Automatic Surface Generation in Computer Aided Design*

Christoph Hoffmann
John Hopcroft
TR 85-661
January 1985

Department of Computer Science
Cornell University
Ithaca, New York 14853

* This work was supported in part by the National Science Foundation under grant ECS 83-12096 and MCS 82-17996.

1. Introduction

It is widely recognized that computer aided design and other engineering tasks will require computer representations of physical objects. These representations must be capable of supporting a wide variety of activities including graphical display, simulations, finite element techniques, etc. The methods of representation in use today include wire-frames, octrees, surface representations and constructive solid geometry[1-3,5,7,8]. Each method has inherent advantages and disadvantages.

Regardless of the relative merits of current representations for the design process, many objects one wishes to design contain surfaces whose sole purpose it is to connect other surfaces smoothly. Often, such surfaces are unimportant to the design at large, yet occupy much of the designer's time and effort since they can be mathematically complicated and their shape and placement has to be in precise relation to the surfaces to be connected. We therefore concentrate in this work on laying the ground work for a system that automatically deduces and places these shapes, freeing the designer from much drudgery and routine work.

We have developed a class of surfaces from which a smooth surface can be selected to connect given algebraic surfaces in a completely automatic fashion. The specific advantages of the class include:

(1) the derivation is straight forward,

(2) the method can connect arbitrary, algebraic surfaces,

(3) the degree of these connecting surfaces is small and their shape is easily understood.

In fact, the method can also be applied to patched surfaces, although we do not explore this fact in detail in this paper.

We demonstrate the use of our automatic surface derivation by means of a concrete design. Here we have to

(1) identify typical situations in which surfaces may be derived, and

(2) provide concise, intuitive, and flexible syntactic structures for specifying where these derivations are to take place.

As a sample design, we have chosen the gate valve body, shown in Figure 1. In it, we derive automatically all fillet surfaces, all edge roundings, and the joining surfaces arising when reducing or increasing pipe diameters. We refer to these surface types collectively as *blending surfaces.*

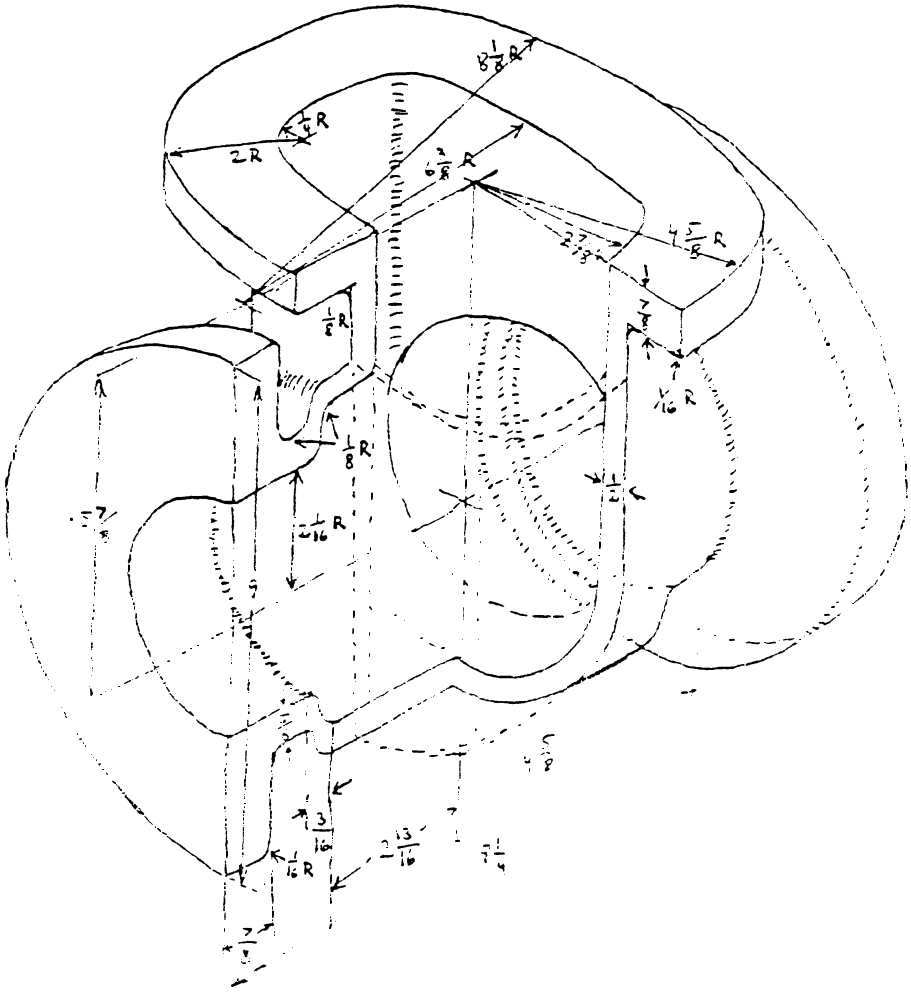

Figure 1

Gate Valve Body

In this paper, we represent surfaces by algebraic equations. An important consideration when doing so is the degree of the algebraic equations that arise. We are concerned about the degree since we do not wish to commit ourselves to specific methods for solving algebraic equations. The equation degree can enter exponentially into the running time of many symbolic computations we might wish to perform, and would also affect adversely the simplicity of numerical approximations, should approximative methods be preferred over symbolic ones. Furthermore, the number of coefficients a surface equation may have grows dramatically with its degree. A three variable equation of degree 16, for example, may have up to 969 coefficients. If the surface depends on several parameters then many more coefficients are required. A degree 16 surface in 3-space depending on only two parameters can have up to $\binom{21}{5}$ or 20349 coefficients.

Our central concern is that the system be able to deduce automatically blending surfaces whose purpose is to smooth out intersections of other shapes or to join other shapes in a standardized manner. Since a blending surface is constrained to be tangent to two other surfaces in general position, it has, in general, a higher degree than either of the other two surfaces. It is crucial to obtain low degree blending surfaces, so that they may be derived simply and admit simple internal manipulation algorithms, if one is not to predispose the implementation to rely on specific solution techniques.

Here we can state positively that the automatic derivation of a blending surface results in a new surface of very low degree. For example, when blending two degree 2 surfaces, a degree 4 surface suffices. In this regard, our derivation method contrasts pleasantly with traditional techniques from differential geometry [6]. In particular, when smoothing the intersection of two cylinders, one could consider a family of spheres of fixed or varying radii that are in contact with both cylinders, and take as blending surface the envelope of the family. This results, in the simplest case, in a polynomial of degree 16. Another possible strategy for obtaining blending surfaces is to rotate a plane through the axis of one of the cylinders and prescribe the intersection curve of the blending surface with this plane, say as a circle. This results in a parametric definition of the surface whose implicitization is beyond the resources of MACSYMA [4] running on a VAX 780. In contrast, our surfaces can be derived very easily without the aid of a computer.

Ultimately, a precise language is needed to specify unambiguously the shape and position of all components to be connected, by automatically derived joints or otherwise. The design of such a language is not the central concern of this paper. Consequently, many language issues deliberately remain undeveloped. Rather, we restrict our attention to classifying a number of typical joints and parameterize their shape and orientation.

Since a general, detailed notation can easily frustrate the intuitive spatial ability of the designer, it is crucial to give a simple and clear form to typical situations. We do this by concentrating on three concepts: *attach, smooth* and *join*. The simplest of these is *attach*. It refers to combining, without penetration, two objects by mating identical features in each object. This entails appropriately positioning the second object in space so that mating can be achieved. In particular, the designer is not required to explicitly position the objects. After the objects have been joined, new edges will appear in the composit as the intersection of two surfaces, one from either component. These edges may have to be smoothed out by rounding or filleting. In this case, one uses *smooth_attach* in place of *attach*.

Both the *smooth* and *join* operations modify objects by adding or subtracting volumes through which a blending of identified features is achieved. Here the *smoothing* operation blends intersecting surfaces in given position, whereas *join* connects nonintersecting surfaces that may have to be positioned in space. In particular, *join* makes special assumptions about the type of joint to be achieved, thereby restricting its positioning abilities. More general joining operations are not needed in the gate valve design, and so both the mathematical derivations, as well as the textual description of more complicated joints has been omitted.

It is by now standard to describe solids in local coordinate frames. When combining two objects, the relative position of the objects as well as the resulting coordinate system needs to be understood. It is our view, that this should be governed by simple, yet flexible rules that dispense with the need of explicitly positioning objects in routine situations. Consequently, *attach, smooth* and *join* all adopt the coordinate frame of the first object argument for the result. Subsequent object arguments are automatically positioned. In considering the valve design, the reader may verify that there is indeed a minimum of coordinate frame manipulation and movement of objects.

2. The derivation of blending surfaces

Two types of blending surfaces arise frequently in physical objects. The first type of surface is used to smooth the intersetion of two surfaces. The second type is used to form a smooth transition between pipes with different cross sections. In this section we develop the first type of blending surface.

Associated with a surface $g(x,y,z)=0$ is a one parameter family of surfaces $G(s)$ defined by $g(x,y,z)=s$. For certain values of s, $G(s)$ may degenerate to lines, points or even the empty set. For example, if $g(x,y,z)=x^2+y^2+z^2-1$, then $G(s)$ is a sphere of radius $\sqrt{1+s}$. For $s=-1$ the surface degenetrates to a point, and for $s<-1$ the surface has no real points. Similarly, if $g(x,y,z)=x^2+y^2-1$, then $G(s)$ is a cylinder of radius $\sqrt{1+s}$ which degenerates into a line for $s=-1$.

Suppose we are given two cylinders C_1 and C_2 whose axes intersect at right angles. We can obtain a smooth surface as follows. Starting with an enlarged radius for C_2, gradually decrease this radius, simultaneously increasing the radius of C_1. The intersection of the two cylinders defines a curve that sweeps out a surface as the radii change. By suitably controlling the relative rate of change of the two radii, a blending surface is obtained. The construction is illustrated in Figure 2.

To be precise, let g and h be two surfaces whose intersection is to be smoothed and let $G(s_1)$ and $H(s_2)$ be the associated families of surfaces. We assume that $g(x,y,z)-s \neq h(x,y,z)$ for all s. Varying the values of s_1 and s_2 correspond to varying the radii in the example of the two intersecting cylinders. A function $f(s_1,s_2)=0$ is used to relate the values of s_1 and s_2. The implicit equation F for the surface is obtained by eliminating s_1 and s_2 from the three equations. Thus $F(x,y,z)=f(g(x,y,z),h(x,y,z))$.

To achieve the desired result, f must be choosen so as to satisfy three conditions.

(1) There must exist a point $(0,b)$ on f such that the surface $H(b)$ intersects $g(x,y,z)=0$ in a space curve. Similarly, a point $(a,0)$ must exist on f such that $G(a)$ intersects $h(x,y,z)$ in a space curve.

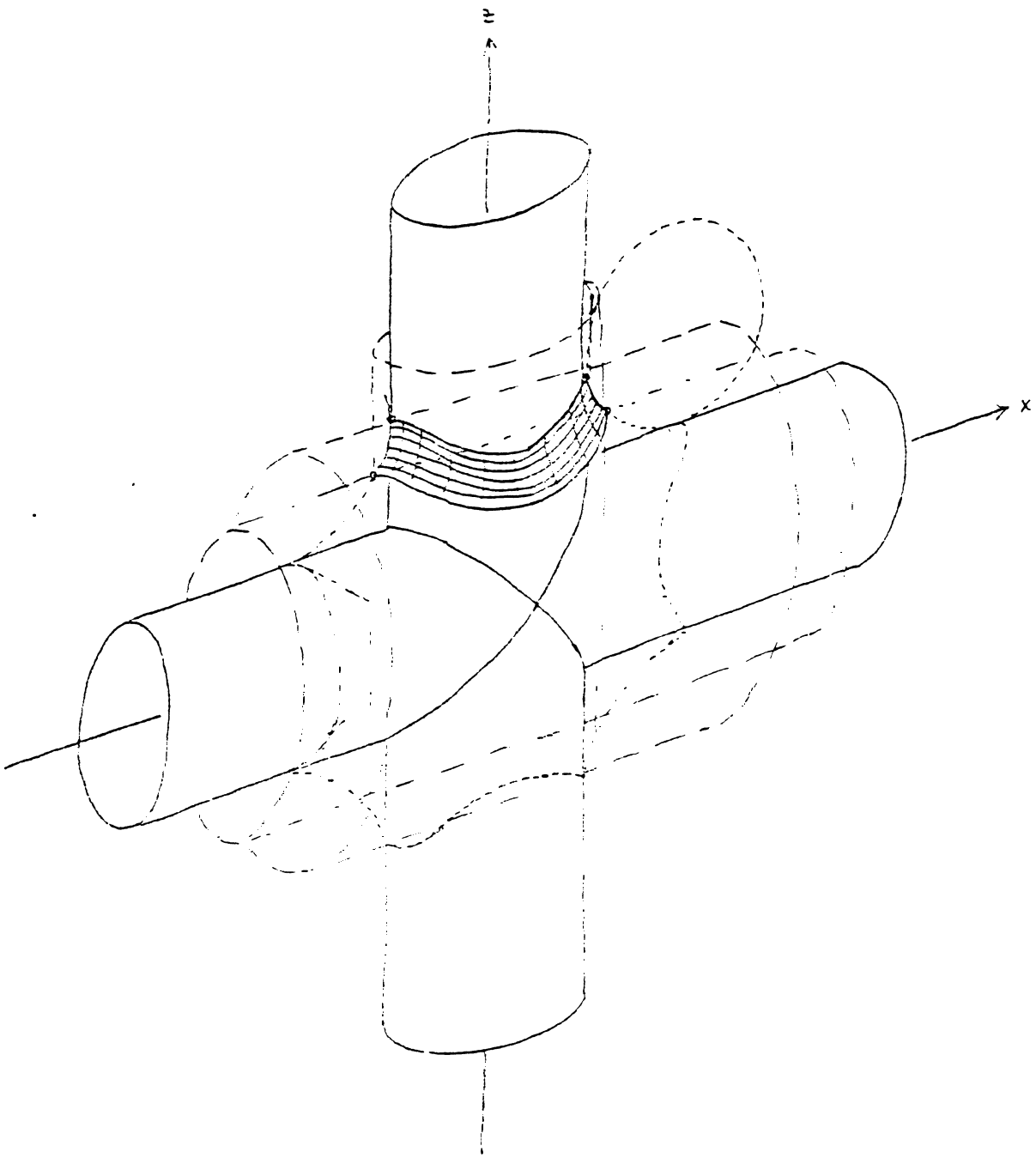

Figure 2

Blending Surface as Intersection

of Surface Families $G(s_1)$ and $H(s_2)$

(2) The coordinate axes must be tangent to f in the points $(0,b)$ and $(a,0)$.

(3) The function f must smoothly connect the points $(0,b)$ and $(a,0)$, and for each point (u,v) of the arc, $G(u)$ must intersect $H(v)$ in a space curve.

Intuitively, condition (1) entails that F intersects each of the surfaces g and h in a space curve. Specifically, F intersects g in the curve $G(0)\cap H(b)$ and F intersects h in the curve $G(a)\cap H(0)$. Condition (2) entails that F is tangent to g and h in the respective curves of intersection, and condition (3) states that F is well behaved in between. These facts are formalized in the following theorem.

Theorem 1: The surface $F=f(g,h)$ intersects g in the curve $G(0)\cap H(b)$ and intersects h in the curve $G(a)\cap H(0)$ and is tangent to the surfaces in the respective intersection curves.

Proof: The intersection of F with g and h is evident. We show that for every point on the intersection curve, the tangent planes to the two surfaces coincide. For this it suffices to show that the partial derivatives at the point agree up to a constant. The partial derivatives of F are:

$$F^{(x)}=f^{(g)}g^{(x)}+f^{(h)}h^{(x)}$$
$$F^{(y)}=f^{(g)}g^{(y)}+f^{(h)}h^{(y)}$$
$$F^{(z)}=f^{(g)}g^{(z)}+f^{(h)}h^{(z)}$$

If (x,y,z) is a point on $G(0)\cap H(b)$, i.e., of $F\cap g$, then by condition (2), $f^{(h)}=0$. Hence, for this point,

$$\begin{bmatrix}F^{(x)}\\F^{(y)}\\F^{(z)}\end{bmatrix}=f^{(g)}\begin{bmatrix}g^{(x)}\\g^{(y)}\\g^{(z)}\end{bmatrix}$$

Now f is not constant, hence $f^{(g)}\neq 0$, and so the tangent planes are the same. A similar argument shows tangency of F with h in the curve $F\cap h=G(a)\cap H(0)$.

Since we desire a blending surface of low degree and since g and h are given surfaces, the degree of f should be low. One choice for $f(s_1,s_2)$ is $(s_1-r)^2+(s_2-r)^2-r^2+0$. (here $a=b=r$.)

The surface F is then given by $(g-r)^2+(h-r)^2-r^2=0$. In the case in which g and h are quadric surfaces, the blending surface is seen to be of degree at most 4.

An Example

Consider the simple case of two cylinders, each of radius 1, whose axes intersect at right angles.

(1) $g(x,y)=x^2+y^2-1=0$

(2) $h(y,z)=y^2+z^2-1=0$

The associated families of surfaces are

$$G(s_1)=x^2+y^2-1-s_1=0$$

$$H(s_2)=y^2+z^2-1-s_2=0$$

We assume that the space curves in which the blending surface F is to intersect with the two cylinders are the intersection of $G(1)$ with h and of $H(1)$ with g. We connect (0,1) and (1,0) in the s_1-s_2 plane with a circle of radius 1, $(s_1-1)^2+(s_2-1)^2-1=0$. This yields the blending surface

$$x^4+2y^4+z^4+2x^2y^2+2y^2z^2-4x^2-8y^2-4z^2+7=0$$

It is possible to verify analytically, that the surface is tangent to both cylinders and smoothly connects them, as shown in Figure 3.

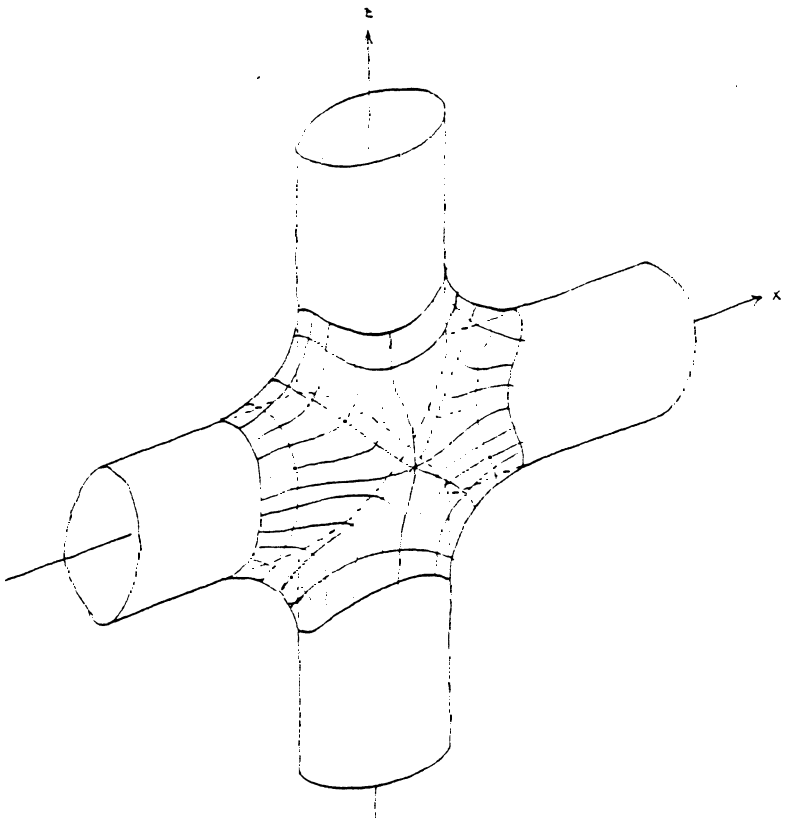

Figure 3

Blending Surface smoothing Outside

$((x^2+y^2-1)-1)^2+((y^2+z^2-1)-1)^2-1=0$

The role of the points $(0,b)$ and $(a,0)$ becomes evident when a and b are chosen as $-1/2$ and connected by an arc of a circle of radius $1/2$. One may verify that the resulting surface is

$$x^4+2y^4+z^4+2x^2y^2+2y^2z^2-3x^2-6y^2-3z^2+17/4=0$$

which also connects the two cylinders smoothly but is in the *interior* of both cylinders. The surface is sketched in Figure 4.

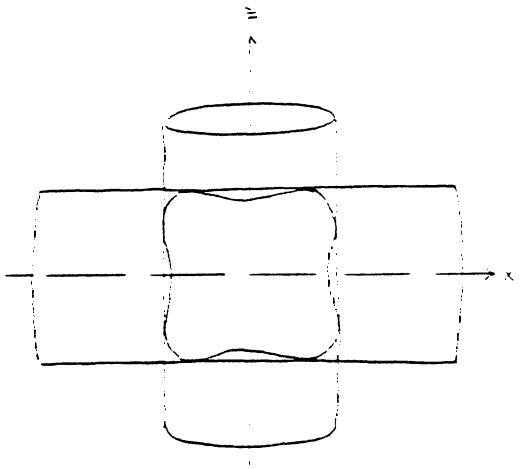

Figure 4

Blending Surface smoothing Inside

$((x^2+y^2-1)-1/4)^2+((y^2+z^2-1)-1/4)^2-1/16=0$

Similarly, a circle in the second s_1-s_2 quadrant leads to a surface that is in the interior of g and the exterior of h. A circle in the fourth quadrant leads to a surface that is in the interior of h and the exterior of g.

Considering these other positions for f illustrates the need to choose a and b such that the respective surfaces intersect. Choosing $a=1$ and $b=-2$ connected by an ellipse, for example, yields a surface that is tangent to h but does not touch g, since $H(-2)$ has no real points. In fact, this surface has two bubbles that "plug" the cylinder g in two places. If the points are $a=1$ and $b=-1$, then the bubbles touch g, each in one point, since $H(-1)$ has degenerated into a straight line.

3. Homotopy methods for transitional joints

In this section we develop smooth surface transitions between surfaces such as coaxial cylinders with different cross sections. The method constructs a family $G(t)$ of surfaces from two given surfaces g and h by means of a homotopy. The surfaces in the family $G(t)$ are intersected with surfaces in an auxiliary family of surfaces $H(t)$. For simplicity we take H to be a family of planes.

The intersection of G and H is a curve that sweeps out the desired surface as s varies. (See Figure 5.)

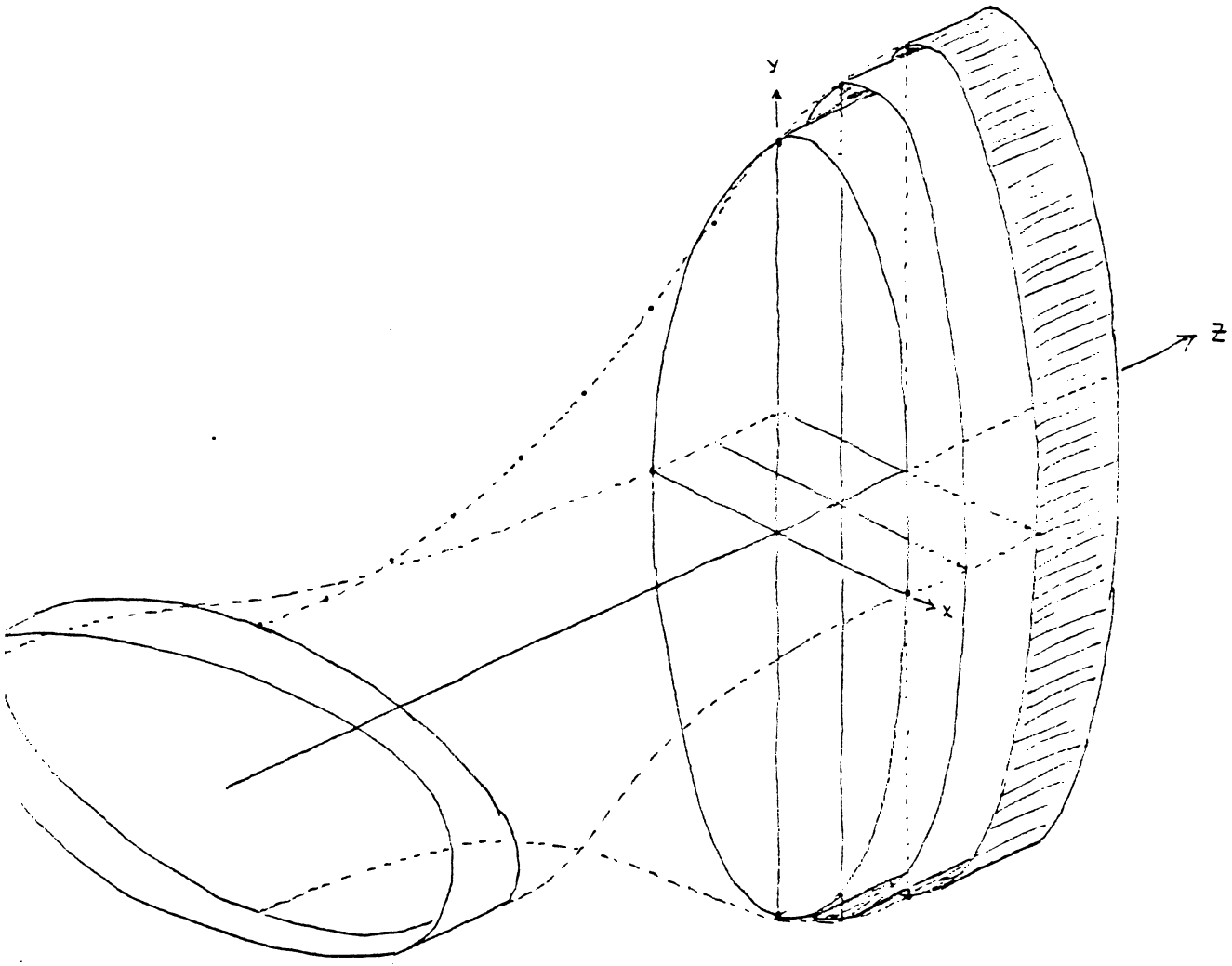

Figure 5

Blending surface as Intersection

of a Homotopy Family with Auxiliary Planes

Consider the homotopy

$$G(x,y,z,t)=s_1(t)g(x,y,z)+s_2(t)h(x,y,z)=0$$

and the auxiliary family of planes, $H(t)=z-t$. The intersection of corresponding surfaces in these

two families yields the blending surface

$$F(x,y,z)=s_1(z)g(x,y,z)+ s_2(z)h(x,y,z)=0$$

The functions s_1 and s_2 must satisfy the following conditions, so that F smoothly joins g and h:

(1) There is a value z_1 such that $s_2(z_1)=0$ and $s_2'(z_1)=0$.

(2) There is a value z_2 such that $s_1(z_2)=0$ and $s_1'(z_2)=0$.

(3) Let $s_1(z_1)=b$ and $s_2(z_2)=a$. In the s_1-s_2 plane, the points $(0,b)$ and $(a,0)$ are smoothly connected by the arc $(s_1(t),s_2(t))$, where $z_1\leq t\leq z_2$, and each surface $G(x,y,z,t)$ intersects the plane $z=t$ in a space curve.

Briefly, conditions (1) and (2) ensure that F is tangent to g and h in the $z=z_1$ and $z=z_2$ planes, respectively, and condition (3) states that in between these planes F is well behaved. Note the similarity with the conditions in the previous section.

Theorem 2: The surface $F=s_1g+s_2h$ is tangent to the surface g at $z=z_1$ and is tangent to the surface h at $z=z_2$.

Proof: Assume that conditions (1), (2) and (3) are satisfied. In the $z=z_1$ plane, we have $F(x,y,z)=bg(x,y,z)$, which means that the curve in which g intersects the $z=z_1$ plane is also the curve in which F intersects the plane. Now

$$F^{(x)}=s_1g^{(x)}+ s_2h^{(x)}$$

$$F^{(y)}=s_1g^{(y)}+ s_2h^{(y)}$$

$$F^{(z)}=s_1' g+ s_2' h+ s_1g^{(z)}+ s_2h^{(z)}$$

With $z=z_1$, we have for every point (x,y,z_1) on the intersection of F with g that $g(x,y,z_1)=0$, $s_2(z_1)=0$, and $s_2'(z_1)=0$. Hence

$$F^{(x)}(z=z_1)=bg^{(x)}(z=z_1)$$

$$F^{(y)}(z=z_1)=bg^{(y)}(z=z_1)$$

$$F^{(z)}(z=z_1)=bg^{(z)}(z=z_1)$$

For each point in which F intersects g, the tangent planes of the surfaces are the same and so F is tangent to g. A similar derivation establishes that F is tangent to h in the $z=z_2$ plane.

Example:

Let g and h be the surfaces to be joined. The homotopy $G(t)=t^2h+(1-t)^2g$ and the auxiliary surfaces $H(t)=z-t$ yield the surface $F(x,y,z)=z^2h+(z-1)^2g=0$. In this manner the two quadrics are joined by a degree 4 surface.

In the case where g and h are the elliptic cylinders $x^2+4y^2=4$ and $9x^2+y^2=9$, F is the surface

$$10x^2z^2+5y^2z^2-2x^2z-8y^2z+x^2+4y^2-13z^2+8z-4=0$$

shown in Figure 6.

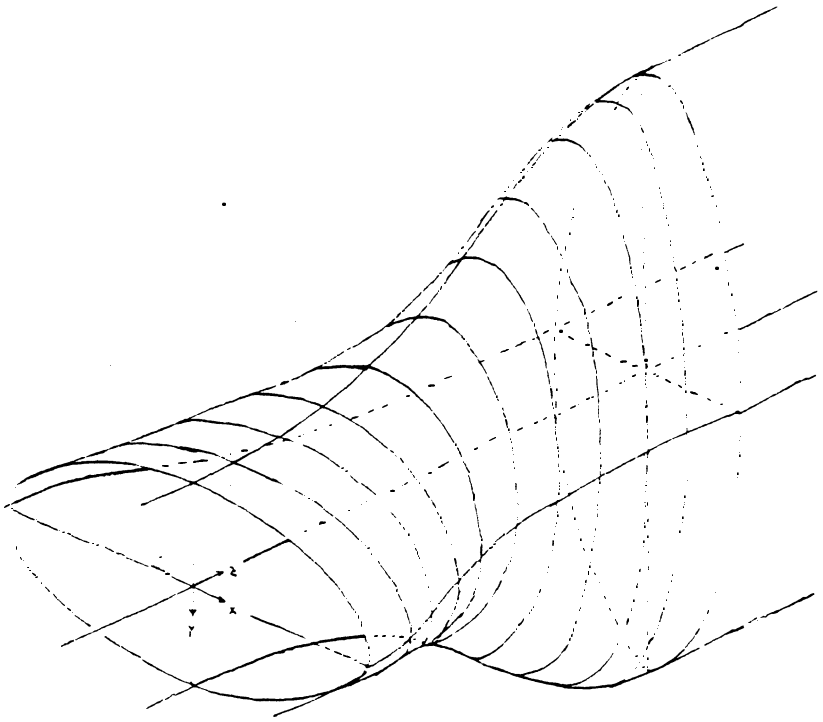

Figure 6

Blending Surface joining two Elliptic Cylinders

$(x^2+4y^2-4)(z-1)^2+(9x^2+y^2-9)z^2=0$

Example: Consider joining the circular cylinder $x^2+y^2-1=0$ with the parabolic cylinder $2x^2-y=0$. The surface derived to join the two is

$$F(x,y,z)=3x^2z^2+y^2z^2-2x^2z-2y^2z-yz^2+x^2+y^2-z^2+2z-1=0$$

Since the parabolic cylinder is open whereas the circular cylinder is not, the surface must open up in the positive y direction. It is shown in Figure 7.

Since the parabolic cylinder is open whereas the circular cylinder is not, the surface must open up in the positive y direction. It is shown in Figure 7.

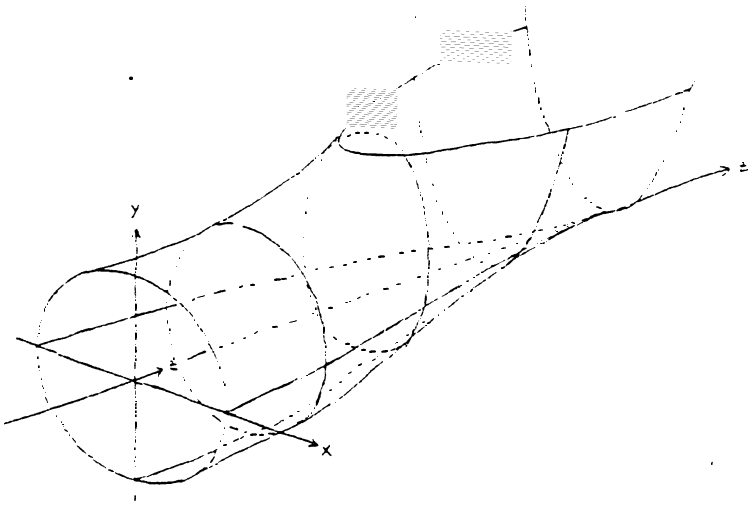

Figure 7

Blending Surface joining Circular Cylinder

and Parabolic Cylinder

$(x^2+y^2-1)(z-1)^2 + (2x^2-y)z^2 = 0$

The cross sections of the surface with the $z=t$ plane are elliptic for $0 \leq t < 1$, with increasing axis length in the $x=0$ plane. In the $z=1$ plane, the parabola is the limit curve of these ellipses. For $z>1$, the surface has again elliptic sections.

One should observe that a smooth joint derived with the homotopy method may not meet other design criteria such as minimum cross sectional area. Our experience indicates that in practice most surfaces derived are satisfactory, but undesirable results can occur in situations, such as illustrated in Figures 8 and 9. In Figure 8, the joining surface has a singularity.

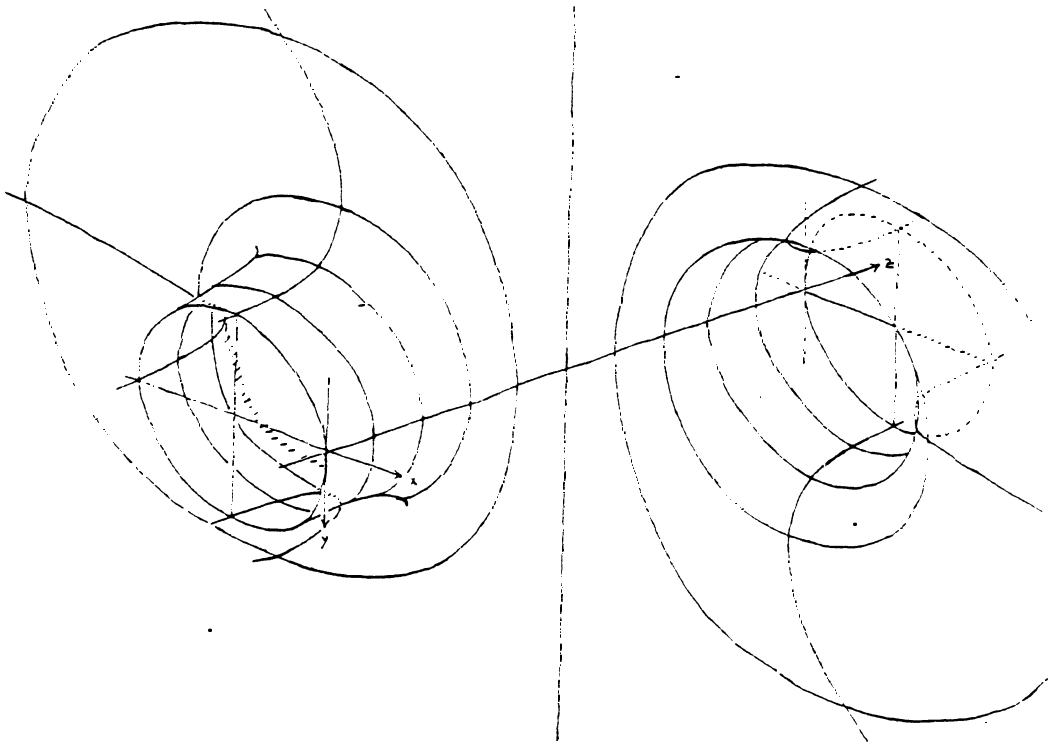

Figure 8

Singular Blending Surface

Homotopy Method

$$((x+1)^2+y^2-1)(z-1)^2+(-(x-1)^2-y^2+1)z^2=0$$

The reason here is that the inside of one cylinder was taken to be between the surface and its axis of symmetry, whereas in the second cylinder the outside is between the surface and its axis. This is typically a sign error that would not occur in an automated system. In Figure 9, the joining surface is pinched to a point because two mantle lines coincide, but on opposite surface sides.

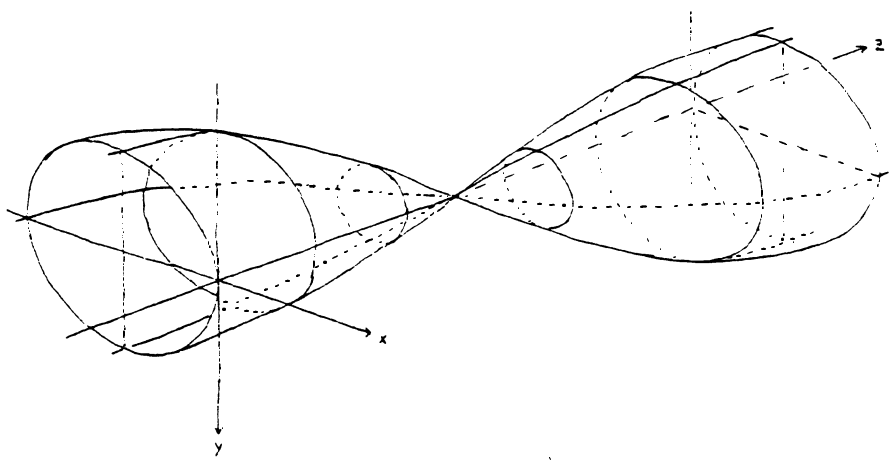

Figure 9

Pinched Blending Surface

Homotopy Method

$$((x+1)^2+y^2-1)(z-1)^2+((x-1)^2+y^2-1)z^2=0$$

This difficulty is intrinsic to the homotopy scheme we use here, and can be corrected by a scheme in which the homotopy is not purely additive.

4. The Gatevalve Body

We use the specification of a gate valve body as a vehicle for illustrating design using automatic surface generation. The gate valve is specified in a textual language. Language statements are definitional as opposed to executable statements in conventional programming languages. The complete description of the valve is given to illustrate that a designer can indeed make use of automatic surface generation to eliminate consideration of complex but unimportant parts of the design, thereby allowing him to concentrate his efforts on functionally important parts.

The valve body consists of a cylindrical chamber to which an intake and an outlet assembly are attached coaxially, one on each side. At right angle to the chamber axis is the bonnet stem, a pipe-flange combination of oval cross section, to which the valve bonnet (not designed here) is fastened, and through which the shut-off mechanism is guided. The chamber diameter equals the major diameter of the ovoid, which is the cross section of the valve stem, and may differ from intake and outlet assembly diameters.

Each assembly consists of a pipe-flange combination. The flange's outer diameter and thickness are functionally dependent on the pipe's wall thickness and size of the bolts with which the valve is to be fastened to a run of pipe. (For simplicity, we do not express the bolt dependence and give all flanges a uniform thickness of 7/8.) The connection between the various assemblies is made by joints that have to mediate between the differing dimensions. These joints are specified explicitly by the joint length and implicitly by the radius differential of the two pipes, as outlined in Section 3.

The valve body in our example is a simplified design. Usually the chamber diameter is reduced to the intake and outlet diameters in two steps with the first reduction accommodating screwed in valve seats. Moreover, the chamber and bonnet stem have two tracks along the inside on which the shut-off mechanism rides. The addition of these features introduces further complexity to the design definition, but it does not add fundamentally new considerations.

4.1. Preliminary Definitions

Since many subassemblies of the gate valve are made from pipes, we begin by defining the generic object *pipe*. The object is constructed from a system defined generic object ***ycylinder(r)*** and two half spaces, which are given explicitly. Here, ***ycylinder(r)*** is an infinite cylinder of radius r whose axis is the y-axis. The cylinder is a solid, given by the equation

$$x^2 + z^2 \leq r^2.$$

Note that definitions are made in the form of assignments to formal names. Features of the pipe are named by other definitions included in the definition of *pipe*. These subdefinitions are nested in the where-clause following the expression defining the pipe, and may be referred to from the outside as, e.g., ***pipe.top***.

```
pipe(radius, thickness, length) := (cyl1 - cyl2) ∩ H1 ∩ H2 where
  begin
    cyl1 := ycylinder(radius+thickness);
    cyl2 := ycylinder(radius);
    H1 := {y ≥ 0};
    H2 := {y ≤ length};
    top := (cyl1 - cyl2) ∩ {y = length};
    bottom := (cyl1 - cyl2) ∩ {y = 0};
    outside := cyl1.surface ∩ H1 ∩ H2;
    top.in_edge := top ∩ cyl2;
    bottom.in_edge := bottom ∩ cyl2
  end;
```

The definition of *pipe* is used to define a new generic object, ***flanged_pipe***, which is constructed from two pipes, one of which serves as a flange. Since the flange is conceptualized as a pipe, the flange's thickness is the the pipe's length (fl), whereas the radius difference is the wall thickness (ft). Note that ft functionally depends on the wall thickness of the pipe to which the flange is attached. Similarly, fl could be made functionally dependent on the bolt size of the bolts used to fasten the valve to the line. Note that fl enters parametrically into the definition of ***flanged_pipe***, although this is not made explicit.

The two objects are combined using the operation *smooth_attach,* which attaches them non-penetratingly to each other mating two identical features. The operation also smoothes all edges that are the intersection of surfaces not belonging to the same original component. By convention,

the resulting object has as intrinsic coordinate system that of the first argument. The third parameter r to *smooth_attach* controls the placement of the tangency points (a,0) and (0,b). These points are placed such that the resultant surface approximates, in certain cross sections, a circle of radius r. The third parameter to *smooth* has the same meaning.

```
flanged_pipe(radius, thickness, length) :=
      smooth_attach(nipple.top.in_edge, sflange.bottom.in_edge, 1/8) where
   begin
      ft := 3*thickness+hole_diameter;
      flange := pipe(radius, ft, fl);
      sflange := smooth(flange.outside, flange.bottom, 1/16);
      nipple := pipe(radius, thickness, length);
      bottom := nipple.bottom;
      top := sflange.top
   end;
```

Two flanged pipes are needed: one as intake assembly, the other as outlet assembly. Both are derived from the same definition, but one of them is rotated by 180 degrees when attached to the valve chamber. A flanged pipe is also needed for the stem of the valve body, but as its cross section is an ovoid, new definitions are required.

We define first the ovoid shape, and then with it, the ovoid pipe. Here the system defined object *zcylinder*(r) is used, which is a solid cylinder of radius r whose axis is the z-axis. The ovoid is the intersection of three cylinders whose axes lie parallel in the x=0 plane, with all intersection edges smoothed out. The function *ymove* moves the axis of the *zcylinder* a prescribed distance in the y direction. Here, $r1$ and $r2$ are the cylinder radii, and $r3$ is the radius for smoothing edges. The operation ***smooth*** is used when combining objects by union or intersection and smoothes those edges not belonging to the same original component. The definition of *flanged_ovoid* is completely analogous to the definition of *flanged_pipe*, and is used to define the bonnet stem.

In defining the chamber of the valve body, a tee is constructed from two pipes of different shape. The through pipe, of circular shape, is defined with radius equal to the major radius of the ovoid, which is the cross section of the stem pipe. The pipes are positioned with the origin of the coordinate system at the intersection of the two pipe axes. The stem pipe rises in the positive direction of the z-axis, and the through pipe is in the direction of the y-axis. Both pipes have the same

wall thickness. In constructing the tee, we cut into *tpipe* an opening corresponding to the inside of the stem pipe. Similarly, the stem pipe is clipped at the bottom so as not to penetrate into the inside of *tpipe*. The resulting shapes are combined with a set union. Note that there is an overlap.

Combining the chamber and the bonnet stem is a simple *attach* due to the fact that we have an exact match of the corresponding cross sections. No smoothing is needed. The intake and outlet assemblies are connected by *join,* which has to construct an adaptive joint of prescribed length that connects two nonintersecting pipes of different wall thickness and inside diameter.

Finally, all remaining parameters are assigned in order to design a specific valve body. Alternatively, they could have remained partially unassigned when designing generically.

```
pipe(radius, thickness, length) := (cyl1 - cyl2) ∩ H1 ∩ H2 where
   begin
      cyl1 := ycylinder(radius+thickness);
      cyl2 := ycylinder(radius);
      H1 := {y ≥ 0};
      H2 := {y ≤ length};
      top := (cyl1 - cyl2) ∩ {y = length};
      bottom := (cyl1 - cyl2) ∩ {y = 0};
      outside := cyl1.surface ∩ H1 ∩ H2;
      top.in_edge := top ∩ cyl2;
      bottom.in_edge := bottom ∩ cyl2
   end;

flanged_pipe(radius, thickness, length) :=
      smooth_attach(nipple.top.in_edge, sflange.bottom.in_edge, 1/8) where
   begin
      ft := 3*thickness+hole_diameter;
      flange := pipe(radius, ft, fl);
      sflange := smooth(flange.outside, flange.bottom, 1/16);
      nipple := pipe(radius, thickness, length);
      bottom := nipple.bottom;
      top := sflange.top
   end;

ovoid(r1, r2, r3) := smooth(cyl1 ∩ cyl2 ∩ cyl3, r3) where
   begin
      cyl1 := ymove(-(4+9/16), zcylinder(r2));
      cyl2 := ymove(4+9/16, zcylinder(r2));
      cyl3 := zcylinder(r1);
   end;

ovoidpipe(r1, r2, r3, thickness, length) := ovoid1 - ovoid2 ∩ H1 ∩ H2 where
   begin
      t := thickness;
      ovoid1 := ovoid(r1+t, r2+t, r3+t);
      ovoid2 := ovoid(r1, r2, r3));
      H1 := {z ≥ 0};
      H2 := {z ≤ l};
      top := (ovoid1 - ovoid2) ∩ {z = length};
      bottom := (ovoid1 - ovoid2) ∩ {z = 0};
      outside := ovoid1.surface ∩ H1 ∩ H2;
      top.in_edge := top ∩ ovoid2;
      bottom.in_edge := bottom ∩ ovoid2
   end;

flanged_ovoid(r1, r2, r3, thickness, length) :=
      smooth_attach(nipple.top.in_edge, sflange.bottom.in_edge, 1/8) where
   begin
      ft := 3*thickness + hole_diameter;
      nipple := ovoidpipe(r1, r2, r3, thickness, length);
      flange := ovoidpipe(r1, r2, r3, ft, fl);
      sflange := smooth(flange.outside, flange.bottom, 1/16);
      bottom := nipple.bottom;
      top := sflange.top
   end;
```

```
intake := flanged_pipe(in_radius, thickness, in_length);

outlet := flanged_pipe(out_radius, thickness, out_length);

chamber(chamber_length, chamber_height) := smooth(clipped_tpipe ∪ clipped_stem, 1/8) where
   begin
      stem := ovoidpipe(r1, r2, r3, stem_thickness, chamber_heigth);
      tpipe := ymove(-chamber.length/2, pipe(r1, stem_thickness, chamber_length));
      ovoid1 := ovoid(r1, r2, r3) ∩ {z ≥ 0};
      pipe1 := ycylinder(r1);
      clipped_tpipe := tpipe - (tpipe ∩ ovoid1);
      clipped_stem := stem - (stem ∩ pipe1);
      top := stem.top;
      left := tpipe.bottom;
      right := tpipe.top
   end;

bonnet_stem := flanged_ovoid(r1, r2, r3, stem_thickness, stem_length);

center := attach(chamber.top, bonnet_stem.bottom);

assembly := B where
   begin
      A := join(center.left, intake.bottom, joint_length);
      B := join(A.center.right, outlet.bottom, joint_length)
   end;

fl := 7/8;
joint_length := 3/16;
in_radius := 2+1/16;
in_length := 3/4;
out_radius := 2+1/16;
out_length := 3/4;
thickness := 5/8;
hole_diameter := 9/16;
r1 := 2+7/8;
r2 := 6+3/8;
r3 := 1/4;
chamber_length := 5+5/8;
chamber_heigth := 3+3/4;
stem_thickness := 1/2;
stem_length :=1;
```

5. Conclusions

We have developed the mathematics needed to support automatic generation of connecting and smoothing surfaces in the design of objects. Not only is the derivation of these surfaces straightforward, but also the surfaces are of low algebraic degree.

A sample design is used to demonstrate the feasibility of using these surfaces as a basic tool to reduce design complexity and to free the designer from the burden of unnecessary detail. In conjunction with a disciplined use of parametrized design, our example also shows that textual languages can be used to modify shape extensively in a concise and convenient way: only the parameters of the valve body need to be changed in order to accommodate, for example, different line sizes, different wall thickness, or different overall dimensions of the valve.

Our design notation is a step towards the textual component of a comprehensive, user-friendly design language. It needs to be perfected and increased in scope as does the underlying mathematics. But a textual design language also should be complemented by a powerful, interactive graphics component, allowing the designer to add to or modify his design graphically or textually, as he sees fit. These issues need to be explored further.

References

1. Brown, C.M., PADL-2: A Technical Summary, IEEE Computer Graphics and Applications 2:2, March 1982, 69-84.

2. Boyce, J.W., and J.E. Gilchrist, GMSolid: Interactive Modeling for Design and Analysis of Solids, IEEE Computer Graphics and Applications 2:2, March 1982, 27-40.

3. Eastman, C.M. and K. Preiss, A Review of solid shape modelling based on integrity verification, Computer Aided Design 16:2, March 1984, 66-80.

4. The Mathlab Group, Laboratory for Computer Science, MIT MACSYMA Reference Manuel, version 10, printed by Symbolics, inc, 1983.

5. Meagher, D.J., Geometric Modelling Using Octree Encoding, Computer Graphics & Image Proc., 19:22, June 1982, 129-147.

6. Pogorelov, A.V., Differential Geometry, P. Noordhoff N.V., Groningen, Netherlands, 1957.

7. Requicha, A.A.G., and H.B. Voelcker, Solid Modeling: A Historical Summary and Contemporary Assessment, IEEE Computer Graphics and Applications, March 1982, 9-24.

8. Veenman, P., ROMULUS_The Design of a Geometric Modeller, in Geometric Modelling Seminar, W.A. Carter, ed., P-80-GM-01, CAM-1, Inc., Bournemouth, UK, Nov. 1979, 127-152.