Soft Computing manuscript No.
(will be inserted by the editor)

Managing uncertainty and discontinuous condition numbers in geometric computation

Peihui Shao · Neil F. Stewart

Revised version: August 8, 2012.
Received: date / Accepted: date

Abstract We consider operations on subdivision surfaces under the strict robustness requirement that these floating-point computations return an object with the same topological form as the true solution. The problems involved may however be ill-conditioned, and defined in terms of uncertain data, and even supplementary interval arithmetic may not ensure robustness. Trapping mechanisms are therefore proposed to resolve this difficulty.

Keywords Uncertainty · robustness · geometric computation · finite-precision arithmetic · interval arithmetic · subdivision surfaces

1 Introduction

The robustness problem in geometric modelling has been studied over a period of decades. It raises many interesting theoretical questions (Hoffmann and Stewart 2005; Yap 2008), and it is also of great practical importance, since the cost of unreliable computation in this application area is measured in billions of dollars (Brunnermeier and Martin 1999; Farouki 1999).

The work of the second author was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

Peihui Shao
Dép't. IRO, Univ. de Montréal
CP6128, Succ. CentreVille
Montréal, H3C 3J7, Canada
E-mail: shaopeih@iro.umontreal.ca

Neil F. Stewart
Dép't. IRO, Univ. de Montréal
CP6128, Succ. CentreVille
Montréal, H3C 3J7, Canada
E-mail: stewart@iro.umontreal.ca

In the context of computation using ordinary IEEE arithmetic (IEEE 2008), the problem has traditionally been studied as one of error control. Unfortunately, most analyses carried out have neglected to include certain of the fundamental ideas of error analysis (Dahlquist and Björck 2008), such as uncertainty in the data, and the condition of the underlying problem. In fact, even a clear definition of a criterion to measure error is often lacking (Hoffmann and Stewart 2005; Andersson et al. 2007). If the problem is reformulated to take these ideas into account, with an error criterion that requires correct topological form, then it immediately becomes clear that the problem may be ill-conditioned, and the question of practical concern becomes one of managing uncertainty. This uncertainty arises, for example, out of uncertainty in the input data, storage and roundoff error due to the use of finite-precision floating-point arithmetic, and (for certain methods of representing geometric objects) the approximation of high-degree polynomials by polynomials of low degree. Given the practical question just mentioned, it is of interest to formulate appropriate algorithms for the management of the uncertainty in the input to geometric algorithms, and management of uncertainty that may be created by such algorithms.

In this paper we describe an interactive approach for the robust implementation of geometric operations on objects defined by subdivision surfaces, in the case when we require correct topological form of the result, when there is uncertainty in the data, and when the computations are done using ordinary IEEE floating-point arithmetic, with supplementary use of interval arithmetic (Moore et al. 2009; CXSC 2012; MPFI 2012; P1788 2012). It is shown that in this context, an interactive approach is forced upon us: for the implementation

to provide correct results, it is necessary that a higher-level process provide (on request) information about the topological form of the result. This is done by having the algorithm throw an exception, to be caught by a higher-level process, when supplementary topological information is required.

Since the boundaries of subdivision-surface objects can vary in an almost arbitrary manner, it may seem unduly optimistic, at first glance, to seek rigorously correct topological form. The reason we can hope to do this, of course, is that we permit the algorithm to trap and request supplementary information. We do not, however, want the algorithm to do this indiscriminately. The goal is then to find an algorithm that will request supplementary information only when it is demonstrably impossible to proceed in any other way.

We are ultimately interested in applying our approach to the general problem of computing Boolean operations on geometric objects defined by subdivision-surface meshes, but in order to study the main idea without having it obscured by complicated details, we consider a simpler problem. Specifically, we give an algorithm to compute the planar cut of an object defined by a subdivision-surface mesh—more precisely, the object defined by Loop subdivision (Loop 1987; Andersson and Stewart 2010). The planar-cut problem arises frequently in practice (Grandine 2000), it is far from trivial, and yet it is simple to visualize (see Figure 1, which comes from (CGAL 2012)). The algorithm uses an adaptive version of Loop subdivision, based on RG (Red-Green) triangulation (Meister and Struckmeier 2002, p. 180), and a multi-step rule proposed in (Puppo and Panozzo 2009, Sec. VI-A).

In order to avoid possible confusion, we distinguish between the often discussed problem of interactive editing of subdivision surfaces (Zorin et al. 1997; Amresh et al. 2002; Chen et al. 2007), and the problem discussed here. Interactive editing is a process for specifying surfaces, while the algorithms we discuss are applied to surfaces that have already been defined, except possibly for uncertainty in their defining data. The interactive approach we propose refers to algorithms for Boolean operations, or for the solution of the planar-cut problem, and this interactivity has nothing to do with interactive surface editing.

Algorithms for Boolean operations on geometric objects defined by subdivision surfaces have been proposed previously in the literature, including for example the surface-based approach of (Biermann et al. 2001), and the voxel-based approach of (Lai and Cheng 2007). Another reference, for the general problem of surface intersections, is (Patrikalakis and Maekawa 2002). Our algorithm can be viewed as a version of the one in (Biermann et al. 2001), simplified because we restrict our attention to Boolean intersection, in the case when one of the two objects to be intersected is a half space. As noted in the next paragraph, the actual algorithm used is not the subject of the paper. The main point is how to deal with the uncertainty in the input data, and with the requirement of correct topological form (which is in contrast to (Biermann et al. 2001), where it is only required that the solution be topologically valid). For sameness of topological form we take the classical requirement that there should be a homeomorphism linking the true and computed solutions: see the discussion in (Andersson et al. 1995).

One of the advantages of using a subdivision-surface representation, rather than the traditional trimmed-NURBS representation, is that many of the difficulties of dealing with inconsistent data (Andersson et al. 2007) are avoided. This is true in particular for the analysis of error related to the approximation of high-degree polynomials by polynomials of low degree, mentioned above.

In Section 2 of the paper it is observed that many geometric problems, such as the problem of Boolean intersection, are problems for which the condition number is discontinuous, but for which it is useful to continue the solution process through the discontinuity. Then, in Section 3, it is observed that the cutting-plane problem (which is closely related to a very special case of Boolean intersection) is also such a problem, and we give an algorithm to solve it. The geometric methods underlying the presented algorithm are not the subject of the paper: in fact our algorithm relies almost exclusively on ideas previously presented in the literature (Loop 1987; Meister and Struckmeier 2002; Puppo and Panozzo 2009; Wu 2005; Wu and Peters 2004; Hohmeyer 1991; Piegl and Tiller 1995). The main contribution of the paper is in Section 4, where we discuss the idea of an exception mechanism to deal with the possible ambiguity of the topological form of the result. In particular, we show that the interactive approach is forced upon us. If the algorithm throws an exception, and asks for help in determining the topological form, it is usually because a small perturbation of the input data could lead to a change in the topological form of the solution, and our goal is to find algorithms which make such a request for supplementary topological information *only* when it is necessary. Section 5 concludes the paper, with some remarks on our prototype implementation, and some remarks on future work.

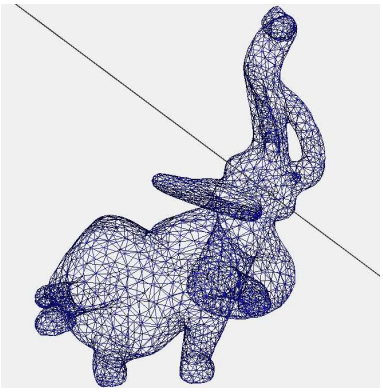

Fig. 1 Intersection of a plane and a locally-planar mesh.

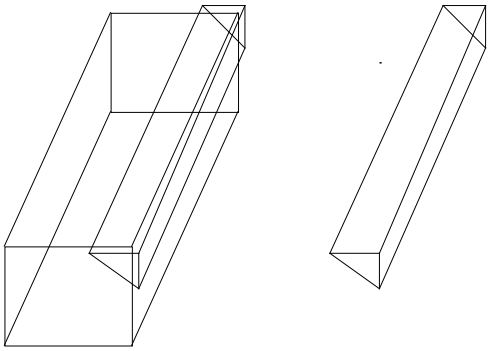

Fig. 2 Bevel specified by a Boolean difference.

2 Uncertainty and condition numbers

In the context of Boolean operations on geometric objects, in the presence of uncertainty, and with a requirement of correct topological form, a mechanism for interaction with a higher-level process is unavoidable.

Consider for example the problem of specifying a bevel in a feature-based solid-modelling system (Spitz and Rappoport 2004). The semantics of such specifications rely on the use of Boolean operations: for example, in Figure 2, the bevel is specified as the set difference between the block and the narrow wedge-shaped object. Both of these objects are defined with uncertainty, possibly resulting from previous operations that used finite-precision arithmetic (say, a rigid motion), or from some other source. Calculation of the set difference can easily lead to an object with topological form different from that of the correct result, and the correct topological form must therefore be imposed by a higher-level process (in the present example, by a "persistent-naming" mechanism, which notifies the algorithm of overlaps in the input objects (Spitz and Rappoport 2004)).

Other examples can easily be given (Andersson et al. 2007). For example, if a flexible springclip is to be joined to an object at one end of the clip, but left separated from the object at its opposite end, and if again there is uncertainty in the defining data, then it will be impossible using finite-precision arithmetic to resolve the topological form without supplementary information.

Since there may be other sources of uncertainty in the data defining the problem, besides those related to preliminary use of floating-point arithmetic, the level of data uncertainty will often be much larger than floating-point roundoff error. An example of another source of uncertainty is the interactive input of objects on a screen. This example also illustrates another possible "higher-level process", namely a procedure for the interactive entry of objects. In this case the correct topological form would be obtained from the human user.

These ideas can be placed in the context of a standard error analysis (Dahlquist and Björck 2008) by viewing the error as infinite if the topological form is incorrect. This means that the computational problem is ill-conditioned for inputs for which small changes in the input data can lead to a change in topological form, and the condition number therefore has an infinite discontinuity at data points corresponding to these inputs. On the other hand, we wish to continue the computational process through the discontinuity, *i.e.*, we do not want to abandon the computation in the case of such ill-condition. External intervention to resolve the ambiguity is consequently necessary. Our criterion for a good algorithm will be that external intervention will be requested only when it is necessary, *i.e.*, only when small perturbations of the uncertain input data could lead to a change in topological form. We seek to define algorithms that meet this criterion as nearly as possible.

3 Planar cut of a locally-planar mesh

A (triangular) *logical mesh* M is a finite collection of faces, each defined by an unordered set of three vertices $\{\ell_0, \ell_1, \ell_2\}$, along with their associated edge sets $\{\{\ell_0, \ell_1\}, \{\ell_1, \ell_2\}, \{\ell_2, \ell_0\}\}$ ((Andersson and Stewart 2010, p. 10). The logical mesh is *locally planar* if each edge belongs to at most two faces and if, for any vertex ℓ, the j faces ϕ_i incident at ℓ $(i = 0, \ldots, j-1)$ can be ordered in such a way that ϕ_i meets ϕ_{i+1} at an edge containing ℓ for $i = 0, 1, \ldots, j-2$. The locally-planar mesh is in addition a *mesh without boundary* if, for each vertex ℓ, the corresponding ordered faces satisfy the condition that ϕ_{j-1} meets ϕ_0 along an edge. Note that "locally planar" does not mean geometric

planarity: no geometric information is specified in a logical mesh. The valence of vertex ℓ, by definition equal to the number of edges incident at ℓ, is denoted by n.

If we now proceed to associate with each vertex ℓ ($\ell = 0, \ldots, L-1$) a control point $p_\ell \in \mathbb{R}^3$, and collect the p_ℓ (written as 1×3 row vectors) in an $L \times 3$ matrix

$$p = \begin{bmatrix} p_0 \\ \vdots \\ p_{L-1} \end{bmatrix}_{(L\times 3)}, \tag{1}$$

then $\mathcal{M} = (M, p)$ is called a *polyhedral mesh* (Andersson and Stewart 2010, p. 15). This is the geometric mesh in $\mathbb{R}^3$. An example of a connected, triangular, locally-planar polyhedral mesh without boundary is illustrated in wire-frame format in Figure 1, above.

In this paper we restrict our attention to (finite) locally planar meshes without boundary. The hypothesis of local planarity is implicit in most discussions of subdivision surfaces, although it is usually not explicitly specified. More general subdivision schemes involving meshes that are not necessarily locally planar have been studied, for example in (Ying and Zorin 2001).

Under weak conditions, application of the Loop subdivision method (Loop 1987; Andersson and Stewart 2010), to a locally-planar triangular mesh without boundary, converges uniformly to a smooth limit surface $S = S(u, v)$, which can be para- metrized locally by two variables u and v. If the control point p_ℓ corresponding to each vertex of each triangle in the mesh is associated with its limit position, the triangles imply a decomposition of S into triangular curvilinear patches with the limit positions at the patches' corners (Andersson and Stewart 2010, p. 111). Further, if the vertices of the triangle are regular ($n = 6$), then the patch can be expressed as a polynomial over the triangle (in fact, it is a quartic Bézier surface (Schweitzer 1996; Lai 1992)). Even at extraordinary (non-regular) vertices, the surface is C^1.

Let the normal vector $n_c \in \mathbb{R}^3$ and the scalar σ define the cutting plane $P_c = \{x \in \mathbb{R}^3 : n_c \cdot x = \sigma\}$. The *contouring problem* (Grandine 2000) can be defined as computing the intersection between the plane P_c and the surface S. In fact, however, we will study a slightly more elaborate problem, namely, the *planar-cut problem*. For simplicity, we assume that the initial mesh is connected, so that the corresponding limit surface is a connected set, and we assume that the surface does not selfintersect. (Non-selfintersection of single patches, and non-intersection of both adjacent and non-adjacent triangular surface patches, can actually be checked computationally using the methods of (Andersson et al. 1998; Grinspun and Schröder 2001).) Let $Int(S)$ be the finite region in $\mathbb{R}^3$ defined by S. The planar-cut problem involves the computation of the boundary B of the intersection between $Int(S)$ and the half space $H_c = \{x \in \mathbb{R}^3 : n_c \cdot x \geq \sigma\}$. These are sets in $\mathbb{R}^3$: they are illustrated in Figure 3. In Figure 1, which shows an approximation to a surface, if the normal n_c defining the cutting plane (indicated by a line in the figure) points downward and to the left, then the trunk and the top of the head of the elephant are cut off, and the resulting opening is covered by a planar surface.

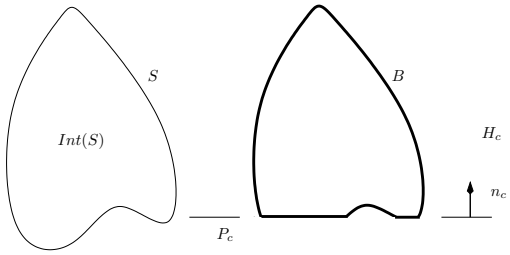

Fig. 3 *Left:* cross-sections of S and $Int(S)$; *Right:* cross-section of boundary B (thick line).

As already mentioned, Loop subdivision takes a triangular mesh and produces a sequence of refined triangular polyhedral meshes that converge uniformly to the surface S (Andersson and Stewart 2010). Note that the limit surface is completely determined by the control points in the given mesh, and the fact that Loop subdivision is used. The adaptive version of Loop subdivision described in Section 4.1, based on Red-Green (RG) triangulation, changes the approximating meshes that are calculated as the process proceeds, but it does not change the limit surface.

The planar-cut problem is defined by the initial mesh $\mathcal{M}^0$, the vector n_c, and the scalar σ; an admissible approximate solution to the problem is a well-formed (conformal) triangular mesh which is *topologically the same as* the boundary B of $Int(S) \cap H_c$. Since small perturbations of the geometric input data (the control points p_ℓ of the initial mesh $\mathcal{M}^0$) may cause the topological form of B to change, the problem is ill conditioned. For example, in Figure 4, if all of the control points p_ℓ undergo an identical translation, upward and away from the plane $P_c = \{x \in \mathbb{R}^3 : n_c \cdot x = \sigma\}$, as indicated by the vertical arrow in the figure, the topological form of B will change. This follows because the surface will be translated by an amount equal to the translation in the control points, due to the affine invariance of the subdivision process (Andersson and Stewart 2010, p. 39).

An approximate and high-level description of our algorithm is given now (details appear in the next section of the paper). The initial mesh $\mathcal{M}^0$ is assumed given. In a first phase, triangles in $\mathcal{M}^0$ near the plane P_c are refined using RG triangulation, until a desired

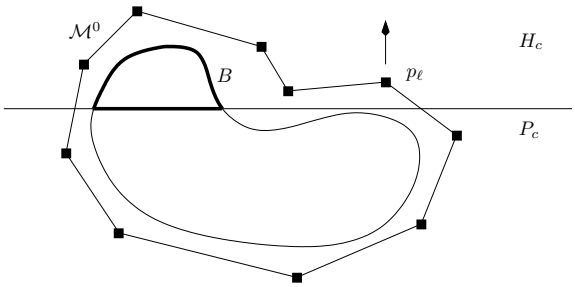

Fig. 4 Cross-section of B: risk of change of topological form.

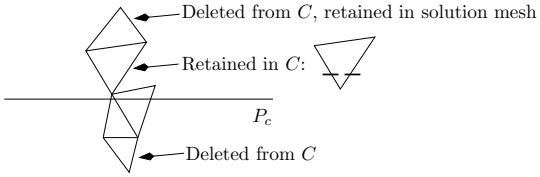

Fig. 5 Labels on triangles in C.

error criterion, based on distance between triangle vertices and the plane P_c, in the direction n_c, is satisfied. Nearness to P_c is determined by using the Wu-Peters bounding box (Wu 2005). An important side effect of this first phase is the creation of a list C (the notation is intended to suggest "close") of triangles which, according to the Wu-Peters bounding-box, might possibly have an associated triangular surface patch that intersects P_c. Although Red-Green triangulation is not described until Section 4.1, below, we remark here that green triangles are treated as pairs, with the Wu-Peters test applied to the red mother triangle. The red-green mesh obtained at the end of this initial phase is denoted $\mathcal{M}^{\nu_0}$.

A triangle is said to generate a simple intersection if the intersection between its triangular surface patch and P_c consists of a single curve cutting across the patch. (The curve must not pass through a corner of the triangular surface patch.) In the second phase, the algorithm processes the triangles in C, subdividing $\mathcal{M}^{\nu_0}$ if necessary, replacing triangles that do not generate a simple intersection.

If subdivision occurs, the set C is updated. Subdivision may result in smaller triangles for which the associated triangular surface patch cannot intersect P_c. These triangles are deleted from C. Triangles generating a simple intersection are labelled to indicate the pair of edges joined by a curve which forms the intersection between the triangular surface patch and P_c. There are three cases, as illustrated in Figure 5. In the figure we have deliberately shown a triangle, with labels specifying intersections of edges, that does not quite intersect P_c. We do this to emphasize that that it is the intersections associated with the associated triangular surface patch that are relevant, and not the intersections with the triangle itself.

In a third phase, the intersection path, between the mesh and P_c, is traced, using the elements of C. (The reason for separating the second and third phases is that the process of elimination of non-simple intersections can in principle lead to subdivision of neighbouring triangles. If the second and third phases were done together, for each triangle, the intersection curve of the neighbouring triangle might already have been traced, and this information would have to be transferred to the children of the triangle. In the context of red-green subdivision, this is not a straightforward process.)

The fourth and final phase of the algorithm is to merge the modified mesh with a conformal mesh approximating the planar set $Int(S) \cap P_c$. This conformal planar mesh is computed using constrained Delaunay triangulation (Shewchuk 2000). The merged mesh is an admissible approximate solution in the sense defined above.

Some further remarks should be made here, concerning the second phase of the algorithm. For the planar-cut problem, to specify the topological form of the result it is necessary and sufficient to specify the topological form of $S \cap P_c$. The topological form of $\mathcal{M}^{\nu_0} \cap P_c$, however, may not be the same as that of $S \cap P_c$. Consequently, to obtain an admissible approximate solution, it may be necessary to modify $\mathcal{M}^{\nu_0}$, by subdivision, in order to find an equivalent mesh $\mathcal{M}^{\nu}$ (one with the same limit surface S) for which the topological form of $\mathcal{M}^{\nu} \cap P_c$ is the same as that of $S \cap P_c$. This task is essentially accomplished in the second phase, by decomposing the intersecting part of the mesh into triangles with simple intersections.

It may happen, however, that even after repeated subdivision, such a decomposition cannot be found, so that it is not possible to determine the topological form of $S \cap P_c$. If the subdivision reaches a triangle size that is on the order of the data uncertainty, the algorithm traps and asks to be informed of the topological form of $S \cap P_c$. Thus, if the algorithm traps, there exists a perturbation of S, approximately equal to the size of the data uncertainty, which could change the topological form. Because of the subdivision, however, this does not necessarily mean that a perturbation of the *original input data* could lead to a change in topological form. Restating our previously stated goal, we would like to establish weak conditions guaranteeing that the algorithm will trap and ask for additional information only in the case when a perturbation of the original data, approximately equal in size to the data uncertainty, could

lead to a change in topological form. This question is discussed further in Section 5.

4 The uncertainty-management process

In this section we present the techniques used by our algorithm to take account of the inherent uncertainty, as outlined at the end of the previous section. We begin in Section 4.1 by describing the basic subdivision process used. We then present certain fundamental techniques used in the algorithm, including the Wu-Peters bounding box, a specialization of the Hohmeyer test for detection of loop intersections, and the Variation Diminishing Property for Bézier curves. Finally, the algorithm to trace the graph defining the topology of $S \cap P_c$ is described in Section 4.6.

4.1 Red-Green Loop triangulation

Our algorithm uses the method of RG triangulation (Meister and Struckmeier 2002). Our implementation of this method uses a multi-step rule, suggested in (Puppo and Panozzo 2009, Sec. VI-A), to determine appropriate geometric values for the vertices $p_\ell \in \mathbb{R}^3$, and their successors $p_\ell(\lambda)$, where the non-negative integer λ specifies the level of subdivision ($p_\ell(0) \equiv p_\ell$). For $\lambda = 0$ we have $\ell = 0, \ldots, L-1$, while for positive λ the index ℓ runs over a larger range. Note that here, "successor" refers to successive values of the vertex in the subdivision process.

(The ambiguity in the notation p_ℓ, due to the fact that the indexing depends on the level, should lead to no confusion. The vertices are not stored in indexed form: rather, they are part of a standard half-edge data structure for meshes. Thus, the notation p_ℓ simply refers to some specific vertex in the mesh.)

Each subdivision of a triangle effected by ordinary Loop subdivision uses the primal $pT4$ schema (Andersson and Stewart 2010, p. 17), in which a new logical vertex is introduced in each edge, and the three new vertices are joined by new edges. Thus, each triangle is decomposed into four new triangles. A consequence of this fact is that if the mesh is subdivided uniformly, the amount of data to be stored increases proportionally to 4^λ. Since this function increases very quickly with λ, various methods of adaptive subdivision have been proposed for practical use. Thus, the RG triangulation method used in our algorithm permits triangles in one part of the mesh to be subdivided while triangles in another part are not.

The main difficulty with adaptive subdivision is that if a triangle is subdivided, but its neighbour is not, then a non-conformal mesh results, as illustrated in Figure 6 (left), where one triangle has been subdivided to level λ, and the other to level $\lambda + 1$. To correct this, a new (green) edge is introduced, and the triangles formed by this new edge are labelled green, in contrast to those obtained by the $pT4$ subdivision scheme, which are labelled red. Introduction of a new edge in this way is referred to as *green refinement*, which is in contrast to the standard subdivision, referred to as *red refinement*. See Figure 6 (right).

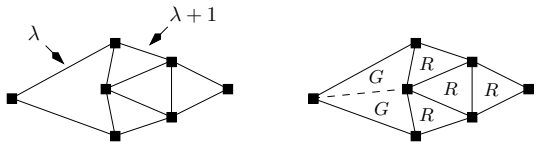

Fig. 6 Mesh made conformal by green subdivision.

Given certain triangles marked for refinement, a conformal mesh can be maintained using the following algorithm (Meister and Struckmeier 2002):

1. Eliminate all green refinements by restoring the mothers of green triangles. If a green triangle was marked for refinement, the restored mother is also marked for refinement.
2. Refine all red triangles marked for refinement using the $pT4$ schema.
3. While there exist triangles with more than one non-conforming vertex, refine them using the $pT4$ schema.
4. Apply green refinement to all triangles having exactly one non-conforming vertex.

The algorithm is implemented recursively. Green and red faces are tagged as green or red in the half-edge data structure representing the mesh. In our context the application of green refinement, in Step 4, and the elimination of green refinement in Step 1 of the next subdivision, can be eliminated. One green refinement is necessary at the very end of the process to make the result conformal.

Proposition 1 (Meister and Struckmeier 2002, p. 181). *The RG triangulation algorithm terminates with a conforming triangulation.*

There remains one other difficulty, however, in the context of Loop subdivision. Any particular vertex ℓ of valence n may have adjacent triangles that have been subdivided at (up to $\lceil \frac{n+1}{2} \rceil$) different levels, as illustrated in Figure 7. This means that the geometric value $p_\ell \in \mathbb{R}^3$ is ill-defined. We resolve the ambiguity by storing the value of p_ℓ at the level λ_0 at which the vertex was inserted; this value is denoted by $p_\ell(\lambda_0)$. For a vertex inserted as a result of subdivision at level λ that

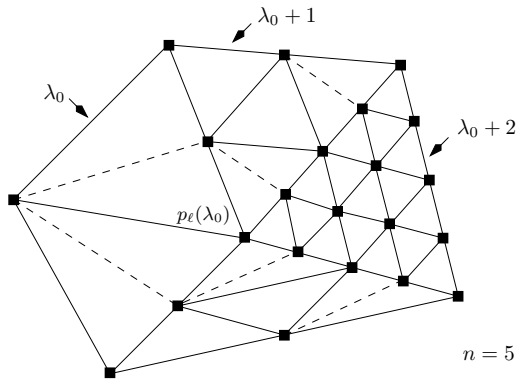

Fig. 7 Adjacent triangles subdivided at different levels.

does not keep the mesh conformal (as in the case of the middle vertex in Figure 6, left), this means that we store $p_\ell(\lambda + 1)$, just as we would in the case of conformal insertion of a vertex. We also store $p_\ell(\infty)$, the limiting value of the vertex on the surface S, which can be calculated (Puppo and Panozzo 2009) by

$$p_\ell(\infty) = \left(1 - \frac{8\alpha_n}{3+8\alpha_n}\right) p_\ell(\lambda_0) + \frac{8\alpha_n}{n(3+8\alpha_n)} \sum_{i=1}^{n} p_\ell^i(\lambda_0).$$

Here, $p_\ell^i(\lambda_0)$, $i = 1, \ldots, n$, denotes the geometric value at one of the n one-neighbours of $p_\ell(\lambda_0)$ at subdivision level λ_0, and

$$\alpha_n = \frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\left[\frac{2\pi}{n}\right]\right)^2.$$

(A one-neighbour of a vertex is another vertex in the mesh, joined to the original by a single edge.) Unavailable one-neighbours (see Figure 7 for examples) are computed on the fly, and not retained in the data structure.

We note in passing that in the example of Figure 7 we must have $\lambda_0 = 0$, since a vertex cannot be inserted with valence $n = 5$.

Proposition 2 (Puppo and Panozzo 2009, Sec. VI-A). *Storage of the two values $p_\ell(\lambda_0)$ and $p_\ell(\infty)$ permits subsequent calculation of p_ℓ at any level $\lambda_0 + \lambda$ by means of the multistep rule*

$$p_\ell(\lambda_0 + \lambda) = \gamma_n(\lambda) p_\ell(\lambda_0) + (1 - \gamma_n(\lambda)) p_\ell(\infty),$$

where

$$\gamma_n(\lambda) = \left(\frac{5}{8} - \alpha_n\right)^\lambda.$$

The values of $p_\ell(\lambda_0)$, $p_\ell^i(\lambda_0)$, $p_\ell(\infty)$ and $p_\ell(\lambda_0 + \lambda)$ are computed using ordinary floating-point arithmetic. The $p_\ell(\lambda_0)$ are computed (using Loop subdivision at Steps 2 and 3 of the RG triangulation algorithm) as convex combinations of values at the previous level of subdivision, and the error in their calculation can be bounded tightly using the standard model for floating-point arithmetic (Dahlquist and Björck 2008, Ch. 2). Similarly, it is straightforward (Dahlquist and Björck 2008, p. 107) to take account of the other errors involved in the calculation of $p_\ell(\lambda_0 + \lambda)$ (the details are omitted). Thus, making the mesh available to the higher-level process, we can replace the $p_\ell(0) \equiv p_\ell$ by the subdivided mesh in the problem definition, and affirm that the method has solved a problem corresponding to a surface that has suffered perturbations, which are assumed to be smaller than those corresponding to the data uncertainty. Our point of view is that of the backward error analysis (Dahlquist and Björck 2008, p. 113): the algorithm will give a correct result for a problem which has been perturbed by less than the data uncertainty, provided only that possible changes in topological form are detected on the fly.

We note here that alternate weights are often used in the Loop subdivision formulas (Andersson and Stewart 2010, Note 18, p. 328; CGAL 2012). The statement of Proposition 2 can easily be modified to correspond to this alternate choice of weights.

Figure 8 shows the results of an RG triangulation using the Loop method with the multistep rule of Proposition 2. The mesh is viewed at a very slight angle to the cutting plane (shown in white), which traverses the figure diagonally from top-left to bottom-right. The cutting of triangles near the top and bottom of the figure seems unambiguous, in contrast to the cutting of the triangles in the centre of the figure. Recall, however, that it is the cutting of the associated triangular surface patches that is of primary concern. Dealing with this distinction is the main task of the second phase of the algorithm, described in the next subsection. Examples illustrating the distinction are also given there.

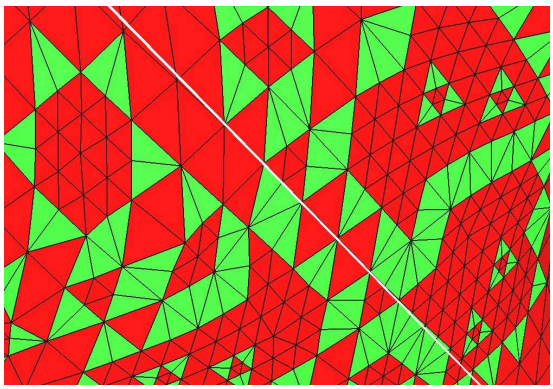

Fig. 8 Result of Red-Green Loop triangulation.

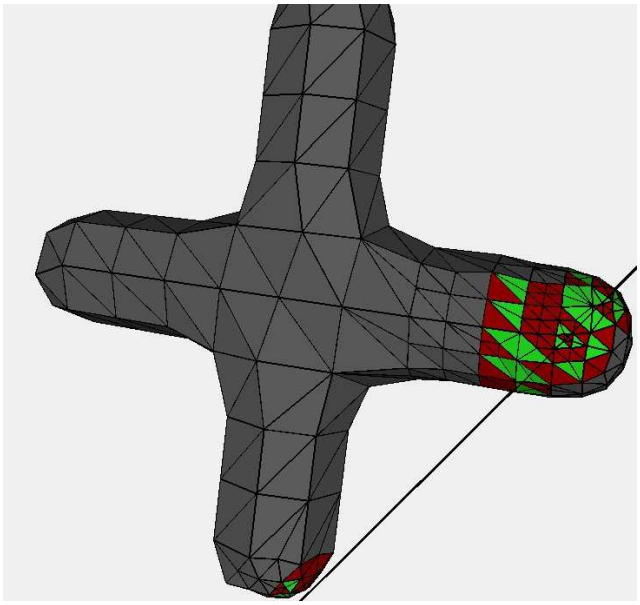

Fig. 9 An example of C (triangles shown in red and green).

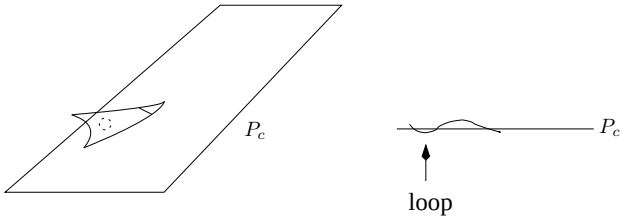

Fig. 10 Possibility of a loop intersection with P_c.

4.2 Triangles generating simple intersections

The initial phase of the subdivision, described in the previous section, is unremarkable. A Wu-Peters bounding box (Wu 2005) provides a criterion to decide which triangles intersecting P_c should be subdivided first, and to estimate the error in the approximation. The Wu-Peters bound is obtained by extracting a linear function from each component of the patch, such as the x-component, and finding a bound of the form

$$\begin{aligned} x^+ &= l(u,v) + \textstyle\sum_{i=3}^{n+5} \max\{d_i,0\}b_i^+ + \min\{d_i,0\}b_i^- \\ x^- &= l(u,v) + \textstyle\sum_{i=3}^{n+5} \max\{d_i,0\}b_i^- + \min\{d_i,0\}b_i^+ \end{aligned} \quad (2)$$

where $l(u,v)$ is the linear function. (In order to define d_i, b_i^+, b_i^-, and $l = l(u,v)$, the parametric domain is defined locally to be a portion of the plane (Wu 2005; Wu and Peters 2004). The surface is then defined in terms of auxiliary basis functions having a linear precision property, and b_i^+ and b_i^- are precomputed linear upper and lower bounds for these basis functions. The function $l(u,v)$ is a linear function approximating the patch, and the d_i can be interpreted as the errors in this approximation corresponding to control points neighbouring the patch. A complete description is given in (Andersson and Stewart 2010, pp. 280-283).) Subdivision is carried to a deeper level in regions of the mesh near P_c, until an externally supplied error criterion is satisfied. This produces the mesh $\mathcal{M}^{\nu_0}$ and the list C mentioned in Section 3. The bounds in (2) are reminiscent of formulas for interval arithmetic, and interval arithmetic is in fact used to compute C.

Figure 9 shows an example, which uses a base mesh from (CGAL 2012). The triangles in C, which according to the Wu-Peters test might intersect the plane P_c, are shown in red and green. In this figure, the mesh is again viewed at a very slight angle to the cutting plane.

The algorithm's next task, in the second phase, is to identify triangles, in $\mathcal{M}^{\nu_0}$ or in some subdivided mesh $\mathcal{M}^{\nu}$ with a possibly slightly perturbed limit surface S, in such a way that the topological form of $\mathcal{M}^{\nu} \cap P_c$ is exactly the same as that of $S \cap P_c$, and to replace them by triangles having only simple intersections. This is not a straightforward problem. For example, a triangle that does not intersect P_c may nonetheless have an associated triangular surface patch which cuts the plane in a curve traversing the patch, in an internal loop within the patch, or both. The last-mentioned case is illustrated in perspective and in cross-section, respectively, in Figure 10 (left and right). Similarly, a triangle might intersect P_c in a straight line, while the associated triangular surface patch intersects P_c in both a curve and a loop, only in a curve, only in a loop, or not at all. Small changes in the data may change the situation from one case to another, and we wish to correctly determine which case holds using finite-precision and interval arithmetic.

As already suggested, the principal approach used to resolve these ambiguities is to produce an equivalent mesh $\mathcal{M}^{\nu}$ by appropriate further subdivision of $\mathcal{M}^{\nu_0}$. It is possible, however, that even after further subdivision it is not possible to determine the topological form of $S \cap P_c$. In this case an exception is thrown, in order to request the required information. As mentioned at the end of Section 3, the algorithm should not throw an exception if there is no ambiguity in the topological form of the result. Consider, for example, the situation illustrated in Figure 11, where S intersects P_c (shown in perspective) at a sharp angle. If the surface does not vary quickly, relative to the level of uncertainty, then there is no ambiguity in the topological form, even though some of the triangle elements (vertices and edges) in $\mathcal{M}^{\nu}$ might be very close to the plane, and therefore the algorithm should not trap. Fur-

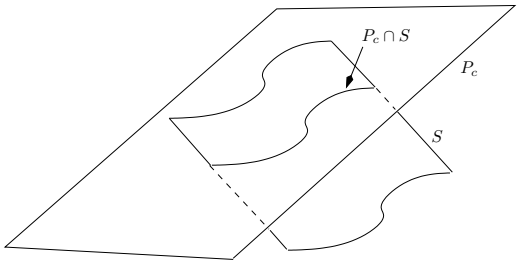

Fig. 11 No ambiguity in the topological form.

thermore, we would like to avoid further subdivision whenever possible, to reduce the computational cost of the algorithm. Consequently, the algorithm does not merely subdivide repeatedly in the areas of possible intersection with P_c until some very small triangle size is attained. Instead, it uses the tests described in subsequent subsections, which detect the topological nature of the intersections between the triangular surface patch and P_c. The tests may succeed, even for fairly large triangles, so that the need for repeated subdivision will usually be avoided. On the other hand, there is no guarantee that the algorithm described here avoids this.

It is reasonable here to introduce a simplifying assumption, which restricts slightly the class of problems for which the method is applicable. In general there might be planar subsurfaces of S that lie in P_c, and to permit specification of this kind of case, by a higher-level process catching the exception, it might in general be useful to use Selective Geometric Complexes (Rossignac and O'Connor 1989) with cells of dimension 2 in the specification of $S \cap P_c$, for purposes of communicating with the higher-level process. (An alternative description of this idea is that $S \cap P_c$ might be represented as a hypergraph with hyperedges sharing three vertices.) For simplicity, however, it is probably sufficient in practice to consider this case to be a loop intersection, and require, for purposes of communicating with the higher-level process, that the intersection of $\mathcal{M}^\nu$ and P_c does not involve two-dimensional sets. Note that, in any case, triangular surface patches can be constrained to lie in the plane by specifying that one-neighbour vertices of the associated triangle lie in the plane. The question of the level of generality provided in the interface with the higher-level process is discussed further in Section 5.

4.3 Detection of loops within a surface patch

In order to eliminate differences between the topological form of $\mathcal{M}^\nu \cap P_c$, and that of $S \cap P_c$, the second phase of the algorithm first tries to eliminate extraneous intersections of the triangular surface patch which have the form of loops, as in Figure 10.

If the vertices of the triangular surface patch were known to lie on the same side of P_c, then a bounding-box approach, in conjunction with repeated subdivision, might be sufficient to confirm that there are no extraneous loop intersections. This does not work, however, if the vertices are on opposite sides of the plane, since a bounding-box approach will simply indicate that there is at least one intersection. Also, again, we would like to remove extraneous intersections without subdivision, if this is possible.

To exclude the possibility of this kind of extraneous intersection (loops), we can make use of bounds on the Gauss map, which is defined as the map that takes a point on the surface to the corresponding unit surface normal.

Proposition 3 *To exclude loop intersections in the triangular surface patch, it is sufficient to check that neither n_c nor $-n_c$ is contained in N_1, the convex hull of the image of the Gauss map of the triangular surface patch.*

Proof The statement follows directly from (Hohmeyer 1991, Theorem 1) (and the notation N_1 has been chosen to be consistent with that of the cited theorem). Indeed, if the stated condition is satisfied, then there exists a plane $\{x \in \mathbb{R}^3 : P_2 \cdot x = \sigma\}$ defined by $P_2 \in \mathbb{R}^3$ such that $P_2 \cdot n_c > \sigma$ and $P_2 \cdot n > \sigma$ for $n \in N_1$. (Note that here we have used the condition on $-n_c$.) Further, there is a plane defined by some $P_1 \in \mathbb{R}^3$, separating the image of the Gauss map from n_c, so that $P_1 \cdot n_c < \sigma$ but $P_1 \cdot n > \sigma$ if $n \in N_1$. It then follows from Theorem 1 in (Hohmeyer 1991) that any intersection of the triangular surface patch with P_c is a curve, a point, or a set of curves and points; and, further, that all isolated point intersections are at the boundaries of the surface patches, the curves do not contain any singularities, and no intersection curve forms a loop.

Note that the stated condition is not a necessary condition. As described in Section 4.5, if at a certain level of subdivision it is not possible to exclude the possibility of a loop intersection, the algorithm will subdivide further, and test again.

A fail-safe estimate of N_1 can be obtained using the method of (Grinspun and Schröder 2001, Secs. 3.1-3.4), which is based on the method of (Stam 1998) for the

direct parametric evaluation of Loop subdivision surfaces. Many details relevant to the implementation are described in (Stam 1998, Sec. 4). The central idea of the method is to decompose the triangular parametric domain, of the triangular surface patch, into a sequence of subdomains for which the patch is regular ($n = 6$); and, to change coordinates so that the subdivision can be expressed in terms of the eigenbasis of the subdivision matrix, so that subdivision becomes a simple scaling.

4.4 Multiple intersections on surface-patch boundary

Suppose now that the test of the previous subsection has confirmed that there is no loop intersection of the triangular surface patch and the plane. We now wish to verify that there is *exactly one intersection along a certain boundary of a triangular surface patch.* The algorithm of Section 4.5 excludes the possibility of zero or multiple intersections by using a certain property of Bézier curves, and proceeding as in the case of loop intersections if the test fails, *i.e.*, by subdividing and repeating the test. The result is a modified set C composed only of triangles generating simple intersections. The property mentioned can be described as follows.

A triangular surface patch in Loop subdivision, in the regular case (all vertices have valence $n = 6$) can be expressed as a quartic Bézier surface, defined by the twelve one-neighbours of the mesh triangle (Schweitzer 1996, p. 21; Lai 1992). Further, an edge of a Bézier patch is a Bézier curve, and the control points of the curve are just the five control points along the border of the triangular array of Bézier control points (Andersson et al. 1998). If we join these five points by line segments, to produce a control polygon for the curve, it follows from the Variation Diminishing Principle for Bézier curves that the curve intersects P_c no more frequently than the Bézier control polygon.

Proposition 4 (Piegl and Tiller 1995, p. 12). *If the control polygon does not intersect P_c, then the Bézier curve does not intersect P_c. Also, if the control polygon intersects P_c exactly once, then the curve intersects P_c at most once.*

Since a Bézier curve interpolates its endpoints (Piegl and Tiller 1995), the cases of zero and one intersection can easily be distinguished by checking whether the endpoints are on opposite sides of the plane. All of these tests can be verified in a fail-safe manner using interval arithmetic. Again, however, these tests provide only a *sufficient condition.* If the sufficient condition is not satisfied then a subdivision step must be carried out.

4.5 Algorithm giving patches with simple intersections.

The algorithm is described in the pseudocode given below. We suppose that $\mathcal{M}^{\nu_0}$ is non-empty. If the list C is initially empty, then the planar-cut algorithm should return either $\mathcal{M}^{\nu_0}$ or $\emptyset$, depending on whether $\mathcal{M}^{\nu_0}$ is entirely above, or entirely below, the plane P_c.

Green triangles are treated as pairs, with tests applied to the red mother triangle.

If C is not empty, the algorithm below either

- certifies a triangle in C as generating only simple intersections; or
- replaces the triangle in C by child triangles eligible for certification, or deletes such child triangles because their corresponding triangular surface patch cannot intersect P_c; or
- throws an exception in order to obtain
  - certification; or
  - the information that the triangle can be deleted from C.

Certification of a triangle includes labelling of the edges, to indicate a single intersection on each of a pair of edges. This unfortunately destroys the monotonicity property of certification. The property of having no loop intersections is inherited by any child triangles created by subdivision. On the other hand, to satisfy the single-intersection test on a given edge, it is necessary to do several things:

- the sufficient condition provided by the Variation Diminishing Principle must be satisfied;
- the triangle vertices must have the same above-below status, with respect to the plane P_c, as the corresponding vertices of the associated triangular patch;
- the end points of the edge must be regular.

The first two of these conditions are verified using interval arithmetic. Although the property that an edge does not have multiple intersections is inherited by children created by subdivision, the ability to confirm all conditions using interval arithmetic is not. This means that in principle it is possible that a child of a certified triangle cannot be certified (see pseudocode, below). Convergence of the algorithm is guaranteed, however, by the geometric reduction in triangle size due to subdivision. Similarly, whether or not the various tests eventually pass, the fact that triangle size is eventually reduced to the size of data uncertainty means that if the algorithm throws an exception, it is because there exists a an additional perturbation of S of that size which could change the topological form.

Pseudocode:

while (there exist uncertified triangles in C)
if (size of triangle is smaller than uncertainty level)
If only the above-below status of one or more endpoints of an edge of a triangle is incorrect, perturb the triangle endpoints in a direction parallel to n_c, to correct the situation. Otherwise, throw an exception, asking the higher-level process to specify the topology of the intersection curves on the triangle.
else
if (Wu-Peters guarantees there is no intersection)
Delete the triangle from C.
else
if (either the loop test or single-intersection test fails)
Apply RG refinement. Mark the child triangles as uncertified, and put them in C in place of the mother triangle.
else
Label the edge containing the single edge intersection, and certify the triangle.

Finally, the algorithm scans the vertices of the mesh. If the limit position of a vertex cannot be isolated from P_c using interval arithmetic, the vertex is perturbed in a direction parallel to n_c to correct the situation.

The algorithm can be made more efficient by making use of the fact that the loop test does not have to be repeated for triangles that are the children of a triangle that has already passed the test. (This monotonicity property for the loop test was mentioned above, within this subsection.)

In the case when an exception is thrown, specification of the topology may involve an indication that there is no intersection, so that the triangle can be deleted from C, or it may involve specification of multiple edge intersections, as in the case of a saddle point. This is discussed briefly in Section 5.

4.6 Tracing the graph defining the topology of $S \cap P_c$

If the higher-level process is permitted only to specify triangles that have single edge intersections, the tracing procedure is straightforward. The intersection of the edge curve is known to exist, and we can compute an approximation to the intersection between the triangle and P_c. Following the triangle adjacency links within the mesh permits us to continue the process in the adjacent triangle. Since we are assuming that each edge of the triangular surface patch has zero or one (non-composite) roots, the intersection path forms a closed loop.

In the more general case more elaborate procedures must be implemented. This question is also discussed briefly in the next section.

5 Discussion and future work

In previous sections we have described our general approach to the problem described in Section 3. There remain, however, many interesting research questions, and several major issues to be resolved in the light of experimentation. In this section we first give a brief discussion of our prototype implementation, followed by a summary of the major issues just mentioned. These issues will be dealt with in detail in the first author's PhD thesis.

5.1 The prototype implementation

The major components of the algorithm described above have been implemented, but not always in full generality. In particular, the fail-safe calculation of N_1, mentioned at the end of Section 4.3, is replaced by an estimate. Study of the modifications necessary (Grinspun and Schröder 2001, Sec. 3.5) to obtain tight bounds for N_1 constitutes an interesting research topic in itself, but it is a topic that is largely independent of the one discussed here.

Our prototype implementation is written in C++ using standard tools, such as *OpenGL* and *QT*. We also make use of the *CGAL* library (CGAL 2012) for basic subdivision, and for other tasks, such as checking the condition of Proposition 3. Similarly, we used the *MPFI* package (MPFI 2012) for interval arithmetic.

The prototype implementation does not achieve interactive execution rates. To illustrate, for the model of Figure 9, which has (after a preliminary subdivision to isolate extraordinary vertices) 304 faces, 456 edges and 154 vertices, the solution required about 540 *milliseconds* to compute the planar cut, when no traps occurred. Although these times could quite likely be improved with a GPU implementation, it is important to emphasize that efficiency is not our main concern. Our point of view is different here: rather than requiring interactive solution rates, we have made correct topological form a tight constraint. We then try to minimize solution times subject to this constraint.

It is unlikely that the approach described in this paper would be of interest in the context of applications such as gaming, even if interactive rates could be achieved. It is in traditional CAD applications that correct topological form can be crucial (Hoffmann and Stewart 2005; Farouki 1999; Andersson et al. 2007).

5.2 Future work

The main questions left open are related to the design of the interface for exception handling, and the elimination of cases where the algorithm traps even though there exists no small perturbation of the data which could lead to a change in the topological form. Also, it may be possible to replace the backward error approach of Section 4 by one based entirely on a direct forward error analysis.

One aspect of the interface design is the level of generality to be treated. For example, in the specification of the topology, we might permit the higher-level process to specify only closed-loop topologies; alternatively, we might also permit the specification of saddle points, as mentioned at the end of Section 4.5. The former choice would limit the generality of the cases that could be handled, while the latter would require the implementation of a more elaborate tracing procedure, as mentioned at the end of Section 4.6. In the latter case it would be possible for the higher-level process to specify triangle deletion, or a saddle-point intersection involving four incident path segments. Even the generality of the latter case would rarely be needed in practice, but in principle, the specification of even more general types of topology might be permitted. An example is the possibility of permitting specification of planar intersections of the mesh and P_c, as mentioned at the end of Section 4.2.

As stated at the end of Section 3, our goal is to establish weak conditions guaranteeing that the algorithm will trap and ask for additional information only in the case when a perturbation of the original data, approximately equal in size to the data uncertainty, could lead to a change in the topological form. But (again as stated at the end of Section 3) the fact that the algorithm traps only if there exists a perturbation of the surface S itself which would change the topological form does not necessarily *guarantee* that there exists a perturbation of the original data that has this effect. In fact, the latter condition is very likely in practice, but in future work, we would like to narrow the gap between "guarantee" and "very likely". This possibility is discussed in the remaining paragraphs of the paper.

In this context, the test of Proposition 3 has as its intuitive basis the idea that if the surface (viewed as a function defined locally over P_c) is nearly horizontal, then because of undulations in the surface, a translation of the original control points will lead to a change in topological form. If the Wu-Peters bounding box has size on the order of the uncertainty, this means that if the algorithm has thrown an exception, it has done so in a case when there is no other choice: a perturbation of the original control points, on the order of the uncertainty in the data, would have led to a change in topological form.

The test does not, however, provide perfectly tight necessary conditions. For example, it could fail because of a saddle point, or because the bound on the surface normal provided by N_1 is not tight, so that the surface normal cannot be equal to $\pm n_c$, even though we are unable to guarantee this.

The consequence of such gaps between the strength of conditions is not algorithm failure, but unnecessary trapping. This will happen very rarely, but nonetheless, it would be better to eliminate such behaviour to whatever extent possible. In fact, it should be possible to design stronger tests, for example by giving conditions which assure that there exists a local relative minimum (Luenberger 1973, Sec. 6.2), so that again a translation of the original control points will guarantee a change in topological form. On the other hand, the computational cost of such tests must be examined.

References

Amresh, A., Farin, G., and Razdan, A. (2002). Adaptive subdivision schemes for triangular meshes. In *Hierarchical and Geometric Methods in Scientific Visualization*, pages 319–327. Springer-Verlag.

Andersson, L.-E., Dorney, S. M., Peters, T. J., and Stewart, N. F. (1995). Polyhedral perturbations that preserve topological form. *Comput. Aided Geom. Des.*, 12(9):785–799.

Andersson, L.-E., Peters, T. J., and Stewart, N. F. (1998). Self-intersection of composite curves and surfaces. *Computer Aided Geometric Design*, 15:507–527.

Andersson, L.-E. and Stewart, N. F. (2010). *Introduction to the Mathematics of Subdivision Surfaces*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Andersson, L.-E., Stewart, N. F., and Zidani, M. (2007). Error analysis for operations in solid modeling in the presence of uncertainty. *SIAM J. Sci. Comput.*, 29:811–826.

Biermann, H., Kristjansson, D., and Zorin, D. (2001). Approximate boolean operations on free-form solids. In *Proceedings of SIGGRAPH '01*, pages 185–194, New York, NY, USA. ACM.

Brunnermeier, S. B. and Martin, S. A. (1999). Interoperability cost analysis of the U.S. automotive supply chain. Technical report, Research Triangle Institute, Center for Economics Research.

CGAL (2012). Computational geometry algorithms library. Accessed 2012. http://www.cgal.org.

Chen, Z., Luo, X., and Ling, R. (2007). An adaptive subdivision method based on limit surface normal. In *Tenth IEEE International Conference on CAD and CG*, pages 65–70.

CXSC (2012). CXSC. Accessed 2012. http://www2.math.uni-wuppertal.de/~xsc/xsc/cxsc.html.

Dahlquist, G. and Björck, A. (2008). *Numerical Methods in Scientific Computing, Volume 1*. Society for Industrial and Applied Mathematics, 3 edition.

Farouki, R. T. (1999). Closing the gap between CAD model and downstream application.

Grandine, T. A. (2000). Applications of contouring. *SIAM Rev.*, 42(2):297–316.

Grinspun, E. and Schröder, P. (2001). Normal bounds for subdivision-surface interference detection. In *Proceedings of the conference on visualization '01*, VIS '01, pages 333–340, Washington, DC, USA. IEEE Computer Society.

Hoffmann, C. and Stewart, N. (2005). Accuracy and semantics in shape-interrogation applications. *Graphical Models*, 67(5):373–389.

Hohmeyer, M. E. (1991). A surface intersection algorithm based on loop detection. In *Proceedings of the first ACM symposium on solid modeling foundations and CAD/CAM applications*, SMA '91, pages 197–207, New York, NY, USA. ACM.

IEEE (2008). *IEEE Standard for Floating-Point Arithmetic*. IEEE, New York. IEEE Std 754-2008.

Lai, M.-J. (1992). Fortran subroutines for B-nets of box splines on three- and four-directional meshes. *Numerical Algorithms*, 2:33–38.

Lai, S. and Cheng, F. (2007). Robust and error controllable Boolean operations on free-form solids represented by Catmull-Clark subdivision surfaces. *Computer Aided Design and Applications*, 1-4:487–496.

Loop, C. (1987). Smooth Subdivision Surfaces Based on Triangles. Master thesis, Department of Mathematics, University of Utah, Utah, USA.

Luenberger, D. G., editor (1973). *Introduction to Linear and Nonlinear Programming*. Addison-Wesley.

Meister, A. and Struckmeier (2002). *Hyperbolic Partial Differential Equations*. Springer Vieweg.

Moore, R. E., Kearfott, R. B., and Cloud, M. J. (2009). *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

MPFI (2012). Multiprecision interval arithmetic library (MPFI). Accessed 2012. http://perso.ens-lyon.fr/nathalie.revol/software.html.

P1788 (2012). IEEE interval standard p1788. Accessed 2012. http://grouper.ieee.org/groups/1788.

Patrikalakis, N. M. and Maekawa, T. (2002). *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Piegl, L. and Tiller, W. (1995). *The NURBS Book*. Springer-Verlag, London, UK.

Puppo, E. and Panozzo, D. (2009). RGB subdivision. *IEEE transactions on visualization and computer Graphics*, 15:295–310.

Rossignac, J. and O'Connor, M. (1989). SGC: a dimension-independent model for pointsets with internal structures and incomplete boundaries. In M. Wosny, J. Turner, K. P., editor, *Geometric Modeling for Product Engineering*, pages 145–180. North-Holland.

Schweitzer, J. E. (1996). *Analysis and Application of Subdivision Surfaces*. PhD thesis, Department of Computer Science and Engineering, University of Washington.

Shewchuk, J. R. (2000). Mesh generation for domains with small angles. In *Proceedings of the 16th annual symposium on computational geometry*, SCG '00, pages 1–10, New York, NY, USA. ACM.

Spitz, S. and Rappoport, A. (2004). Integrated feature-based and geometric cad data exchange. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*, SM '04, pages 183–190, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Stam, J. (1998). Evaluation of loop subdivision surfaces. In *CD-ROM Proceedings of SIGGRAPH'98*. ACM.

Wu, X. (2005). An accurate error measure for adaptive subdivision surfaces. In *Proceedings of the international conference on shapes and solids*, pages 51–56.

Wu, X. and Peters, J. (2004). Interference detection for subdivision surfaces. In *Proceedings of the 5th international World Wide Web conference*, pages 96–107.

Yap, C. (2008). Reliable implementation of real number algorithms: Theory and practice. In Hertling, P., Hoffmann, C. M., Luther, W., and Revol, N., editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, chapter Theory of Real Computation According to EGC, pages 193–237. Springer-Verlag, Berlin, Heidelberg.

Ying, L. and Zorin, D. (2001). Nonmanifold subdivision. In *Proceedings of the conference on visualization '01*, VIS '01, pages 325–332, Washington, DC, USA. IEEE Computer Society.

Zorin, D., Schröder, P., and Sweldens, W. (1997). Interactive multiresolution mesh editing. In *Proceedings of the 24th annual conference on computer graphics and interactive techniques*, SIGGRAPH '97, pages 259–268, New York, NY, USA. ACM Press/Addison-

Wesley Publishing Co.