz+y=0$ and $z^2+xz+y=0$ by using the `intersectplot` command. Left: in the box $[-3,3]^3$. Right: in the box $[-5,5]^3$.

only intersect at the points $(-1,-2/3,-2)$ and $(-1,2,-2)$. Using the `intersectplot` command in the boxes $[-3,3]^3$ and $[-1.5,-0.5]\times[-1,2.5]\times[-2.5,-1.5]$ produces the empty plot in both cases while our implementation produces the only two points in the intersection curve.

Third issue to mention concerns the quality of the output produced by the `intersectplot` command when the box does not fit adequately with the intersection curve. Figure 3 shows the output produced by the `intersectplot` command when computing the intersection curve of the quadrics

$$p(x,y,z) = -16xy + 24xz + 4y^2 + 8yz - 16z^2 + 16x + 16y - 40z - 32 = 0$$

$$q(x,y,z) = -8xy + 4xz + 2y^2 + 8yz - 8z^2 - 8x + 16y - 20z - 16 = 0$$

considering different boxes. We can conclude here that, depending on the size of the box, the quality and accuracy of the output change drastically.

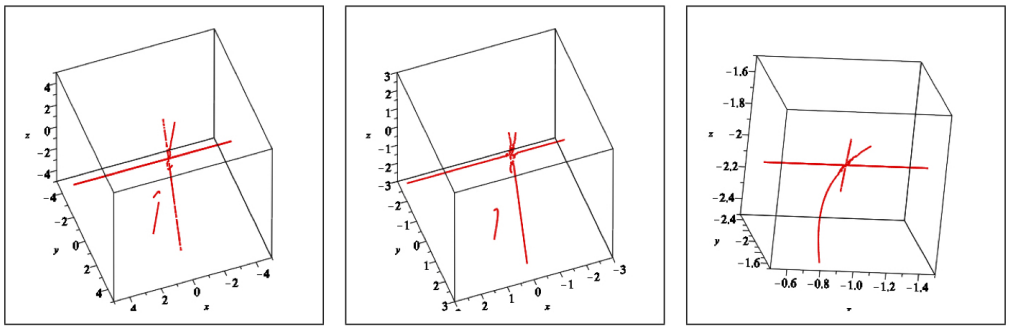


Fig. 3. Computing the intersection curve of $p(x,y,z)=0$ and $q(x,y,z)=0$ by using the `intersectplot` command in three different boxes.

The output of our Maple implementation when computing the the intersection curve of $p(x,y,z)=0$ and $q(x,y,z)=0$ in shown in Figure 4. Procedure `par` tries to compute a closed form for some of the components of the intersection

curve and, when involving radicals, determines the those intervals where such a description can be evaluated. Procedure `sing` computes three lists of points: the first one contains, if any, the tangential intersection points, the second one the lifting to the intersection curve of the singular points of the cutcurve, if any, and the third one a discretisation for the components of the intersection curve without a parameterisation in closed form available.

```
par(-16*x*y+24*x*z+4*y^2+8*y*z-16*z^2+16*x+16*y-40*z-32,-8*x*y+4*x*z+2*y^2+8*y*z-8*z^2-8*
x+16*y-20*z-16)
```

$$[[[x, 4x+2, -2], [x, -2, -2]]]$$

```
sing(-16*x*y+24*x*z+4*y^2+8*y*z-16*z^2+16*x+16*y-40*z-32,-8*x*y+4*x*z+2*y^2+8*y*z-8*z^2-8*
x+16*y-20*z-16)
```

$$\left[[\,], \left[\left[x, 2x, \frac{5x}{4} - \frac{5}{4} + \frac{\sqrt{9x^2-2x-7}}{4}\right], \left[x, 2x, \frac{5x}{4} - \frac{5}{4} - \frac{\sqrt{9x^2-2x-7}}{4}\right], \left[\left[-\frac{16}{9}, -\frac{7}{9}\right], [1, 2]\right]\right], [\,]\right]$$

Fig. 4. Maple output when computing the intersection curve of $p(x, y, z) = 0$ and $q(x, y, z) = 0$ by using our Maple implementation.

Figure 5 plots the intersection curve of $p(x, y, z) = 0$ and $q(x, y, z) = 0$ by using as input the information presented in figure 4 together with two different views placing the intersection curve onto the two considered quadrics.

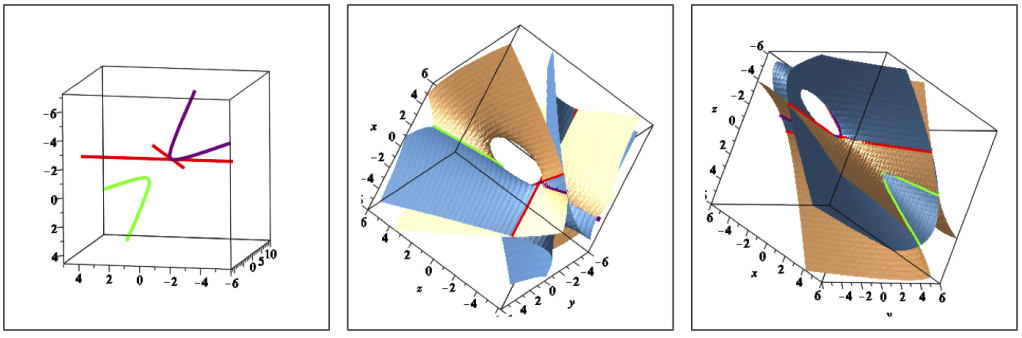


Fig. 5. The intersection curve of $p(x, y, z) = 0$ and $q(x, y, z) = 0$ by using our Maple implementation together with the two quadrics (two views).

Finally, figure 6 presents the output of our Maple implementation when computing the intersection curve for three pairs of quadrics. Last example shows a concrete case where the intersection curve is discretised.

Figure 7 shows, for two concrete examples, the shape of the intersection curve when some (or all) of the components components of the cutcurve have been discretised.

```
par((-16*x^2+8*x*z+8*y^2-8*z^2+48*x-32*y-24*z)*(-1/8), (-12*x*z+4*y^2+2*z^2+8*x-16*y-4*z)*(1/2));
```

$$\left[\left[\left[x, 2+2\sqrt{x^2+1}, 2x\right], \left[x, 2-2\sqrt{x^2+1}, 2x\right], \left[x, 2+\frac{2\sqrt{4x^2-10x+4}}{3}, \frac{2x}{3}-\frac{4}{3}\right], \left[x, 2-\frac{2\sqrt{4x^2-10x+4}}{3}, \frac{2x}{3}-\frac{4}{3}\right], \left[\left[-\frac{1}{2}, \frac{1}{2}\right], [2,3]\right]\right]\right]$$

```
sing((-16*x^2+8*x*z+8*y^2-8*z^2+48*x-32*y-24*z)*(-1/8), (-12*x*z+4*y^2+2*z^2+8*x-16*y-4*z)*(1/2));
```

$$[[\,], [[-1, 2-2\sqrt{2}, -2], [-1, 2+2\sqrt{2}, -2]], [\,]]$$

```
> sing(32*x^2-48*x*y+64*x*z+14*y^2-34*z^2+32*x-8*y-72*z-16, 48*x^2-48*x*y+24*x*z+9*y^2-9*z^2-16*x+12*y-12*z)
```

$$\left[[0.3819660113, 0.3519093630, -0.1573786518], [0.6666666667, 2.000000000, -0.6666666672], [1.500000000, 2.000000000, -6.629999999\,10^{-10}], [2.618033989, 6.314757311, 2.824045318]\right], \left[\left[-1, -\frac{14}{3}+\frac{4\sqrt{5}}{3}, -\frac{10}{3}+\frac{4\sqrt{5}}{3}\right], \left[-1, -\frac{14}{3}+\frac{4\sqrt{5}}{3}, -\frac{2}{3}-\frac{4\sqrt{5}}{3}\right], \left[-1, -\frac{14}{3}-\frac{4\sqrt{5}}{3}, -\frac{2}{3}+\frac{4\sqrt{5}}{3}\right], \left[-1, -\frac{14}{3}-\frac{4\sqrt{5}}{3}, -\frac{10}{3}-\frac{4\sqrt{5}}{3}\right]\right], [\,]\right]$$

```
> par((32*x^2-48*x*y+64*x*z+14*y^2-34*z^2+32*x-8*y-72*z-16, 48*x^2-48*x*y+24*x*z+9*y^2-9*z^2-16*x+12*y-12*z)
```

$$\left[\left[\left[x, 4x-\frac{2}{3}+\frac{4\sqrt{5}\,x}{5}-\frac{8\sqrt{5}}{15}, -\frac{2}{3}+\frac{(-12x+8)\sqrt{5}}{15}\right], \left[x, 4x-\frac{2}{3}-\frac{4\sqrt{5}\,x}{5}+\frac{8\sqrt{5}}{15}, -\frac{2}{3}+\frac{(12x-8)\sqrt{5}}{15}\right], \left[x, \frac{8x}{3}-2+\frac{8\sqrt{5}\,x}{15}-\frac{4\sqrt{5}}{5}, \frac{(8x-12)\sqrt{5}}{15}+\frac{4x}{3}-2\right], \left[x, \frac{8x}{3}-2-\frac{8\sqrt{5}\,x}{15}+\frac{4\sqrt{5}}{5}, \frac{(-8x+12)\sqrt{5}}{15}+\frac{4x}{3}-2\right]\right]\right]$$

```
> par(48*x^2-96*x*y+8*x*z+44*y^2-4*y*z+4*z^2+80*x-72*y+16*z+48, -24*x^2+52*x*z+2*y^2-10*y*z-18*z^2+40*x-4*y-40*z-24)
```

$$[[\,]]$$

```
> sing(48*x^2-96*x*y+8*x*z+44*y^2-4*y*z+4*z^2+80*x-72*y+16*z+48, -24*x^2+52*x*z+2*y^2-10*y*z-18*z^2+40*x-4*y-40*z-24)
```

$$[[\,], [\,], [[-4.344526811, -4.048555471, -2.388763429], [-4.344526811, -3.982907043, -2.388004965], [-4.297136261, -4.008781568, -2.368323547], [-4.297136261, -3.918472459, -2.367251752], [-4.249745710, -3.965228355, -2.347843176], [-4.249745710, -3.857809940, -2.346534136], [-4.202355159, -3.919538652, -2.327339500], [-4.202355159, -3.799277195, -2.325835208], [-4.154964609, -3.872346607, -2.306818594], [-4.154964609, -3.742239614, -2.305148742], [-4.107574058, -3.823974265, -2.286283225], [-4.107574058, -3.686374787, -2.284471828], [-4.060183507, -3.774615326, -2.265735067], [-4.060183507, -3.631489666, -2.263803136], [-4.012792957, -3.724381755, -2.245174690], [-4.012792957, -3.577471891, -2.243142028], [-3.965402406, -3.673342909, -2.224602254], [-3.965402406, -3.524252141, -2.222488401], [-3.918011856, -3.621533607, -2.204017390], [-3.918011856, -3.471795200, -2.201842483], [-3.870621305, -3.568968925, -2.183419630], [-3.870621305, -3.420086052, -2.181204924], [-3.823230754, -3.515641398, -2.162808175], [-3.823230754, -3.369132566, -2.160576723], [-3.775840204, -3.461512538, -2.142181529], [-3.775840204, -3.318972671, -2.139959226], [-3.728449653, -3.406517395, -2.121537836], [-3.728449653, -3.269671349, -2.119354412], [-3.681059102, -3.350549380, -2.100874519], [-3.681059102, -3.221335421, -2.098765127], [-3.633668552, -3.293422865, -2.080187297], [-3.633668552, -3.174150194, -2.078195539], [-3.586278001, -3.234820415, -2.059469443], [-3.586278001, -3.128433026, -2.057652492], [-3.538887450, -3.174121845, -2.038708540], [-3.538887450, -3.084804478, -2.037148758], [-3.491496900, -3.109712611, -2.017873131], [-3.491496900, -3.044878505, -2.016715622], [-3.444106349, -3.025333350, -1.996662147], [-3.444106349, -3.024914767, -1.996654509]]]$$

Fig. 6. Three examples of using Maple to compute the intersection curve of two quadrics.

4 Conclusions

We have presented our Maple implementation of the algorithm in [11] for computing the intersection curve of two quadrics. Compared with the Maple `intersectplot` command it brings, for this particular case, several advantages concerning the accuracy of the result. Its output brings more information since the cutcurve is completely characterised allowing for example to identify the tangential intersection points among many other things. Regarding efficiency our algorithm is slower than the Maple `intersectplot` command but in most cases our implementation requires less than one second when computing the intersection curve of the two considered quadrics. It is worth to remark here that there are lot of potential improvements in the implementation we have reported here.

We think that the Maple `intersectplot` command should improve its accuracy by adding several particular cases where the intersection curve is determined by an ad-hoc algorithm and the case of two quadrics is one of these possibilities.

In [12] we have reported our implementation in GeoGebra of the algorithm in [11]. Compared with the implementation in Maple reported here it is worth

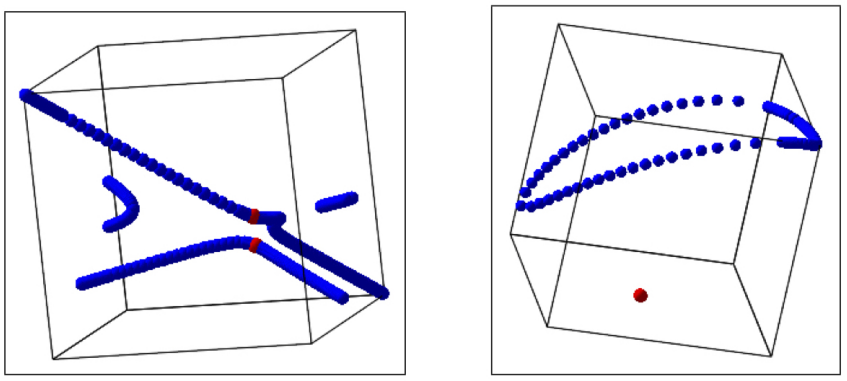


Fig. 7. Plotting the intersection curve with our Maple implementation when the components are discretised. Red points: lifting of the singular points of the cutcurve.

to mention that Maple is clearly more adequate that GeoGebra thanks to how easy is to perform the symbolic and numerical analysis of the cutcurve which is the cornerstone of the algorithm we are dealing with. The only point where GeoGebra outperforms Maple concerns the lifting of the branches of the cutcurve around a tangential intersection point: the determination of these points is quite complicated but they are very easy to lift in GeoGebra without computing them explicitely (in Maple this is needed): by continuity the lifting of the points produced by the `Locus` GeoGebra function around these singular points produces automatically their lifting to the intersection curve (and it is not necessary to compute them explicitly).

References

1. Basu, S., Pollack, R., Roy, M.-F. (2006). Algorithms in Real Algebraic Geometry. Springer–Verlag. http://doi.org/10.1007/3-540-33099-2
2. Dupont, L., Lazard, S., Lazard, D., Petitjean, S. (2003). Near-optimal parameterization of the intersection of quadrics. Proceedings of the Annual Symposium on Computational Geometry. http://doi.org/10.1145/777829.777830
3. Dupont, L., Lazard, D., Lazard, S., Petitjean, S. (2008). Near–optimal parameterization of the intersection of quadrics: I. The generic algorithm. Journal of Symbolic Computation. http://doi.org/10.1016/j.jsc.2007.10.006
4. Dupont, L., Lazard, D., Lazard, S., Petitjean, S. (2008). Near–optimal parameterization of the intersection of quadrics: II. A classification of pencils. Journal of Symbolic Computation. http://doi.org/10.1016/j.jsc.2007.10.012
5. Dupont, L., Lazard, D., Lazard, S., Petitjean, S. (2008). Near–optimal parameterization of the intersection of quadrics: III. Parameterizing singular intersections. Journal of Symbolic Computation. http://doi.org/10.1016/j.jsc.2007.10.007
6. Farouki, R. T., Neff, C. A., OConnor, M. A. (1989). Automatic parsing of degenerate quadric-surface intersections. ACM Transactions on Graphics. http://doi.org/http://doi.acm.org/10.1145/77055.77058
7. Levin, J. (1976). A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. Communications of the ACM. http://doi.org/10.1145/360349.360355

8. Levin, J. Z. (1979). Mathematical models for determining the intersections of quadric surfaces. Computer Graphics and Image Processing. http://doi.org/10.1016/0146-664X(79)90077-7
9. Mourrain, B., Técourt, J. P., Teillaud, M. (2005). On the computation of an arrangement of quadrics in 3D. Computational Geometry: Theory and Applications. http://doi.org/10.1016/j.comgeo.2004.05.003
10. Schömer, E., Wolpert, N. (2006). An exact and efficient approach for computing a cell in an arrangement of quadrics. Computational Geometry: Theory and Applications. http://doi.org/10.1016/j.comgeo.2004.02.007
11. Trocado, A., Gonzalez-Vega, L. (2018). On the intersection of two quadrics: Submitted for publication. Available at `arXiv:1903.06983v2`.
12. Trocado, A., Gonzalez-Vega, L., Dos Santos, J. M. (2019). Intersecting Two Quadrics with GeoGebra. Lecture Notes in Computer Science. http://doi.org/10.1007/978-3-030-21363-3
13. Wang, W., Goldman, R., Tu, C. (2003). Enhancing Levin's method for computing quadric-surface intersections. Computer Aided Geometric Design. http://doi.org/10.1016/S0167-8396(03)00081-5

Chapter 5

Conclusions

We have introduced a new approach to deal with the computation of the intersection curve of two quadrics. The main ingredients of this approach are a detailed analysis of the cutcurve, its singular points and of its relation with the silhouette curves together with the using of an uniform way to perform the lifting of the cutcurve to the intersection curve of the two considered quadrics.

Concerning the analysis of the cutcurve we classify its singular points in two different types depending on how they will be lifted. Those belonging to the line $p_1(x, y) = q_1(x, y)$ are easy to compute and difficult to lift (but just solving a degree two equation) and those not in that line which are more complicated to be determined but easier to lift.

This approach is not intended to classify the intersection curve between the two considered quadrics. Its main goal is to produce in a very direct way a description of the intersection curve which is topologically correct. This is the reason why we allow in the lifting of the cutcurve, when possible, the use of radicals or we rely on the discretisation of the branches of the cutcurve (uniquely determined by the points computed in that curve).

The implementation in GeoGebra showed to be functional but demonstrated slow when there were many singular points of the cutcurve. Using this process, singular points that come from tangential intersection were not computed and these points can be produced by the `locus` GeoGebra function. Some technical issues were detected by the lack of precision of the GeoGebra locus command during the cutcurve representation that did not invalidate the correct functioning of the algorithm. In some cases GeoGebra may produce better results if it was possible to increase the accuracy of the `locus` functionality.

Regarding the implementation of the algorithm in Maple, its performance was only analysed in the most complicated cases, where both quadrics are defined by two degree 2 polynomials in z. In some of these cases it was possible to obtain the exact parameterization of the intersection curve even using radicals. It is worth to remark here that some points not determined by Maple's `intersectplot` command were possible to be computed by using our implementation. In some cases, when it was not possible to parameterize the cutcurve, a discretization of its branches was used.

In the near-future the algorithm performance will be optimized and its implementation in Maple will consider not only the general case, namely considering quadrics defined by polynomials of degree less than 2 in variable z.

Chapter 6

The Appendix

```
with(RegularChains):with(ChainTools):with(RealDomain):
par := proc (f::algebraic, g::algebraic)
#Definition
local p1, q1, p0, q0, inter, lift, yi, line, yline, i, R, S0, yS0,fz1, fz2, gz1, gz2,
z0f1, z0f2, z0g1, z0g2, Omega, zreg, xline, xi, d1, d2, l1, lifr, x0, y0, param0,
param, qq, tang, tau0,rad1,rad2,zero,intv,j,py,pz,k:
#Initialization
p1 := coeff(f, z, 1): p0 := coeff(f, z, 0): q1 := coeff(g, z, 1):
q0 := coeff(g, z, 0):
d1 := simplify(discrim(f, z)):
d2 := simplify(discrim(g, z)):
S0 := resultant(f, g, z):
yS0 := ListTools:-MakeUnique([solve(S0, y)]):
param := []: tang := []:
fz1 := -(1/2)*p1+(1/2)*sqrt(p1^2-4*p0):fz2 := -(1/2)*p1-(1/2)*sqrt(p1^2-4*p0):
gz1 := -(1/2)*q1+(1/2)*sqrt(q1^2-4*q0):gz2 := -(1/2)*q1-(1/2)*sqrt(q1^2-4*q0):
line := p1-q1:
yline := solve(line = 0, y):
rad1:=[]:zero:=[]:intv:=[]:param0:=[]:k:=0:
#Test if S0 can be factorized
k:=0:
#if there is one factor of degree 3 or 1 with degree 4 with (and no factor possible)
if nops(factors(S0)[2])=1 and degree(factors(S0)[2][1][1],y)=4 then
 k:=k+1:
else
  for i from 1 to nops(factors(S0)[2]) do
     if degree(factors(S0)[2][i][1],y)=3 or factors(S0)[2][i][2]=3 then
        k:=k+1:
     fi:
  od:
fi:
if k=0 then
```

```
#All points singular
if line = 0 then
  rad1:=discrim(p0-q0,y):
  zero:=ListTools:-MakeUnique([solve(rad1=0,x)]):
  if rad1=0 then
     rad1:=[]:
     zero:=[]:
  end if:
  for i to nops(yS0) do
     param0 := [op(param0), [x, yS0[i], simplify(expand(subs(y = yS0[i],fz1)))],
          [x, yS0[i], simplify(expand(subs(y = yS0[i], fz2)))]]:
     zero:=[op(zero),evalf(solve(expand(subs(y=yS0[i],p1^2-4*p0))))]:
  end do:
#If there is no zero
  if zero=[] then
     param:=param0:
  else
    #Evaluation zone
    zero:=sort(zero):
    for i to nops(yS0) do
      rad2:=expand(subs(y=yS0[i],p1^2-4*p0)):
      #k count the number of intervals that has positive value
      #intv is to add intervals that are selected
      intv:=[]:k:=0:
      #Only one zero
      if nops(zero)=1 then
         py:=evalf(subs(x=zero[1]-1,rad1)):
         pz:=evalf(subs(x=zero[1]-1,rad2)):
         if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                        intv:=[op(intv),[zero[1]-1,zero[1]]]:k:=k+1:
         end if:
         py:=evalf(subs(x=zero[1]+1,rad1)):
         pz:=evalf(subs(x=zero[1]+1,rad2)):
         if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                        intv:=[op(intv),[zero[1],zero[1]+1]]:k:=k+1:
         end if:
         #If at left and right of zero all positive then all points are selected
     #Only more than one zero
      else if nops(zero)>1 then
            for j to nops(zero) do
               #Analyse first zero
               if j=1 then
                  py:=evalf(subs(x=zero[j]-1,rad1)):
                  pz:=evalf(subs(x=zero[j]-1,rad2)):
                  if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                    intv:=[op(intv),[zero[j]-1,zero[j]]]:k:=k+1:
                  end if:
               end if:
               #Analyse midle
                if j>1 and j<nops(zero) then
```

```
                    py:=subs(x=(zero[j-1]+zero[j])/2,rad1):
                    pz:=subs(x=(zero[j-1]+zero[j])/2,rad2):
                    if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                       intv:=[op(intv),[zero[j-1],zero[j]]]:k:=k+1:
                    end if:
                 end if:
                 #Analyse last zero
                 if j=nops(zero) then
                     py:=evalf(subs(x=(zero[j-1]+zero[j])/2,rad1)):
                     pz:=evalf(subs(x=(zero[j-1]+zero[j])/2,rad2)):
                     if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                         intv:=[op(intv),[zero[j-1],zero[j]]]:k:=k+1
                     end if:
                     py:=evalf(subs(x=zero[j]+1,rad1)):
                     pz:=evalf(subs(x=zero[j]+1,rad2)):
                     if signum(subs(x=zero[j]+1,rad1))>=0 and
                     signum(subs(x=zero[j]+1,rad2))>=0 and Im(py)=0 and Im(pz)=0 then
                         intv:=[op(intv),[zero[j],zero[j]+1]]:k:=k+1:
                     end if:
                 end if:
              end do:
           end if:
        end if:

     #If there are intervals and not all, compute the parameterization with intervals
     if intv<>[] and k<>nops(zero)+1 then
           param:=[op(param),[x, yS0[i], simplify(subs(y = yS0[i],fz1))],
           [x, yS0[i], simplify(subs(y = yS0[i], fz2))],intv]
    end if:

     #If all points are selected
     if k=nops(zero)+1 then param:=[op(param),
     [x, yS0[i], expand(subs(y = yS0[i],fz1))], [x, yS0[i],
     expand(subs(y = yS0[i], fz2))]]
     end if:
     end do:
     end if:

#Compute parameterization if is a vertical line
else if coeff(p1-q1, y, 1) <> 0 then
     zreg := simplify(-(q0-p0)/(q1-p1)):
     param := []: param0:[];
     #Verify all factor of S0
     for i to nops(factors(S0)[2]) do
        param0:=[solve(factors(S0)[2][i][1],y)];
        if degree(factors(S0)[2][i][1],y)=2 and nops(param0)<>0 then
          rad1:=discrim(factors(S0)[2][i][1],y);
          zero:=[solve(rad1)];
          #Compute every y of every factor of S0
            #Evaluation zone
```

```
            zero:=sort(zero):
            #k count the number of intervals that has positive value
            #intv is to add intervals that are selected
             intv:=[]: k:=0:
            #Only one or no zero
            if nops(zero)<=1 and coeff(rad1,x,2)>0 then
                param:=[op(param),[x, param0[1], simplify(subs(y = param0[1],zreg))],
                [x, param0[2], simplify(subs(y = param0[2], zreg))]]:
            #Only more than one zero
            else if nops(zero)=2 then
                   if coeff(rad1,x,2)>0 then
                        intv:=[op(intv),[zero[1]-1,zero[1]],[zero[2],zero[2]+1]]:
                   else intv:=[op(intv),[zero[1],zero[2]]]:
                   end if:
                   param:=[op(param),[x, param0[1], simplify(subs(y = param0[1],zreg))],
                        [x, param0[2], simplify(subs(y = param0[2], zreg))],intv];
                end if:
             end if:
             #If only one point is selected
             if nops(zero)=1 and coeff(rad1,x,2)<0 then
                    param:=[op(param),[x, param0[1], simplify(subs(y = param0[1],zreg))],
                                    [x, param0[2], simplify(subs(y = param0[2], zreg))]]:
             end if:
       else if solve(factors(S0)[2][i][1],y)<>yline and nops(param0)<>0 then
               param:=[op(param),[x,solve(factors(S0)[2][i][1],y),
                    simplify(subs(y = solve(factors(S0)[2][i][1],y),zreg))]]:
            end if:
       end if:
       end do:

#Compute parameterization if the line is vertical
else if coeff(p1-q1, x, 1) <> 0 then
     zreg := simplify(-(q0-p0)/(q1-p1)):
     param:=[]:
     #Verify all factor of S0
    for i to nops(factors(S0)[2]) do
        param0:=[solve(factors(S0)[2][i][1],y)]:
        #If factor is of degree two
        if degree(factors(S0)[2][i][1],y)=2 and nops(param0)<>0 then
          rad1:=discrim(factors(S0)[2][i][1],y):
          zero:=[solve(rad1)]:
          #Compute every y of every factor of S0
              #Evaluation zone
               zero:=sort(zero):
            #Only one or no zero
            if nops(zero)<=1 and coeff(rad1,x,2)>0 and nops(param0)<>0 then
               param:=[op(param),[x, param0[1], simplify(subs(y = param0[1],zreg))],
                         [x, param0[2], simplify(subs(y = param0[2], zreg))]]:
           #Only more than one zero
              else if nops(zero)=2 then
```

```
                    if coeff(rad1,x,2)>0 then
                         intv:=[op(intv),[zero[1]-1,zero[1]],[zero[2],zero[2]+1]]:
                    else intv:=[op(intv),[zero[1],zero[2]]]:
                    end if:
                    param:=[op(param),[x, param0[1], simplify(subs(y = param0[1],zreg))],
                         [x, param0[2], simplify(subs(y = param0[2], zreg))],intv];
                   end if:
               end if:
               #If only one point is selected
               if nops(zero)=1 and coeff(rad1,x,2)<0 then
                      param:=[op(param),[x, param0[1], simplify(subs(y = param0[1],zreg))],
                                        [x, param0[2], simplify(subs(y = param0[2], zreg))]]:
               end if:
          else
                    if solve(factors(S0)[2][i][1],y)<>yline and nops(param0)<>0 then
                       param:=[op(param),[x,solve(factors(S0)[2][i][1],y),
                                  simplify(subs(y = solve(factors(S0)[2][i][1],y),zreg))]]:
                    end if:
          end if:
     end do:

     else
        for i to nops(yS0) do param := [op(param), [x, yS0[i],
              simplify(subs(y = yS0[i], fz1))], [x, yS0[i], simplify(subs(y = yS0[i], fz2))]]
         end do :
     end if:

ListTools:-MakeUnique(param):
end if:
param := ListTools:-MakeUnique(param):
end if:
end if:
[param]
end proc:


with(RegularChains):with(ChainTools):with(RealDomain):with(algcurves):with(plots):
sing := proc (f::algebraic, g::algebraic)

#Definition
local p1, q1, p0, q0, inter, lift, yi, line, yline, i, R, S0, yS0, Rxy, src, U0, U1,
U11, U10, V0, V1, V11, V10, tau, fz1, fz2, gz1, gz2, z0f1, z0f2, z0g1, z0g2,zreg,
xline, xi, d1, d2, l1, lifr, x0, y0, param0, param, qq, tang, tau0,rad1,rad2,zero,
intv,j,py,pz,k,r1,r2,pl,pl1,pl2,py2,silh,dd,liftpl,UV1100,UV,UU0,VV0,points:
#Initialization
p1 := coeff(f, z, 1):
p0 := coeff(f, z, 0):
q1 := coeff(g, z, 1):
q0 := coeff(g, z, 0):
d1 := simplify(discrim(f, z)):
```

```
d2 := simplify(discrim(g, z)):
S0 := resultant(f, g, z):
yS0 := ListTools:-MakeUnique([solve(S0, y)]):
Rxy := PolynomialRing([x, y]):
U0 := SubresultantChain(S0, diff(S0, x), y, Rxy)[subresultant_chain_vector][1]:
U1 := SubresultantChain(S0, diff(S0, x), y, Rxy)[subresultant_chain_vector][2]:
U11 := coeff(U1, y, 1): U10 := coeff(U1, y, 0):
V0 := SubresultantChain(S0, diff(S0, y), y, Rxy)[subresultant_chain_vector][1]:
V1 := SubresultantChain(S0, diff(S0, y), y, Rxy)[subresultant_chain_vector][2]:
V11 := coeff(V1, y, 1): V10 := coeff(V1, y, 0):
qq := []: param := []: tang := []:
fz1 := -(1/2)*p1+(1/2)*sqrt(p1^2-4*p0):fz2 := -(1/2)*p1-(1/2)*sqrt(p1^2-4*p0):
gz1 := -(1/2)*q1+(1/2)*sqrt(q1^2-4*q0):gz2 := -(1/2)*q1-(1/2)*sqrt(q1^2-4*q0):
line := p1-q1:
rad1:=[]:zero:=[]:intv:=[]:param0:=[]:k:=0:pl:=[]:silh:=[]:pl2:=[]:liftpl:=[]:
#All points singular

if line = 0 then
  rad1:=discrim(p0-q0,y):
  zero:=ListTools:-MakeUnique([solve(rad1=0,x)]);
  if rad1=0 then
     rad1:=[]:
     zero:=[]:
  end if:
  for i to nops(yS0) do
     param0 := [op(param0), [x, yS0[i], expand(subs(y = yS0[i],fz1))],
                 [x, yS0[i], expand(subs(y = yS0[i], fz2))]]:
     zero:=[op(zero),evalf(solve(expand(subs(y=yS0[i],p1^2-4*p0))))]:
  end do:

#If there is no zero
  if zero=[] then
     param:=param0:
  else
    #Evaluation zone
    zero:=sort(zero):
    for i to nops(yS0) do
      rad2:=expand(subs(y=yS0[i],p1^2-4*p0)):
      #k count the number of intervals that has positive value
      #intv is to add intervals that are selected
      intv:=[]:k:=0:
      #Only one zero
      if nops(zero)=1 then
         py:=evalf(subs(x=zero[1]-1,rad1)):
         pz:=evalf(subs(x=zero[1]-1,rad2)):
         if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                        intv:=[op(intv),[zero[1]-1,zero[1]]]:k:=k+1:
         end if:
         py:=evalf(subs(x=zero[1]+1,rad1)):
         pz:=evalf(subs(x=zero[1]+1,rad2)):
```

```
    if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                    intv:=[op(intv),[zero[1],zero[1]+1]]:k:=k+1:
    end if:
    #If at left and right of zero all positive then all points are selected
#Only more than one zero
 else if nops(zero)>1 then
       for j to nops(zero) do
          #Analyse first zero
          if j=1 then
             py:=evalf(subs(x=zero[j]-1,rad1)):
             pz:=evalf(subs(x=zero[j]-1,rad2)):
             if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
               intv:=[op(intv),[zero[j]-1,zero[j]]]:k:=k+1:
             end if:
          end if:
          #Analyse midle
           if j>1 and j<nops(zero) then
              py:=subs(x=(zero[j-1]+zero[j])/2,rad1):
              pz:=subs(x=(zero[j-1]+zero[j])/2,rad2):
              if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                 intv:=[op(intv),[zero[j-1],zero[j]]]:k:=k+1:
              end if:
           end if:
           #Analyse last zero
           if j=nops(zero) then
               py:=evalf(subs(x=(zero[j-1]+zero[j])/2,rad1)):
               pz:=evalf(subs(x=(zero[j-1]+zero[j])/2,rad2)):
               if Im(py)=0 and Im(pz)=0 and signum(py)>=0 and signum(pz)>=0 then
                    intv:=[op(intv),[zero[j-1],zero[j]]]:k:=k+1
               end if:
               py:=evalf(subs(x=zero[j]+1,rad1)):
               pz:=evalf(subs(x=zero[j]+1,rad2)):
               if signum(subs(x=zero[j]+1,rad1))>=0 and
                  signum(subs(x=zero[j]+1,rad2))>=0 and Im(py)=0 and Im(pz)=0 then
                    intv:=[op(intv),[zero[j],zero[j]+1]]:k:=k+1:
               end if:
           end if:
        end do:
      end if:
  end if:

#If there are intervals and not all selected, compute the parameterization with intervals
if intv<>[] and k<>nops(zero)+1 then
   param:=[op(param),[x,yS0[i],subs(y=yS0[i],fz1)],
       [x,yS0[i],subs(y=yS0[i],fz2)],intv] end if:
#If all points are selected
if k=nops(zero)+1 then
    param:=[op(param),[x,yS0[i],expand(subs(y=yS0[i],fz1))],
        [x, yS0[i], expand(subs(y = yS0[i], fz2))]] end if:
end do:
```

```
     end if:
#Not vertical line
else if coeff(p1-q1, y, 1) <> 0 then
     zreg := simplify(-(q0-p0)/(q1-p1)): yline := solve(line = 0, y):
     inter := [solve(simplify(subs(y = yline, d1-d2)) = 0, x)]:
     #If all line is in the cutcurve
     if simplify(subs(y = yline, d1-d2)) = 0 then
        #Determine intervals if the line is in the cutcurve
        intv:=[]:
        #Define rad2 as the substitution of the line in the radical that lifts the curve
        rad2:=expand(subs(y=yline,p1^2-4*p0)):
        #Compute the roots of the rad
        zero:=ListTools:-MakeUnique([solve(rad2)]):
        #If there are no zeros of rad2 and all points are selected
        if nops(zero)=0 and coeff(rad2,x,2)>0 then
             qq := [[x, yline, simplify(subs({y = yline}, fz1))],
                    [x, yline, simplify(subs({y = yline}, fz2))]]:
        end if:
        #If there are no zeros of rad2 and no points are selected
        if nops(zero)=0 and coeff(rad2,x,2)<0 then
             qq:=[]:
        end if:
        #If there is only one zero of rad 2 and all points are selected
        if nops(zero)=1 and coeff(rad2,x,2)>0 then
             qq := [[x, yline, simplify(subs({y = yline}, fz1))],
                    [x, yline, simplify(subs({y = yline}, fz2))]]:
        end if:
        #If there is only one zero of rad 2 and no points are selected
                       the is only one value of x that has z coordinate
        if nops(zero)=1 and coeff(rad2,x,2)<0 then
             #Verify if this pois is in the validation region and compute its coordinates
             if 0<=signum(expand(subs({x=zero,y=subs(x=zero,yline)},d1))) and
                 0<=signum(expand(subs({x=zero,y=subs(x=zero,yline)},d2)))  then
                   qq:=[[zero,y=subs(x=zero,yline),subs({x=zero,y=subs(x=zero,yline)},fz1)],
                        [zero,y=subs(x=zero,yline),subs({x=zero,y=subs(x=zero,yline)},fz2)]]:
             end if:
        end if:
        #If there are 2 zeros
        zero:=sort(zero):
        if nops(zero)=2 then
             k:=0:
             pz:=evalf(subs(x=zero[1]-1,rad2)):
             if Im(pz)=0 and signum(pz)>=0 then
                    intv:=[op(intv),[zero[1]-1,zero[1]]]:k:=k+1:
             end if:
             pz:=evalf(subs(x=(zero[1]+zero[2])/2,rad2)):
             if Im(pz)=0 and signum(pz)>=0 then
                    intv:=[op(intv),[zero[1],zero[2]]]:k:=k+1:
             end if:
             pz:=evalf(subs(x=zero[2]+1,rad2)):
```

```
            if Im(pz)=0 and signum(pz)>=0 then
                   intv:=[op(intv),[zero[2],zero[2]+1]]:k:=k+1:
            end if:
            if intv<>[] and k<>nops(zero)+1 then
                qq:=[op(qq),[x,yline,subs(y=yline,fz1)],
                    [x,yline,subs(y=yline,fz2)],intv] end if:
            if k=nops(zero)+1 then qq:=[op(qq),[x,yline,subs(y=yline,fz1)],
                                               [x,yline,subs(y=yline,fz2)]] end if:
        end if
     else if nops(inter) <> 0 then
         for i to nops(inter) do
           yi := simplify(subs(x = inter[i], yline)):
           if 0 <= signum(expand(subs({x = inter[i], y = yi}, d1))) and
                0 <= signum(expand(subs({x = inter[i], y = yi}, d2))) then
                  qq := [op(qq),[inter[i],yi,simplify(subs({x=inter[i],y=yi},fz1))],
                     [inter[i], yi, simplify(subs({x = inter[i], y = yi}, fz2))]]
           end if:
         end do:
         end if:end if:
#Compute singular points if the line is vertical
else if coeff(p1-q1, x, 1) <> 0 then
   zreg := simplify(-(q0-p0)/(q1-p1)):
   xline := solve(line = 0, x):
   inter := [solve(simplify(subs(x = xline, S0)) = 0, y)]:
  #Vertical line inside the cutcurve
   if subs(x = xline, S0) = 0 then
     qq := [[xline,y,simplify(subs({x=xline},fz1))],
            [xline,y,simplify(subs({x=xline},fz2))]]:
  #Vertical line outside the cutcurve
   else if nops(inter) <> 0 then
       for i to nops(inter) do
          #Test if all common points to the line and the cutcurve are inside
                 the validation region
          if signum(subs({x = xline, y = inter[i]}, d1))>=0 and
                   0 <= signum(subs({x = xline, y = inter[i]}, d2)) then
                #Compare if z coordinate is the same
                z0f1 := simplify(subs({x = xline, y = inter[i]}, fz1)):
                z0f2 := simplify(subs({x = xline, y = inter[i]}, fz2)):
                z0g1 := simplify(subs({x = xline, y = inter[i]}, gz1)):
                z0g2 := simplify(subs({x = xline, y = inter[i]}, gz2)):
                if z0f1=z0g1 or z0f1=z0g2 then qq:=[op(qq),[xline, inter[i],z0f1]]:fi:
                if z0f2=z0g1 or z0f2=z0g2 then qq:=[op(qq),[xline, inter[i],z0f2]]:fi:
          end if:
        end do
        else qq := []:
        end if:
   end if:
end if:
end if:
```

```
#Determine tangential points:
   #Define tau
   if coeff(p1-q1, y, 1)<>0 then
      tau0:= simplify(subs(y = yline, d1-d2)):
      if tau0<>0 then
        tau:= simplify(tau0/gcd(tau0,diff(tau0,x))):
      else tau:=tau0:
      end if:
   else tau:=p1-q1:
   end if:
   if gcd(U10*V11-V10*U11,tau)<>0 then
          #Remove squares from U10*V11-V10*U11
          UV1100:=U10*V11-V10*U11:
          if UV1100<>0 then UV:=simplify(UV1100/gcd(UV1100,diff(UV1100,x))):
          else UV:=0 fi:
          if U0<>0 then UU0:=simplify(U0/gcd(U0,diff(U0,x))):
          else UU0:=0: fi:
          if V0<>0 then VV0:=simplify(V0/gcd(V0,diff(V0,x))):
          else VV0:=0: fi:
   end if:
   x0:=ListTools:-MakeUnique([fsolve(gcd(gcd(UV,UU0),VV0),x)]):
   if nops(x0) <> 0 then
      for i to nops(x0) do
        k:=0:
        if evalf(subs(x = x0[i], V11))<>0 and evalf(subs(x=x0[i],gcd(UV,tau)))<>0 then
                y0 := evalf(subs(x = x0[i], -V10/V11)):
                if nops(qq)>0 then
                    for j from 1 to nops(qq) do
                      if evalf((x0[i]-evalf(qq[j][1]))^2+(y0-evalf(qq[j][2]))^2)<=0.000001^2
                          then k:=k+1 fi:
                    od:
                fi:
                if k=0 then tang:=[op(tang),[x0[i],y0,subs({x=x0[i],y=y0},-(q0-p0)/(q1-p1))]]:fi:
        end if:
     end do:
    end if:
qq := ListTools:-MakeUnique(qq):
tang := ListTools:-MakeUnique(tang):
param := ListTools:-MakeUnique(param):
end if:
#Test if S0 can be factorized
k:=0:
#if there is one factor of degree 3 or 1 with degree 4 with (and no factor possible)
if nops(factors(S0)[2])=1 and degree(factors(S0)[2][1][1],y)=4 then
 k:=k+1:
else
  for i from 1 to nops(factors(S0)[2]) do
     if degree(factors(S0)[2][i][1],y)=3 or factors(S0)[2][i][2]=3 then
        k:=k+1:
     fi:
```

```
  od:
fi:

#If k=0, S0 can be factorized do all procedures
if k=1 then
     zero:=ListTools:-MakeUnique([fsolve(discrim(S0,y))]);
     zero:=sort(zero):
     if nops(zero)=1 then
       #compute number of branches to the left and right of the zero
        r1:=nops([fsolve(subs(x=op(zero)-1,S0))]):
        r2:=nops([fsolve(subs(x=op(zero)+1,S0))]):
        if r1>=1 and r2>=1 then
            for i from 0 to 19 do
               py:=[fsolve(subs(x=op(zero)-1+i*1/20,S0))];
               for j from 1 to nops(py) do
                   if subs(x=op(zero)-1+i*1/20,y=py[j],d1)>=0 and
                         subs(x=op(zero)-1+i*1/20,y=py[j],d2)>=0 and Im(py[j])=0 then
                      pl:=[op(pl),[op(zero)-1+i*1/20,py[j]]];
                   fi:
               od:
               py2:=[fsolve(subs(x=op(zero)+1-i*1/20,S0))];
               for j from 1 to nops(py2) do
                   if subs(x=op(zero)+1-i*1/20,y=py2[j],d1)>=0 and
                         subs(x=op(zero)+1-i*1/20,y=py2[j],d2)>=0 and Im(py2[j])=0 then
                      pl:=[op(pl),[op(zero)+1-i*1/20,py2[j]]];
                   fi:
               od:
            od:
        fi:
        #Only branches to the left
        if r1>=1 and r2=0 then
           for i from 0 to 19 do
              py:=[fsolve(subs(x=op(zero)-1+i*1/20,S0))]:
              for j from 1 to nops(py) do
                 if subs(x=op(zero)-1+i*1/20,y=py[j],d1)>=0 and
                       subs(x=op(zero)-1+i*1/20,y=py[j],d2)>=0 and Im(py[j])=0 then
                    pl:=[op(pl),[op(zero)-1+i*1/20,py[j]]];
                 fi:
              od:

           od:
        fi:
        #Only branches to the right
        if r1=0 and r2>=1 then
            for i from 0 to 19 do
                py2:=[fsolve(subs(x=op(zero)+1-i*1/20,S0))]:
                for j from 1 to nops(py2) do
                    if subs(x=op(zero)+1-i*1/20,y=py2[j],d1)>=0 and
                          subs(x=op(zero)+1-i*1/20,y=py2[j],d2)>=0 and Im(py[j])=0 then
                       pl:=[op(pl),[op(zero)+1+i*1/20,py2[j]]]:
```

```
                fi:
             od:
          od:
       fi:
       #If the zero is not coincident with singular points line
       if r1=0 and r2=0 then
           if subs(x=op(zero),line)<>0 then
              py:=[fsolve(subs(x=op(zero),S0))]:
              for i from 1 to nops(py) do
                  if signum(subs({x=zero[1],y=py[i]},d1))>=0 and
                      signum(subs({x=zero[1],y=py[i]},d1))>=0 and Im(py[i])=0 then
                     pl:=[op(pl),[zero[1],py[i]]]:
                  fi:
              od:
           fi:
       fi:
   else if nops(zero)>1 then
          for j to nops(zero) do
              if j=1 then
                  r1:=nops([fsolve(subs(x=zero[j]-1,S0))]):
                  if r1<>0 then
                      for i from 0 to 19 do
                       py:=[fsolve(subs(x=zero[j]-1+i*1/20,S0))]:
                         for k from 1 to r1 do
                            if signum(subs({x=zero[j]-1+i*1/20,y=py[k]},d1))>=0 and
                                signum(subs({x=zero[j]-1+i*1/20,y=py[k]},d1))>=0 and
                                                                  Im(py[k])=0 then
                                  pl:=[op(pl),[zero[j]-1+i*1/20,py[k]]]:
                            fi:
                         od:

                      od:
                  fi:
              fi:
           if j>1 and j<nops(zero) then
                r1:=nops([fsolve(subs(x=(zero[j-1]+zero[j])/2,S0))]):
                if r1<>0 then
                   for i from 1 to 19 do
                      py:=[fsolve(subs(x=zero[j-1]+i*(zero[j]-zero[j-1])/20,S0))]:
                      for k from 1 to r1 do
                        if signum(subs({x=zero[j-1]+i*(zero[j]-zero[j-1])/20,y=py[k]},d1))>=0
                           and
                           signum(subs({x=zero[j-1]+i*(zero[j]-zero[j-1])/20,y=py[k]},d1))>=0
                           and Im(py[k])=0 then
                           pl:=[op(pl),[zero[j-1]+i*(zero[j]-zero[j-1])/20,py[k]]]:
                        fi:
                      od:
                   od:
                 fi:
              fi:
```

```
                if j=nops(zero) then
                     r1:=nops([fsolve(subs(x=(zero[j-1]+zero[j])/2,S0))]):
                    if r1<>0 then
                    for i from 1 to 19 do
                      py:=[fsolve(subs(x=zero[j-1]+i*(zero[j]-zero[j-1])/20,S0))]:
                    for k from 1 to r1 do
                      if signum(subs({x=zero[j-1]+i*(zero[j]-zero[j-1])/20,y=py[k]},d1))>=0
                         and
                         signum(subs({x=zero[j-1]+i*(zero[j]-zero[j-1])/20,y=py[k]},d1))>=0
                         and Im(py[k])=0 then
                         pl:=[op(pl),[zero[j-1]+i*(zero[j]-zero[j-1])/20,py[k]]]:
                        fi:
                     od:
                     od:
                     fi:
                     r1:=nops([fsolve(subs(x=zero[j]+1,S0))]):
                     if r1<>0 then
                        for i from 0 to 19 do
                           py:=[fsolve(subs(x=zero[j]+1-i*1/20,S0))]:
                           for k from 1 to r1 do
                               if signum(subs({x=zero[j]+1-i*1/20,y=py[k]},d1))>=0
                                  and
                                  signum(subs({x=zero[j]+1-i*1/20,y=py[k]},d1))>=0
                                  and Im(py[k])=0 then
                                    pl:=[op(pl),[zero[j]+1-i*1/20,py[k]]]:
                               fi:
                           od:
                        od:
                     fi:

                end if:

            od:
         fi:
     fi:
 if nops(zero)=0 then
    for i from 0 to 19 do
       py:=[fsolve(subs(x=-5+i*1/2,S0))]:
       for k from 1 to nops(py) do
          pl:=[op(pl),[-5+i*1/2,py[k]]]:
       od:
    od:
 fi:

liftpl:=[]:
#Vertical lines
     if nops(zero)>0 then
        for i from 1 to nops(zero) do
            #If the vertical line is the line of singular points
            if subs(x=zero[i],line)=0 then
```

```
                if subs(x=zero[i],S0)=0 then
                    for j from -10 to 10 do
        #Lifiting like singular points from vertical line comparing z coordinates
                            z0f1 := simplify(subs({x=zero[i],y=j},fz1)):
                            z0f2 := simplify(subs({x=zero[i],y=j},fz2)):
                            z0g1 := simplify(subs({x=zero[i],y=j},gz1)):
                            z0g2 := simplify(subs({x=zero[i],y=j},gz2)):
                            if z0f1=z0g1 or z0f1=z0g2 then liftpl:=[op(liftpl),
                                [zero[i],j,simplify(subs({x=zero[i],y=j},z0f1))]]: fi:
                            if z0f2=z0g1 or z0f2=z0g2 then liftpl:=[op(liftpl),
                                [zero[i],j,simplify(subs({x=zero[i],y=j},z0f2))]]: fi:

                     od:
                fi:

            #If the vertical line is inside the cutcurve
            else if subs(x=zero[i],S0)=0 then
                    for j from -10 to 10 do
                        pl:=[op(pl),[zero[i],j]]:
                    od:
                #Compute points from vertical line inside the cutcurve
                else pl1:=[fsolve(subs(x=zero[i],S0),y)]:
                    for j from 1 to nops(pl1) do
                        if evalf(subs({x=zero[i],y=pl1[j]},line))<>0 then
                            pl:=[op(pl),[zero[i],pl1[j]]]:
                        fi:
                    od:
                fi:
            fi:
        od:
     fi:

#lifting
    if nops(pl)>0 and nops(qq)>0 then
         for i from 1 to nops(pl) do
              k=0:
               if 0<=signum(expand(subs({x = pl[i][1], y = pl[i][2]}, d1))) and
                      0 <= signum(expand(subs({x = pl[i][1], y = pl[i][2]}, d2))) then
              for j from 1 to nops(qq) do
                 #Check if every point is near to some singular point
                    if evalf(sqrt((qq[j][1]-pl[i][1])^2+(qq[j][2]-pl[i][2])^2))<0.1 and
                         evalf(sqrt((qq[j][1]-pl[i][1])^2+(qq[j][2]-pl[i][2])^2))>0 then
                      k=k+1:
                   #Lifiting like singular points comparing z coordinates
                      z0f1 := simplify(subs({x =pl[i][1], y =pl[i][2]},fz1)):
                      z0f2 := simplify(subs({x =pl[i][1], y =pl[i][2]},fz2)):
                      z0g1 := simplify(subs({x =pl[i][1], y =pl[i][2]},gz1)):
                      z0g2 := simplify(subs({x =pl[i][1], y =pl[i][2]},gz2)):
                      if z0f1=z0g1 or z0f1=z0g2 then
                            liftpl:=[op(liftpl),[pl[i][1],pl[i][2],
```

```
                if z0f2=z0g1 or z0f2=z0g2 then
                          liftpl:=[op(liftpl),[pl[i][1],pl[i][2],
                          simplify(subs({x=pl[i][1],y=pl[i][2]},z0f2))]]: fi:
                 #if the point is far from every singular point
                 else liftpl:=[op(liftpl),[pl[i][1],pl[i][2],
                   simplify(subs({x=pl[i][1],y=pl[i][2]},-(q0-p0)/(q1-p1)))]]
                 fi:
            od:
            fi:
      od:
   fi:
   if nops(pl)>0 and nops(qq)=0 then
      for i from 1 to nops(pl) do
          if 0<=signum(expand(subs({x = pl[i][1], y = pl[i][2]}, d1))) and
                 0 <= signum(expand(subs({x = pl[i][1], y = pl[i][2]}, d2))) then
             liftpl:=[op(liftpl),[pl[i][1],pl[i][2],
                        simplify(subs({x=pl[i][1],y=pl[i][2]},-(q0-p0)/(q1-p1)))]]:
          fi:
      od:
   fi:
fi:
[tang,qq,liftpl]
end proc:
```

Bibliography

[1] N. Arbain and N. A. Shukor. The Effects of GeoGebra on Students Achievement. *Procedia - Social and Behavioral Sciences*, 2015. ISSN 18770428. doi: 10.1016/j.sbspro.2015.01.356. URL https://doi.org/10.1016/j.sbspro.2015.01.356.

[2] R. Barnhill, G. Farin, M. Jordan, and B. Piper. Surface/surface intersection. *Computer Aided Geometric Design*, 4:3–16, jul 1987. URL https://doi.org/10.1016/0167-8396(87)90020-3.

[3] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. 2006. ISBN 3540009736 (alk. paper). doi: 10.1007/3-540-33099-2. URL https://doi.org/10.1007/3-540-33099-2.

[4] E. Berberich, M. Hemmer, L. Kettner, E. Schömer, and N. Wolpert. An exact, complete and efficient implementation for computing planar maps of quadric intersection curves. 2005. doi: 10.1145/1064092.1064110. URL https://doi.org/10.1145/1064092.1064110.

[5] E. Berberich, P. Emeliyanenko, A. Kobel, and M. Sagraloff. Exact symbolic-numeric computation of planar algebraic curves. *Theoretical Computer Science*, 2013. ISSN 03043975. doi: 10.1016/j.tcs.2013.04.014. URL https://doi.org/10.1016/j.tcs.2013.04.014.

[6] E. Boender. A survey of intersection algorithms for curved surfaces. *Computers and Graphics*, 1991. ISSN 00978493. URL https://doi.org/10.1016/0097-8493(91)90037-I.

[7] W. S. Brown and J. F. Traub. On Euclid's Algorithm and the Theory of Subresultants. *Journal of the ACM (JACM)*, 1971. ISSN 1557735X. doi: 10.1145/321662.321665. URL https://doi.org/10.1145/321662.321665.

[8] E. W. Chionh, R. N. Goldman, and J. R. Miller. Using Multivariate Resultants to Find the Intersection of three Quadric Surfaces. *ACM Transactions on Graphics (TOG)*, 1991. ISSN 15577368. doi: 10.1145/116913.116917. URL https://doi.org/10.1145/116913.116917.

[9] D. Cox, J. Little, and D. O'Shea. Ideals, Varieties, and Algorithms: an introduction to computational algebraic geometry and commutative algebra. Undergraduate Texts in Mathematics. *Springer Verlag*, 1992. ISSN 978-1-4757-2183-6. URL https://www.springer.com/gp/book/9783319167206.

[10] D. N. Diatta, F. Rouillier, and M. F. Roy. On the computation of the topology of plane curves. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, 2014. ISBN 9781450325011. doi: 10.1145/2608628.2608670. URL doi.org/10.1145/2608628.2608670.

[11] L. Diković. Applications geogebra into teaching some topics of mathematics at the college level. *Computer Science and Information Systems*, 2009. ISSN 18200214. doi: 10.2298/CSIS0902191D. URL http://www.doiserbia.nb.rs/img/doi/1820-0214/2009/1820-02140902191D.pdf.

[12] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics: I. The generic algorithm. *Journal of Symbolic Computation*, 2008. ISSN 07477171. doi: 10.1016/j.jsc.2007.10.006. URL https://doi.org/10.1016/j.jsc.2007.10.006.

[13] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics: II. A classification of pencils. *Journal of Symbolic Computation*, 2008. ISSN 07477171. doi: 10.1016/j.jsc.2007.10.012. URL https://doi.org/10.1016/j.jsc.2007.10.006.

[14] L. Dupont, D. Lazard, S. Lazard, and S. Petitjean. Near-optimal parameterization of the intersection of quadrics: III. Parameterizing singular intersections. *Journal of Symbolic Computation*, 2008. ISSN 07477171. doi: 10.1016/j.jsc.2007.10.007. URL https://doi.org/10.1016/j.jsc.2007.10.007.

[15] R. T. Farouki, C. A. Neff, and M. A. O'Connor. Automatic parsing of degenerate quadric-surface intersections. In *ACM Transactions on Graphics*, 1989. doi: 10.1145/77055.77058. URL http://doi.acm.org/10.1145/77055.77058.

[16] N. Geismann, M. Hemmer, and E. Schömer. Computing a 3-dimensional cell in an arrangement of quadrics: Exactly and actually! In *Proceedings of the Annual Symposium on Computational Geometry*, 2001. doi: 10.1145/378583.378689. URL https://doi.org/10.1145/378583.378689.

[17] R. N. Goldman and J. R. Miller. Combining algebraic rigor with geometric robustness for the detection and calculation of conic sections in the intersection of two natural quadric surfaces. In *Proceedings of the 1st ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications, SMA 1991*, 1991. ISBN 0897914279. doi: 10.1145/112515.112545. URL https://doi.org/10.1145/112515.

[18] L. González-Vega. A subresultant theory for multivariate polynomials. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, ISSAC '91, pages 79–85, New York, NY, USA, 1991. ACM. ISBN 0-89791-437-6. doi: 10.1145/120694.120705. URL http://doi.acm.org/10.1145/120694.120705.

[19] L. Gonzalez-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Computer Aided Geometric Design*, 19(9):719–743, 2002. ISSN 01678396. doi: 10.1016/S0167-8396(02)00167-X. URL https://doi.org/10.1016/S0167-8396(02)00167-X.

[20] L. Gonzalez-Vega and I. Rúa. Solving the implicitization, inversion and reparametrization problems for rational curves through subresultants. *Computer Aided Geometric Design*, 26(9):941–961, dec 2009. ISSN 0167-8396. doi: 10.1016/J.CAGD.2009.07.003. URL https://www.sciencedirect.com/science/article/abs/pii/S0167839609000764.

[21] L. Gonzalez-Vega and A. Trocado. Using Maple to compute the intersection curve of two quadrics: improving the intersect plot command. In *Communications in Computer and Information Science*, 2019.

[22] J. Hall and G. Chamblee. Teaching Algebra and Geometry with GeoGebra: Preparing Pre-Service Teachers for Middle Grades/Secondary Mathematics Classrooms. *Computers in the Schools*, 2013. ISSN 07380569. doi: 10.1080/07380569.2013.764276. URL http://doi.org/10.1080/07380569.2013.764276.

[23] M. E. Hohmeyer. *Robust and Efficient Surface Intersection for Solid Modeling*. PhD thesis, EECS Department, University of California, Berkeley, May 1992. URL https://www2.eecs.berkeley.edu/Pubs/TechRpts/1992/CSD-92-681.pdf.

[24] J. K. Johnstone and C. K. Shene. Computing the intersection of a plane and a natural quadric. *Computers and Graphics*, 1992. ISSN 00978493. doi: 10.1016/0097-8493(92)90045-W. URL https://doi.org/10.1016/0097-8493(92)90045-W.

[25] S. Khouyibaba. Teaching mathematics with technology. In *Procedia - Social and Behavioral Sciences*, 2010. doi: 10.1016/j.sbspro.2010.12.210. URL https://doi.org/10.1016/j.sbspro.2010.12.210.

[26] S. Lazard, L. M. Peñaranda, and S. Petitjean. Intersecting quadrics: An efficient and exact implementation. In *Computational Geometry: Theory and Applications*, 2006. ISBN 1581138857. doi: 10.1016/j.comgeo.2005.10.004. URL https://doi.org/10.1016/j.comgeo.2005.10.004.

[27] J. Levin. A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. *Communications of the ACM*, 1976. ISSN 00010782. doi: 10.1145/360349.360355. URL https://doi.org/10.1145/360349.360355.

[28] J. Z. Levin. Mathematical models for determining the intersections of quadric surfaces. *Computer Graphics and Image Processing*, 11(1):73–87, sep 1979. ISSN 0146-664X. doi: 10.1016/0146-664X(79)90077-7. URL https://doi.org/10.1016/0146-664X(79)90077-7.

[29] Y.-B. Li. A new approach for constructing subresultants. *Applied Mathematics and Computation*, 183(1):471–476, 2006. ISSN 0096-3003. doi: https://doi.org/10.1016/j.amc.2006.05.120. URL http://www.sciencedirect.com/science/article/pii/S0096300306005807.

[30] J. R. Miller. Geometric Approaches to Nonplanar Quadric Surface Intersection Curves. *ACM Transactions on Graphics (TOG)*, 1987. ISSN 15577368. doi: 10.1145/35039.35041. URL https://doi.org/10.1145/35039.35041.

[31] J. R. Miller and R. N. Goldman. Geometric Algorithms for Detecting and Calculating All Conic Sections in the Intersection of Any 2 Natural Quadric Surfaces. *Graphical Models and Image Processing*, 1995. ISSN 10773169. doi: 10.1006/gmip.1995.1006. URL https://doi.org/10.1006/gmip.1995.1006.

[32] B. Mourrain, J. P. Técourt, and M. Teillaud. On the computation of an arrangement of quadrics in 3D. In *Computational Geometry: Theory and Applications*, 2005. doi: 10.1016/j.comgeo.2004.05.003. URL http://doi.org/10.1016/j.comgeo.2004.05.003.

[33] R. A. Saha, A. F. M. Ayub, and R. A. Tarmizi. The effects of GeoGebra on mathematics achievement: Enlightening Coordinate Geometry learning. In *Procedia - Social and Behavioral Sciences*, 2010. doi: 10.1016/j.sbspro.2010.12.095. URL https://doi.org/10.1016/j.sbspro.2010.12.095.

[34] T. S. A. . Salleh and E. Zakaria. Enhancing Students' Understanding in Integral Calculus through the Integration of Maple in Learning. *Procedia - Social and Behavioral Sciences*, 2013. ISSN 18770428. doi: 10.1016/j.sbspro.2013.10.734. URL https://doi.org/10.1016/j.sbspro.2013.10.734.

[35] R. F. Sarraga. Algebraic methods for intersections of quadric surfaces in GMSOLID. *Computer Vision, Graphics and Image Processing*, 1983. ISSN 0734189X. doi: 10.1016/0734-189X(83)90066-X. URL https://doi.org/10.1016/0734-189X(83)90066-X.

[36] E. Schömer and N. Wolpert. An exact and efficient approach for computing a cell in an arrangement of quadrics. *Computational Geometry: Theory and Applications*, 2006. ISSN 09257721. doi: 10.1016/j.comgeo.2004.02.007. URL https://doi.org/10.1016/j.comgeo.2004.02.007.

[37] J. R. Sendra and F. Winkler. Algorithms for rational real algebraic curves. *Fundam. Inf.*, 39(1,2):211–228, Apr. 1999. ISSN 0169-2968. URL http://dl.acm.org/citation.cfm?id=2378083.2378093.

[38] C. K. Shene and J. K. Johnstone. On the Lower Degree Intersections of two Natural Quadrics. *ACM Transactions on Graphics (TOG)*, 1994. ISSN 15577368. doi: 10.1145/195826.197316. URL doi.org/10.1145/195826.197316.

[39] A. Trocado and L. González-Vega. Computing the intersection of two quadrics through projection and lifting. *CoRR*, abs/1903.06983, 2019. URL http://arxiv.org/abs/1903.06983.

[40] A. Trocado, L. Gonzalez-Vega, and J. M. Dos Santos. Intersecting two quadrics with GeoGebra. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019. ISBN 9783030213626. doi: 10.1007/978-3-030-21363-3_20. URL https://doi.org/10.1007/978-3-030-21363-3_20.

[41] J. von zur Gathen, J. Gerhard, J. von zur Gathen, and J. Gerhard. Fundamental algorithms. In *Modern Computer Algebra*. 2013. doi: 10.1017/cbo9781139856065.004. URL https://doi.org/10.1017/cbo9781139856065.004.

[42] W. Wang, B. Joe, and R. Goldman. Computing quadric surface intersections based on an analysis of plane cubic curves. *Graphical Models*, 2002. ISSN 15240703. doi: 10.1016/S1077-3169(02)00018-7. URL https://doi.org/10.1016/S1077-3169(02)00018-7.

[43] W. Wang, R. Goldman, and C. Tu. Enhancing Levin's method for computing quadric-surface intersections. *Computer Aided Geometric Design*, 20(7):401–422, oct 2003. ISSN 0167-8396. doi: 10.1016/S0167-8396(03)00081-5. URL https://www.sciencedirect.com/science/article/abs/pii/S0167839603000815.

[44] I. Wilf and Y. Manor. Quadric-surface intersection curves: shape and structure. *Computer-Aided Design*, 1993. ISSN 00104485. doi: 10.1016/0010-4485(93)90018-J. URL https://doi.org/10.1016/0010-4485(93)90018-J.

[45] N. Wolpert. *An exact and efficient approach for computing a cell in an arrangement of quadrics*. PhD thesis, Universität des Saarlandes, Saarbrücken, 2002. URL http://hdl.handle.net/11858/00-001M-0000-000F-2EFC-A.

[46] C. Yap. *Fundamental problems of algorithmic algebra*. Oxford University Press, 2000. ISBN 0-19-512516-9.