Extensions: Extrapolation Methods for CAD

Hans J. Wolters
Client and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2000-37
March, 2000

CAD/CAM, solid modeling, B-Splines

Many operations within a solid modeling application, notably applying thickness (shelling) or blending edges (filleting), encounter difficulties during topology resolution. In order to create a solid object, certain faces have to be intersected but the geometry is such that no intersection curve can be computed. The solution is to "extend" one or both faces. This means that one has to extrapolate the underlying curves or surfaces. This operation causes instability since extrapolation is inherently an unstable process. An additional difficulty is the selection of a strategy to compute the extension amount. Furthermore, there are additional restrictions relating to continuity across the extension boundary. In this talk I will illustrate by examples some modeling situations where extensions are necessary. I present the methods currently used, and illustrate their advantages and disadvantages. Subsequently, I will demonstrate a solution for primitives such as cylinders, cones, spheres and tori. I will conclude by suggesting approaches which could avoid some of the current pitfalls.

Internal Accession Date Only

Extensions: Extrapolation Methods for CAD

Hans J. Wolters

Client and Media Systems Lab
Hewlett Packard Laboratories
1501 Page Mill Rd
Palo Alto, CA 94304

Abstract. Many operations within a solid modeling application, notably applying thickness (shelling) or blending edges (filleting), encounter difficulties during topology resolution. In order to create a solid object, certain faces have to be intersected but the geometry is such that no intersection curve can be computed. The solution is to "extend" one or both faces. This means that one has to extrapolate the underlying curves or surfaces. This operation causes instability since extrapolation is inherently an unstable process. An additional difficulty is the selection of a strategy to compute the extension amount. Furthermore, there are additional restrictions relating to continuity across the extension boundary. In this talk I will illustrate by examples some modeling situations where extensions are necessary. I present the methods currently used, and illustrate their advantages and disadvantages. Subsequently, I will demonstrate a solution for primitives such as cylinders, cones, spheres and tori. I will conclude by suggesting approaches which could avoid some of the current pitfalls.

§1. Introduction

At present almost all engineering design tasks are performed with the help of a CAD system or — more generally — mechanical design automation (MDA) software. Most all of the commercial modelling packages converged to certain standard respresentations. The geometry is represented as NURBS curves and surfaces, where truly rational representations are used only for primitives such as circular and elliptic arcs, cylinders, cones, tori and spheres. Most modelers also moved from strictly CSG representations to a hybrid model where the topology is expressed as a BRep, and the sequence of operations is stored in a CSG-like tree. As mentioned above, our focus is on solid modelling applications where it is essential to maintain a valid topological solid after each operation. We restrict ourselves here to the manifold setting.

In more technical terms we define a solid as a 3-manifold with a compact boundary which is consistently oriented. This allows us to include objects with finite surfaces without excluding objects with infinite volumes. The reader

who is unfamiliar with these concepts should consult a textbook on solid modelling such as [2,3] or the excellent survey [4].

The requirement that the validity of the solid be maintained after any operation is a source for many of the robustness issues encountered in solid modelling. For example, it is often difficult to find a crisp intersection between surfaces even though the intersection curve is needed to close the solid. In this article I will focus on the problem of extensions. In a nutshell, we often need to extend surface patches in order to create intersections. This extension operator is equivalent to performing extrapolation. Very little information can be found in the literature regarding this topic; the notable exception is [5]. This survey article is meant to fill the void.

The outline of this article is as follows. In section 2, I will describe three operations which almost always lead to the need for extending curves and surfaces. Section 3 will present the approaches used in practice and discuss their advantages and disadvantages. Furthermore, I will describe one modification which leads to significant improvements when extending quadratic Bézier patches such as cylinders, cones, spheres and tori. In Section 4 we will suggest alternative approaches to circumvent the need for extensions and encourage some future work.

§2. Extensions in Solid Modeling

The need for extensions arises quite frequently when modeling parts. I will explain the need for extensions when performing three of the most common operations, namely

- Shelling
- Blending
- Drafting

These three operators are local operators, meaning that only a region of the solid is modified. Shelling is the process of applying thickness to a part. The steps to be performed are as follows: The intial step is to offset all the surfaces with prescribed offset distance, the thickness. This distance can vary considerably. The result is illustrated in Figure 1.

In step 2, surfaces need to be intersected to form edges. Here extensions might be needed in order to compute crisp surface intersections. Trimming back the surfaces in Step 3 yields the final result. The alert reader might have noticed that the true offset is the Minkowski sum, and vertices should really correspond to arcs. This would avoid computing extensions all together. However, this result is not desired in practice.

Blending or filleting is the process of rounding sharp edges. Hereby an additional face is constructed which meets the adjoining faces with G^1 continuity as illustrated in Figure 2. This surface is typically constructed as a loft interpolating circular or elliptic cross sections. Extensions are needed for vertex resolution when multiple filleted edges meet at a corner or for extending features when the blend face interferes with an existing feature. For more information on blending, see the survey article by Varady et al. [6].

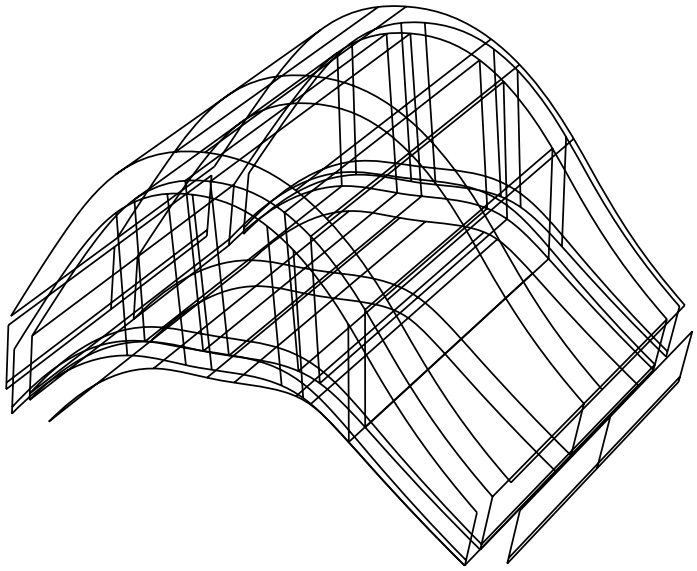

Fig. 1. Offsetting outward results in free edges which need to be resolved.

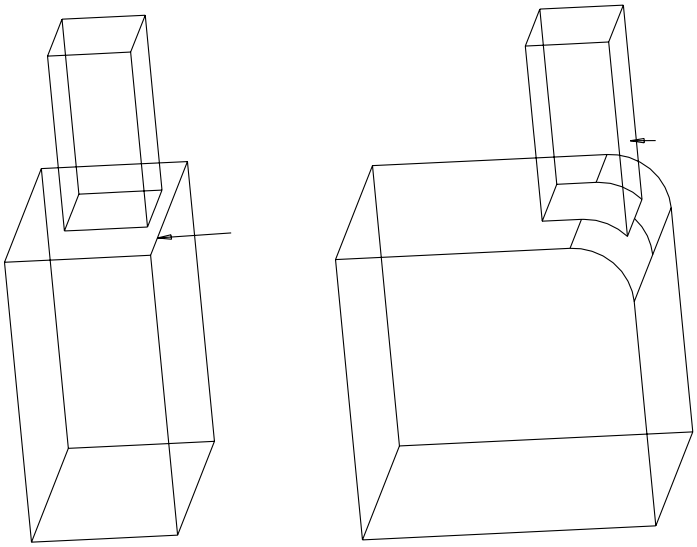

Fig. 2. Blending with features present: The edge denoted by an arrow in the left figure is filleted. The face denoted by the arrow in the right figure needs to be extended.

A draft operation in solid modeling consists of changing the solid such that certain faces are slightly angled, see Figure 3. This is necessary for plastic parts manufactured by injection molding. In order to be able to pull the part out of the mold, there needs to be some room such that the faces do not stick to the mold wall. Hence this operation is mandated purely for manufacturability. Extensions are needed here as well as Figure 3 illustrates.

In conclusion, one can see that the success of these operations depends on being able to produce the geometry which is required for the successful resolution of the topology. Specifically, this means that one has to be able

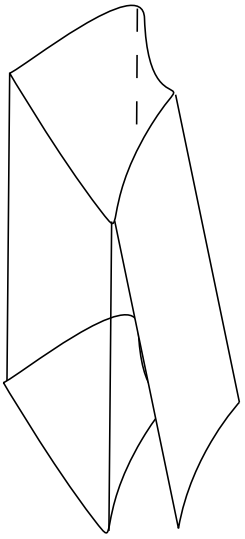

Fig. 3. Draft: The angled face replaces the front face and, hence the bottom face needs to be extended.

to produce intersection curves. The vast majority of failures can be traced to the failure of producing these curves due to bad geometry generated by extensions.

§3. Extension Methods

In this section we will survey the extension methods which are typically employed in commercial systems, and we will evaluate their strengths and weaknesses. Additionally, we will present a modification which allows to extend quadric surfaces while maintaining their current parametrization. Extensions need to fulfill certain requirements to be useful. Special surfaces such as cylinders, cones, spheres, tori should be maintained. Ideally the existing degree of continuity across the extension boundary should be kept as well; however, this requirement is mostly relaxed and only G^1 continuity is required. The shape of the resulting surface should be predictable, and the extensions should result in well-defined surface intersections. Note that the requirements differ in one crucial point: the first two requirements can be enforced, whereas the last two can not when using extrapolation based methods. We will revisit this topic in Section 4.

Subsequently, we assume as given a B-Spline surface $\boldsymbol{s}(u,v)$ of degree d with control points $\boldsymbol{s}_{ji}, i = 1,\ldots,n, j = 1,\ldots,m$ and knot vectors $\mathbf{u}$ and $\mathbf{v}$. We assume the parameter domain of the surface to be $[a_1,b_1]\times[a_2,b_2]$. We assume that we want to extend across the boundary $u = b_1$ such that the new bound is $\hat{u}$.

Natural Extension

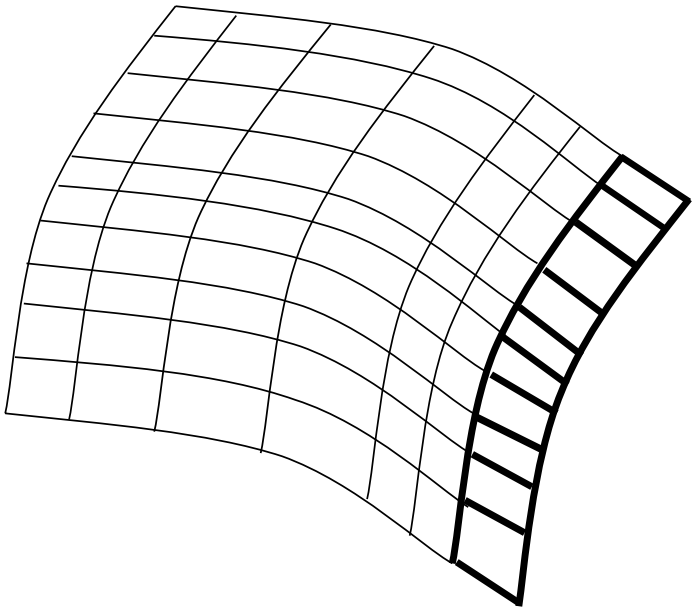

Fig. 4. Construction of linear extensions.

The natural extension approach is the straightforward extrapolation approach in a B-Spline or Bézier setting. Consider subdivision of a given Bézier curve $\boldsymbol{c}(t)$ with $t \in [a, b]$. It is known that the control points for any subcurve $\boldsymbol{d}(t)$ with $t \in [a, s]$ are given by the intermediate points of the de Casteljau algorithm:

$$\boldsymbol{d}_i = \boldsymbol{c}_0^i(s).$$

Of course the formula is still valid if s is lying outside the original parameter interval, here $[a, b]$. Natural extension is performed by applying this formula with a value $\hat{s} \notin [a, b]$. This results in extrapolation. Furthermore, the control points are not computed as convex combinations of the previous layer control points as before; hence attributes like the convex hull property are lost. The derivation presented here directly generalizes to the setting we are considering. In the B-Spline case the control points are the intermediate points generated by the de Boor algorithm with the new extension parameter $\hat{u}$. Pseudocode on how to compute these intermediate points can be found in the book by Farin [1]. We can easily extend this method to surfaces by repeatedly applying the curve algorithm to rows or columns of control points. It is worthwhile to point out that this method is inherently very unstable. However, it is often used in practice even though it should be avoided.

Linear extension

The simplest form of extension is using the derivatives across the extension boundary to infer the new geometry, see Figure 4.

This method is known as linear extension. In our case we extend each row of control points linearly in the direction given by

$$\boldsymbol{v}_i = \boldsymbol{d}_{n,i} - \boldsymbol{d}_{n-1,i}.$$

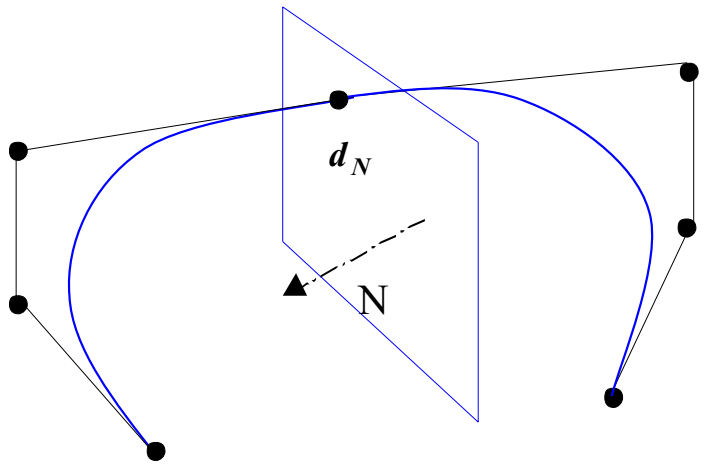

Fig. 5. Reflection of control points.

Note that now one additional segment is being created, and hence we increase the number of control points in each row. Typically, we have the freedom to achieve C^1 continuity by appropriate scaling. If we define

$$\alpha = \frac{\hat{u} - u_{n+d+1}}{u_{n+d+1} - u_{n+d}},$$

then by setting

$$\boldsymbol{d}_{n+1,i} = \boldsymbol{d}_{n,i} + \alpha \boldsymbol{v}_i,$$

we have achieved C_1 continuity at $u^* = u_{n+d+1}$ as can readily be verified. If we denote as $\boldsymbol{D}_+$ the partial of $\boldsymbol{s}$ at u^* computed from the (new) right segment and with $\boldsymbol{D}_-$ the corresponding partial computed from the left segment, we derive:

$$\boldsymbol{D}_+ = \frac{\boldsymbol{d}_{n+1,i} - \boldsymbol{d}_{n,i}}{\hat{u} - u_{n+d+1}} = \frac{\boldsymbol{v}_i}{\hat{u} - u_{n+d+1}} \frac{\hat{u} - u_{n+d+1}}{u_{n+d+1} - u_{n+d}} = \frac{\boldsymbol{v}_i}{u_{n+d+1} - u_{n+d}} = \boldsymbol{D}_-.$$

The other control points are usually placed equidistantly on the tangent line. Again rational surfaces are treated in homogeneous space, and it is possible that negative weights are created. One can remedy this by inserting knots appropriately.

Reflection Extension

This method has been introduced in [5]. The idea here is that more predictable results can be obtained by just mirroring the existing geometry across the normal plane. This is shown in Figure 5.

The basic operation of reflecting control points suffices when dealing with nonrational surfaces. The basic reflection operation can be formalized as follows: We again define $\boldsymbol{v}_i$ as in the case for linear extensions. Let us denote by $\hat{\boldsymbol{v}}_i$ the normalized vector. Then we have

$$\boldsymbol{d}_{n+j,i} = \boldsymbol{d}_{n-j,i} - 2 < \hat{\boldsymbol{v}}_i, \boldsymbol{d}_{n-j,i} - \boldsymbol{d}_{n,i} > \hat{\boldsymbol{v}}_i, \quad j = 1, \ldots, K,$$

where K is the number of new control points to be computed. It can be easily seen that G^1 continuity is preserved across the boundary $u = b$. A more subtle point to consider is continuity in v across a knot v_l with full multiplicity. If the original surface is C^1 continuous across v_l, we would like to ensure that the new part of the surface fulfills that condition as well. Control points generated by the simple extension equation above will *not* inherit C^1 continuity from the generating geometry. This is due to the fact that the normalization introduces a nonlinearity into the reflection formula. Only if we have the same reflection plane for the three rows affected is C^1 continuity across v_l achievable. G^1 continuity is achievable if one chooses the weight functions in the continuity equations appropriately. In the case of rational surfaces, more work is required in any case. The continuity conditions for adjacent rational patches are somewhat complex: Suitable weights and scalar values have to be computed by inserting the model space control points into the equations for G^1 continuity as stated in [5]. In addition, it is now not clear that G^1 continuity across a knot v_l with full multiplicity can be obtained. The authors in [5] ignored the complications arising by this configuration. Note that this problem is closely related to twist incompatibility issues when one is computing the new corner points.

Extending special surfaces

In most CAD systems, rational surfaces are only used to represent surfaces such as cones, cylinders, spheres and tori. In this case, one can create an extension surface which is again a special patch by solving a simple system of equations without invoking the machinery of rational continuity conditions across boundaries: We make use of the fact that we are dealing with quadratic Bezier patches. So let us assume that we are given a biquadratic patch $\boldsymbol{p}(u, v)$ which again shall be extended across $u = b$. Let us denote the new patch by $\boldsymbol{q}$. The boundary control points are generated by simple reflection in model space. The weights are just copied, hence we guarantee positive weights. It remains to generate the point $\boldsymbol{q}_{11}$.We make use of the fact that the two end derivatives of the isoparametric curve formed by $\boldsymbol{q}_{01}$,$\boldsymbol{q}_{11}$, and $\boldsymbol{q}_{21}$ are the same up to a scale factor as the corresponding derivatives of the curve denoted by $\boldsymbol{p}_{01}$,$\boldsymbol{p}_{11}$, and $\boldsymbol{p}_{21}$. Furthermore, the scale factor λ is identical for both derivatives. This gives rise to a simple system of 6 equations in 4 unknowns: Denote by

$$\boldsymbol{D}_0 := \frac{\partial \boldsymbol{p}}{\partial u}(u = a, v)\frac{w_{01}}{w_{11}}\frac{u_4 - u_1}{2},$$
$$\boldsymbol{D}_1 := \frac{\partial \boldsymbol{p}}{\partial u}(u = b, v)\frac{w_{21}}{w_{11}}\frac{u_4 - u_1}{2}.$$

Then we obtain the equations

$$\lambda \boldsymbol{D}_0 - \boldsymbol{q}_{11} = -\boldsymbol{q}_{01},$$
$$\lambda \boldsymbol{D}_1 + \boldsymbol{q}_{11} = \boldsymbol{q}_{21}.$$

We know that a solution must exist, and by looking at the system, we can readily determine λ - for example by adding equation 4 to equation 1. Having determined λ, $\boldsymbol{q}_{11}$ follows trivially.

Hence we have presented an approach to compute the correct control points in the circular direction of a special surface such as cone, cylinder, sphere or torus.

Summary

We have presented the three commonly used extension methods. Let us summarize the advantages and disadvantages of each: Starting with the natural extension, its advantages are that maximal continuity is preserved and special surfaces retain their characteristic. The disadvantages are that depending on the original surface parameterization the results can be undesirable even when only extending a relatively small amount. Furthermore, since extension is performed in homogeneous space, weights can easily become negative even for special surfaces.

Linear extension is the most predictable method in that it resembles a ruled segment that joins the original surface with G^1 or C^1 continuity. However special surfaces are not preserved, and again we might produce negative weights when dealing with rational surfaces.

Reflection extensions yield positive weights for rational surfaces. The resulting surface is related to the original surface in a predictable fashion, at least for a modest extension amount. It is possible to create G^2 continuous surfaces across the extension boundary. However, one might lose continuity in the other direction when knots with full multiplicity are present.

The combination scheme derived above combines reflection extension with a direct computation of inner Bézier points. This method has been developed especially for the extension of rational quadratic Bézier patches, and hence it preserves special surfaces without introducing negative weights.

Implementation Issues

In order to implement a topology resolution system based on extensions, there are some other complicating factors to consider. First, the amount of extension has to be determined. In general this amount is given in parameter space. Depending on the parameterization, parameter space and model space might not correspond well. As a consequence, it is difficult to even predict the extension amount in model space without careful analysis of the given parameterization. Usually one needs to perform extensions in a loop by ways of callbacks. The process flow is as follows:

1) extend

2) test for intersection

3) if intersection found then process, else goto 1)

Of course, it is necessary to monitor this iteration. When finding an intersection curve requires extensions of significant amount, it is likely that

the result is not acceptable. This is particularly true for shelling operations. When dealing with trimmed surfaces, the desired result might differ: In some cases it might be valid to extend the untrimmed surface; in other cases the trimming information must be preserved.

§4. Alternatives

We have seen that extension approaches are the weak link in topology resolution algorithms. Inherently, the problem of extrapolation is ill-defined. An alternative solution worth exploring is to reverse the process and to establish the desired intersection curve first. Given this intersection curve and optionally a tangent ribbon, one can construct a cubic Bézier patch blending between the intersection curve and the extension boundary and tangents constructing a G^1 continuous extension. Variations of this approach are possible as well: one might even prescribe four boundary curves and construct the extension patch by Coons blending techniques. If one needs to establish a vertex by performing multiple intersections, one could again establish intersection curves first, and then perform additional intersections to establish the vertex. To the authors knowledge, such an approach is currently not implemented in any commercial modeler.

§5. Conclusion

This survey article presented the methods used for extending curves and surfaces. Since they are all based on extrapolation, the algorithms are unstable and can lead to undesirable results. As a consequence, the topology cannot be resolved, and a valid solid cannot be produced. This leads to the failure of the entire local topology operation such as shelling, blending or drafting and loss of productivity for the end user who typically has to perform time-consuming steps to get the desired result. We have shown that all the algorithms have inherent weaknesses, and we have put forward a suggestion for alternative approaches. It is the authors hope that this paper motivates some much-needed further work in this area.

Acknowledgments. I wish to thank my former colleague Tim Strotman at SDRC for many fruitful discussions, and for helping me with his knowledge. Thanks to Sreedharan Vijayan for generating Figure 2.

References

1. Farin, G. E., *Curves and Surfaces for CAGD*, Academic Press, San Diego, 1996.
2. Hoffmann, C. M., *Geometric and Solid Modeling*, Morgan Kaufmann, San Mateo, 1989.
3. Mäntylä, M., *An Introduction to Solid Modeling*, Computer Science Press, Rockville, 1988.

4. Requicha, A. G., and Rossignac, J. R., Solid Modeling and Beyond, IEEE Computer Graphics and Applications, **12**, Nr. 5, (1992), 31–44.
5. Shetty, S., and White, P. R., Curvature continuous extensions for rational B-Spline curves and surfaces, Computer Aided Design, **23** (1991) 484–491.
6. Vida, J., Martin, R. and Varady, T., A survey of blending methods that use parametric surfaces, Computer Aided Design, **26** 1994 341–365.