IsoGeometric Analysis with Subdivision Surfaces

Pieter J. Barendrecht
Eindhoven University of Technology
January 19, 2013

Abstract

In this introductory project we have applied IsoGeometric Analysis (IGA) on geometries modeled with Catmull-Clark subdivision surfaces. The fundamental concept of subdivision surfaces is the repeated subdivision of the control net, which basically represents a rough draft of an envisioned geometry. In the limit this control net converges to a surface referred to as the limit surface, which can be completely described by (piecewise) basis functions. This means that subdivision surfaces can be used in conjunction with IGA.

Using a combination of Blender and Matlab, we have created a workflow that allows us to successively import coarsely modeled objects, subdivide them, calculate their limit surface and eventually apply analysis (e.g. elasticity analysis) on them. Although this environment is rather static and still requires some manual intervention, it shows that the two topics go hand-in-hand very well.

This report discusses the absolute basics of both univariate and bivariate subdivision, along with a few related topics. Using the concept of the limit surface, the connection to IGA is made. The combination of these two areas of research is ultimately illustrated with an example.

1. Introduction

Subdivision surfaces are a powerful modeling technique, much used in animated movies and games. However, it has been virtually absent in Computer Aided Engineering (CAE) for a long time. One of the reasons might be that it can be rather difficult to model a geometry exactly as intended (e.g. design an object from a blueprint). Another cause might be the difficulty in evaluating points on the resulting surface, called the *limit surface*.

Nowadays, these inconveniences are practically resolved. There are extensive possibilities regarding shape control [14], NURBS-compatible subdivision surfaces [6] (NURBS [17] are still the standard in CAD), and stable methods to evaluate any point on the limit surface, whether it is regular, extraordinary [34], a point near the boundary or near a crease [14]. This means that subdivision surfaces can now be combined with analysis. A relatively new Finite Element (FEA) technique, IsoGeometric Analysis (IGA) [11], seems the right candidate for this conjunction [5].

The intention of this introductory project was to briefly study a popular subdivision scheme and subsequently use it for analysis. It has, however, gotten slightly out of hand, resulting in an interactive Matlab environment to experiment with different schemes, shape control and other related topics like reverse subdivision. The topic is of such interest that the research described in this report will be continued and extended in my MSc thesis. Therefore, we have chosen to only focus on the basics of a single scheme for this report – Catmull-Clark [8].

In this text, it is assumed that the reader has a basic knowledge of CAGD (e.g. Bézier and B-Spline curves and standard knot-insertion/refinement algorithms). If not, [31] and [18] are excellent references to catch up. Good books discussing FEA and IGA are [3], [28] and [19]. Regarding the topic of subdivision – there are many well-written papers and surveys [24, 7, 29], but unfortunately not many books. Both [1] and [30] have been regularly consulted for this project.

To conclude this section, an overview of the contents. We will start more or less from scratch with univariate subdivision. The main concepts will be introduced, and are extended to the bivariate case in the subsequent section. Following up is the most important part, discussing the limit surface. Since the limit surface forms the connection between subdivision and analysis, it is only logical to look at this topic in a little more detail. Next is an overview of IGA, followed by its application to subdivision surfaces. We conclude with some thoughts on the implementation and a look ahead regarding future research.

2. Subdivision curves

Let us begin with an arbitrary curve $C(t) = P^T A(t)$ where P is a vector of *control points*, and $A(t)$ a vector of *basis functions*. For many reasons it could be desirable to describe *the exact same curve* in a different way as $D(t) = Q^T B(t)$. For instance, a change of basis could be useful (e.g. using the Bernstein basis instead of the B-spline basis when applying *Bézier decomposition*), or a different set of control points (e.g. for a better control of the curve when using it to model an object), where each new control point is a linear combination of the old control points.

Note that when a different set of basis functions is chosen, a different set of control points must be used as well, and vice versa. This can be easily seen when writing the new control points as a linear combination of the old ones: $Q = SP$, where S is the matrix containing the linear combinations. It follows that $Q^T = P^T S^T$, and therefore the curve $D(t)$ can be written as $P^T S^T B(t)$. Since $D(t) = C(t)$ by definition, it results in the expression $A(t) = S^T B(t)$.

In the case of binary subdivision (whether it is univariate or bivariate), we're interested in finding a *two-scale* relation for the original basis functions, and use this relation to compute the new set of control points. For special cases, the matrix S relating Q to P can be found directly, but in the general case we have to resort to the two-scale relation (also called *scaling* or *refinement* relation).

We start by taking a look at both these approaches, and briefly touch the topic of convergence. The subsequent section will expand the concepts to surfaces.

The graphical method to find the new set of control points is due to the Lane-Riesenfeld algorithm [22], which extends the de Rham-Chaikin algorithm [9] to arbitrary degree B-splines. To illustrate the approach, the cubic case is depicted in Figure 1.

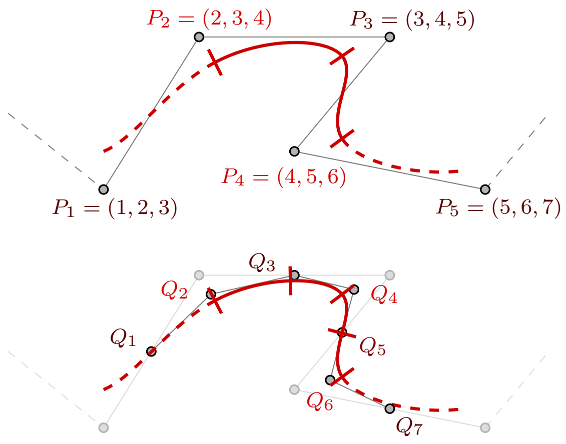

Fig. 1 *Lane-Riesenfeld for $d = 3$. Each control point has an associated label, containing the parameters for the blossom. Labels for Q_i are ommitted.*

Note that the mathematical interpretation of the Lane-Riesenfeld algorithm is midpoint knot-refinement, which can be observed when making use of the blossom (i.e. the polar form of the curve) [27, 25].

The line linearly interpolating the control points is called the *control polygon*, which is the key concept of subdivision. Note that the control polygon of the new set of control points already looks a lot more like the actual B-spline curve than the old control polygon. If we were to apply the same method repeatedly, the control polygon would (in the limit) converge to the actual curve – the *limit curve*. This means that instead of drawing the actual curve, we could just take its initial control polygon, subdivide it a few times, and display the result instead. While this is not that useful for curves, it certainly is for surfaces. It is used a lot in the animation and gaming industry (e.g. the power of the computer being used determines the smoothness of the objects in a game). As of yet, it is not widely used in CAD.

In matrix notation, the new control points Q_i can be obtained from the old control points P_i by multiplying them with a subdivision matrix S. For the cubic case ($d = 3$) this results in

$$\begin{pmatrix} \vdots \\ Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \\ \vdots \end{pmatrix} = \frac{1}{8} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 4 & 4 & 0 & 0 & 0 & \cdot \\ \cdot & 1 & 6 & 1 & 0 & 0 & \cdot \\ \cdot & 0 & 4 & 4 & 0 & 0 & \cdot \\ \cdot & 0 & 1 & 6 & 1 & 0 & \cdot \\ \cdot & 0 & 0 & 4 & 4 & 0 & \cdot \\ \cdot & 0 & 0 & 1 & 6 & 1 & \cdot \\ \cdot & 0 & 0 & 0 & 4 & 4 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \vdots \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ \vdots \end{pmatrix}$$

Note that this matrix S directly follows from the application of the Lane-Riesenfeld algorithm in Figure 1 (see also Figure 2).

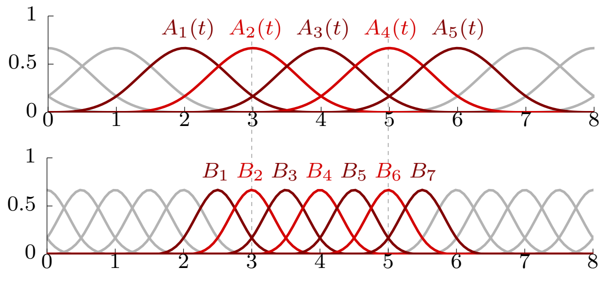

Fig. 2 *Unrefined (top) and refined (bottom) B-Spline basis. $A_i(t)$ are the accompanying basis functions for P_i, likewise $B_i(t)$ for Q_i.*

When studying the dependencies of the new control points, it can be observed that there are in fact two different cases – there are two different *stencils*. A stencil [30] illustrates how much a new control point depends on which old control points. The ones for the example are $^1/_8\,[4\ 4]$ and $^1/_8\,[1\ 6\ 1]$. These stencils represent an affine combination (in this and most other cases even a convex combination) – for convenience the factor $1/8$ could therefore be omitted.

The more general approach is finding the *mask*. A mask [30] is the opposite of a stencil – it shows how much an old control point contributes to which new control points. For splines, the two-scale relation for the original basis functions can be used to find the mask. Using the geometric definition of box splines (including uniform B-splines), the refinement relation is readily obtained (see Figure 2 and 3).

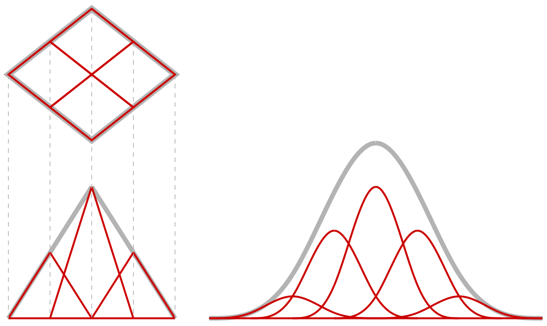

Fig. 3 *Graphical representation of the two-scale relations for $d = 1$ (using the projection of a two-dimensional box) and $d = 3$. The associated masks are $1/2\,(1\ 2\ 1)$ and $1/8\,(1\ 4\ 6\ 4\ 1)$.*

It appears that the new basis functions are scaled and shifted versions of the old ones (which is basically the definition of a two-scale relation) – for example, $A_3(t)$ equals $1/8(B_2(t) + 4B_3(t) + 6B_4(t) + 4B_5(t) + B_6(t))$, which is equivalent to $1/8(A_3(2t) + 4A_3(2t-1)+6A_3(2t-2)+4A_3(2t-3)+A_3(2t-4))$. See Figure 2 and 3. Note that the coefficients add up to 2, because the mask corresponds to a *binary* subdivision scheme. Again, the factor $1/8$ could therefore be omitted. A geometrical interpretation of a coefficient in a mask is simply the number of subcubes projected onto the same part of the support.

A different approach for finding the two-scale relation for B-splines relies on the Fourier transformation [26]. Other methods of obtaining the mask include the z-transform (i.e. using the *generating polynomial* of a scheme) [23]. A really intuitive approach is the *arrow* method [15]. Combined with mask convolution, this is a very powerful technique.

It is straightforward to obtain the stencils from a mask (the sequences of every other coefficient in a mask constitute the stencils), so for [1 4 6 4 1] we have $\left(\boxed{1}\ 4\ \boxed{6}\ 4\ \boxed{1}\right) = [1\ 6\ 1]$ and $\left(1\ \boxed{4}\ 6\ \boxed{4}\ 1\right) = [4\ 4]$. The stencils can then be used to construct the subdivision matrix S.

It is important to note that in the literature the term *subdivision matrix* is used for (at least) three different matrices. The first one is the bi-infinite one as illustrated above. Additionally, because the basis functions of these subdivision schemes have local support, only a small part of the matrix is enough to derive the properties of the scheme (i.e. the smoothness or continuity). For this kind of analysis, the curve in the neighborhood of a point of interest has to be known. For the cubic case, this leads to the 5×5 matrix

$$\frac{1}{8}\begin{pmatrix} 1 & 6 & 1 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 6 & 1 & 0 \\ 0 & 0 & 4 & 4 & 0 \\ 0 & 0 & 1 & 6 & 1 \end{pmatrix}$$

which is used to obtain the control points defining a cubic segment left and right of the point of interest, in other words its *two-ring neighborhood.* When only interested in the evaluation of the scheme at a certain point, an even smaller matrix suffices. In the cubic case, this is the 3×3 matrix

$$\frac{1}{8}\begin{pmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{pmatrix}$$

which corresponds to the *one-ring neighborhood* of the point of interest.

To conclude this section, let us think about what will happen when the same matrix (either the 5×5 or the 3×3 one) is repeatedly (i.e. n times) applied to respectively 5 or 3 control points: $R = S^n P$. Instead of iteratively applying S for a number of n times, we could try to compute S^n and use the resulting matrix to find R directly. Using the *eigendecomposition* of $S = V \Lambda V^{-1}$, it follows that $S^n = V \Lambda^n V^{-1}$. Obtaining the eigenstructure for these small matrices is relatively easy, but for larger ones we will need special techniques, as explained in the next section.

Looking at the expression $R = S^n P$ for the 3×3 matrix, this reduces (for $n \to \infty$) to

$$\begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix} = \frac{1}{6}\begin{pmatrix} 1 & 4 & 1 \\ 1 & 4 & 1 \\ 1 & 4 & 1 \end{pmatrix}\begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

so any three consecutive control points of the control polygon of a uniform cubic B-spline will converge to a single point. Rewriting this point to $\frac{1}{2}(\frac{1}{3}P_1 + \frac{2}{3}P_2) + \frac{1}{2}(\frac{2}{3}P_2 + \frac{1}{3}P_3)$ and using blossoming, it is not difficult to see that this point indeed lies on the actual B-spline curve.

3. Subdivision surfaces

The first bivariate subdivision schemes were those of Doo-Sabin [16] and Catmull-Clark [8]. The foundation of these schemes is constituted by the tensor product of the Lane-Riesenfeld algorithm for respectively $d = 2$ and $d = 3$ (note that this is *not* historically correct, as [22] was published several years later than [16] and [8]). To illustrate this, let us take the tensor products of the two stencils for the $d = 3$ case:

$$\begin{bmatrix} 4 & 4 \end{bmatrix} \otimes \begin{bmatrix} 4 & 4 \end{bmatrix} = \begin{bmatrix} 16 & 16 \\ 16 & 16 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 6 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 24 & 4 \\ 4 & 24 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 6 & 1 \end{bmatrix} \otimes \begin{bmatrix} 4 & 4 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 24 & 24 \\ 4 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 6 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 6 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 6 & 1 \\ 6 & 36 & 6 \\ 1 & 6 & 1 \end{bmatrix}$$

These stencils also follow from the mask by taking every other coefficient on every other row, as illustrated for the second stencil:

$$\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ \boxed{4} & 16 & \boxed{24} & 16 & \boxed{4} \\ 6 & 24 & 36 & 24 & 6 \\ \boxed{4} & 16 & \boxed{24} & 16 & \boxed{4} \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

Note that there are in fact three unique stencils – the second and thirds ones are rotational variants. These three stencils are illustrated in Figure 4.

The first one describes the centroid of a quadrilateral face – the resulting new vertex is called a *face point*. The second stencil creates an *edge point*, whereas the third one *updates* the position of an old vertex, resulting in a *vertex point*.

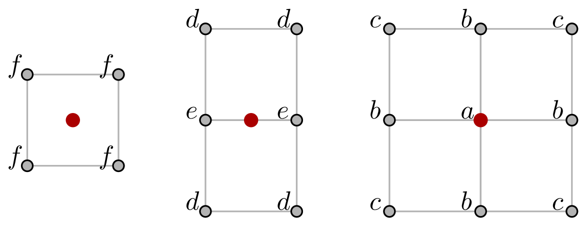

Fig. 4 *The three stencils, resulting in a new face point, a new edge point and an updated vertex point.*

To avoid any confusion, the coefficients d, e and f are just the values from the stencils:

$$d = \frac{4}{64} = \frac{1}{16}, \; e = \frac{24}{64} = \frac{3}{8}, \text{ and } f = \frac{16}{64} = \frac{1}{4}.$$

The same goes for a, b and c – we will come back to this later.

Similarly as for the univariate case, repeatedly applying the subdivision matrix (i.e. the one corresponding to the one-ring neighborhood) to a set of 3-by-3 neighboring control points results in convergence to a single point. This can be observed when S is constructed from the stencils and subsequently used to compute S^n. Every row of S consists of exactly one stencil, with its coefficients moved to the right places (see Figure 4).

$$S = \frac{1}{64}\left(\begin{array}{c|cccc|cccc} 36 & 6 & 6 & 6 & 6 & 1 & 1 & 1 & 1 \\ \hline 24 & 24 & 4 & 0 & 4 & 4 & 0 & 0 & 4 \\ 24 & 4 & 24 & 4 & 0 & 4 & 4 & 0 & 0 \\ 24 & 0 & 4 & 24 & 4 & 0 & 4 & 4 & 0 \\ 24 & 4 & 0 & 4 & 24 & 0 & 0 & 4 & 4 \\ \hline 16 & 16 & 16 & 0 & 0 & 16 & 0 & 0 & 0 \\ 16 & 0 & 16 & 16 & 0 & 0 & 16 & 0 & 0 \\ 16 & 0 & 0 & 16 & 16 & 0 & 0 & 16 & 0 \\ 16 & 16 & 0 & 0 & 16 & 0 & 0 & 0 & 16 \end{array}\right)$$

Again, we can use the eigendecomposition to find S^n for $n \to \infty$. Because of the rotational symmetry of the scheme, S is actually (apart from the first row and column) a *block-circulant* matrix containing 2×2 blocks of size 4×4. Using the 4×4 Discrete Fourier Transformation (DFT) matrix F_4 and its inverse F_4^{-1}, the eigenvalues of each 4×4 block M are easy to find: $F_4 M F_4^{-1} = \Lambda$ [12]. Note that F_4^{-1} is in fact the conjugate transpose of F_4, i.e. F_4^*. As an example, let us look at the first 4×4 block, i.e.

$$M = \frac{1}{64}\begin{pmatrix} 24 & 4 & 0 & 4 \\ 4 & 24 & 4 & 0 \\ 0 & 4 & 24 & 4 \\ 4 & 0 & 4 & 24 \end{pmatrix}.$$

The 4×4 DFT matrix F_4 is defined as

$$F_4 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -\mathrm{i} & -1 & \mathrm{i} \\ 1 & -1 & 1 & -1 \\ 1 & \mathrm{i} & -1 & -\mathrm{i} \end{pmatrix}$$

which leads us to the (diagonal) matrix of eigenvalues Λ

$$F_4 M F_4^* = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{3}{8} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{3}{8} \end{pmatrix}.$$

Calculating the eigenvalues of all the blocks at once can be done using the matrix $I_2 \otimes F_4$, where I_2 is the 2×2 identity matrix. Finally, pre-multiplying the resulting matrix by a permutation matrix W and post-multiplying it by W^{-1} it is possible to obtain a *block-diagonal* matrix. This matrix is still *similar* (i.e. it has the same eigenvalues) to the original block-circulant matrix. The eigenvalues of the block-diagonal matrix are readily computed, since they are the union of the eigenvalues of the individual blocks. More details on this process (i.e. involving the first row and column of S and finding the eigenvectors) can be found in [12, 34]. In the next section we will take a closer look at the limit surface.

In order to create a subdivision scheme that can be used on a non-regular mesh (containing general n-gons and *extraordinary* or *irregular* vertices), the subdivision rules need to be generalized. From Figure 4 it follows that the application of the first stencil results in the average value (the *face point*) of the 4 vertices defining the quadrilateral. An obvious choice is to generalize this to the average value of the n vertices defining an n-gon. An *edge point* can then be defined as the average value of the adjacent two new *face points* and the two old vertices constituting the edge.

We continue with the third stencil. This one seems rather problematic, considering that we have to allow both n-gons and extraordinary vertices at the same time. However, studying the results after one application of the first and second stencil, it is easy to see that all n-gons are quadrangulated. Figure 5 shows the result of this *primal quadrisection.*

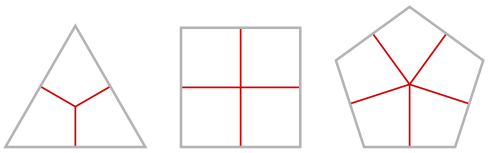
Fig. 5 *Quadrisection of a few different n-gons. Note that the valence of the new face point is n.*

Therefore, assuming that all faces are quads, we can focus on just the extraordinary points. We can recognize three types of vertices in the regular stencil (see Figure 4), so it makes sense to continue working with the three different coefficients a, b and c in the general case (see Figure 6).

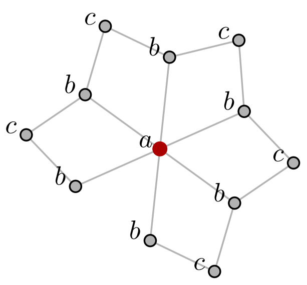

Fig. 6 *Catmull-Clark stencil for an extraordinary vertex.*

The optimal values for these coefficients depend on the valence of the central point. There have been many suggestions for these coefficients (mainly by using DFT to obtain the eigenstructure of the subdivision matrix of the *two-ring* neighborhood) – Catmull and Clark just generalized them from the geometrical interpretation of the third stencil [8], resulting in an intuitive scheme:

$$V_{new} = \frac{n-3}{n} V_{old} + \frac{1}{n^2} \sum F + \frac{2}{n^2} \sum M \qquad (1)$$

where F are the *face points* and M the midpoints of the incident edges of the vertex V_{old}. This results in the following expressions for the coefficients:

$$a_n = 1 - \frac{7}{4n}, \; b_n = \frac{3}{2n^2} \text{ and } c_n = \frac{1}{4n^2}.$$

Since it is sufficient to only consider quadrilaterals, the coefficients d, e and f are the same as before. Note that the relation in (1) is often written as

$$V_{new} = \frac{n-2}{n} V_{old} + \frac{1}{n^2} \sum F + \frac{1}{n^2} \sum P$$

where P are the vertices at the end of the incident edges of V_{old}. Additionally, it can also be written as

$$V_{new} = \frac{n-3}{n} V_{old} - \frac{1}{n^2} \sum F + \frac{4}{n^2} \sum E$$

where E are the *edge points* of the incident edges of V_{old} – it is quite suitable for implementation.

Using the first form in (1), it is particularly easy to apply *reverse* Catmull-Clark subdivision (that is, on a mesh that is the result of a regular Catmull-Clark subdivision), by isolating V_{old}:

$$V_{old} = \frac{1}{n-3} \left(nV_{new} - \frac{1}{n} \sum F - \frac{2}{n} \sum M \right)$$

The strategy is to find all *original* vertices with $n \neq 3$, which can only be done if it is known which current vertices are the *updated* vertices with respect to the coarser mesh (which can be taken care of by ordering the newly calculated face points, edge points and vertex points in a logical way). Finally, using the vertices already computed, the ones corresponding to $n = 3$ can be found. Note that this method might fail if there are too many (or even exclusively) old vertices with valence 3.

To conclude, it should be mentioned that there are tools available that allow the user to shape the resulting geometry by adding *sharpness* to the edges and vertices [14]. See Figure 7 for an example. The rules for these features derive from special stencils for boundary edges and vertices, which will be briefly discussed in the next section.

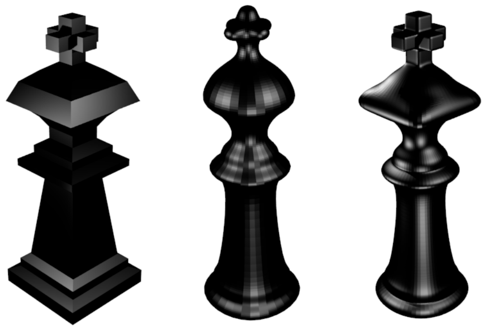
Fig. 7 *Left: Coarse model of a chess piece. Middle: Regular Catmull-Clark subdivision. Right: Catmull-Clark subdivision using selective sharpness. Created in* MATLAB*, rendered with Blender 2.6.*

4. The limit surface

As implicitly mentioned in the previous section, the limit surface corresponding to the regular faces in the control net consists of uniform bi-cubic B-Spline patches. However, we would also like to evaluate points on patches corresponding to non-regular faces in the control net – the faces containing an extraordinary vertex. Therefore, the first question is whether the subdivision matrices representing the one-ring neighborhoods of these extraordinary vertices have the same convergence behavior as for the regular vertices.

For a non-defective subdivision matrix, a necessary and sufficient condition for convergence to a single point is a simple (i.e. with multiplicity 1) dominant eigenvalue having value 1. When all entries of the matrix are non-negative, this is not difficult to check. Because the rows of the subdivision matrix represent affine combinations (i.e. each row sums up to unity), S has an eigenvector v_1 consisting of all ones, accompanied by an eigenvalue λ_1 having value 1. Gershgorin's circle theorem provides an upper bound of value 1 for all eigenvalues, whereas the Perron-Frobenius theorem for non-negative irreducible matrices guarantees a simple dominant eigenvalue. The one-ring neighborhood matrix S is irreducible because the entries of its first row and column are all positive, resulting in a strongly connected associated digraph.

Since the $(2v+1)\times(2v+1)$ subdivision matrix S for Catmull-Clark is non-defective for all valences v, we know that S has $2v+1$ linear independent eigenvectors. Saving the *right* eigenvectors v_i as columns in V, and the corresponding eigenvalues λ_i in a diagonal matrix Λ, we can write $SV = V\Lambda$. Because the columns of V are linearly independent, V^{-1} exists and thus we can write $V^{-1}S = \Lambda V^{-1}$, which means that every row of V^{-1} is in fact a *left* eigenvector w_i.

Repeatedly applying S to the vertices constituting the one-ring neighborhood of an extraordinary vertex leads to $Q = S^nP$, where P and Q are both vectors of $2v+1$ control points. This means that we can write P as a vector in the space spanned by the $2v+1$ eigenvectors v_i, so $P = \sum v_i h_i$ where h_i are row vectors containing the coefficients (recall that each control point in P has an x, y and z component). In matrix form this can be written as $P = VH$, and therefore the coefficients can be computed as $H = V^{-1}P$. This results in $Q = S^nP = S^n\sum v_ih_i = \sum S^nv_ih_i = \sum \lambda_i^n v_ih_i$. When $n \to \infty$, only $\lambda_1^n = 1$ remains, along with its accompanying eigenvector v_1 consisting of all ones. This means that the point of convergence is simply h_1, which is the product of the first row of V^{-1} (i.e. the left eigenvector w_1) and the original control points P.

Using Stam's approach [34], we assume that each face in the control net has *at most* one extraordinary vertex (which is easily satisfied by applying Catmull-Clark subdivision once or at most twice). A patch corresponding to a face with one extraordinary vertex is defined by $K = 2v + 1 + 7$ control points P_0, where v is the valence of the extraordinary vertex. This is illustrated Figure 8 for $v = 5$.

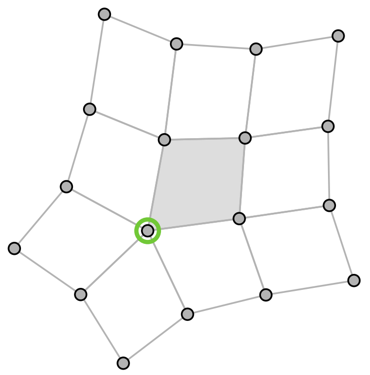

Fig. 8 *The K control points for the case $v = 5$. The extraordinary vertex of the face is highlighted in green.*

Now when we apply our subdivision scheme to these K control points, we end up with $M = K+9 = 2v + 1 + 7 + 9$ new ones. They describe the exact same patch, but looking a little closer, we observe that the old patch has been subdivided into *four* subpatches. Of these subpatches, *three* are regular – *which means that they can be evaluated!* – and *one* is non-regular, i.e. again corresponding to a face with one extraordinary point. See Figure 9.

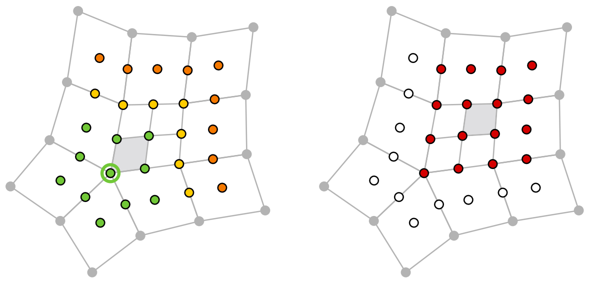

Fig. 9 *Left: The M new control points for the case $v = 5$ – the non-regular face is highlighted. The colors of the control points correspond to the matrices below. Right: The 16 control points $B_{2,1}$ defining a regular subpatch highlighted in red.*

The non-regular subpatch is of course again defined by K control points P_1, calculated as $P_1 = AP_0$, whereas the M control points $\bar{P}_1$ defining all four subpatches (constituting one patch of the previous level) are obtained via $\bar{P}_1 = \bar{A}P_0$. Both A and $\bar{A}$ are so-called *extended* subdivision matrices:

$$A = \begin{pmatrix} S & 0 \\ S_{11} & S_{12} \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} S & 0 \\ S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}$$

The matrices S_{ij} are comprised of regular stencils (i.e. corresponding to the case $v = 4$) [34]. It follows that $P_n = A^n P_0$ and $\bar{P}_n = \bar{A} A^{n-1} P_0$.

Repeating the step above results in a tiling of the parameter domain $\Omega = [0,1] \times [0,1]$. Each tile Ω_k^n corresponds to a subdivision level n and a sector k as depicted below in Figure 10.

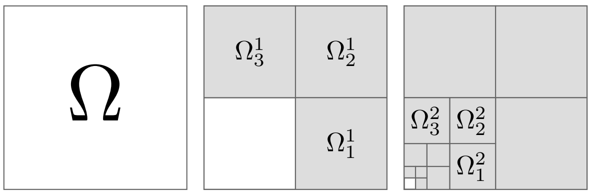

Fig. 10 *The unit square Ω decomposed into tiles Ω_k^n, where $k \in \{1,2,3\}$ and $n \in \mathbb{N}^+$.*

The 16 control points $B_{k,n}$ for a subpatch at level n in sector $k \in \{1,2,3\}$ are obtained from the M control points in $\bar{P}_n$ by using a $(16 \times M)$ *picking matrix* X_k, i.e. $B_{k,n} = X_k \bar{P}_n$. The subpatch corresponding to the tile Ω_k^n is therefore described by

$$s_{k,n}(s,t) = B_{k,n}^T b(s,t) = \bar{P}_n^T X_k^T b(s,t) \qquad (2)$$

where $b(s,t)$ are the regular bivariate cubic B-Splines. Of course we don't compute $\bar{P}_n$ by repeatedly applying A and then $\bar{A}$ – these control points are computed *directly* from the original control points P_0. We like to call this approach *n steps of virtual subdivision.* Using $A = V \Lambda V^{-1}$ we can write

$$\bar{P}_n = \bar{A} V \Lambda^{n-1} V^{-1} P_0 \qquad (3)$$

which subsequently leads to the expression

$$s_{k,n}(s,t) = \hat{P}_0^T \Lambda^{n-1} \left(X_k \bar{A} V \right)^T b(s,t). \qquad (4)$$

where $\hat{P}_0 = V^{-1} P_0$, i.e. the coefficients for writing P_0 as a vector in the space spanned by eigenvectors (see earlier in this section).

A point $(u,v) \in \Omega$ on an extraordinary patch is thus computed as follows. First, the maximum of u and v is determined as $m = \max(u,v)$. If $m = 0$, we return the point of convergence h_1 as described above. Otherwise, the virtual subdivision level n is determined by $n = \text{ceil}(-\log_2(m))$. In order to determine $k \in \{1,2,3\}$, we use n to scale (u,v) by a factor 2^{n-1}. Together, n and k determine the tile Ω_k^n corresponding to a certain subpatch. Since the B-Spline basis functions are defined on Ω, we have to map Ω_k^n to Ω in order to obtain the right point of evaluation (s,t). Take for example $(u,v) = (.35, .20)$, which results in $n = 2$. Scaling by $2^{n-1} = 2$ results in $(\bar{u},\bar{v}) = (.70, .40)$, so $k = 1$. The point (s,t) then follows from a simple transformation, in this case $(s,t) = (2\bar{u} - 1, 2\bar{v}) = (.40, .80)$. See Figure 11.

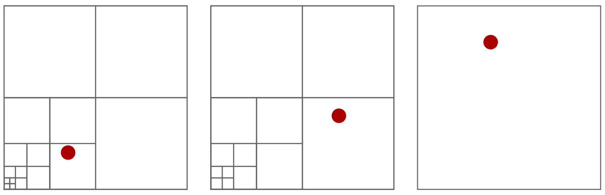

Fig. 11 *The point $(u,v) = (.35,.20)$ transformed to $(\bar{u},\bar{v}) = (.70,.40)$ transformed to $(s,t) = (.40,.80)$.*

At first sight this might seem as lot of required computations. However, since most of these steps don't depend on the value of n (i.e. the number of virtual subdivisions) or the set of control points P_0, they can be pre-calculated. For each valence v we could therefore compute the K eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_K)$, the matrix V^{-1} and the matrix $D_k = (X_k \bar{A} V)^T$ for $k \in \{1,2,3\}$, and save them to a file. Note that the dimensions of D_k are $(K \times 16)$. A more concise notation to express a point on the patch associated with tile Ω_k^n is therefore

$$s_{k,n}(s,t) = \hat{P}_0^T \Lambda^{n-1} D_k b(s,t). \qquad (5)$$

In practice it means that this data has to be loaded from a file before evaluating a mesh (the control polygon). Next, for each non-regular face in the mesh, described by the control points P_0, the coefficients $\hat{P}_0$ have to be determined *only once*. Any point $(u,v) \in \Omega$ on the associated patch can then be transformed to the correct point of evaluation (s,t) and evaluated using the relation in (5). Although it is possible to use (5) to evaluate patches associated with regular faces in the mesh, it is slightly faster to directly use the B-Spline basis functions $b(u,v)$ for this case.

At this point, we can evaluate any point of a patch associated with an *internal* face of the mesh. However, a mesh does not have to be a *closed manifold* – there could be a boundary. Furthermore, the boundary faces could be either regular or non-regular. Because our geometry of interest could be modeled in a way without any non-regular boundary faces (see Figure 12), we have not studied the evaluation of such patches in full detail. In short, the main difficulty lies in the fact that the associated subdivision matrices are often defective (i.e. the eigenvectors are not linearly independent). This means that S is not diagonalizable as $V \Lambda V^{-1}$, which complicates the calculation of S^n. Fortunately there is another kind of matrix decomposition called the *Jordan normal form*, which can be used to compute S^n. The subdivision matrix can then be written as $S = W J W^{-1}$, where J is a block-diagonal matrix (containing m blocks where m is the number of linearly independent eigenvectors) and W the matrix containing the *generalized* eigenvectors as columns.

Note that the same approach can be used for internal edges with sharp edges or vertices – see [33] for more details.

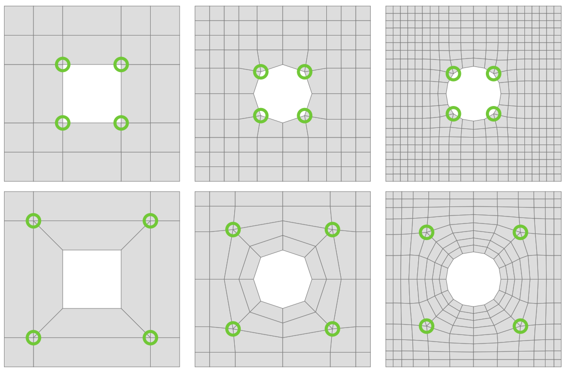

Fig. 12 *The types and locations of extraordinary vertices (highlighted) might depend on the modeling method. The coarse meshes are depicted on the left, followed by two iterations of Catmull-Clark subdivision.*

The boundary faces of a mesh require special stencils for the calculation of both edge-points and vertex-points (face points don't pose a problem). An intuitive way to look at this problem is to decompose the mesh into its boundary curve and its inner part, see Figure 13.

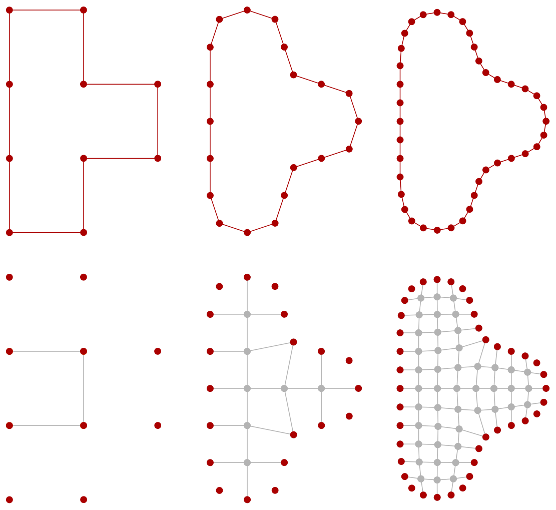

Fig. 13 *Simple geometry decomposed into its boundary curve and inner part, followed by two iterations of Catmull-Clark.*

It is then easy to see that we can use the univariate stencils for the boundary – after all, it is a *curve*. This means that an edge point on the boundary is calculated as the mean value of its two neighboring vertices (i.e. using the stencil [4 4]), whereas a vertex point is calculated using the stencil [1 6 1]. Strictly speaking, there is one additional concern – application of the regular stencil for an edge-point comprising two boundary faces could lead to folds in the resulting surface (e.g. if the mesh contains a concave section). In [4], an improved stencil is suggested.

In case of a regular boundary face, which is defined by 12 control points (see Figure 18), we can compute 20 new control points using a 20×12 subdivision matrix $\bar{E}$. These 20 points can then be used to evaluate *two* subpatches. The control points defining a single subpatch are obtained by using the 16×20 picking matrix X_L or X_R. See Figure 14.

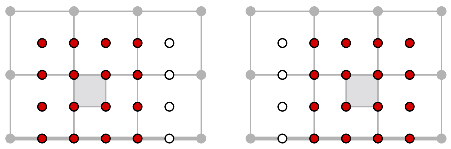

Fig. 14 *Application of the matrix* $\bar{E}$ *results in* 20 *new control points, defining two subpatches.*

Note that these two subpatches only constitute $^1/_2$ of the original patch, whereas for the internal case, $^3/_4$ of the original patch could be evaluated after applying the matrix $\bar{A}$. This leads to a different tiling of the domain Ω, as is depicted in Figure 15.

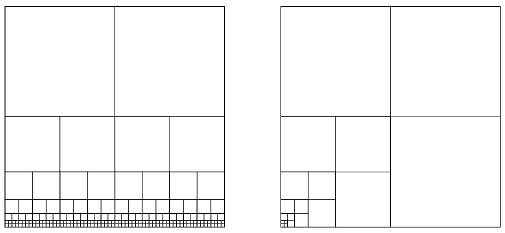

Fig. 15 *Tiling of the domain* Ω *for the regular boundary case compared to the internal case. Note that the latter has a constant number of three subpatches per level.*

It is clear that in this case the number of subpatches at level n is *not* constant, but 2^n. In order to compute the control points defining a subpatch Ω_k^n, we therefore have to use *two* different 12×12 matrices E_L and E_R. The subdivision matrix E_L computes the 12 left-most control points, whereas E_R computes the 12 right-most ones. See Figure 16.

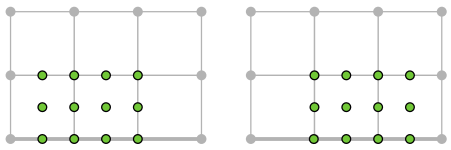

Fig. 16 *Application of the subdivision matrices* E_L *and* E_R *to the original control points.*

It follows that the control points defining a *group of two subpatches* at level n are found by repeatedly applying E_L and E_R in an alternating fashion, followed by one application of $\bar{E}$. The alternating pattern can be derived from the binary tree associated with the tiling depicted in Figure 15. A left branch in this tree corresponds to the application of A_L, a right branch to the application of A_R. Note that each *leaf* in the tree corresponds to a set of 12 control points, i.e. the situation depicted in Figure 14. As example, the control points $\bar{P}_{n=5}$ defining the group of two subpatches highlighted in Figure 17 are found by $\bar{P}_5 = \bar{E}E_RE_LE_LE_RP_0$ (compare this to the relation in (3)). Subsequently applying X_L or X_R results in the control points defining a single subpatch Ω_k^n.

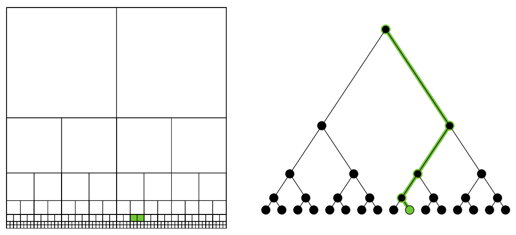

Fig. 17 *The sequence of E_L and E_R follows from the path in the associated binary tree.*

Regrettably, this approach is rather inefficient. Using the eigendecomposition of E_L and E_R is of no use, since they appear in an arbitrary alternating pattern. In addition, the case of a corner face is even more complicated! The tiling for this case reveals a recursive relation for $\#_n$, the number of subpatches at level n. Starting with $\#_1 = 1$, the relation reads $\#_n = 3 + 2\#_{n-1}$. The resulting associated tree is a combination of a binary and a ternary tree.

Fortunately there is a much easier method for evaluating points on patches associated with a boundary or a corner face. This technique, which is briefly mentioned in [21, 33], makes use of so called *phantom* control points. For the regular boundary case, the idea is to combine the original 12 control points with 4 phantom points, such that the regular patch $s(u, v)$ corresponding to these 16 control points is precisely the same as the boundary patch. The key for finding these phantom points is to equate the boundary curve of the boundary patch to the isoparameter curve $s(u, 0)$ of the regular patch. This results in the relation $P_i = 2P_{i+4} - P_{i+8}$ for $i \in \{1, 2, 3, 4\}$, as illustrated in Figure 18. If a *coordinate-free* approach [13] is used, this relation should be rewritten to $P_i = P_{i+4} + (P_{i+4} - P_{i+8})$.

An equivalent way of deriving these phantom points is by equating the results from the stencils applied on the boundary, i.e. the results from the bivariate stencils should equal the results from the univariate stencils. As example, the bivariate edge-point stencil [4 4; 24 24; 4 4] applied to P_1, P_2, P_5, P_6, P_9 and P_{10} should have the same result as the univariate edge-point stencil [4 4] applied to P_5 and P_6. See Figure 18.

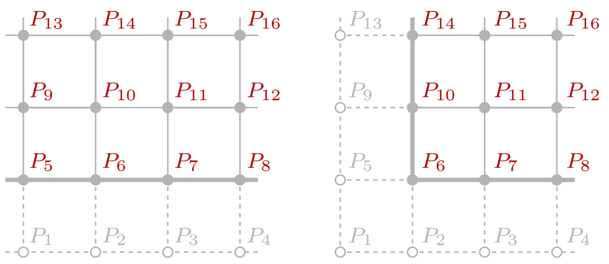
P_{13} P_{14} P_{15} P_{16}
P_9 P_{10} P_{11} P_{12}
P_5 P_6 P_7 P_8
P_1 P_2 P_3 P_4
P_{13} P_{14} P_{15} P_{16}
P_9 P_{10} P_{11} P_{12}
P_5 P_6 P_7 P_8
P_1 P_2 P_3 P_4

Fig. 18 *A boundary face and a double boundary (corner) face. The boundary is indicated by the thick line.*

A similar approach can be followed for a corner face, but *only* when we assume that the corner vertex P_6 is marked as *sharp*. This means that its position is fixed, resulting in the interpolation of this point. The phantom points P_2, P_3 and P_4 can then be calculated from $P_i = 2P_{i+4} - P_{i+8}$ and the phantom points P_5, P_9 and P_{13} from $P_i = 2P_{i+1} - P_{i+2}$. Finally, the phantom point P_1 follows from equating the point $s(0, 0)$ of the regular patch to the control point P_6, resulting in $P_1 = 4P_7 + 4P_{10} + P_{11} - 8P_6$, or $P_1 = P_{11} + 4(P_7 - P_6) + 4(P_{10} - P_6)$ for a coordinate-free approach. See Figure 18.

To conclude, note that we are in fact using only 12 basis functions for the evaluation of regular boundary patches, and 9 basis functions for the corner patches. In a later section we will look at these basis functions in some more detail.

5. IsoGeometric Analysis in a nutshell

IsoGeometric Analysis (IGA) is a relatively new form of FEA that is currently receiving a lot of interest in the academic world. The concepts of the technique were first discussed in [20], and later in the book [11]. IGA ultimately attempts to unite the worlds of CAGD and FEA – the similarities between the two are abundant, but the different terminology still causes a lot of confusion.

The central notion of IGA are *basis functions*. As we have seen, geometries can be modeled using basis functions (either directly by using NURBS, or indirectly by using subdivision surfaces). The key observation is that IGA uses the same basis functions for *both* the geometry and the analysis. This implies that we don't need an expensive mesh generator anymore, since the mesh now *follows* from the geometry. Note that although the *isoparametric* concept (which is often used in FEA) might sound similar, it is fundamentally different.

Since a geometry modeled with Catmull-Clark surfaces is C^2 continuous at regular parts and C^1 continuous near non-regular parts, the basis functions are suitable to use for analysis. Note that subdivision surfaces have been combined with analysis before, for instance in [10].

With regard to the analysis, the basis functions span the *trial space* – the approximation to the solution of the differential equation to be solved is an element of this space. Because of the continuity across the element boundaries, the approximation will have a higher continuity than is the case in classical FEA (which commonly uses the Lagrange basis functions).

The employment of IGA in conjunction with subdivision surfaces is demonstrated in the next section through the use of an example.

6. IGA and subdivision surfaces

We start this section with a recap of the different basis functions. Figure 19 illustrates the regular uniform cubic B-Splines $B_r(u)$, along with the basis functions $B_b(u)$ for the boundary case.

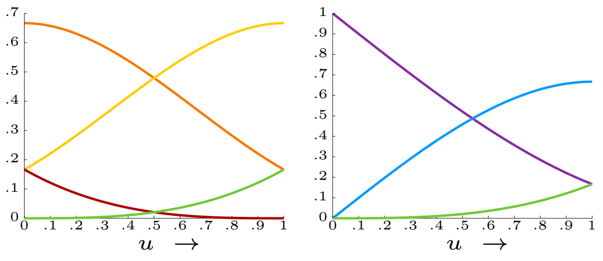

Fig. 19 *The basis functions $B_r(u)$ and $B_b(u)$. Notice that both sets of basis functions form a partition of unity, i.e. $\sum B_r(u) = 1$ and $\sum B_b(u) = 1$.*

The latter basis functions can be obtained as follows. It is given that a single column of three control points P (based on Figure 18), accompanied by three *unknown* basis functions $B_b(u)$, describe a certain curve $P^T B_b(u)$. Furthermore, we know that the four control points Q, accompanied by the *known* basis functions $B_r(u)$, describe the exact *same* curve $Q^T B_r(u)$ – note that the first control point Q_1 is a phantom point. The matrix C, such that $Q = CP$, is readily derived as

$$C = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad C^T = \begin{pmatrix} 2 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

As explained in the section on subdivision curves, we can then use C^T to relate the basis functions $B_b(u)$ to $B_r(u)$ as $B_b(u) = C^T B_r(u)$.

The bivariate basis functions now follow as set out below:

Regular patches
The regular bivariate cubic B-Splines, defined by the tensor product $B_{rr}(u,v) = B_r(u) \otimes B_r(v)$.

Boundary patches
The tensor product $B_{rb}(u,v) = B_r(u) \otimes B_b(v)$. Note that this operation is *not* commutative!

Corner patches
The tensor product $B_{bb}(u,v) = B_b(u) \otimes B_b(v)$. Note that the corner vertex will be interpolated.

Non-regular patches
Rewriting the equations (2) and (3) as $s_{k,n}(s,t) = P_0^T V^{-T} \Lambda^{n-1} V^T \bar{A}^T X_k^T b(s,t)$, it follows that these basis functions – *restricted* to a tile Ω_k^n – are described by $B_{nr}(u,v)\big|_{\Omega_k^n} = V^{-T} \Lambda^{n-1} V^T \bar{A}^T X_k^T b(s,t)$.

Switching to Finite Element Analysis, we can now set up the abstract Bubnov-Galerkin setting (G):

$$(G) \begin{cases} \text{Find } u^h \in S^h \text{ such that} \\ B(w^h, u^h) = l(w^h) \ \forall w^h \in V^h \end{cases}$$

where V^h represents our discrete test space and S^h the discrete trial space constructed from V^h as $S^h = \{v^h + q^h | v^h \in V^h\}$, where the lifting function q^h satisfies the eventual Dirichlet boundary conditions. Note that the operators $B(w^h, v^h)$ and $l(w^h)$ are, respectively, the *bilinear* form and the *linear* form.

In our case the space V^h is spanned by the basis functions $B_i(u,v)$ discussed above, so $V^h = \text{span}\,(B_i(u,v))$.

Let us move on to a specific problem. Our geometry or interest is based on Figure 12, and is chosen because it contains all four types of patches described above. It represents a *block of material* including some imperfections (e.g. grains) – see Figure 20.

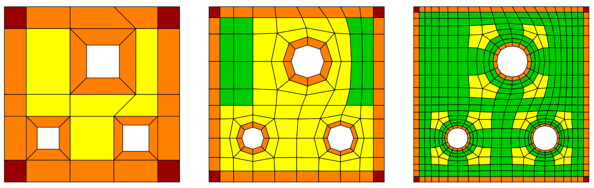

Fig. 20 *The original geometry followed by two iterations of Catmull-Clark. The color green corresponds to regular patches, yellow to extraordinary patches, orange to boundary patches and red to corner patches.*

The initial mesh clearly contains faces with *more* than one extraordinary vertex, so we apply an iteration of Catmull-Clark subdivision. After this single iteration the conditions for evaluating the limit surface are satisfied, see Figure 21.

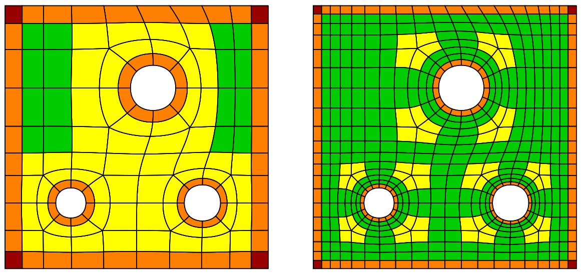

Fig. 21 *The limit surface after one and two iterations of the Catmull-Clark algorithm.*

For our analysis we consider isotropic linear elastic behavior:

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} = \mathbf{0} & \text{on } \Omega \\ \boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda \operatorname{tr}(\boldsymbol{\varepsilon})\, \boldsymbol{I} \\ \boldsymbol{\varepsilon} = \nabla^s \mathbf{u} \\ \mathbf{u} = (\boldsymbol{F}^m - \boldsymbol{I})\,\mathbf{x} + \tilde{\mathbf{u}} & \text{on } \Gamma \end{cases}$$

where μ and λ are the Lamé parameters using $\nu = .3$ and $E = 1$. $\boldsymbol{F}^m$ is the *macroscopic* deformation gradient, defined as

$$\boldsymbol{F}^m = \begin{pmatrix} 1 & .05 \\ .05 & 1 \end{pmatrix}$$

which means that we shear the block along the $x = y$ diagonal. Finally, $\tilde{\mathbf{u}}$ represents a *microscopic* fluctuation which is applied *periodically* on both the left and right sides and the top and bottom sides – after all our geometry is just a small block of material that is eventually repeated in all directions, thus constituting a larger unit.

Note that the description of the model can be written in the abstract form mentioned above (see for example [19]).

After solving for the displacement $\mathbf{u}$, we can compute the strains $\boldsymbol{\varepsilon}$ and stresses $\boldsymbol{\sigma}$. The stress field in the x-direction, $\boldsymbol{\sigma}_{xx}$, is plotted in Figure 22.

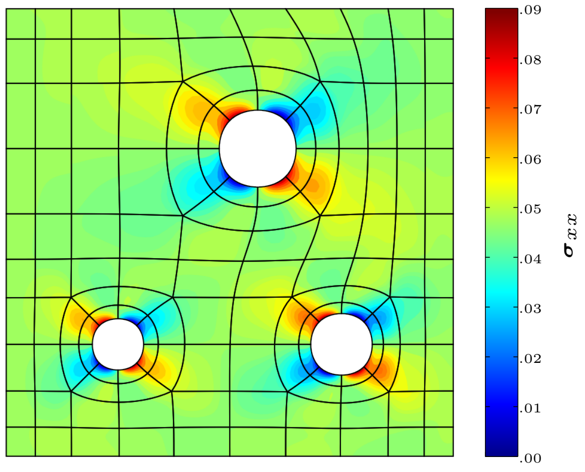

Fig. 22 *Plot of $\boldsymbol{\sigma}_{xx}$ for the coarse case. A vector version is available online (click on the plot).*

To conclude, we apply the same analysis on a globally refinement geometry. Since the test space (and therefore the trial space) is now *richer*, we obtain an improved description of the stress field – see Figure 23.

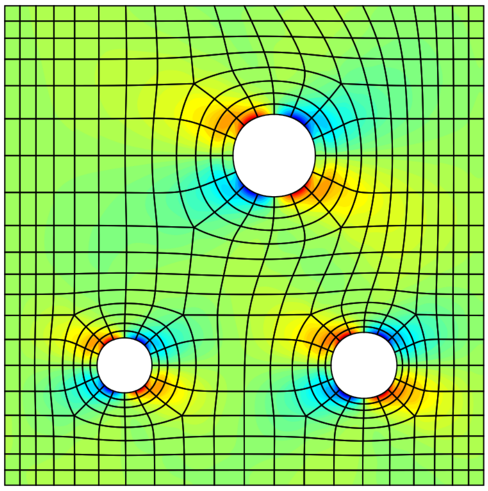

Fig. 23 *Plot of $\boldsymbol{\sigma}_{xx}$ for the globally refined case. A vector version is available online.*

7. Implementation

We conclude this report with a short overview of the workflow. The first step in the process was the implementation of the Catmull-Clark scheme. Although there are several good implementations available (see for example CGAL and OpenMesh), we chose to implement the scheme in MATLAB ourselves, inter alia for a better comprehension of the scheme. After a rather inefficient attempt using a face-vertex mesh, we switched to a half-edge mesh. The half-edge data structure provides easy access to neighboring components.

There are several ways to implement subdivision. An existing mesh can either be modified (by adding components to the mesh) or completely rewritten into a new mesh. Moreover, subdivision schemes can be split into refinement operators and smoothing operators, leading to a so-called *factored* approach [36]. The other method is to apply each stencil (representing both operators) at once. Our main implementation uses the rewriting method by directly applying the stencils. In addition, due to some overlap with another project, the schemes were also implemented using stellar operators (see [2] for the accompanying report). It uses the modifying method, and by the nature of the stellar operators [35], the splitting approach.

Next up was the implementation of an import script for .OBJ files, exported from BLENDER. Since .OBJ uses a face-vertex format to store a mesh, we had to write a script to convert it to a half-edge mesh. Additionally we added an export script for .OBJ to be able to render our results in Blender.

Over time the implementation in Matlab was extended to an interactive environment, in order to be able to indicate the sharpness of both vertices and edges. We also took a look at other subdivision schemes (e.g. Doo-Sabin, Loop, Kobbelt's $\sqrt{3}$ and Velho's $4-8$ scheme) which are not discussed in this report. Functionality for reverse Catmull-Clark was added later.

An interesting topic for future research is studying the difference between CPU and GPU implementations (either using OpenGL, OpenCL or CUDA).

8. Conclusion

After a succinct discussion of subdivision schemes, the limit surface and IGA, the example demonstrated that the two topics go together quite well. Only a more dynamic and generally applicable workflow can provide a decisive answer, but these early signs certainly look promising. In the conclusion of my MSc thesis I hope to give a somewhat more extensive review.

Intriguing future research topics include the extension of the discussed concepts to general volumetric subdivision methods (e.g. box spline solids). Taking a detailed look at convergence rates and comparing them to other numerical methods might also be of interest.

Additionally, implementing adaptive mesh refinement should be very useful. Triangular schemes are probably the most suitable for this purpose, although the extension to volumes might be slightly problematic (subdivision of a tetrahedron is not completely trivial).

Another option would be to implement an alternative method to Stam's approach like the one presented in [32] and subsequently compare the two methods (e.g. in the extend of efficiency). An advantage of this alternative method is that it works for any subdivision scheme, not just for polynomial ones. A possible drawback might be the fact that the limit surface can only be evaluated at points on a predefined grid.

Some possible future directions with regard to the implementation were already mentioned in the previous section. In addition it might be practical to make use of the Python interface of Blender, which provides the possibility to access the mesh directly. The recent switch to BMesh makes this idea even more appealing.

Acknowledgements

First and foremost I would like to thank my supervisor Clemens Verhoosel for his close involvement in this project. Additionally my gratitude goes to Göktürk Kuru for the many fruitful discussions we had. Finally a word of thanks to the Inkscape and Blender communities and the people at Math.StackExchange!

References

[1] L. Andersson and N. Stewart. *Introduction to the mathematics of subdivision surfaces.* Society for Industrial and Applied Mathematics (SIAM), 2010.

[2] P. Barendrecht. Describing subdivision schemes with Lindenmayer systems. Eindhoven University of Technology, 2012.

[3] K. Bathe. *Finite element procedures.* Prentice hall Englewood Cliffs, NJ, 1996.

[4] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 113–120. ACM Press/Addison-Wesley Publishing Co., 2000.

[5] D. Burkhart, B. Hamann, and G. Umlauf. Isogeometric finite element analysis based on Catmull-Clark subdivision solids. In *Computer Graphics Forum*, volume 29, pages 1575–1584. Wiley Online Library, 2010.

[6] T. Cashman. *NURBS-compatible subdivision surfaces.* PhD thesis, University of Cambridge, 2010.

[7] T. Cashman. Beyond catmull–clark? A survey of advances in subdivision surface methods. In *Computer Graphics Forum.* Wiley Online Library, 2012.

[8] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978.

[9] G. Chaikin. An algorithm for high-speed curve generation. *Computer graphics and image processing*, 3(4):346–349, 1974.

[10] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin-shell finite element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000.

[11] J. Cottrell, T. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA.* Wiley, 2009.

[12] P. Davis. *Circulant matrices.* Chelsea Publishing Company, 1994.

[13] T. DeRose. A coordinate-free approach to geometric programming. In *Theory and practice of geometric modeling*, pages 291–305. Springer-Verlag New York, Inc., 1989.

[14] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94. ACM, 1998.

[15] N. Dodgson, U. Augsdörfer, T. Cashman, and M. Sabin. Deriving box-spline subdivision schemes. *Mathematics of Surfaces XIII*, pages 106–123, 2009.

[16] D. Doo and M. Sabin. Behaviour of recursive subdivision surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

[17] G. Farin. *NURBS: from projective geometry to practical use.* AK Peters, Ltd., 1999.

[18] R. Goldman. *Pyramid algorithms: A dynamic programming approach to curves and surfaces for geometric modeling.* Morgan Kaufmann Pub, 2003.

[19] T. Hughes. *The finite element method: Linear static and dynamic finite element analysis.* Dover Publications, 2000.

[20] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39):4135–4195, 2005.

[21] D. Lacewell and B. Burley. Exact evaluation of Catmull-Clark subdivision surfaces near B-spline boundaries. *Journal of Graphics, GPU, and Game Tools*, 12(3):7–15, 2007.

[22] J. Lane and R. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):35–46, 1980.

[23] D. Levin. Using Laurent polynomial representation for the analysis of non-uniform binary subdivision schemes. *Advances in Computational Mathematics*, 11(1):41–54, 1999.

[24] W. Ma. Subdivision surfaces for CAD – an overview. *Computer-Aided Design*, 37(7):693–709, 2005.

[25] S. Mann. A blossoming development of splines. *Synthesis Lectures on Computer Graphics and Animation*, 1(1):1–108, 2006.

[26] U. Masami and S. Lodha. Wavelets: An elementary introduction and examples. 1995.

[27] L. Ramshaw. *Blossoming: A connect-the-dots approach to splines.* Digital Systems Research Center, 1987.

[28] B. Reddy. *Introductory functional analysis: with applications to boundary value problems and finite elements.* Springer, 1997.

[29] M. Sabin. Recent progress in subdivision: a survey. *Advances in Multiresolution for Geometric Modelling*, pages 203–230, 2005.

[30] M. Sabin. *Analysis and Design of Univariate Subdivision Schemes*, volume 6. Springer, 2010.

[31] D. Salomon. *Curves and surfaces for computer graphics.* Springer, 2006.

[32] S. Schaefer and J. Warren. Exact evaluation of non-polynomial subdivision schemes at rational parameter values. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pages 321–330. IEEE, 2007.

[33] J. Smith, D. Epps, and C. Séquin. Exact evaluation of piecewise smooth catmull-clark surfaces using Jordan blocks. 2004.

[34] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 395–404. ACM, 1998.

[35] L. Velho. Stellar subdivision grammars. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 188–199. Eurographics Association, 2003.

[36] J. Warren and S. Schaefer. A factored approach to subdivision surfaces. *Computer Graphics and Applications, IEEE*, 24(3):74–81, 2004.