Intersection Curves Between Quadric Surfaces of Revolution

Yin LIU

B.Eng. (Beijing University of Science and Technology), 1982

Department of Mechanical Engineering
McGill University
Montréal

May, 1995

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

ISBN 0-612-05458-6

Abstract

A novel algebraic method is used to determine the intersection between two bi-quadratic surfaces. The underlying idea is to deal with functional decomposition by reducing the order of high-degree functions and thereby overcome the problems common to other methods.

Une méthode algébrique nouvelle se servait de déterminant à l´intersection des deux surfaces quatres. L´idée trait avec decomposition functionnelle de rédui l´ordre de haut degré functions et ainsi triomph de la probléme que il a fait des remarques autre méthodes.

Acknowledgements

I am grateful to Professor P. J. Zsombor-Murray, my research supervisor, for his guidance and encouragement.

My thanks are extended Mr. Ming Jian, for his many useful suggestions and great assistance. Thanks are also due to my fellow graduate student, Mr. Ren-Rong Zhang, who helped me to face difficulty with good humor.

I am indebted to my mother Shu-Zhen Kong, my wife Wan-Ying Shi and my daughter Bing-Shan Liu, for the love, understanding and support which helped me to persevere.

To My Wife Wan-Ying and daughter Bing-Shan

Contents

List of Figures

Chapter 1

Introduction

Intersection is a fundamental process in computational geometry, needed to build and interrogate models of complex shapes in the computer. We need intersection computation primarily to evaluate set operations on primitive volumes in creating boundary representations of complex artifacts. Such capability helps in the design representation of complex objects, in finite-element discretization, in computer animation, in feature recognition, and in simulation and control of manufacturing processes. Similarly, intersection is useful in scientific visualization to provide methods for visualizing implicitly defined objects and to contour multivariate functions representing some property of a system.

1.1 Background and Basic Terminology

1.1.1 Surface-intersection methods

A good surface-intersection technique has to balance three conflicting goals: *efficiency*, *robustness*, and *accuracy*.

Typically, a numerical algorithm is efficient, but is not fully robust and so may fail in certain cases. Furthermore, the accuracy a numerical method can deliver varies with the surface degree, with the local surface geometry at the intersection curve, and with the angle at which the surfaces intersect. Algorithms based on exact arithmetic, on the other hand, are fully robust and accurate, but are usually slow. Perhaps the goals of efficiency, robustness, and accuracy cannot be met simultaneously without some compromises, and we might have to negotiate those compromises judiciously, appropriate to the particular application.

Surface-intersection methods can be classified in three main categories: algebraic, marching, and subdivision. Most of these methods were developed in the context of polynomial surfaces.

1.1.2 Algebraic methods [1] – [7]

Algebraic methods rely on the derivation of a governing equation describing the intersection of two surfaces. For polynomial surfaces, the resulting equation is an algebraic curve $f(u,v) = 0$, where f is a polynomial in two parametric variables u and v. This equation can, for example, be obtained by substituting the three Cartesian coordinate expressions of a rational polynomial surface $\mathbf{r} = \mathbf{r}(u,v)$ into the equation of an implicit algebraic surface $f(\mathbf{r}) = 0$. In theory, we can handle the intersection between two rational polynomial parametric surfaces by obtaining an algebraic (implicit polynomial) representation for one of the surfaces, for example, with methods discussed by Hoffmann [5] and Miller [6]. The relatively high degree of this algebraic representation and the subsequent substitution of the second rational polynomial surface into this high-degree equation lead to an algebraic curve of even higher degree.

Detecting the topological configuration of a high-degree algebraic curve with integer or real coefficients is a difficult problem that we can approach with cylindrical algebraic decomposition. Hoffmann [5] provides an overview. These methods, implemented in rational arithmetic, are topologically reliable but it is not clearly understood how to process algebraic curves with general, real number coefficients found in practice.

1.1.3 Marching methods [8] – [11]

Marching methods involve generating point sequences of an intersection curve branch by stepping from a given point on the required curve in a direction prescribed by the curve's local differential geometry as described by Bajaj [11]. However, such methods are by themselves incomplete in that they require starting points for every branch of the solution. Starting points are usually obtained using subdivision methods like those developed by Barnhill & Kersey [10]. Marching methods also require a variable stepping size appropriate for the local length scales of the problem. Variable step size was considered by Kriezis, Patrikalakis & Wolter [8]. Incorrect step size might lead to erroneous connectivity of solution branch or even to endless looping in the presence of closely spaced features.

1.1.4 Subdivision methods [12] – [17]

In their most basic form, subdivision methods involve recursive decomposition of the problem into a simpler, similar problem until we reach a level of simplicity that allows direct solution. Unlike marching methods, subdivision techniques do not require starting points. This is an important advantage. A disadvantage of subdivision techniques used in intersection curve evaluation is that, in actual implementations with finite subdivision steps, correct connectivity of solution branches in the vicinity of singular or near-singular points is difficult to guarantee, small loops might be missed, or extraneous loops might be present in the solution approximation. Furthermore, if we use subdivision methods for high-precision evaluation of the entire intersection set, they lead to data proliferation and are consequently slow and unattractive. Many CAD/CAM applications, for example, the off-line programming of a robotic welding operation which must follow a seam represented by the intersection curves, require high accuracy. Pure subdivision methods are impractical for this purpose.

1.2 An idea to improve analytic methods

1.2.1 Common problem

Common problems of intersection methods include the difficulty in handling singularities and surface overlap, as shown in Figs. 1.1 & 1.2, respectively, as well as identification of closely spaced features and small loops. The above algorithmic difficulties are further compounded by the numerical error in

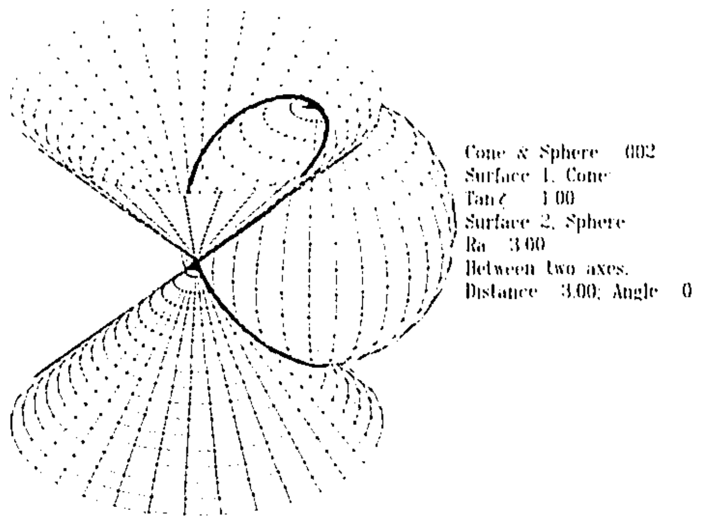


Figure 1.1: Intersection Between Cone and Sphere with Singularity

finite precision computations.

A novel algebraic method to overcome the problems is introduced below.

1.2.2 New analytic method

The brief summary, § 1.1.1 - § 1.1.4, above, implies that algebraic methods are more efficient than others. But serious problems arise due to high-degree equations. So a novel approach, based on a particular subdivision method, is proposed so as to reduce the degree. This new method also relies on the derivation of a particular form of governing equation to describe the intersection of two surfaces. The resulting equation is not an algebraic curve equation ($f_c(u,v) = 0$), but an equation which represents coincident points on coincident curves, one on each surface, which is of the form ($f_p(u_1,u_2) = 0$), where f_p is a polynomial in u_1, u_2. This equation can be obtained by substi-

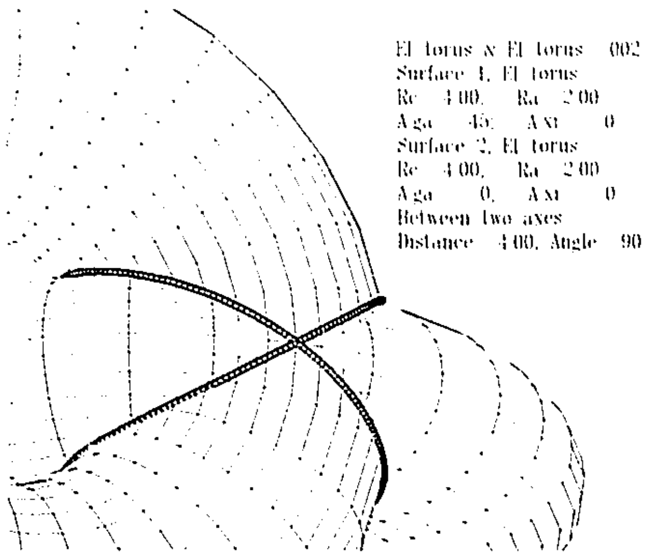


Figure 1.2: Intersection Between Two Cyclids with Overlap

tution of the three Cartesian coordinate relationships between two surfaces ($\mathbf{r}_1(u_1, v_1) = \mathbf{r}_2(u_2, v_2)$). In theory, we can handle the intersection between two polynomial parametric surfaces by obtaining an algebraic representation for intersecting points between two sets of curves on the surfaces. The degree of a points of intersection equation is only half that of an algebraic curve equation. For example, say an algebraic curve equation describing intersection of two surfaces is of degree 8. That means that it has a maximum of 8 solutions. But, the equivalent intersecting point equation has a degree of 4 and hence a maximum of 4 solutions exist. With this novel algebraic method we gain advantages in addition to reducing the degree. These include:

1. Through the introduction of two articulated rigid body mechanism families as surface generators, we obtain considerable geometric insight. This makes it easy to build surfaces and to classify surface species within each family. For example, we can set up a simple, variable parameter RP mechanism model to generate ruled surfaces of revolution. Here and in what follows, the abbreviations, R $\equiv$ revolute, P $\equiv$ prismatic, S $\equiv$ spheric, C $\equiv$ cylindric and U $\equiv$ hooke or bi-revolute, are used to specify joint type. The species in this first family include cylinder, cone and hyperboloid. On the other hand, we gain insight useful in the study of closed, sometimes overconstrained mechanisms. For example, an intersecting point equation obtained from a pair of parallel connected RR mechanism models, of the second family whose species include cyclids where the sphere and torus are special cases, describes the motion of a RRSRR closed chain or single loop. Consider that such mechanisms may be studied by systematically varying the RR mechanism parameters to determine the conditions under which the S joint is required to furnish less than 3 dof mobility. Such a study might provide additional understanding as regards certain types of overconstrained 6R, 5R and 4R mechanisms.

2. It makes it easy to handle singularities and surface overlap. Curves of intersection are mapped in two steps.The first step is to solve for u_1 (or u_2) with the intersecting points equation ($f_p(u_1, u_2) = 0$). The second step is to solve for v_1 (or v_2) by backsubstituting into an equation, $f_1(v_1) = 0$ (or $f_2(v_2) = 0$), which can be obtained in the same way as

the intersecting points equation, where f_i is a second degree polynomial in v_1 (or v_2). These equations are explicit. So solution branch points and other types of singularity pose no particular problem. When the solutions of both equations have a pair of repeated roots, it means that the intersection curves overlap or crossover.

3. The intersection curves can be computed accurately and quickly because this method finds the roots of equations directly.

It should not be overlooked, however, that the method has a few disadvantages:

1. It cannot be applied to every pair of surfaces,

2. For every pairing of surface species, different algorithms have to be formulated, and

3. The code is lengthy and may be difficult to implement in microcontrolers used for online trajectory generation.

Chapter 2

The functions of surfaces

2.1 Functions of surfaces of revolution

Referring to Fig. 2.1, any point d on a surface can be found when its generating curve is given. The position vector $\mathbf{d}$ is defined as

$$\mathbf{d} = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \begin{bmatrix} (x_c + R_c)\cos\beta - y_c \sin\beta \\ (x_c + R_c)\sin\beta + y_c \cos\beta \\ z_c + Z \end{bmatrix} \tag{2.1}$$

where (R_c, β, Z) are the cylindrical base coordinates, origin o_c, of the frame in which the generating curve is described.

2.2 Functions of surfaces swept by a straight line

Referring to the RP mechanism model shown in Fig.(2.2), point d on a surface can be found when the joint variables, (β and v), of the surface are given.

The position vector $\mathbf{d}$ is defined by eqn.(2.1).

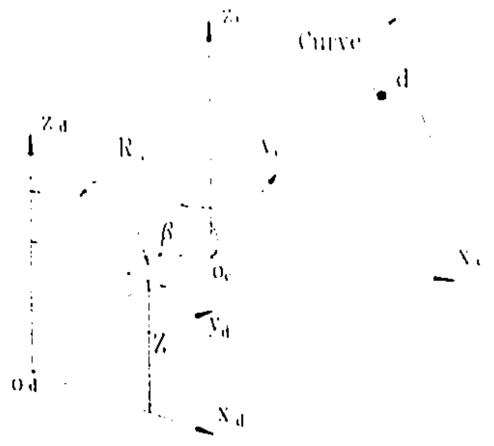


Figure 2.1: Generating a Curved Surface of Rotation

Again, R_c is the radius of revolution and Z is the cylindroaxial coordinate locating origin o_c in frame o_d.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} 0 \\ -v \sin\zeta \\ v \cos\zeta \end{bmatrix} \tag{2.2}$$

2.2.1 Cylinder

For the surface of a cylinder,

$$\begin{aligned} R_c &\neq 0 \\ \zeta &= 0 \end{aligned}$$

2.2.2 Cone

For the surface of a cone,

$$\begin{aligned} R_c &= 0 \\ \zeta &\neq 0 \end{aligned}$$

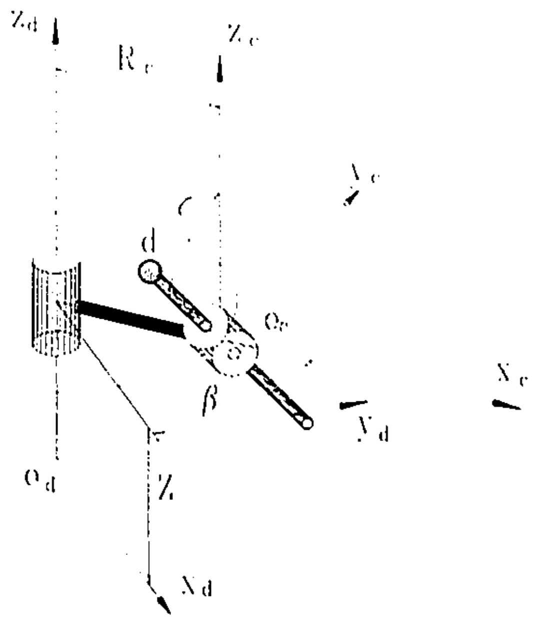


Figure 2.2: The Surface of Hyperboloid, Cone or Cylinder

2.2.3 Hyperboloid

For the surface of a hyperboloid,

$$\begin{aligned} R_c &\neq 0 \\ \zeta &\neq 0 \end{aligned}$$

2.3 Functions of surfaces swept by a circle

Referring to the RR mechanism model shown in Fig.(2.3), point d on a surface can be found when the angular displacement parameters, i.e., joint variables, (β and α), are given. **d**, the position vector of d, is defined by eqn.(2.1) where R_c is the length of the common perpendicular between

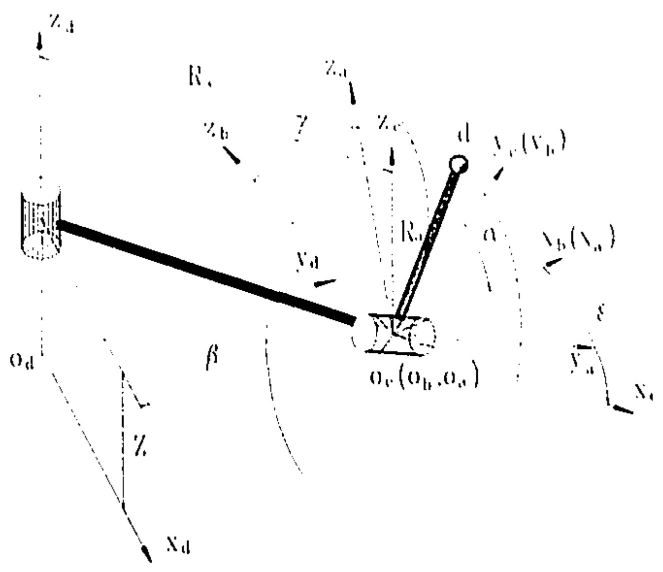


Figure 2.3: The Surface of Torus or Sphere

the RR pair's axes and Z is the offset displacement of the equatorial plane, along the polar axis, measured from the base frame origin. The point d on a circle with radius R_a is defined by rotating around axis y_a with angle α in coordinate a. The coordinate a rotates around axis x_a with angle γ in coordinate a to be translated into coordinate b, then the coordinate b rotates around axis y_b with angle ξ in coordinate b to be translated into coordinate c. So, the position d in the coordinate c can be defined by given R_a, α, γ and ξ.

$$\mathbf{c} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} R_a(\cos\alpha\cos\xi - \sin\alpha\cos\gamma\sin\xi) \\ R_a\sin\alpha\sin\gamma \\ R_a(\cos\alpha\sin\xi + \sin\alpha\cos\gamma\cos\xi) \end{bmatrix} \tag{2.3}$$

Here R_a is the distance to d measured perpendicular to the axis of the mobile revolute, from o_c, the point of intersection between the axis and the common perpendicular. γ is an angle which defines the aspect ratio of the cyclid's form. This is the angle between the plane of the circle of radius R_a and the local meridianal plane through o_c. ξ is an angle to define the axial orientation of the elliptical projection of the circle, traced by $\mathbf{d}$, on the plane (z_d, x_c). Note that $\mathbf{c}$ is the position vector of point d measured in the frame with origin o_c.

2.3.1 Sphere

For the surface of a sphere, $R_c = 0$ and $0 \leq \beta \leq \pi$.

2.3.2 Torus and Cyclid

For the surface of a torus or cyclid, $R_c \neq 0$ and $0 \leq \beta \leq 2\pi$.

Chapter 3

The intersection functions

3.1 Normal function

When considering the intersection between two surfaces of revolution one will be called the primary surface. Its axis will be colinear with the z−axis of the global Cartesian frame and the origin of this frame will be coincident with o_{d1}, the primary surface local origin. The secondary surface origin, o_{d2}, is located a distance Y from o_{d1} and the line joining the origins is the common perpendicular between the surface axes of revolution. The angle between axes is ϕ. In Fig.(3.1) we see the vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ which locate a typical point on the curve of intersection. $\mathbf{d}_1$ and $\mathbf{d}_2$ are related by the following equation.

$$\begin{bmatrix} x_{d1} \\ y_{d1} \\ z_{d1} \end{bmatrix} = \begin{bmatrix} x_{d2}\cos\phi - z_{d2}\sin\phi \\ y_{d2} + Y \\ x_{d2}\sin\phi + z_{d2}\cos\phi \end{bmatrix} \tag{3.1}$$

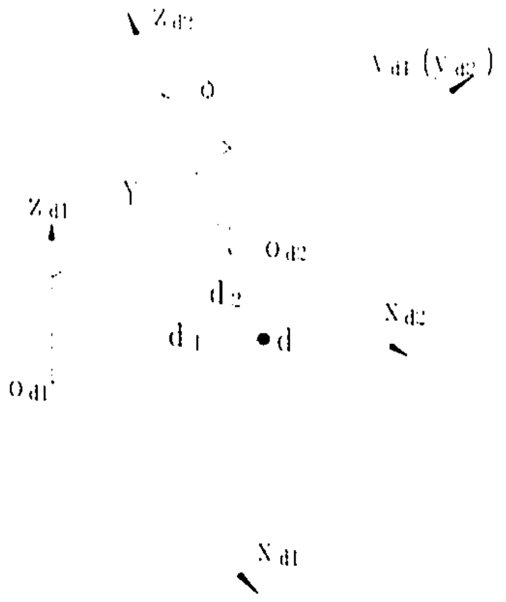


Figure 3.1: The Relationship Between Two Surfaces of Revolution

3.2 Parameters, known and unknown

When two surfaces intersect, if we obtain the parameters, β and α(or v), of any surface, we can map the intersecting curve. There are three functions in eqn.(3.1), which are fewer than the four unknowns, β_1 and α_1(or v_1) in the primary surface and β_2 and α_2(or v_2) in the secondary surface. So, we have to fix one of these unknowns to solve for three others. There are a number of ways to do this. A direct calculation method, where the three functions are reduced to one so as to solve for a single unknown, will be introduced in next chapter. This method provides good precision in a fast algorithm.

Chapter 4

General functions of intersection

4.1 General functions of intersection with known, α_2 or u_2

4.1.1 Functions

The main point is to eliminate the unknowns of β_1 and β_2 and to obtain a function, which includes the single unknown, α_1(or v_1). We proceed as follows.

Square eqn.(3.1) to obtain

$$x_{d1}^2 + y_{d1}^2 + z_{d1}^2 = x_{d2}^2 + y_{d2}^2 + z_{d2}^2 + Y^2 + 2Y y_{d2} \tag{4.1}$$

Square eqn.(2.1) and substitute into eqn.(4.1).

$$\begin{aligned} e_1 &= 2Y y_{d2} \\ &= (x_{c1} + R_{c1})^2 \\ &+ y_{c1}^2 + (z_{c1} + Z_1)^2 \\ &- [(x_{c2} + R_{c2})^2 + y_{c2}^2 + (z_{c2} + Z_2)^2 + Y^2] \end{aligned} \tag{4.2}$$

From the third function in eqn.(3.1) and the third in eqn.(2.1),

$$\begin{aligned} c_2 = x_{d2}\sin\phi &= z_{d1} - z_{d2}\cos\phi \\ &= z_{c1} + Z_1 - (z_{c2} + Z_2)\cos\phi \end{aligned} \tag{4.3}$$

From the first and second functions in eqn.(2.1),

$$x_{d2}^2 + y_{d2}^2 = (x_{c2} + R_{c2})^2 + y_{c2}^2 \tag{4.4}$$

Multiply eqn.(4.2) by $\sin\phi$ and multiply eqn.(4.3) by $2Y$. Then add and substitute the result into eqn.(4.4). The functions, which only include one unknown, α_1(or v_1), can be written as

$$\begin{aligned} [x_{c1}^2 + y_{c1}^2 + z_{c1}^2 &+ 2R_{c1}x_{c1} + 2Z_1 z_{c1} + d_1]^2 \sin^2\phi \\ &+ 4Y^2(z_{c1} + d_2)^2 + 4d_3Y^2\sin^2\phi = 0 \end{aligned} \tag{4.5}$$

$$d_1 = R_{c1}^2 + Z_1^2 - [(x_{c2} + R_{c2})^2 + y_{c2}^2 + (z_{c2} + Z_2)^2 + Y^2] \tag{4.6}$$

$$d_2 = Z_1 - (z_{c2} + Z_2)\cos\phi \tag{4.7}$$

$$d_3 = -[(x_{c2} + R_{c2})^2 + y_{c2}^2] \tag{4.8}$$

4.1.2 Secondary Surface

1. For the surface of a cyclid or sphere,

$$\begin{bmatrix} x_{c2} \\ y_{c2} \\ z_{c2} \end{bmatrix} = \begin{bmatrix} R_{a2}(\cos\alpha_2\cos\xi_2 - \sin\alpha_2\cos\gamma_2\sin\xi_2) \\ R_{a2}\sin\alpha_2\sin\gamma_2 \\ R_{a2}(\cos\alpha_2\sin\xi_2 + \sin\alpha_2\cos\gamma_2\cos\xi_2) \end{bmatrix} \tag{4.9}$$

2. For a hyperbolic, conical or cylindrical surface,

$$\begin{bmatrix} x_{c2} \\ y_{c2} \\ z_{c2} \end{bmatrix} = \begin{bmatrix} 0 \\ -v_2\sin\zeta_2 \\ v_2\cos\zeta_2 \end{bmatrix} \tag{4.10}$$

4.1.3 Primary Surface

1. For the surface of a cyclid or sphere,

$$\begin{bmatrix} x_{c1} \\ y_{c1} \\ z_{c1} \end{bmatrix} = \begin{bmatrix} R_{a1}(\cos\alpha_1\cos\xi_1 - \sin\alpha_1\cos\gamma_1\sin\xi_1) \\ R_{a1}\sin\alpha_1\sin\gamma_1 \\ R_{a1}(\cos\alpha_1\sin\xi_1 + \sin\alpha_1\cos\gamma_1\cos\xi_1) \end{bmatrix} \tag{4.11}$$

Substitute eqn.(4.11) into eqn.(4.5) and define

$$t = \tan\frac{\alpha_1}{2} \tag{4.12}$$

Then differentiate the result to obtain

$$\begin{aligned} (n_1t^2 &+ n_2t + n_3)^2 + 4Y^2(n_4t^2 + n_5t + n_6)^2\sin^2\phi \\ &+ 4d_3Y^2(t^2+1)^2\sin^2\phi = 0 \end{aligned} \tag{4.13}$$

where

$$n_1 = R_{a1}^2 + d_1 - 2R_{a1}(Z_1\sin\xi_1 + R_{c1}\cos\xi_1) \tag{4.14}$$

$$n_2 = 4R_{a1}\cos\gamma_1(Z_1\cos\xi_1 - R_{c1}\sin\xi_1) \tag{4.15}$$

$$n_3 = R_{a1}^2 + d_1 + 2R_{a1}(Z_1\sin\xi_1 + R_{c1}\cos\xi_1) \tag{4.16}$$

$$n_4 = d_2 - R_{a1}\sin\xi_1 \tag{4.17}$$

$$n_5 = 2R_{a1}\cos\gamma_1\cos\xi_1 \tag{4.18}$$

$$n_6 = d_2 + R_{a1}\sin\xi_1 \tag{4.19}$$

Differentiating eqn.(4.13) gives

$$m_4t^4 + m_3t^3 + m_2t^2 + m_1t + m_0 = 0 \tag{4.20}$$

where

$$m_4 = n_1^2\sin^2\phi + 4Y^2n_4^2 + 4d_3Y^2\sin^2\phi \tag{4.21}$$

$$m_3 = 8Y^2n_5n_4 + 2n_2n_1\sin^2\phi \tag{4.22}$$

$$\begin{aligned} m_2 = {} & 4Y^2(2n_6n_4 + n_5^2) + 8d_3Y^2\sin^2\phi \\ & + \sin^2\phi(2n_3n_1 + n_2^2) \end{aligned} \tag{4.23}$$

$$m_1 = 2n_3n_2\sin^2\phi + 8Y^2n_6n_5 \tag{4.24}$$

$$m_0 = n_3^2\sin^2\phi + 4Y^2n_6^2 + 4Y^2d_3\sin^2\phi \tag{4.25}$$

According to eqn.(4.12), when $\mid \alpha_1 \mid \rightarrow \pi$, which is possible in many instances of surface intersection, then $\mid t \mid \rightarrow \infty$. In such cases we can not obtain a satisfactory solution with eqn.(4.20). In fact, when $\mid \alpha_1 \mid$ is close to π the solution will be badly conditioned. Under such conditions, t will be replaced by its reciprocal. In the range

$$\alpha_1 = 2\arctan(t); \qquad -\frac{\pi}{2} \leq \alpha_1 < \frac{\pi}{2} \tag{4.26}$$

we will use

$$\frac{1}{t} = \tan\frac{\alpha_1}{2} \tag{4.27}$$

so as to obtain

$$m_0 t^4 + m_1 t^3 + m_2 t^2 + m_3 t + m_4 = 0 \tag{4.28}$$

$$\alpha_1 = \pi - 2\arctan(t); \qquad \frac{\pi}{2} \le \alpha_1 < \frac{3\pi}{2} \tag{4.29}$$

2. Similarly, for a hyperbolic, conical or cylindrical surface,

$$\begin{bmatrix} x_{c1} \\ y_{c1} \\ z_{c1} \end{bmatrix} = \begin{bmatrix} 0 \\ -v_1 \sin\zeta_1 \\ v_1 \cos\zeta_1 \end{bmatrix} \tag{4.30}$$

The result obtained by differentiating eqn.(4.5) is

$$\begin{aligned}(n_1 z_{c1}^2 + n_2 z_{c1} + d_1)^2 \sin^2\phi &+ 4Y^2(z_{c1} + d_2)^2 \\ &+ 4d_3 Y^2 \sin^2\phi = 0 \end{aligned} \tag{4.31}$$

$$n_1 = 1 + \tan^2\zeta_1 \tag{4.32}$$

$$n_2 = 2Z_1 \tag{4.33}$$

Differentiation of eqn.(4.31) results in

$$m_4 z_{c1}^4 + m_3 z_{c1}^3 + m_2 z_{c1}^2 + m_1 z_{c1} + m_0 = 0 \tag{4.34}$$

$$m_4 = n_1^2 \sin^2\phi \quad (4.35)$$

$$m_3 = 2n_2 n_1 \sin^2\phi \quad (4.36)$$

$$m_2 = 4Y^2 + \sin^2\phi(2d_1 n_1 + n_2^2) \quad (4.37)$$

$$m_1 = 2d_1 n_2 \sin^2\phi + 8Y^2 d_2 \quad (4.38)$$

$$m_0 = d_1^2 \sin^2\phi + 4Y^2 d_2^2 + 4Y^2 d_3 \sin^2\phi \quad (4.39)$$

4.2 Solving for β_2

To obtain the solution for α_1(or v_1), we need to solve for β_2 or β_1 to map the intersecting curve.

4.2.1 The solution for $Y \neq 0$ and $\sin\phi \neq 0$

from eqn.(2.1) and 4.2), the solution of β_2 can be written as

$$s_1 = e_1(x_{c2} + R_{c2})\sin\phi - 2e_2 Y y_{c2} \quad (4.40)$$

$$s_2 = e_1 y_{c2}\sin\phi + 2e_2 Y(x_{c2} + R_{c2}) \quad (4.41)$$

$$\begin{cases} \beta_2 = \arctan(\frac{s_1}{s_2}); & s_2 > 0 \\ \beta_2 = \arctan(\frac{s_1}{s_2}) + \pi; & s_2 < 0 \end{cases} \quad (4.42)$$

From the result of eqn.(4.42), β_2 is chosen as

$$-\frac{\pi}{4} \leq \beta_2 < \frac{\pi}{4}; \quad or \quad \frac{3\pi}{4} \leq \beta_2 < \frac{5\pi}{4} \quad (4.43)$$

$$\begin{cases} \beta_2 = \frac{\pi}{2} - \arctan(\frac{s_2}{s_1}); & s_1 > 0 \\ \beta_2 = \frac{3\pi}{2} - \arctan(\frac{s_2}{s_1}); & s_1 < 0 \end{cases} \tag{4.44}$$

From the result of eqn.(4.44), β_2 is chosen as

$$\frac{\pi}{4} \le \beta_2 < \frac{3\pi}{4}; \quad or \quad \frac{5\pi}{4} \le \beta_2 < \frac{7\pi}{4} \tag{4.45}$$

4.2.2 The solution for $Y = 0$ or $\sin\phi = 0$

There are, in this case, two solutions for any given α_2(or v_2) so as to produce α_1(or v_1). From eqn.(2.1), use

$$t = \tan\frac{\beta_2}{2} \tag{4.46}$$

with

$$m_2 t^2 + m_1 t + m_0 = 0 \tag{4.47}$$

where

$$\beta_2 = 2\arctan(t); \qquad -\frac{\pi}{2} \le \beta_2 < \frac{\pi}{2} \tag{4.48}$$

However, use

$$\frac{1}{t} = \tan\frac{\beta_2}{2} \tag{4.49}$$

with

$$m_0 t^2 + m_1 t + m_2 = 0 \tag{4.50}$$

where

$$\beta_2 = \pi - 2\arctan(t); \qquad \frac{\pi}{2} \le \beta_2 < \frac{3\pi}{2} \tag{4.51}$$

1. For $Y = 0$,

$$m_2 = -(x_{c2} + R_{c2}) \sin\phi - e_2 \tag{4.52}$$

$$m_1 = -2y_{c2} \sin\phi \tag{4.53}$$

$$m_0 = (x_{c2} + R_{c2}) \sin\phi - e_2 \tag{4.54}$$

2. For $\sin\phi = 0$,

$$m_2 = -2Yy_{c2} - e_1 \tag{4.55}$$

$$m_1 = 4Y(x_{c2} + R_{c2}) \tag{4.56}$$

$$m_0 = 2Yy_{c2} - e_1 \tag{4.57}$$

Chapter 5

Special functions of intersection

When $\phi = 0$ and $Y = 0$ (primary surface and secondary surface share the same axis), the intersecting curves are circles, perpendicular to the axes of revolution of the two generating curves. The radius, R, and centre points, O, are defined as

$$R = \sqrt{(x_{c1} + R_{c1})^2 + y_{c1}^2} \tag{5.1}$$

$$\begin{bmatrix} O_x \\ O_y \\ O_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ z_{c1} + Z_1 \end{bmatrix} \tag{5.2}$$

Coordinates (x_{c1},y_{c1},z_{c1}) can be obtained by using eqn.(3.1) and solving for α_1(or v_1).

5.1 Primary surfaces swept by a straight line

(hyperboloid, cone or cylinder)

For primary surface of hyperboloid, cone or cylinder and secondary suface of hyperboloid, cone or cylinder, the solution for v_1 can be written as

$$m_2 v_1^2 + m_1 v_1 + m_0 = 0 \tag{5.3}$$

where

$$m_2 = \tan^2 \zeta_1 - \tan^2 \zeta_2 \tag{5.4}$$

$$m_1 = -2(Z_1 - Z_2) \tan^2 \zeta_2 \tag{5.5}$$

$$m_0 = R_{c1}^2 - R_{c2}^2 - (Z_1 - Z_2)^2 \tan^2 \zeta_2 \tag{5.6}$$

5.2 Primary surfaces swept by a circle

(cyclids or spheres)

For primary surface of a cyclid or sphere, there are two solutions for α_1.

5.2.1 Secondary surfaces swept by a straight line

(hyperboloid, cone or cylinder)

For a primary surface of cyclid or sphere and secondary suface of hyperboloid, cone or cylinder, the solution for α_1 can be written as

$$m_4 t^4 + m_3 t^3 + m_2 t^2 + m_1 t + m_0 = 0 \tag{5.7}$$

where t is defined as

$$t = \tan \frac{\alpha_1}{2} \tag{5.8}$$

where

$$\alpha_1 = 2\arctan(t); \qquad -\frac{\pi}{2} \le \alpha_1 < \frac{\pi}{2} \tag{5.9}$$

t is defined as

$$\frac{1}{t} = \tan\frac{\alpha_1}{2} \tag{5.10}$$

to obtain

$$m_0 t^4 + m_1 t^3 + m_2 t^2 + m_3 t + m_4 = 0 \tag{5.11}$$

where

$$\alpha_1 = \pi - 2\arctan(t); \qquad \frac{\pi}{2} \le \alpha_1 < \frac{3\pi}{2} \tag{5.12}$$

such that

$$\begin{aligned}
m_4 &= s_6 - s_1^2\tan^2\zeta_2 - s_3^2 - 2R_{a1}s_4 && (5.13)\\
m_3 &= -4R_{a1}\cos\gamma_1[\cos\xi_1(s_1\tan^2\zeta_2 + s_3) + s_5] && (5.14)\\
m_2 &= 2s_6 - 2(Z_1^2 - R_{a1}^2\sin^2\xi_1) - 4R_{a1}^2\cos^2\gamma_1\cos^2\xi_1 \\
&\quad -2(s_1s_2 + 2R_{a1}^2\cos^2\gamma_1\cos^2\xi_1)\tan^2\zeta_2 && (5.15)\\
m_1 &= -4R_{a1}\cos\gamma_1[s_5 + \cos\xi_1(Z_1 + R_{a1}\sin\xi_1 + s_2\tan^2\zeta_2)] && (5.16)\\
m_0 &= s_6 + 2R_{a1}s_4 - (R_{a1}\sin\xi_1 + Z_1)^2 - s_2^2\tan^2\zeta_2 && (5.17)
\end{aligned}$$

and

$$\begin{aligned}
s_1 &= Z_1 - Z_2 - R_{a1}\sin\xi_1 && (5.18)\\
s_2 &= Z_1 - Z_2 + R_{a1}\sin\xi_1 && (5.19)\\
s_3 &= Z_1 - R_{a1}\sin\xi_1 && (5.20)\\
s_4 &= R_{c1}\cos\xi_1 + Z_1\sin\xi_1 && (5.21)\\
s_5 &= R_{c1}\sin\xi_1 - Z_1\cos\xi_1 && (5.22)\\
s_6 &= R_{a1}^2 + R_{c1}^2 + Z_1^2 - R_{c2}^2 && (5.23)
\end{aligned}$$

5.2.2 Secondary surfaces swept by a circle

(cyclids or spheres)

For a primary surface of cyclid or sphere and secondary suface of cyclid or sphere, the solution of α_1 can be written as

$$m_4t^4 + m_3t^3 + m_2t^2 + m_1t + m_0 = 0 \tag{5.24}$$

where t is defined as

$$t = \tan\frac{\alpha_1}{2} \tag{5.25}$$

where

$$\alpha_1 = 2\arctan(t); \qquad -\frac{\pi}{2} \leq \alpha_1 < \frac{\pi}{2} \tag{5.26}$$

t is defined as

$$\frac{1}{t} = \tan\frac{\alpha_1}{2} \tag{5.27}$$

to obtain

$$m_0t^4 + m_1t^3 + m_2t^2 + m_3t + m_4 = 0 \tag{5.28}$$

where

$$\alpha_1 = \pi - 2\arctan(t); \qquad \frac{\pi}{2} \leq \alpha_1 < \frac{3\pi}{2} \tag{5.29}$$

such that

$$\begin{aligned}
m_4 &= \cos^2\gamma_2(s_{10}^2 - 4R_{a2}^2R_{c2}^2) + s_9^2 && (5.30)\\
m_3 &= 2s_{11}s_9 + 2s_{12}s_{10}\cos^2\gamma_2 && (5.31)\\
m_2 &= 2s_7s_9 + s_{11}^2 + \cos^2\gamma_2(2s_8s_{10} + s_{12}^2 - 8R_{a2}^2R_{c2}^2) && (5.32)
\end{aligned}$$

$$m_1 = 2s_8s_{12}\cos^2\gamma_2 + 2s_7s_{11} \tag{5.33}$$

$$m_0 = \cos^2\gamma_2(s_8^2 - 4R_{a2}^2R_{c2}^2) + s_7^2 \tag{5.34}$$

and

$$s_1 = R_{a1}^2 + R_{c1}^2 + Z_1^2 - R_{a2}^2 - R_{c2}^2 - Z_2^2 \tag{5.35}$$

$$s_2 = Z_1 - Z_2 + R_{a1}\sin\xi_1 \tag{5.36}$$

$$s_3 = Z_1 - Z_2 - R_{a1}\sin\xi_1 \tag{5.37}$$

$$s_4 = s_1 + 2R_{a1}(Z_1\sin\xi_1 + R_{c1}\cos\xi_1) - 2Z_2s_2 \tag{5.38}$$

$$s_5 = 4R_{a1}\cos\gamma_1[(Z_1 - Z_2)\cos\xi_1 - R_{c1}\sin\xi_1] \tag{5.39}$$

$$s_6 = s_1 - 2R_{a1}(Z_1\sin\xi_1 + R_{c1}\cos\xi_1) - 2Z_2s_3 \tag{5.40}$$

$$s_7 = 2R_{c2}s_2\cos\xi_2 - s_4\sin\xi_2 \tag{5.41}$$

$$s_8 = 2R_{c2}s_2\sin\xi_2 + s_4\cos\xi_2 \tag{5.42}$$

$$s_9 = 2R_{c2}s_3\cos\xi_2 - s_6\sin\xi_2 \tag{5.43}$$

$$s_{10} = 2R_{c2}s_3\sin\xi_2 + s_6\cos\xi_2 \tag{5.44}$$

$$s_{11} = 4R_{c2}R_{a1}\cos\gamma_1\cos\xi_1\cos\xi_2 - s_5\sin\xi_2 \tag{5.45}$$

$$s_{12} = 4R_{c2}R_{a1}\cos\gamma_1\cos\xi_1\sin\xi_2 + s_5\cos\xi_2 \tag{5.46}$$

Chapter 6

Samples

6.1 Programming Method

In this section, calculation and drawing of an intersection curve will be described.

6.1.1 Intersection computation

According to § 4 & § 5, given one α_2 or u_2, up to eight solutions can be obtained. So, the main problem in calculating intersection curves concerns how to choose the given parameter value and how to organize the solutions, i.e., so as to generate a sequence of points on the intersection curve.

- Choice of given parameter:

  First one should define the parametric range, its step size and a minimum number of intersection points to be calculated on each loop. Usually the given parameter to be independently varied is chosen as an angle variable, which has a range of $(0 \rightarrow 2\pi)$. The step must be small enough so that a small intersection loop will not be ignored. For example, if a parametric range of $0 \rightarrow 2\pi$ is chosen, the step size might be defined as $\pi/180$ or 1 degree. The minimum number of intersection points required depends on being able to trace a sufficiently smooth intersection curve with the chords joining them.

Then, intersection points are calculated by stepping the parametric value through the range decided upon with the step size chosen. If a complete cycle reveals no solutions, there is no intersection within the resolution of the chosen step size. If intersection points in a cycle produce adjacent chords which vary in direction by more than some prescribed value, intersection points at locally reduced step size may be calculated and inserted, either explicitly or by some suitable interpolation scheme.

- Organizing the solutions:
  Except for simple cases of multiple loops, like say, intersecting cylinders, which can be handled easily, it is more efficient to identify intersection curve branching online, interactively, i.e., with human operator decision and intervention.

Accordingly, a software package using Matlab was developed (see Appendix A).

6.1.2 Curve construction

Intersection curve data can be obtained with the Matlab software package mentioned above. Because AutoCAD supports script files (command list

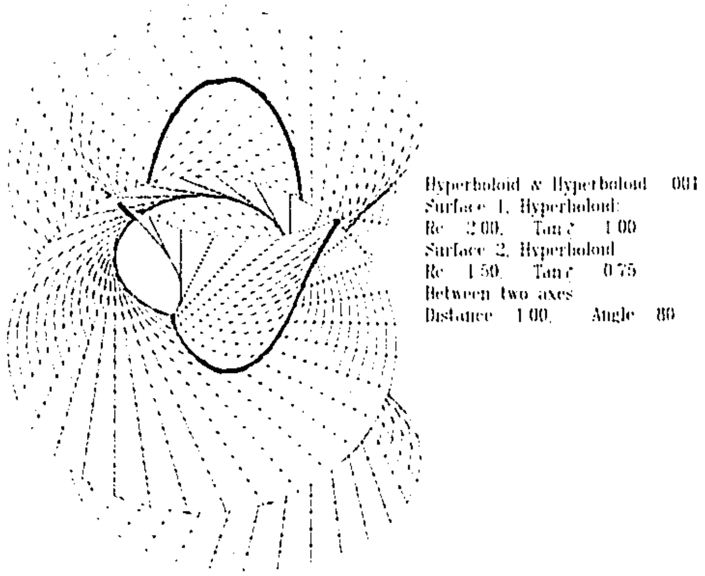


Figure 6.1: The Intersection between Hyperboloids

files), another software package was programmed in BASIC to deal with converting the data into an script file, which is used directly by AutoCAD. For example, this was done to produce the various pictorial illustrations like Fig. 6.1. This is documented in Appendix B.

6.2 Examples

Fig. 6.1 through 6.15 show fifteen different example cases. Fig. 6.6 & 6.12 describe curve overlap while Fig. 6.13 is an illustration of the special case of coincident axes.

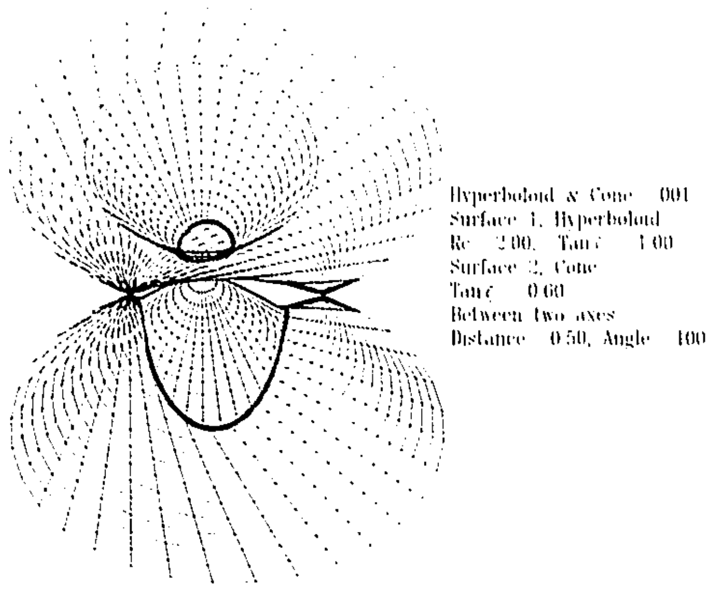


Figure 6.2: The Intersection between Hyperboloid and Cone

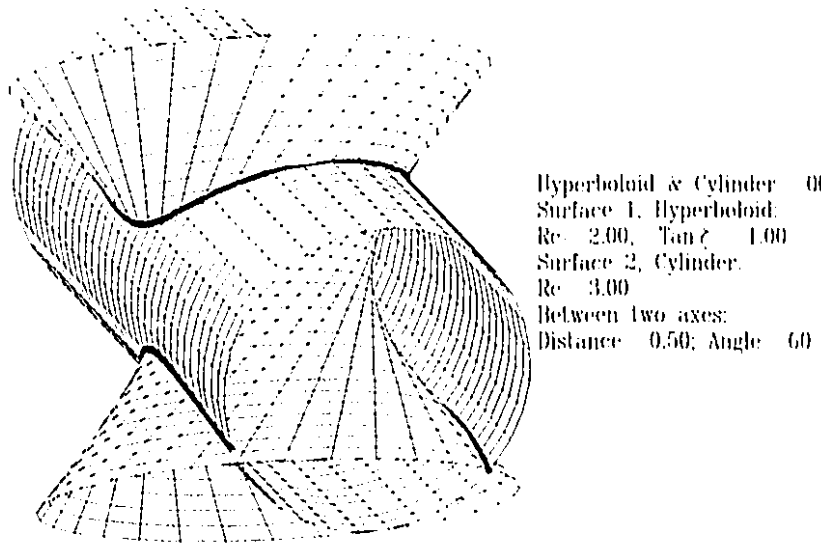
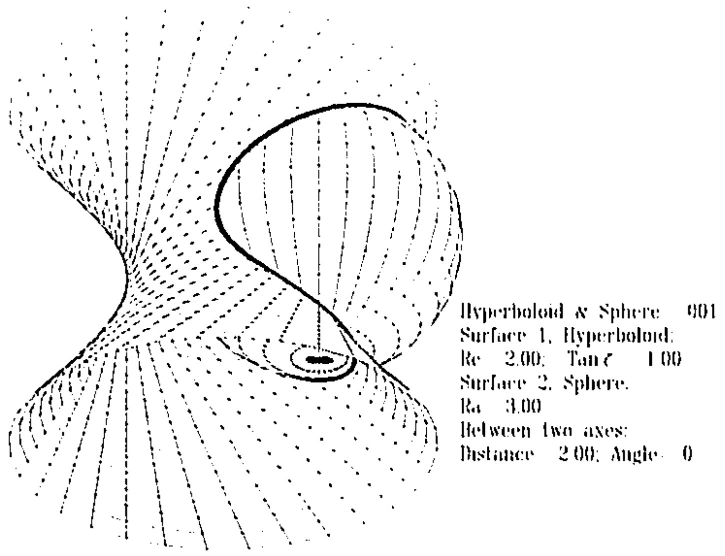
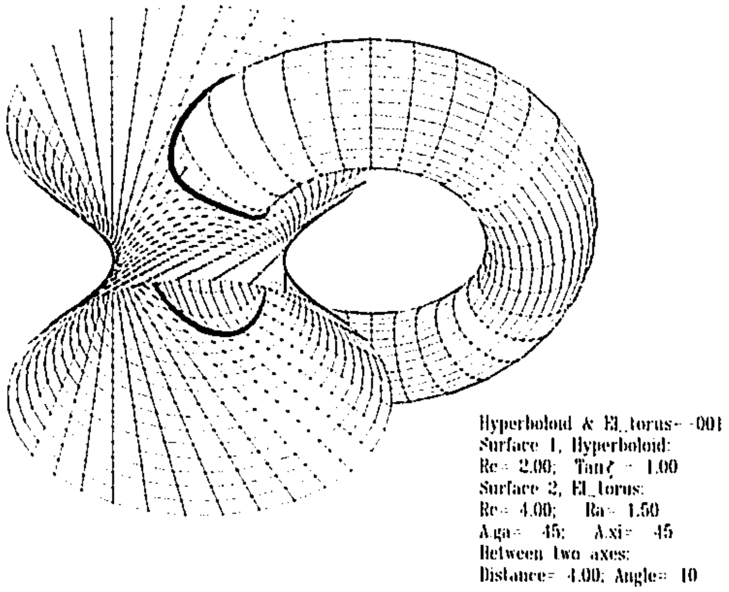


Figure 6.3: The Intersection between Hyperboloid and Cylinder



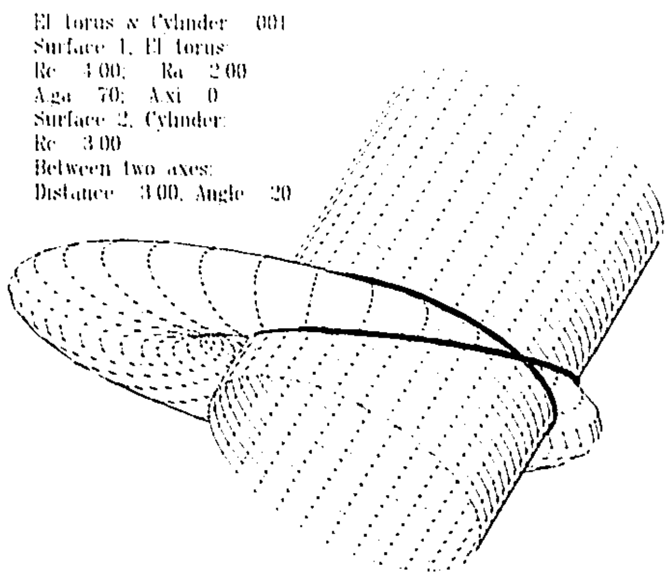
```
Hyperboloid & Sphere   001
Surface 1, Hyperboloid:
Re   2.00;   Tan ζ    1.00
Surface 2, Sphere.
Ra   3.00
Between two axes:
Distance   2.00; Angle   0
```

Figure 6.4: The Intersection between Hyperboloid and Sphere



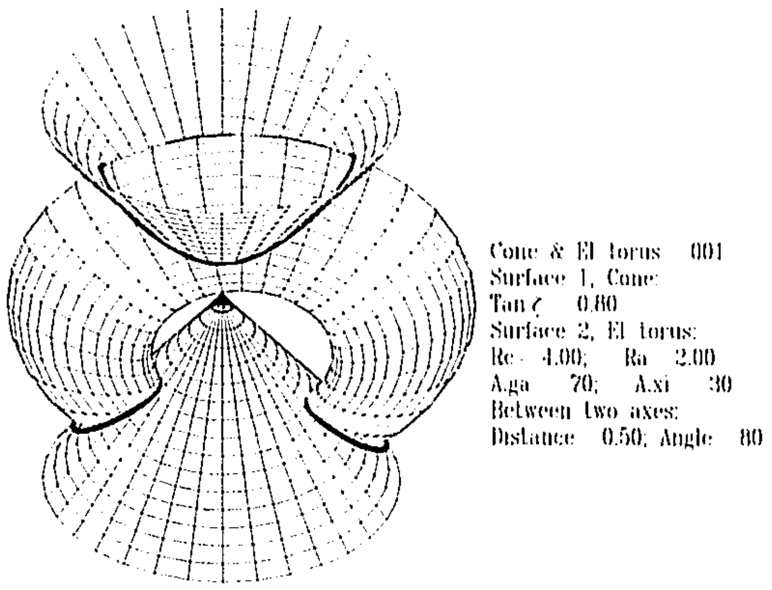
```
Hyperboloid & El_torus- -001
Surface 1, Hyperboloid:
Re= 2.00;   Tanζ = 1.00
Surface 2, El_torus:
Re= 4.00;    Ra= 1.50
Aga=  45;    Axi=  45
Between two axes:
Distance= 4.00; Angle= 10
```

Figure 6.5: The Intersection between Hyperboloid and Cyclid



El torus & Cylinder 001
Surface 1, El torus:
Rc 4.00; Ra 2.00
Aga 70; Axi 0
Surface 2, Cylinder:
Rc 3.00
Between two axes:
Distance 3.00, Angle 20

Figure 6.6: The Intersection between Cyclid and Cylinder



Cone & El torus 001
Surface 1, Cone:
Tan ζ 0.80
Surface 2, El torus:
Rc 4.00; Ra 2.00
Aga 70; Axi 30
Between two axes:
Distance 0.50; Angle 80

Figure 6.7: The Intersection between Cyclid and Cone

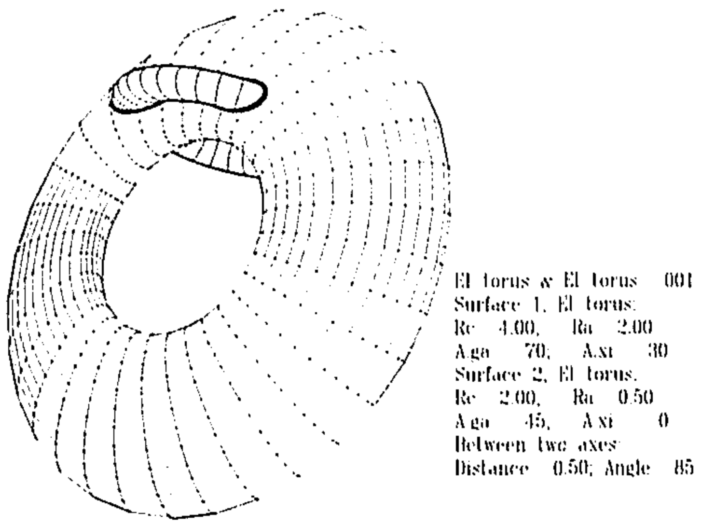


Figure 6.8: The Intersection between Cyclids

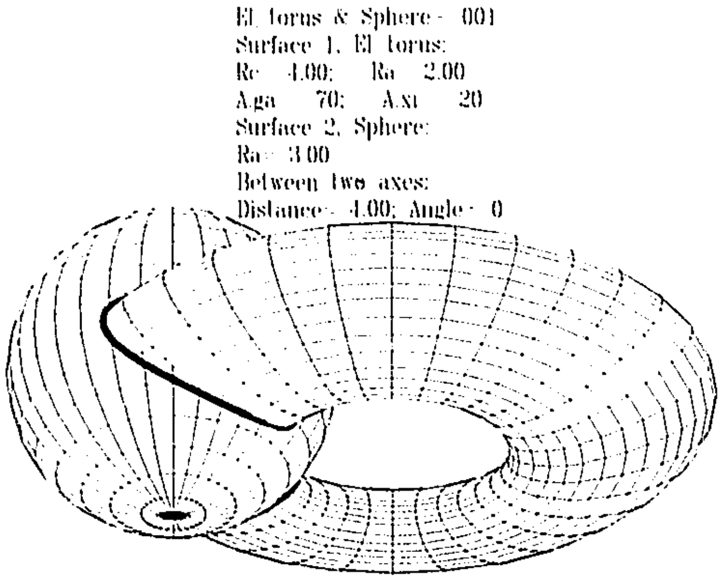


Figure 6.9: The Intersection between Cyclid and Sphere

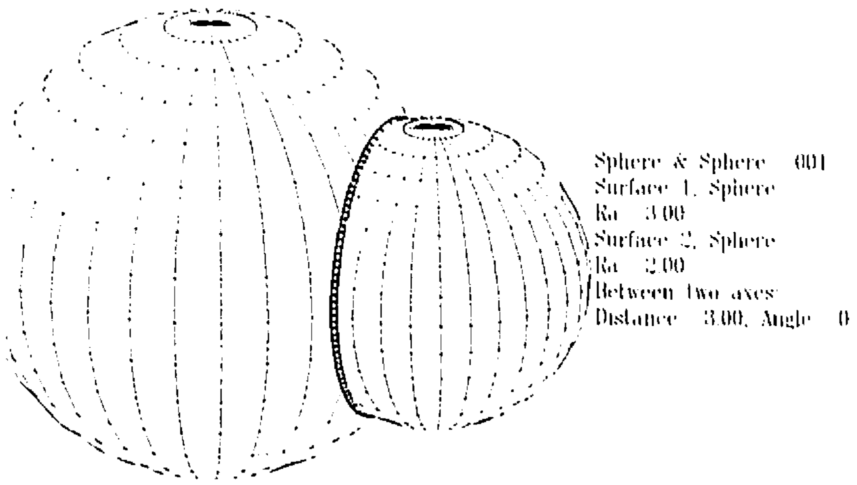


Figure 6.10: The Intersection between Spheres

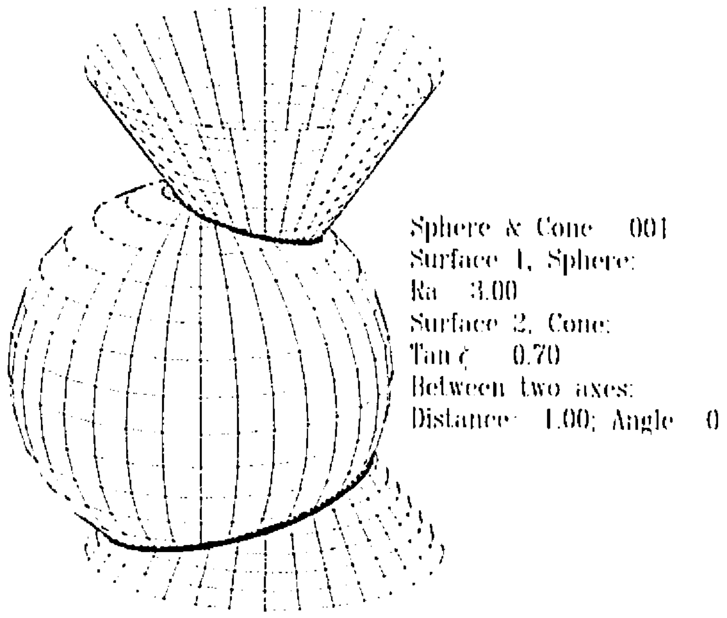


Figure 6.11: The Intersection between Sphere and Cone

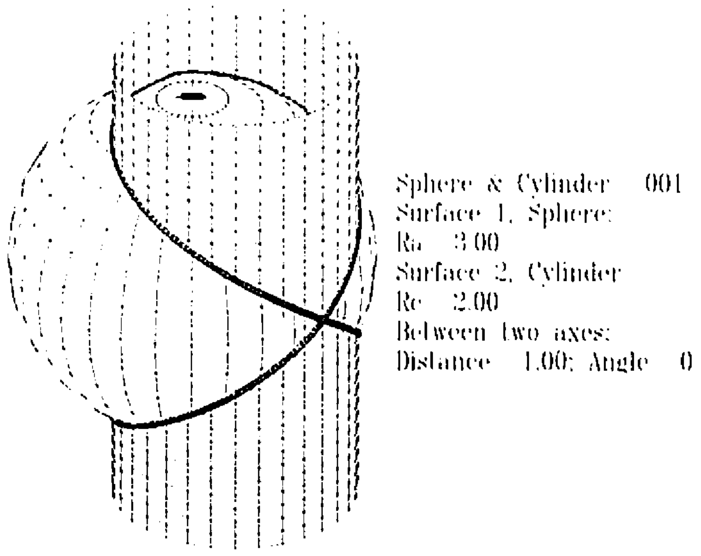

Sphere & Cylinder 001
Surface 1, Sphere:
Ra = 3.00
Surface 2, Cylinder
Rc = 2.00
Between two axes:
Distance = 1.00; Angle = 0

Figure 6.12: The Intersection between Sphere and Cylinder

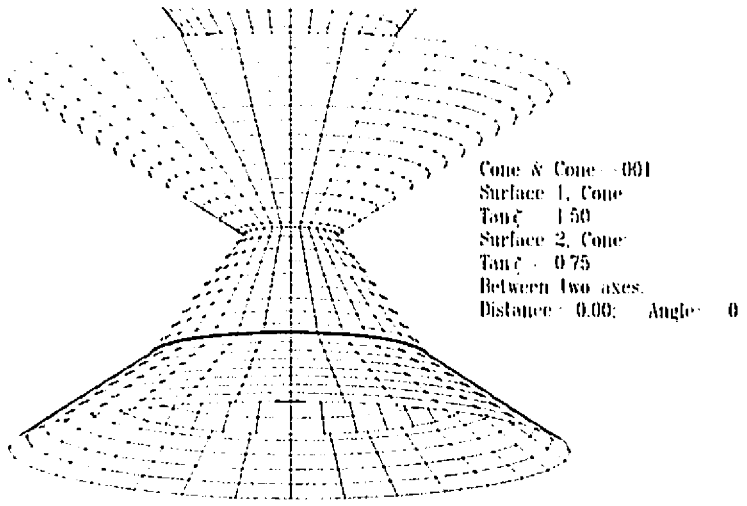

Cone & Cone - 001
Surface 1, Cone
Tanζ = 1.50
Surface 2, Cone
Tanζ = 0.75
Between two axes.
Distance = 0.00; Angle = 0

Figure 6.13: The Intersection between Cones

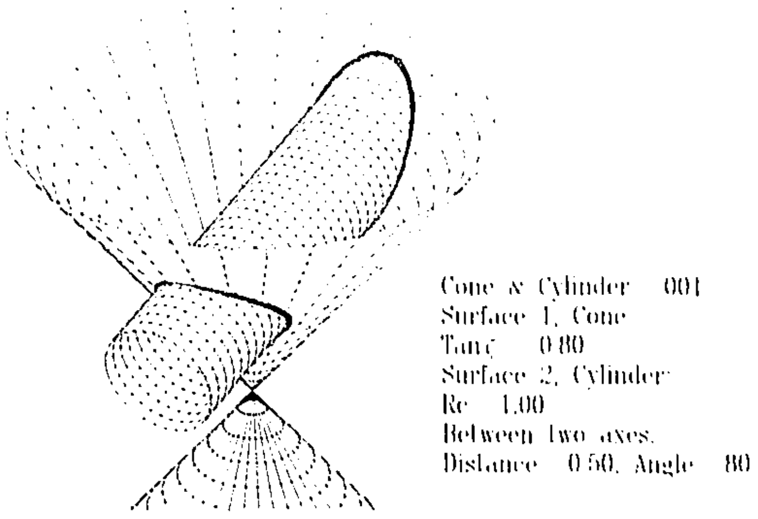

Cone & Cylinder 001
Surface 1, Cone
Tanζ 0.80
Surface 2, Cylinder:
Rc 1.00
Between two axes:
Distance 0.50, Angle 80

Figure 6.14: The Intersection between Cone and Cylinder

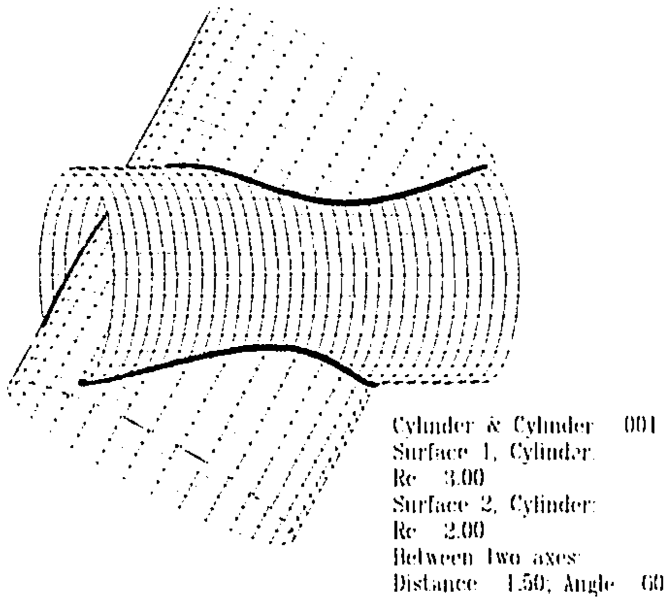

Cylinder & Cylinder 001
Surface 1, Cylinder:
Rc 3.00
Surface 2, Cylinder:
Rc 2.00
Between two axes:
Distance 1.50; Angle 60

Figure 6.15: The Intersection between Cylinders

Chapter 7

Concluding Remarks

7.1 Conclusion

With this novel algebraic method we gain advantages, including important reduction of the polynomial degree. These include:

- The degree of the intersecting points equation is only half of that of the algebraic curve equation.
- Singularity and local tangency between surfaces are easy to handle.
- The resulting closed form solutions are more accurate
- Articulated rigid body mechanism models which generate surfaces are introduced. This makes it easy to build surfaces, and easy to classify groups of surfaces.
- Self intersection does not hinder or complicate solutions.

It should not be overlooked, however, that the method has a few disadvantages:

- It can be applied only to pairs of surfaces of revolution.
- For every pairing of surface species, different algorithms have to be formulated, and
- The code is lengthy and may be difficult to implement in microcontrolers used for online trajectory generation.

According to this novel algebraic method, a set of solutions can be obtained for surface species described in § 4 and § 5. These describe trajectories of motion of some point on a rigid link of some single, closed loop mechanism. There mechanisms are listed in Table 1.

Primary surface	Secondary surface	Type of linkage
Hyperboloid	Hyperboloid	RPSPR
Hyperboloid	Elliptic tori	RPSRR
Hyperboloid	Sphere	RPSU
Hyperboloid	Cone	RPSPR
Hyperboloid	Cylinder	RPSPR
Elliptic tori	Elliptic tori	RRSRR
Elliptic tori	Sphere	RRSU
Elliptic tori	Cone	RRSPR
Elliptic tori	Cylinder	RRSPR
Sphere	Sphere	USU
Sphere	Cone	USPR
Sphere	Cylinder	USPR
Cone	Cone	RPSPR
Cone	Cylinder	RPSPR
Cylinder	Cylinder	RPSPR

Table 1 : Mechanism types

7.2 Contributions

- A new algebraic method has been developed which can be applied to various engineering problem, e.g., the fabrication of mechanical componerts which are composites of surfaces of revolution.

- A software package (see Appendix A), which can be used to solve intersections like the 15 cases listed in Table 1 was created.

- A useful enhancement to Advanced Modeling Extension (AME) in AutoCAD (see Appendix B) has been designed. It can handle intersection curves between more complicated surfaces than those within the capacity of commercial packages available.

- A systematic approach to study the kinematic geometry of the motions produced by 15 kinds of single, closed loop mechanisms was developed.

References

[1] Hedrick, R. W. and Bedi, S., "Method for intersection of parametric surfaces," *Trans. of the Canadian Society for Mech. Engineers*, Vol. 14, No. 3, 1990, pp. 79–84

[2] Sederberg, T. W. and Nishita, T., "Curve intersection using Bezier clipping," *Computer Aided Design*, Vol. 22, No. 9, Nov., 1990, pp. 538–549

[3] Piegl, L., "Geometric method of intersecting natural quadrics represented in trimmed surface form," *Computer Aided Design*, Vol. 21, No. 4, May, 1989, pp. 201–212

[4] Sederberg, T. W., "Algorithm for algebraic curve intersection," *Computer Aided Design*, Vol. 21, No. 9, Nov., 1989, pp. 547–554

[5] Hoffmann, C. M., *Geometric and Solid Modelling: An Introduction,* Morgan Kaufmann, San Mateo, Calif., 1989

[6] Miller, J. R., "Geometric approaches to nonplanar quadric surface intersection curves," *ACM Trans. Graph.* Vol. 6, No. 4, 1987, pp.274–307

[7] Pfeifer, H.-U., "Methods used for intersecting geometric entities in the GPM module for volume geometry," *Computer Aided Design*. Vol. 17, No. 7, 1985, pp. 311–318

[8] Kriezis, G. A., Patrikalakis, N. M. and Wolter, F.-E., "Topological and differential equation methods for surface intersections," *Computer Aided Design*, Vol. 24, No. 1, Jan., 1992, pp. 41-55

[9] Markot, R. P. and Magedson, R. L., "Procedural method for evaluating the intersection curves of two parametric surfaces," *Computer Aided Design*, Vol. 23, No. 6, Jul./Aug., 1991, pp. 395–404

[10] Barnhill, R. E. and Kersey, S. N., "Marching method for parametric surface/surface intersection," *Computer Aided Geometric Design*, Vol. 7, Nos. 1-4, Jun., 1990, pp. 257–280

[11] Bajaj, C. L., "Tracing surface intersections," *Computer Aided Geometric Design*, Vol. 5, No. 4, Nov., 1988, pp. 285–307

[12] Patrikalakis, N. M., "Surface-surface intersections," *IEEE Computer Graphics and Applications*, Vol. 13, No. 1, Jan., 1993, pp. 89–95

[13] Kriezis, G. A., Prakash, P. V. and Patrikalakis, N. M., "A method for intersecting algebraic surfaces with rational polynomial patches," *Computer Aided Design*, Vol. 22, No. 10, Dec., 1990, pp. 645-654

[14] Barnhill, R. E., Farin, G. E., Jordan, M. and Piper, B. R., "Surface/surface intersection," *Computer Aided Geometric Design*, Vol. 4, Nos. 1-2, 1987, pp. 3–16

[15] Lasser, D., "Intersection of parametric surfaces in the Bernstein-Bezier representation," *Computer Aided Design*, Vol. 18 No. 4, 1986, pp. 186 192

[16] Dokken, T., "Finding intersections of B-spline represented geometries using recursive subdivision techniques," *Computer Aided Geometric Design*, Vol. 2, Nos. 1-3, 1985, pp. 189–195

[17] Houghton, E. G., Emmett, R. F., Factor, J. D. and Sabharwal, C. L., "Implementation of divide-and-conquer method for intersection of parametric surfaces," *Computer Aided Geometric Design*, Vol. 2, Nos. 1-3, 1985, pp. 173–183

[18] Rogers, D. F. and Adams, J. A., *Mathematical Elements for Computer Graphics*, McGraw-Hill, New York, 1990

Appendix A

Matlab Program

```
% This function is used to calculate the begining values function [t1,t2,t3,t4]=begin(nr)
ii=[2 2];R=[1 4;0 2];Z=[0 0];Y=4;xii1=0;xii2=0;ga1=30;ga2=0;
ph=0;l1=10;l2=10;ans=[ii';Y;ph;Z';R(1,:)';R(2,:)'];
if ii(1)==1, ans=[ans;l1;R(1,2)];   else  ans=[ans;ga1;xii1];
end
xii1=xii1*pi/180;ga1=ga1*pi/180;ph=ph*pi/180;
if ii(2)==2,
delb=1*pi/180;db=2*pi;bv=0;ans=[ans;ga2;xii2];
else
ll=l2;delb=ll/360;db=ll;bv=-ll/2;ans=[ans;ll;R(2,2)];
end
xii2=xii2*pi/180;ga2=ga2*pi/180;xi1=[sin(xii1) cos(xii1)];
xi2=[sin(xii2) cos(xii2)];gamma1=[sin(ga1) cos(ga1)];
gamma2=[sin(ga2) cos(ga2)];phi=[sin(ph) cos(ph)];
if phi(1)¡0.0000001 & Y==0,
[r,z,jj ]=xyzc(ii,R,Z,xi1,xi2,gamma1,gamma2)
for i=1:jj,
ans=[ans;[0 0 z(i)]'];ans=[ans;[r(i) 0 z(i)]'];ans=[ans;[0 r(i) z(i)]'];
end
else
[t1,t2,t3,t4]= x(ii,R,Y,Z,phi,xi1,xi2,gamma1,gamma2,db,nr,bv,delb);
[ia,ib ]=size(t1);
if ib==0,
else
for i=1:ib,ans=[ans;t1(:,i)];  end
```

```
end
[ia,ib]=size(t2);
if ib==0,
else
for i=1:ib,ans=[ans;t2(:,i)]; end
end
[ia,ib]=size(t3);
if ib==0,
else
for i=1:ib,ans=[ans;t3(:,i)]; end
end
[ia,ib]=size(t4);
if ib==0,
else
for i=1:ib,ans=[ans;t4(:,i)]; end
end
end
chdir c:int
save to1.dat ans ascii

    % This function is used to solve intersecting points (x,y,z)
% in the general cases.

    function [r1,r2,r3,r4]=
x(ii,R,Y,Z,phi,xi1,xi2,gamma1,gamma2,dv,nr,bv,delb)
ni=[];jj1=0;nj=[];mi=0;mn=nr;no=[0 0];
while mi¡nr
r1=[];r2=[];r3=[];r4=[];del=dv/mn;ik=0;kk=0;n=0;mm=[0 0 0 0];d=bv;
[jj,t]=xy(ii,R,Y,Z,phi,xi1,xi2,gamma1,gamma2,d);
if jj==0,
n=1;no(1)=1;
else
ik=ik+1;mm(ik)=mm(ik)+1;n=0;
[r1,r2,r3,r4,n1]=ro4(t,mm,ik,jj,kk,r1,r2,r3,r4,no,jj1,n1);
end
d=d+del;
for i=1:mn,
kk=jj;[jj,t]=xy(ii,R,Y,Z,phi,xi1,xi2,gamma1,gamma2,d);
```

```
if jj==0,
n=1;
if ik¿0,
no(2)=1;
else
end
else
if n==1,
ik=ik+1;n=0;
else
end
mm(ik)=mm(ik)+1;no(2)=0;
[r1,r2,r3,r4,n1]=ro4(t,mm,ik,jj,kk,r1,r2,r3,r4,no,jj1,n1);
jj1=jj;
end
d=d+del;
end
mi=min(mm(1:ik));ma=max(mm(1:ik));mn=mn*2;
if ma==0 & del¡delb,break,end
end
if no(2)==1,
[ni(1),nj(1)]=size(r1);[ni(2),nj(2)]=size(r2);
[ni(3),nj(3)]=size(r3);[ni(4),nj(4)]=size(r4);
i1=max(nj);i2=i1+1;[r1,r2,r3,r4]=ro5(i1,i2,r1,r2,r3,r4,n1);
else
end

    % This function is used to solve the intersecting point (x,y,z)
% in the general cases.
function [kk,t]=xy(ii,R,Y,Z,phi,xi1,xi2,gamma1,gamma2,v)
t2=xyza(ii(2),2,R,xi2,gamma2,v);
if ii(1)==1,
[jj,tt]=xyz2(t2,R,Z,Y,phi,xi1,gamma1);
else
[jj,tt]=xyz1(t2,R,Z,Y,phi,xi1,gamma1);
end
if jj==0,
else
```

```
for i=1:jj,
t4=xyza(ii(1),1,R,xi1,gamma1,tt(i));
c1(i)=(t4(1)+R(1,1))^2+(t4(2))^2+(t4(3)+Z(1))^2-(t2(1)+R(2,1))^2
-(t2(2))^2-(t2(3)+Z(2))^2-Y^2;
c2(i)=t4(3)+Z(1)-(t2(3)+Z(2))*phi(2);
end
if Y==0 | phi(1)==0,
if jj==1,
if Y==0,
c=[-c2-phi(1)*(t2(1)+R(2,1));-2*phi(1)*t2(2);-c2+phi(1)*(t2(1)+R(2,1))];
else
c=[-c1-2*Y*t2(2);4*Y*(t2(1)+R(2,1));-c1+2*Y*t2(2)];
end
qa=[c]';qa1=[qa(3) qa(2) qa(1)];bet0=[[roots(qa)]'];bet1=[[roots(qa1)]'];
else
if Y==0,
c=[-c2-phi(1)*(t2(1)+R(2,1));-2*phi(1)*t2(2) -2*phi(1)*t2(2);
-c2+phi(1)*(t2(1)+R(2,1))];
else
c=[-c1-2*Y*t2(2);4*Y*(t2(1)+R(2,1)) 4*Y*(t2(1)+R(2,1));-c1+2*Y*t2(2)];
end
qa=[c(:,1)]';qb=[c(:,2)]';qa1=[qa(3) qa(2) qa(1)];qb1=[qb(3) qb(2) qb(1)];
bet0=[[roots(qa)]' [roots(qb)]'];bet1=[[roots(qa1)]' [roots(qb1)]'];
end
[ij,be]=ro2(bet0,bet1);
else
qa=c1*(t2(1)+R(2,1))*phi(1)-2*Y*t2(2)*c2;
qb=c1*t2(2)*phi(1)+2*Y*(t2(1)+R(2,1))*c2;
bet0=qa./qb;bet1=qb./qa;[ij,be]=ro3(bet0,bet1,qa,qb);
end
end
xd=(t2(1)+R(2,1))*cos(be)-t2(2)*sin(be);
yd=(t2(1)+R(2,1))*sin(be)+t2(2)*cos(be);
zd=t2(3)+Z(2);x=xd*phi(2)-zd*phi(1);y=yd+Y;
z=xd*phi(1)+zd*phi(2);t=[x;y;z];[ll,kk]=size(t);
end
```

% This function is used to solve the intersecting points of x and y

```
% with z when angle of phi and Y both are near to zero or they both
% are zero.

   function [jj,ij,t]=xyz(ii,R,Y,Z,phi,xi1,xi2,gamma1,gamma2,beta1)
if ii(1)==1,
a(1,1)=1;a(1,2)=0;a(1,3)=0;a(1,4)=-(R(1,2))^2;a(1,5)=2*(R(1,2))^2*Z(1);
a(1,6)=(beta(1))^2-(R(1,1))^2-(R(1,2))^2*(Z(1))^2;
a(2,1)=1;a(2,2)=-2*Y;a(2,3)=0;
a(2,4)=(phi(1))^2-(R(2,2))^2*(phi(2))^2;
a(2,5)=phi(2)*(2*beta(1)*phi(1)+2*(R(2,2))^2*(beta(1)*phi(1)+Z(2)));
a(2,6)=(beta(1))^2*(phi(2))^2-(R(2,1))^2
+(R(2,2))^2*(beta(1)*phi(1)+Z(2))^2+Y^2;
else
a(1,1)=(xi1(2))^2*(gamma1(1))^2+(xi1(1)*gamma1(1)*beta1(2)
-gamma1(2)*beta1(1))^2;
a(1,2)=-2*R(1,1)*((xi1(2))^2*(gamma1(1))^2*beta1(2)
+xi1(1)*gamma1(1)*(xi1(1)*gamma1(1)*beta1(2)-gamma1(2)*beta1(1)));
a(1,3)=2*(xi1(1)*gamma1(1)*beta1(2)
-gamma1(2)*beta1(1))*(xi1(1)*gamma1(1)*beta1(1)
+gamma1(2)*beta1(2));
a(1,4)=(xi1(2))^2*(gamma1(1))^2+(xi1(1)*gamma1(1)*beta1(1)
+gamma1(2)*beta1(2))^2;
a(1,5)=-2*R(1,1)*((xi1(2))^2*(gamma1(1))^2*beta1(1)
+xi1(1)*gamma1(1)*(xi1(1)*gamma1(1)*beta1(1)+gamma1(2)*beta1(2)));
a(1,6)=(R(1,1))^2*(xi1(1))^2*(gamma1(1))^2
+(xi1(2))^2*(gamma1(1))^2*((R(1,1))^2-(R(1,2))^2);
if ii(2)==1,
s1=Z(1)*xi1(2)*gamma1(1)-R(1,1)*xi1(1)*gamma1(1);
s2=(phi(1))^2-(R(2,2))^2*(phi(2))^2;
s3=beta1(1)*gamma1(1)*xi1(1)+beta1(2)*gamma1(2);
s4=beta1(2)*gamma1(1)*xi1(1)-beta1(1)*gamma1(2);
a(2,1)=(xi1(2))^2*(gamma1(1))^2*((phi(2))^2
-(R(2,2))^2*(phi(1))^2)
+2*(1+(R(2,2))^2)*phi(1)*phi(2)*xi1(2)*gamma1(1)
+s2^2*s4^(2);
a(2,2)=2*((R(2,2))^2*Z(2)*s4*phi(2)
+s1*xi1(2)*gamma1(1)*(1+(R(2,2))^2))*xi1(2)*gamma1(1)
-2*(R(2,2))^2*Z(2)*phi(1)*(xi1(2))^2*(gamma1(1))^2+2*s1*s2*s4;
```

```
a(2,3)=2*(s2*s4+(1+(R(2,2))^2)*phi(1)*phi(2)*xi1(2)*gamma1(1))*s3;
a(2,4)=(xi1(2))^2*(gamma1(1))^2+s2*s3^2;
a(2,5)=2*(s1*s2*s3-Y*(xi1(2))^2*(gamma1(1))^2
+(R(2,2))^2*Z(2)*s3*phi(2)*xi1(2)*gamma1(1));
a(2,6)=(xi1(2))^2*(gamma1(1))^2*(Y^2-(R(2,1))^2
-(Z(2))^2*(R(2,2))^2)
+2*(R(2,2))^2*Z(2)*s1*phi(2)*xi1(2)*gamma1(1)+s2*s1^2;
else
s1=Z(1)*xi1(2)*gamma1(1)-R(1,1)*xi1(1)*gamma1(1);
s2=((R(1,2))^2-(R(2,2))^2-(R(1,1))^2
-(R(2,1))^2+(Z(2))^2-(Z(1))^2)*xi1(2)*gamma1(1)
+2*s1*(Z(1)-Z(2)*phi(2));
s3=beta1(1)*gamma1(1)*xi1(1)+beta1(2)*gamma1(2);
s4=2*(R(1,1)*beta1(1)-Y)*xi1(2)*gamma1(1)+2*s3*(Z(1)-Z(2)*phi(2));
s5=beta1(2)*gamma1(1)*xi1(1)-beta1(1)*gamma1(2);
s6=2*(R(1,1)*beta1(2)-Z(2)*phi(1))*xi1(2)*gamma1(1)
+2*s5*(Z(1)-Z(2)*phi(2));
s7=s5*phi(2)-phi(1)*xi1(2)*gamma1(1);
s8=s1*phi(2)-Z(2)*xi1(2)*gamma1(1);
a(2,1)=(s6^2+4*(R(2,1))^2*s7^2)*(gamma2(2))^2
+(2*R(2,1)*s7*xi2(2)-s6*xi2(1))^2*(gamma2(1))^2;
a(2,2)=2*(s2*s6+4*(R(2,1))^2*s8*s7)*(gamma2(2))^2
+2*(2*R(2,1)*s8*xi2(2)-s2*xi2(1))*(2*R(2,1)*s7*xi2(2)
-s6*xi2(1))*(gamma2(1))^2;
a(2,3)=2*(2*R(2,1)*s7*xi2(2)-s6*xi2(1))*(2*R(2,1)*s3*phi(2)*xi2(2)
-s4*xi2(1))*(gamma2(1))^2
+2*(s4*s6+4*s3*(R(2,1))^2*s7*phi(2))*(gamma2(2))^2;
a(2,4)=(s4^2+4*(R(2,1))^2*s3^2*(phi(2))^2)*(gamma2(2))^2
+(2*R(2,1)*s3*phi(2)*xi2(2)-s4*xi2(1))^2*(gamma2(1))^2;
a(2,5)=2*(s2*s4+4*(R(2,1))^2*s8*s3*phi(2))*(gamma2(2))^2
+2*(2*R(2,1)*s8*xi2(2)-s2*xi2(1))*(2*R(2,1)*s3*phi(2)*xi2(2)
-s4*xi2(1))*(gamma2(1))^2;
a(2,6)=(s2^2+4*(R(2,1))^2*s8^2
-4*(R(2,2))^2*(R(2,1))^2*(xi1(2))^2*(gamma1(1))^2)*(gamma2(2))^2
+(2*R(2,1)*s8*xi2(2)-s2*xi2(1))^2*(gamma2(1))^2;
end
end
a[x,y,jj]=ro1(a)
```

```
if jj¿0,
xc1=y*beta1(1)+x*beta1(2)-R(1,1);yc1=y*beta1(2)-x*beta1(1);
z1=sqrt((R(1,1))^2-xc1.^2-yc1.^2);
z2=-sqrt((R(1,1))^2-xc1.^2-yc1.^2);
f1=yc1*gamma1(2)-((z1-Z(1))*xi1(2)-xc1*xi1(1))*gamma1(1);
f2=yc1*gamma1(2)-((z2-Z(1))*xi1(2)-xc1*xi1(1))*gamma1(1);
if abs(f1)¡0.00001 & abs(f2)¡0.00001,
t=[x;y;z1;z2];ij=2;
else
ij=1;
if abs(f1)¡0.00001,
t=[x;y;z1];
else
t=[x;y;z2];
end
end
else
ij=0;
end

    % This function is used to solve the single parameters(a1) of the
% intersecting points of two rotational quadratic surfaces with
% primary surface being a cyclid or a sphere.

    function [jj,t]=xyz1(t2,R,Z,Y,phi,xi,gamma)
d1=(R(1,1))^2+(Z(1))^2-(t2(1)+R(2,1))^2
-(t2(2))^2-(t2(3)+Z(2))^2-Y^2;
d2=Z(1)-(t2(3)+Z(2))*phi(2);
d3=-(t2(1)+R(2,1))^2-(t2(2))^2;
n1=(R(1,2))^2+d1-2*R(1,2)*(Z(1)*xi(1)+R(1,1)*xi(2));
n2=4*R(1,2)*gamma(2)*(Z(1)*xi(2)-R(1,1)*xi(1));
n3=(R(1,2))^2+d1+2*R(1,2)*(Z(1)*xi(1)+R(1,1)*xi(2));
n4=d2-R(1,2)*xi(1);n5=2*R(1,2)*gamma(2)*xi(2);n6=d2+R(1,2)*xi(1);
c(1)=n1^2*(phi(1))^2+4*Y^2*n4^2
+4*d3*Y^2*(phi(1))^2;
c(2)=2*n2*n1*(phi(1))^2+8*Y^2*n5*n4;
c(3)=4*Y^2*(2*n6*n4+n5^2)+8*d3*Y^2*(phi(1))^2
+(phi(1))^2*(2*n3*n1+n2^2);
```

```
c(4)=2*n3*n2*(phi(1))^2+8*Y^2*n5*n6;
c(5)=n3^2*(phi(1))^2+4*Y^2*n6^2
+4*d3*Y^2*(phi(1))^2;
c1(1)=c(5);c1(2)=c(4);c1(3)=c(3);c1(4)=c(2);c1(5)=c(1);t0=roots(c);
t1=roots(c1);[jj,t]=ro2(t0,t1);

    % This function is used to solve the z(c1) of intersecting points of
% rotational quadratic surfaces with primary surface being a H.,
% a cone or a cylinder.
function [jj,t]=xyz2(t2,R,Z,Y,phi,xi,gamma)
d1=(R(1,1))^2+(Z(1))^2-(t2(1)+R(2,1))^2
-(t2(2))^2-(t2(3)+Z(2))^2-Y^2;
d2=Z(1)-(t2(3)+Z(2))*phi(2);
d3=-(t2(1)+R(2,1))^2-(t2(2))^2;
n1=1+(R(1,2))^2;n2=2*Z(1);
c(1)=n1^2*(phi(1))^2;
c(2)=2*n2*n1*(phi(1))^2;
c(3)=4*Y^2+(phi(1))^2*(2*d1*n1+n2^2);
c(4)=2*d1*n2*(phi(1))^2+8*Y^2*d2;
c(5)=d1^2*(phi(1))^2+4*Y^2*d2^2 +4*d3*Y^2*(phi(1))^2;
t1=roots(c);nn=length(t1);kj=0;
for i=1:nn,
if abs(imag(t1(i)))¡0.000001,
t1(i)=real(t1(i));
else
end
if imag(t1(i))==0,
kj=kj+1;tt(kj)=t1(i);
else
end
end
nn=length(tt);jj=0;
for i=1:nn,
if i==1,
jj=jj+1;t(jj)=tt(i);
else
ks=0;
for k=i:-1:2,
if abs(tt(i)-tt(i-1))¡0.000001,
ks=1;
else
end
end
if ks==1,
else
jj=jj+1;t(jj)=tt(i);
end
end
end

    % This function is used to calculate the values of [x(c2),y(c2),z(c2)]
% of another surface of a cyclid, sphere, H.,
% cone or cylinder.
function t2=xyza(ii,ik,R,xi,gamma,v)
t2=[];
if ii==2,
alpha(1)=sin(v);alpha(2)=cos(v);
t2(1)=R(ik,2)*(alpha(2)*xi(2)-alpha(1)*gamma(2)*xi(1));
t2(2)=R(ik,2)*alpha(1)*gamma(1);
t2(3)=R(ik,2)*(alpha(2)*xi(1)+alpha(1)*gamma(2)*xi(2));
else
t2(1)=0;t2(3)=v;t2(2)=v*R(ik,2);
end

    % This function is used to solve the intersecting point (x,y,z)
% in the general case.
Function [jj,t]=xyzb(ii,R,Y,Z,phi,xi1,xi2,gamma1,gamma2,alpha2)
t2=xyza(ii(2),R,alpha2,xi2,gamma2);
if ii(1)==1,
tt=xyz2(t2,R,Z,Y,phi,xi1,gamma1);
else
tt=xyz1(t2,R,Z,Y,phi,xi1,gamma1);
end
jj=0;
for i=1:4,
if imag(tt(1))==0,
jj=jj+1;t3(jj)=tt(i);
else
end
end
if jj==0,
else
for i=1:jj,
t5(1)=sin(t3(i);t5(2)=cos(t3(i);t4(:,i)=xyza(ii(1),R,t5,xi1,gamma1);
e1(i)=(t4(1,i)+R(1,1))^2+(t4(2,i))^2+(t4(3,i)+Z(1))^2
-(t2(1)+R(2,1))^2-(t2(2))^2-(t2(3)+Z(2))^2-Y^2;
e2(i)=t4(3,i)+Z(1)-(t2(3)+Z(2))*phi(2);
end
bet=(e1*(t2(1)+R(2,1)*phi(1)-2*Y*t2(2)*e2)./
(e1*t2(2)*phi(1)+2*Y*(t2(1)+R(2,1))*e2);
bet=atan(bet);beta2(:,1)=sin(bet);beta2(:,2)=cos(bet);
xd=(t2(1)+R(2,1))*cos(det)-t2(2)*sin(bet);
yd=(t2(1)+R(2,1))*sin(det)+t2(2)*cos(bet);
zd=t2(3)+Z(2);x=xd2*phi(2)-zd2*phi(1);y=yd2+Y;
z=xd2*phi(1)+zd2*phi(2);t=[x;y;z];
end

   % This function is used to solve intersecting circles when
% phi=0 and Y=0
function [r,z,jj]=xyzc(ii,R,Z,xi1,xi2,gamma1,gamma2)
v1=[];rx=[];rd=[];r=[];z=[];x=[];y=[];be=[];
if ii(1)==1 & ii(2)==1,
c=[(R(1,2))^2-(R(2,2))^2 2*(R(2,2))^2*(Z(2)-Z(1))
(R(1,1))^2-(R(2,1))^2-(R(2,2))^2*(Z(1)-Z(2))^2];
v1=roots(c);jj=length(v1);rc=R(1,1);
if jj¿1,
for i=2:jj,
rc=[rc;R(1,1)];
end
end
rd=[rc R(1,2)*v1 v1+Z(1)];
else
if ii(2)==1,
s1=Z(1)-Z(2)-R(1,2)*xi1(1);s2=Z(1)-Z(2)+R(1,2)*xi1(1);
```

```
s3=Z(1)-R(1,2)*xi1(1);s4=R(1,1)*xi1(2)+Z(1)*xi1(1);
s5=R(1,1)*xi1(1)-Z(1)*xi1(2);
s6=(R(1,2))^2+(R(1,1))^2+(Z(1))^2-(R(2,1))^2;
c=[s6-2*R(1,2)*s4-(R(2,2))^2*s1^2-s3^2
-4*R(1,2)*gamma1(2)*(xi1(2)*((R(2,2))^2*s1+s3)+s5)];
c=[c 2*s6-2*((Z(1))^2-(R(1,2))^2*(xi1(1))^2)
-4*(R(1,2))^2*(gamma1(2))^2*(xi1(2))^2
-2*(R(2,2))^2*(s1*s2+2*(R(1,2))^2*(gamma1(2))^2*(xi1(2))^2)];
c=[c -4*R(1,2)*gamma1(2)*(s5+xi1(2)*(Z(1)+R(1,2)*xi1(1)
+(R(2,2))^2*s2))];
c=[c s6+2*R(1,2)*s4-(R(1,2)*xi1(1)+Z(1))^2-(R(2,2))^2*s2^2];
else
s12=(R(1,2))^2+(R(1,1))^2+(Z(1))^2-(R(2,2))^2
-(R(2,1))^2-(Z(2))^2;
s1=Z(1)-Z(2)+R(1,2)*xi1(1);s2=Z(1)-Z(2)-R(1,2)*xi1(1);
s3=s12+2*R(1,2)*(Z(1)*xi1(1)+R(1,1)*xi1(2))-2*Z(2)*s1;
s4=4*R(1,2)*gamma1(2)*((Z(1)-Z(2))*xi1(2)-R(1,1)*xi1(1));
s5=s12-2*R(1,2)*(Z(1)*xi1(1)+R(1,1)*xi1(2))-2*Z(2)*s2;
s6=2*R(2,1)*s1*xi2(2)-s3*xi2(1);s7=2*R(2,1)*s1*xi2(1)+s3*xi2(2);
s8=2*R(2,1)*s2*xi2(2)-s5*xi2(1);s9=2*R(2,1)*s2*xi2(1)+s5*xi2(2);
s10=4*R(2,1)*R(1,2)*gamma1(2)*xi1(2)*xi2(2)-s4*xi2(1);
s11=4*R(2,1)*R(1,2)*gamma1(2)*xi1(2)*xi2(1)+s4*xi2(2);
c=[(gamma2(2))^2*(s9^2-4*(R(2,2))^2*(R(2,1))^2)
+s8^2 2*s10*s8+2*s9*s11*(gamma2(2))^2];
c=[c 2*s6*s8+s10^2+(2*s7*s9+s11^2
-8*(R(2,2))^2*(R(2,1))^2)*(gamma2(2))^2];
c=[c 2*s10*s6+2*s7*s11*(gamma2(2))^2 (gamma2(2))^2*(s7^2
-4*(R(2,2))^2*(R(2,1))^2)+s6^2];
end
c1=[c(5) c(4)];
if abs(c(3))+abs(c(2))+abs(c(1))¿0.000001,
c1=[c1 c(3)];
end
if abs(c(2))+abs(c(1))¿0.000001,
c1=[c1 c(2)];
end
if abs(c(1))¿0.000001,
c1=[c1 c(1)];
```

```
end
bet1=roots(c);bet2=roots(c1);[jj,be ]=ro2(bet1,bet2);
if jj==0,
else
for i=1:jj,
rx=xyza(2,1,R,xi1,gamma1,be(i));rd(i,:)=rx+[R(1,1) 0 Z(1)];
end
end
end
if jj==0,
else
x=rd(:,1);y=rd(:,2);r=sqrt(x.^2+y.^2);z=rd(:,3);
end

    % This function is used to solve the x and y of two quadratic
% functions (f1(x^2,x,xy,Y^2,y)=0,f2(x^2,x,xy,y^2,y)=0)
% with parameters a(i,j). (i=2, j=6)
function [x,y,jj]=ro1(a)
if abs(a(2,1))¡0.00001
else
t=a(2,1);
for i=1:6,
a(2,i)=a(1,1)*a(2,i)/t-a(1,i);
end
end
c(1)=-a(1,3)*a(2,3)*a(2,4)+a(1,4)*a(2,3)^2+a(1,1)*a(2,4)^2;
c(2)=-a(1,2)*a(2,3)*a(2,4)+a(1,5)*a(2,3)^2+2*a(1,1)*a(2,4)*a(2,5)
-a(1,3)*a(2,3)*a(2,5)-a(1,3)*a(2,2)*a(2,4)+2*a(1,4)*a(2,2)*a(2,3);
c(3)=a(1,4)*a(2,2)^2+a(1,6)*a(2,3)^2-a(1,3)*a(2,3)*a(2,6)
+2*a(1,1)*a(2,4)*a(2,6)+2*a(1,5)*a(2,2)*a(2,3)-a(1,2)*a(2,2)*a(2,4)
+a(1,1)*a(2,5)^2-a(1,2)*a(2,3)*a(2,5)-a(1,3)*a(2,2)*a(2,5);
c(4)=-a(1,2)*a(2,2)*a(2,5)+a(1,5)*a(2,2)^2-a(1,3)*a(2,2)*a(2,6)
-a(1,2)*a(2,3)*a(2,6)+2*a(1,6)*a(2,2)*a(2,3)+2*a(1,1)*a(2,5)*a(2,6);
c(5)=-a(1,2)*a(2,6)*a(2,2)+a(1,6)*a(2,2)^2+a(1,1)*a(2,6)^2;
q=roots(c);jj=0;
for i=1:4
if imag(q(i))==0
jj=jj+1;y(jj)=q(i);
```

```
else
end
end
x=-(a(2,4)*y.^2+a(2,5)*y+a(2,6))./(a(2,2)+a(2,3)*y);

   % This function is used to solve t and 1/t
function [jj,anl]=ro2(t0,t1)
nn=length(t0);ki=0;
for i=1:nn,
if abs(imag(t0(i)))¡0.000001,
t0(i)=real(t0(i));
else
end
if imag(t0(i))==0
t0(i)=atan(t0(i));
if t0(i)¿-pi/4 & t0(i)¡=pi/4
ki=ki+1;tt2(ki)=t0(i);
else
end
else
end
end
ki=0;nn=length(t1);
for i=1:nn,
if abs(imag(t1(i)))¡0.000001,
t1(i)=real(t1(i));
else
end
if imag(t1(i))==0
t1(i)=(pi/2-atan(t1(i)));
if t1(i)¿pi/4 & t1(i)¡=3*pi/4
ki=ki+1;tt3(ki)=t1(i);
else
end
else
end
end
tt=[tt2 tt3];nn=length(tt);jj=0;
```

```
for i=1:nn,
if i==1,
jj=jj+1;anl(jj)=2*tt(i);
else
if abs(tt(i)-tt(i-1))¡0.000001,
else
jj=jj+1;anl(jj)=2*tt(i);
end
end
end

    % This function is used to solve t and 1/t
function [jj,anl]=ro3(t0,t1,s1,s2)
nn=length(t0);ki=0;
for i=1:nn,
if abs(imag(t0(i)))¡0.000001,
t0(i)=real(t0(i));
else
end
if imag(t0(i))==0
if s2(i)¿0
t0(i)=atan(t0(i));
if t0(i)¿=-pi/4 & t0(i)¡=pi/4
ki=ki+1;tt2(ki)=t0(i);
else
end
else
t0(i)=atan(t0(i))+pi;
if t0(i)¿=3*pi/4 & t0(i)¡=5*pi/4
ki=ki+1;tt2(ki)=t0(i);
else
end
end
else
end
end
ki=0;nn=length(t1);
for i=1:nn,
if abs(imag(t1(i)))¡0.000001,
t1(i)=real(t1(i));
else
end
if imag(t1(i))==0
if s1(i)¿0,
t1(i)=pi/2-atan(t1(i));
if t1(i)¿pi/4 & t1(i)¡3*pi/4
ki=ki+1;tt3(ki)=t1(i);
else
end
else
t1(i)=3*pi/2-atan(t1(i));
if t1(i)¿5*pi/4 & t1(i)¡7*pi/4
ki=ki+1;tt3(ki)=t1(i);
else
end
end
else
end
end
tt=[tt2 tt3];nn=length(tt);jj=0;
for i=1:nn,
if i==1,
jj=jj+1;anl(jj)=tt(i);
else
ks=0;
for k=i:-1:2,
if abs(tt(i)-tt(k-1))¡0.000001,
ks=1;
else
end
end
if ks==1,
else
jj=jj+1;anl(jj)=tt(i);
end
end
```

```
end

% this function is uesed to update intersecting points(x,y,z).
function [r1,r2,r3,r4,n1]=ro4(t,mm,ii,jj,kk,r1,r2,r3,r4,no,jj1,n1)
ni=[];nj=[];
[ni(1),nj(1)]=size(r1);[ni(2),nj(2)]=size(r2);
[ni(3),nj(3)]=size(r3);[ni(4),nj(4)]=size(r4);
nm=max(nj)+1;n2=n1;
if mm(ii)==1,
r1(:,nm+3)=t(:,1);r2(:,nm+3)=t(:,1);n1=1;
if jj¿1,
r2(:,nm+3)=t(:,2);
if jj¿2,
r3(:,nm+3)=t(:,3);r4(:,nm+3)=t(:,3);n1=0;
if jj¿3,
r4(:,nm+3)=t(:,4);
end
end
end
if ii==1 & no(1)==1,
i1=nm+3;i2=nm+2;[r1,r2,r3,r4]=ro5(i1,i2,r1,r2,r3,r4,n1);
no(1)=0;else
if ii==1,
else
i1=nm+3;i2=nm+2;[r1,r2,r3,r4]=ro5(i1,i2,r1,r2,r3,r4,n1);
i1=nm-1;i2=nm;[r1,r2,r3,r4]=ro5(i1,i2,r1,r2,r3,r4,n2);
end
end
else
if kk==1,
r2(:,nm-1)=r1(:,nm-1);kk=2;
end
if kk==3,
d=[];
d=abs(r1(1,nm-1)-t(1,:))+abs(r1(2,nm-1)-t(2,:))+abs(r1(3,nm-1)-t(3,:));
[y,k1]=min(d(1:jj));
d=[];
d=abs(r2(1,nm-1)-t(1,:))+abs(r2(2,nm-1)-t(2,:))+abs(r2(3,nm-1)-t(3,:));
```

```
[y,k2]=min(d(1:jj));
if k1==1,
if k2==2,
r4(:,nm-1)=r3(:,nm-1);
else
r4(:,nm-1)=r2(:,nm-1);
end
else
if kk==2,
if k2==1,
r4(:,nm-1)=r3(:,nm-1);
else
r4(:,nm-1)=r1(:,nm-1);
end
else
if k2==2,
r4(:,nm-1)=r1(:,nm-1);
else
r4(:,nm-1)=r2(:,nm-1);
end
end
end
kk=4;
end
if jj==1,
t(:,2)=t(:,1);jj=2;
end
if jj==3,
if n1==2,
[r1,r2,r3,r4,k1,k2,n1]=ro7(t,nm,jj,r1,r2,r3,r4,2,0);
cc1=t(:,k1);cc2=t(:,k2);
for i=1:3,
if i =k1 & i =k2,
t(:,1)=t(:,i);t(:,2)=t(:,i);
end
end
t(:,3)=cc1;t(:,4)=cc2;
else
```

```
[r1,r2,r3,r4,k1,k2,n1]=ro7(t,nm,jj,r1,r2,r3,r4,1,0);
cc1=t(:,k1);cc2=t(:,k2);
for i=1:3,
if i =k1 & i =k2,
t(:,3)=t(:,i);t(:,4)=t(:,i);
end
end
t(:,1)=cc1;t(:,2)=cc2;
end
jj=4;
end
if kk¿2,
if jj==2,
[r1,r2,r3,r4,k1,k2,n1]=ro7(t,nm,jj,r1,r2,r3,r4,0,0);
[r1,r2,r3,r4,k1,k2,n1]=ro7(t,nm,jj,r1,r2,r3,r4,n1,1);
i2=nm-1;
[r1,r2,r3,r4]=ro6(2,m1,m2,i2,nm,t,r1,r2,r3,r4,n1);
else
[r1,r2,r3,r4,k1,k2,n1]=ro7(t,nm,jj,r1,r2,r3,r4,0,1);
end
else
[r1,r2,r3,r4,k1,k2,n1]=ro7(t,nm,jj,r1,r2,r3,r4,n1,1);
if jj =2,
for i=1:4,
if i =k1 & i =k2, m1=i; end
end
for i=1:4,
if i =k1 & i =k2 & i =m1, m2=i; end
end
i2=nm-1;[r1,r2,r3,r4]=ro6(1,m1,m2,nm,i2,t,r1,r2,r3,r4,n1);
end
if k1==k2,
for i=1:2,
if i =k1,
if n1==1, r2(:,nm)=t(:,i); end
if n1==2, r4(:,nm)=t(:,i); end
end
end
```

```
end
end
end

    % this function is uesed to update intersecting points(x,y,z).
function [r1,r2,r3,r4]=ro5(i1,i2,r1,r2,r3,r4,n1)
if n1==2,
else
r2(:,i2)=r1(:,i1);r1(:,i2)=r2(:,i1);
end
if n1==1,
else
r4(:,i2)=r3(:,i1);r3(:,i2)=r4(:,i1);
end

    % this function is uesed to update intersecting points(x,y,z).
function [r1,r2,r3,r4]=ro6(ld,m1,m2,i1,i2,t,r1,r2,r3,r4,n1)
if n1==1,
if ld==1,
r3(:,i1)=t(:,m1);r4(:,i1)=t(:,m2);
end
r3(:,i2)=r4(:,i1);r4(:,i2)=r3(:,i1);
else
if ld==1,
r1(:,i1)=t(:,m1);r2(:,i1)=t(:,m2);
end
r1(:,i2)=r2(:,i1);r2(:,i2)=r1(:,i1);
end

    % this function is uesed to update intersecting points(x,y,z).
function [r1,r2,r3,r4,k1,k2,no]=ro7(t,nm,jj,r1,r2,r3,r4,n1,n)
if n1==2,
else
d=[];
d=abs(r1(1,nm-1)-t(1,:))+abs(r1(2,nm-1)-t(2,:))+abs(r1(3,nm-1)-t(3,:));
s1=norm(d);[y,i]=min(d(1:jj));
if n==1, r1(:,nm)=t(:,i); end
k1=i;d=[];
```

```
d=abs(r2(1,nm-1)-t(1,:))+abs(r2(2,nm-1)-t(2,:))+abs(r2(3,nm-1)-t(3,:));
s2=norm(d);[y,i]=min(d(1:jj));
if n==1, r2(:,nm)=t(:,i); end
k2=i;
end
if n1==1,
else
d=[];
d=abs(r3(1,nm-1)-t(1,:))+abs(r3(2,nm-1)-t(2,:))+abs(r3(3,nm-1)-t(3,:));
s3=norm(d);[y,i]=min(d(1:jj));
if n==1, r3(:,nm)=t(:,i); end
k1=i;d=[];
d=abs(r4(1,nm-1)-t(1,:))+abs(r4(2,nm-1)-t(2,:))+abs(r4(3,nm-1)-t(3,:));
s4=norm(d);[y,i]=min(d(1:jj));
if n==1, r4(:,nm)=t(:,i); end
k2=i;
end
if n==0,
no=1;[y,i]=min([s1+s2 s1+s3 s1+s4 s2+s3 s2+s4 s3+s4]);
if i==6,
rr(:,:)=r1(:,:);r1(:,:)=r3(:,:);r3(:,:)=rr(:,:);
rr(:,:)=r2(:,:);r2(:,:)=r4(:,:);r4(:,:)=rr(:,:);
end
if i==5,
rr(:,:)=r1(:,:);r1(:,:)=r4(:,:);r4(:,:)=rr(:,:);
end
if i==4,
rr(:,:)=r1(:,:);r1(:,:)=r3(:,:);r3(:,:)=rr(:,:);
end
if i==3,
rr(:,:)=r2(:,:);r2(:,:)=r4(:,:);r4(:,:)=rr(:,:);
end
if i==2,
rr(:,:)=r2(:,:);r2(:,:)=r3(:,:);r3(:,:)=rr(:,:);
end
else
no=n1;
end
```

Appendix B: Solution Verification Program

(BASIC)

```
1000 OPEN "i",#1,"tol.dat"
1010 OPEN "o",#2,"in.scr"
1020 INPUT #1, TI1
1030 INPUT #1, TI2
1040 INPUT #1, YY
1050 INPUT #1, F
1060 INPUT #1, ZZ1
1070 INPUT #1, ZZ2
1080 INPUT #1, RC1
1090 INPUT #1, RA1
1100 INPUT #1, RC2
1110 INPUT #1, RA2
1120 INPUT #1, GA1
1130 INPUT #1, IT1
1140 INPUT #1, GA2
1150 INPUT #1, IT2
1160 N1=1:PI=3.1415926#:IF ABS(YY)+ABS(F)=0! THEN N1=2:
GOTO 1170
1170 PRINT #2, "LAYER m 1 m 2 m 3 c r m 4 c g m 5 ":PRINT #2,
"'_SURFTAB1 32"
```

```
1190 IF TI1¡1.5 THEN GOTO 1230
1200 PRINT #2,"'_SURFTAB2 2"
1210 X1=RC1+RA1*COS(IT1*PI/180):Y1=ZZ1+RA1*SIN(IT1*PI/180):
Z1=0
1220 PRINT #2, USING "__ELLIPSE c #.#####^^^^,#.#####^^^^
@#.#####^^^^¡#.#####^^^^ r #.#####^^^^"; RC1, ZZ1, RA1,
IT1, GA1: GOTO 1260
1230 GA1=GA1/2: X1=RC1:X2=RC1: Y1=ZZ1+GA1: Y2=ZZ1-GA1:
Z1=-GA1*IT1: Z2=-Z1
1240 GOSUB 1910:PRINT #2, "'_SURFTAB2 32"
1250 IF RC1=0! THEN PRINT #2, "ROTATE3D l y -90":X1=-Z1:Z1=0!:
GOTO 1260
1260 N=3:GOSUB 1920:PRINT #2, USING "_REVSURF #.#####^^^^,
#.#####^^^^, #.#####^^^^ 0,0.2,0 ";X1,Y1,Z1
1270 N=2:GOSUB 1920:N=3:GOSUB 1930:N=1:GOSUB 1930
1280 X1=0:Y1=2:Z1=0:X2=0:Y2=-2:Z2=0:GOSUB 1910
1290 IF TI2¡1.5 THEN GOTO 1330
1300 PRINT #2,"'_SURFTAB2 2"
1310 X1=RC2+RA2*COS(IT2*PI/180):Y1=ZZ2+RA2*SIN(IT2*PI/180):
Z1=0
1320 PRINT #2, USING "__ELLIPSE c #.#####^^^^, #.#####^^^^
@#.#####^^^^¡#.#####^^^^ r #.#####^^^^"; RC2, ZZ2, RA2,
IT2, GA2: GOTO 1360
1330 GA2=GA2/2: X1=RC2:X2=RC2: Y1=ZZ2+GA2: Y2=ZZ2-GA2:
Z1=-GA2*IT2: Z2=-Z1
1340 GOSUB 1910:PRINT #2, "'_SURFTAB2 32"
1350 IF RC2=0! THEN PRINT #2, "ROTATE3D l y -90":X1=-Z1:Z1=0!:
GOTO 1360
1360 N=4:GOSUB 1920:PRINT #2, USING "_REVSURF #.#####^^^^,
#.#####^^^^, #.#####^^^^ 0,0.2,0 ";X1,Y1,Z1
1370 PRINT #2, USING "MOVE l 0,0,0 #.#####^^^^,0,0";YY
1380 IF F=0! THEN GOTO 1400
1390 F=-F:PRINT #2, USING "ROTATE3D l x #.#####^^^^";F
1400 N=5:GOSUB 1920:N=4:GOSUB 1930:N=2:GOSUB 1930
1410 NM=0:M=0:MM=0
1420 IF EOF (1) THEN 1630
1430 INPUT #1,Z1
1440 INPUT #1,X1
```

```
1450 INPUT #1,Y1
1460 IF N1=1 THEN GOTO 1520
1470 NM=NM+1:IF NM=1 THEN X2=X1:Y2=Y1:Z2=Z1:GOTO 1510
1480 IF NM=2 THEN D=ABS(Z1):GOTO 1510
1490 NM=0:PRINT #2, USING "ai__torus #.#####^^^^,
#.#####^^^^, #.#####^^^^ #.#####^^^^ 0.05 180 18";
Z2,X2,Y2,D
1500 PRINT #2, "ROTATE3D l x -90"
1510 GOTO 1420
1520 IF ABS(X1)+ABS(Y1)+ABS(Z1)=0! THEN GOTO 1620
1530 IF MM=0 THEN X2=X1:Y2=Y1:Z2=Z1:MM=1:GOTO 1420
1540 X=X1-X2:Y=Y1-Y2:Z=Z1-Z2:D=SQR(X*X+Y*Y+Z*Z)
1550 IF Z=0! THEN ALP2=90*(1+SGN(X)):GOTO 1570
1560 ALP2=ATN(X/Z)*180/PI+SGN(Z)*90
1570 IF Y=0! THEN ALP1=90!:GOTO 1590
1580 ALP1=180/PI*ATN(SQR(X*X+Z*Z)/Y)+90*(1-SGN(Y))
1590 PRINT #2, USING "INSERT turb 0,0,0 0.05 #.#####^^^^,
#.#####^^^^"; D,ALP1
1600 PRINT #2, USING "ROTATE3D l y #.#####^^^^";ALP2
1610 PRINT #2, USING "MOVE l 0,0,0 #.#####^^^^, #.#####^^^^,
#.#####^^^^"; X2, Y2, Z2: X2=X1: Y2=Y1: Z2=Z1: GOTO 1420
1620 MM=0:GOTO 1420
1630 N=3:GOSUB 1940:N=4:GOSUB 1940
1640 CLOSE #1
1650 CLOSE #2
1660 OPEN "o",#2,"in.txt"
1670 INPUT "fig number:",NUM$
1680 IF TI1¿1.5 THEN GOTO 1710
1690 IF RC1=0! THEN TIT1$="Cone":GOTO 1730
1700 IF IT1=0! THEN TIT1$="Clinder":GOTO 1730
ELSE TIT1$="Hyperboloid": GOTO 1730
1710 IF RC1=0! THEN TIT1$="Sphere":GOTO 1730
1720 TIT1$="El_torus"
1730 IF TI2¿1.5 THEN GOTO 1760
1740 IF RC2=0! THEN TIT2$="Cone":GOTO 1780
1750 IF IT2=0! THEN TIT2$="Clinder":GOTO 1780
ELSE TIT2$="Hyperboloid": GOTO 1780
1760 IF RC2=0! THEN TIT2$="Sphere":GOTO 1780
```

```
1770 TIT2$="El_torus"
1780 PRINT #2, TIT1$ " & " TIT2$ "- " NUM$
1790 PRINT #2, "Surface 1, " TIT1$ ":"
1800 PRINT #2, USING "Rc=##.##; Ra=##.##";RC1,RA1
1810 IF TI1¡1.5 OR RC1=0! THEN GOTO 1830
1820 PRINT #2, USING "A.ga=####; A.xi=####";GA1,IT1
1830 PRINT #2, "Surface 2, "TIT2$ ":"
1840 PRINT #2, USING "Rc=##.##; Ra=##.##";RC2,RA2
1850 IF TI2¡1.5 OR RC2=0! THEN GOTO 1870
1860 PRINT #2, USING "A.ga=####; A.xi=####";GA2,IT2
1870 PRINT #2, "Between two axes:"
1880 F=-F:PRINT #2, USING "Distance=##.##; Angle=###";YY,F
1890 CLOSE #2
1900 STOP
1910 PRINT #2,USING "LINE #.##### ^^^^,#.#####^^^^,
#.#####^^^^ #.#####^^^^, #.#####^^^^, #.#####^^^^ ";
X1, Y1, Z1, X2, Y2, Z2: RETURN
1920 PRINT #2,USING "LAYER s # ";N:RETURN
1930 PRINT #2,USING "LAYER f # ";N:RETURN
1940 PRINT #2,USING "LAYER t # ";N:RETURN
```