Basis Functions, IGA and Subdivision

Malcolm Sabin

These notes are a more readable form of the presentation I gave at the ITI conference in September 2017. They have three main sections: the first provides a little terminology and the theory for a better way of looking at the analysis process: the second looks at the IGA claims and supports two of them: the third looks at whether we should be adding subdivision surfaces to the toolkit for shape definition and analysis.

Basis Functions

The context is **Field Analysis**, i.e. the solution of some PDE over an interestingly shaped domain.

When Finite Elements congealed as an academic subject in the late 1950's/early 1960's it was thought that the method was based on splitting the domain into pieces for which the variation of the field would be simple enough that it could be approximated reasonably accurately: hence the name.

It is now more appropriate to forget this idea: the real purpose of doing the splitting was to enable the easy construction of fairly good basis functions (see below) and we should now focus on that purpose, not on the means to that end.

What is a basis function about ?

Actually what we need is not just one basis function* but a **basis**, which is a set of **basis functions** which we shall denote by $\phi_i(P)$, i.e. scalar functions of position in the domain. If this set of functions satisy certain conditions (see below) then they can be used to represent a field over the domain by using the linear form

$$F(P) = \Sigma_i \, F_i \, \phi_i(P)$$

The field can be of many different kinds: displacement vectors for elasticity, electromagnetic fields, a scalar for thermal,..., so the field F at position P is given by the sum of a set of coefficients (of the same type as the field) times the basis functions.

* though over a regular grid regular copies of the same function provide a good way of analysing things

Pretty well all of our analysis techniques fit this pattern.

- if the basis functions are each zero except within some small connected part of the domain we call this Finite Elements
- if the basis functions are themselves solutions of the PDE we call this Boundary Elements
- if the basis functions use local physics to try to match singularities we call it XFEM
- if the basis functions are defined over a regular cartesian grid, then we have methods which are called 'ambient', 'embedded', or 'immersed', depending on how the boundaries of the domain are treated.

Exactly the same idea is used in parametric surface definition, where the domain is now the 2D parameter space, and the coefficients are called 'control points'.

How do they help ?

In all of these methods we can make a linear system for finding the F_i to approximate the PDE solution that we are looking for. There are at least three ways of doing so:

- Galerkin (also known as the 'weak form')

  Here an 'energy', E, is expressed which, when minimised gives the PDE solution, and we solve the linear system

  $$dE/dF_i = 0$$

  The matrix turns out to contain terms which are integrals over the domain of products of basis function derivatives. The attraction is that these derivatives are typically of lower degree than the PDE.
- Collocation, where the PDE expression is forced to take the right value at well-chosen points in the domain. This involves basis function derivatives of the same degree as the PDE. This doesn't involve products and so the matrix condition number is better.
- Least squares collocation, where instead of risking the collocation solution oscillating in between the collocation points, the sum of the squares of the errors in fit at a larger selection of points is minimised. The condition number gets squared again.

Conditions on the basis

The $\phi_i(P)$ do have to satisfy a few conditions if any of these methods is going to work. Call the domain Δ.

A commonly demanded condition is **summation to unity**.

$$\forall P \in \Delta,\ \Sigma_i\, \phi_i(P) = 1$$

but in fact this is only a sufficient condition for convergent solutions. A much sharper condition is

$$\exists \{a_i \in \Re\}, \forall P \in \Delta,\ \Sigma_i\, a_i\, \phi_i(P) = 1$$

where some of the a_i can be zero in this condition. The others can be set equal to unity by just scaling the $\phi_i(P)$, and this is a good idea from the point of view of well-conditioning.

A second condition is like unto it

$$\exists \{P_i \in \Re\}, \forall P \in \Delta,\ \Sigma_i\, P_i\, \phi_i(P) = P$$

This is a necessary condition for the **patch test** in elasticity theory, which is that if boundary conditions are set up to match a situation of constant strain, (linearly varying displacement) then the solution should be exactly that. The fact that this is automatically satisfied when the same basis is used for position as for field was a big advantage for *IsoParametric Elements*.

Criteria for choosing a good basis

If there are other known solutions for the PDE, then in principle we could set up higher order conditions, but quadratically varying displacement is not necessarily a solution of the elasticity PDE, so we haven't. What we actually want is **a good trade-off between accuracy and compute time**.

Properties such as

- support size
- continuity
- convergence rate

are relevant, but one which is just being recognised is **nestedness**. This is what supports adaptivity, whereby better solutions can be achieved relatively cheaply by adding extra functions into the basis in places where the accuracy isn't quite good enough. This has been known about for years in the academic FE community, but the big commercial FE codes don't support it because it is hard to do if you start by thinking about how to split the domain. When you start by thinking about the basis it becomes a lot easier, so this is going to change.

Iso Geometric Analysis

The IGA bandwaggon claims that if we use the same functions as a basis for analysis that we use for definition of geometry, then the world will suddenly become marvellous. There are three sub-arguments, of which only the first is nonsense.

(i) If you can use the same (NURBS) basis functions as the CAD model, then you can sidestep the whole of the meshing problem, which was only about making basis functions anyway. Just use the ones the designer already built.

This is very attractive, (particularly to managers), because the lead time taken by meshing is becoming a dominant cost. Unfortunately it isn't very practical

- because typical NURBS definitions are too dense: you can't afford a model with that much freedom.
- because typical NURBS definitions contain singularities because the designers have never been shown how to make good models
- because typical NURBS definitions are trimmed, and we don't know how to make the corresponding analysis work.
- because a good basis is one which matches the way the field flows, rather than the shape of the surface.
- because the CAD model has a whole load of irrelevant detail which any analyst will want removed.
- because the NURBS definitions are only surfaces, so although we could use these for shell elements, we can't do solids, where the real cost of meshing hits us.

(ii) Basis functions based on B-splines are really good for modelling smooth fields. The classic finite elements, based on polynomials in individual little bits of the domain are typically only $C0$. This means that although the field value matches across an inter-element frontier, the derivatives don't. If the PDE is elliptic, then the true field has no discontinuities of any derivative in the interior, and so freedoms which allow the representation of such discontinuities are basically being wasted. A simple example is that avoiding discontinuities of first derivative by using triquadratic B-splines instead of HE20 serendipity elements in a large solid domain gives a reduction of factorisation time in an explicit solution of a factor of 30. Two minutes instead of an hour, overnight instead of a week.

I have seen academic projects where extremely good results have been produced. These were called IGA, but the basis was chosen to suit the analysis, not downloaded from a CAD file.

(iii) If a basis is chosen for the analysis at an early stage in the design, and used also for the shape, which is then optimised for some design objective, then exporting it to the CAD system for the addition of all that detail which is actually necessary as part of the product, will be straightforward and relatively error-free.

Subdivision Surfaces

Subdivision surfaces (in particular a variant of Catmull-Clark) have become the de-facto standard for computer animation when non-trivial smooth shapes are required.

They have not become available in CAD packages except in a few which focus on styling. There is an apparent question, *'Why doesn't CAD use subdivision ?'* and corollaries *'Should it ?'* and *'Is it time to consider the possibility ?'*.

The increasing importance of analysis and optimisation in design suggest that the answer to at least the last question should be *Yes!*. The argument goes that

- Trimmed NURBS have problems in CAD

  and even more in CAE
- Maybe subdivision surfaces would be better ?
- But subdivision surfaces are not C2..
- Let's look at some facts and see what the real arguments are.

Ths part of this note therefore

(i) tries to identify exactly what the problems are

(ii) tries to see how subdivision might provide some solutions

(iii) tries to identify what problems there still are in the use of subdivision as first class surface entities in CAD.

NURBS problems

NURBS are really good if you have

- a rectangular surface shape
- whose boundaries are aligned with the flow of the shape in the interior
- and you can resist the temptation to use lots of knots*

Examples include - a wing before tip, flaps and slats and engine pylons are added, - a fuselage aft of the cockpit area, - ...

They are not fine if the alignment of the knot lines with the flow of the surface is wrong.

In a situation like a modern car with lots of bulges for the wheel arches, there is a lot of variation in the flow of the surface. This leads the designers to use a patchwork of separate NURBS pieces. These do not form a nice regular rectangular structure (otherwise a single NURBS would have done the job) and so pieces locally reflecting the desired surface flow are patched together at **trim-lines** where the NURBS on the two sides are supposed to meet continuously and smoothly.

* Bézier would use a single bicubic patch for the entire side of a car below the window-line. This would then get a bit more complicated by the addition of the flanging round the edge to make the panel stiffer, but it started life as a single bicubic with no interior knots at all.

Sorry, trimmed NURBS are not capable of this: at a frontier where position and surface normal are supposed to be common, there will usually be a slight mismatch of both, and neither will match the 3D shape of the nominal trim curve. The definition of *Class A* in the auto industry is *'it is class A when the chief designer is happy with it.'* but the definition of $C0$ is *'less than 0.1 mm of mismatch'*, and of $C1$ is *'less than one degree of crease.'*

Yes, cars can be designed to these requirements by using products like ICEM-SURF, but there are hints that maybe limits as large as these are acceptable only because the pressing of steel sheet can iron out small blemishes. Maybe additive manufacture might not be so forgiving.

Because the detail of the problem is not well understood (except that we know there are no simple solutions) there is the temptation to *'reduce the error by increasing the density of data.'*. Every time a knot is added there is the potential for one more inflexion in the surface - or in the curvature plot: The change in curvature due to a rounding error in a control point coordinate goes up as the square of the knot density. This isn't a good answer either, but everybody uses it.

How might subdivision surfaces be better ?

Simply, by letting you run the control mesh of a single surface so that it matches the flow of the shape everywhere.

All boundaries (where you want a crease) become mesh lines, so there is no need to trim there, and in places where the flow con- or di-verges you are able to use **extraordinary points** in the mesh to let the flow do what you need*.

But what problems do subdivision surfaces have ?

The story in the street is that *'subdivision surfaces are not C2 at extraordinary points'*, and this will be detailed below, but in fact there are a number of issues that we have to address before we can use subdivision both for surfaces and for basis functions for analysis.

(i) Catmull-Clark only does cubics

Doo-Sabin did quadratics, but the real question is whether degrees higher than cubic are ever even desirable and whether they are actually used. Incidentally, Tom Cashman's dissertation did indeed lift this restriction completely. He showed how you could not only see the maths, but also write code which was as flexible degree-wise as NURBS.

(ii) You are stuck with equal-intervals, so you need the same density everywhere.

An initial mesh should certainly be an equal-interval one, but the nested nature of the B-spline surfaces which subdivision surfaces consist of except at the extraordinary points means that higher density can be achieved locally wherever you want it. Tom Cashman covered it, but I don't think his solution is the elegant way to go.

* Yes, if you really want to use trimming to cut out a little circular hole, for example, you can still do so. This will be no worse than for NURBS.

(iii) Subdivision surfaces might not have a single parametrisation covering the whole surface.

This is a serious one, not for the users, but for the programmers of surface-handling software. It is really nice to be able to identify a point in the surface by just quoting its u, v parameter values. But the power of making the mesh flow where you want it to brings the power to make surfaces like torusses with fifteen holes (‘*high genus*’) which just do not have a sensible single parametrisation*.

Incidentally, triangulations (STL files) which have the same power of representing high genus objects also cannot be guaranteed a single parametrisation. Our programmers need to find ways of handling this issue.

(iv) Catmull-Clark is not C2 at extraordinary points.

This problem is well-understood by the academics. There do exist subdivision surface schemes which are C2 everywhere: they are rather ugly, and tend to give rather ugly surfaces. However, a simple thought experiment suggests that ‘not C2’ is the wrong name for the problem. There exist fix-ups which cut a small piece of surface out, and then replace it by something really smooth, giving a result which is C2 everywhere. Imagine taking a million steps of Catmull-Clark and then applying such a fix. The hole you are filling is smaller than the radius of an electron and for all practical purposes the surface is exactly the same shape as the non-C2 Catmull-Clark one, but the non-C2ness has been fixed. That cannot be the problem.

The real problem is that Catmull-Clark has shape problems in regions around the extraordinary point, not just at it. We can analyse this in terms of polynomial reproduction, and see that if you want to have something looking really smooth in a small region around an extraordinary point, it is going to converge towards a quadratic, so not being able to reproduce a quadratic is a rather more fundamental problem with Catmull-Clark.

In fact Catmull-Clark doesn’t just have discontinuity of curvature: it has infinite curvature, and unless the original configuration of control points is exactly symmetric, it has infinite negative Gaussian curvature.

The good news is that the reason for this is in the choice of coefficients which Catmull and Clark used when they were defining their scheme. Their target was computer graphics, and the computers they had were minicomputers (PDP11) which did not have floating point hardware. They wanted to make their surface compute fast (frames-per-second is an important measure in those days - still is) and so they avoided wherever possible division by anything but a power of 1/2 which could be programmed as a shift. This is no longer an important constraint on how we choose the coefficients.

* There are tricks which can be played by using a polygon with lots of sides on the projective plane and then identifying different sides, but crossing such a sewn-together single sheet has most of the same problems as crossing from one local parametrisation to another.

There are three coefficients available in the mask of an extraordinary point. We can choose two of them to eliminate the divergent curvature, and the third to reduce to a minimum the remaining ripples in the curvature plot of the surface. You need curvature colouring or reflection lines to be able to see what is left, not just normal rendering.

Even without this tuning the Catmull-Clark basis functions had energy functionals which meant that they could be used in elastic finite element analysis. It didn't just fail outright. With this tuning, density of the analysis mesh gives a good trade-off with accuracy.

(v) Catmull-Clark has boundary conditions which are poor for shape and seriously wrong for analysis.

What happens at the boundary of a Catmull-Clark surface (as described in their original 1978 paper) is unacceptable for analysis, and is probably only just tolerable for graphics. The limit curves corresponding to mesh lines have zero curvature at the boundary. This means that you cannot get positive Gaussian curvature there. By sheer concidence, this ties up with the condition in elasticity where a boundary position is fixed and no moment is applied across the boundary. AND the mesh lines lie exactly perpendicular to the boundary. Even a small breach of this last condition reduces the convergence rate of a solution in a way which serious analysis cannot tolerate.

Fortunately there is a really trivial way of fixing this. It is called *'Bézier Edge Conditions* and in terms of coding it means that there are slightly different coefficients used in the refinement construction near the boundary. Yes, we know what the values should be: there is just a little extra complication in the way that the actual refinement is coded. The user won't really see it (except that ideally the spacing of control points is reduced near the edge): only the programmers.

Are there alternatives ?

The key requirement is to be able to use control meshes which support extraordinary points. This could also be satisfied by what are known as **finite fillings**, where the surface is defined in terms of some specific finite collection of subpatches which the user doesn't see. (Subdivision surfaces use an infinite regress of subpatches and the user shouldn't see those either.)

I'm looking at these, to see if they have any advantages, or if the curvature problems carry across. An important issue is whether they support adaptivity as well as subdivision: my expectation is that it can be supported, but with a fair amount of complication in the data structures and the code.