Smooth mesh interpolation with cubic patches*

J Peters

An algorithm for the local interpolation of a mesh of cubic curves with 3- and 4-sided facets by a piecewise cubic C^1 surface is stated and illustrated by an implementation. Precise necessary and sufficient conditions for oriented tangent-plane continuity between adjacent patches are derived, and the explicit constructions are characterized by the degree of the three scalar weight functions that relate the versal to the two transversal derivatives. The algorithm fully exploits the possibility of reparametrization by choosing all three weight functions nonconstant and not just degree-raising polynomials. The construction is local and consists mainly of averaging. The only systems to be solved are linear and of size 2 × 2. The algorithm guarantees interpolating surfaces without cusps and has a simple, implemented extension to n-sided facets.

geometric design, Bernstein–Bézier form, surfaces, patches

This paper presents an algorithm that constructs a smooth piecewise polynomial surface interpolant to a mesh of cubic curves by splitting and averaging. The algorithm, called SPLAV in the following, is similar to the schemes described by Farin[1], Piper[2], Shirman and Séquin[3], and Jones[4]. It differs in that

- the surface interpolant is of lower degree,
- the surface interpolant can be guaranteed to be free of cusps,
- no systems of equations (larger than 2 × 2) have to be solved,
- the construction extends to arbitrary n-sided mesh facets.

Table 1, taken from Peters[5], gives the details.

SPLAV is a splitting scheme. Compared to blending schemes (e.g. Nielson[6], Charrot and Gregory[7]), splitting schemes use the same number of pieces to cover an n-facet, but generate generically a lower-degree surface.

A problem with cubic mesh interpolation by cubics was pointed out in Piper[2]. This paper characterizes admissible data for cubic mesh interpolation with splitting and exhibits a scheme for generating admissible curve meshes from data points (and their normals).

Center of Mathematical Sciences and Computer Sciences Department, University of Wisconsin – Madison, Madison, WI 53706, USA
Paper received 23 August 1988. Revised: 10 October 1989

Table 1. Smooth interpolation schemes based on splitting

Data	Sides	Degree	Reference
N	3	4	Farin[1]
T	3	4	Piper[2]
M	3, 4	4	Shirman and Séquin[3]
N	n	5	Jones[4]
M	3,4	3	Peters[8] and this paper
M	$n > 4$	4	this paper
N	3	bi3	Peters[9]

N = normal, T = tangents, M = cubic mesh

SPLAV is summarized as follows. Each n-sided mesh facet splits into n subtriangles. If $n = 3$ or $n = 4$ or if the data are symmetric, then a piecewise *cubic* interpolant (in Bernstein–Bézier form) is constructed; otherwise the interpolant is quartic. In either case, the centre coefficients of the patches can be given explicitly, so as to guarantee cusp-free tangent plane continuity across the original mesh curves. With the exception of asymmetric data and $n > 4$ (one banded linear $n \times n$ system), the remaining coefficients are determined as the (simple) average of two or three previously established coefficients. By default, the averaging is symmetric. However, the weights can be used as parameters to deform the interpolant. If a facet has to be covered by quartic patches, the splitting point may be chosen freely instead of by averaging. In all cases, the interpolating surface is parametrically smooth across the splitting curves.

SPLAV is a surface-fitting scheme and hence makes heavy use of reparametrization for the match across the original mesh curves (see Example 1). It is based on the following symmetric formulation of the necessary and sufficient constraints for oriented tangent plane continuity (Lemma 1):

$$\lambda p_u = \mu p_v + \nu q_w \text{ [common tangent plane]} \quad (E)$$

$$p_v \times p_u \neq 0,\ \mu\nu > 0 \text{ [proper orientation]}$$

NOTATION

A mesh curve intersects other mesh curves only at its two endpoints; that is, a mesh curve consists of only one (polynomial) piece. The intersections of the mesh curves are called data points, mesh points or just points. By labelling the mesh curves at each point (counter)

 0010–4485/90/020109–12

clockwise, a mesh facet is procedurally defined as the tuple of mesh curves obtained by following a mesh curve to the end and continuing on with the curve with the next higher label. A patch is the image of a bivariate polynomial piece over a unit domain. Hence one refers to total degree and tensor product patches also as triangular and rectangular patches. In this paper, patches are represented in Bernstein–Bézier form (BB form). Besides stability under evaluation and differentiation, this form gives geometric meaning to its coefficients and easy access to value and derivative information along patch boundaries. Farin[10] and de Boor[11] give a detailed discussion of the basics. A surface is constructed by determining the vector coefficients of the polynomial pieces. For this, it will suffice to look at univariate polynomials, i.e. derivatives of the patches evaluated at a boundary. As a mnemonic help, the BB coefficients of the derivatives in the direction parametrized by u, v and w are denoted as u^i, v^i and w^i, respectively. That is (see Figure 2), u^i, v^i and w^i are difference vectors of Bézier control points. Similarly, the difference polynomials are denoted $\tilde{u}$, $\tilde{v}$ and $\tilde{w}$. Since the main algebraic work consists of multiplying the univariate polynomials, the polynomials are written as an ordered tuple with entries that explicitly list the scalar coefficients $\binom{d}{j}$ together with the control points. That is,

$$\underbrace{\left(a^0, \ldots, \binom{d}{j} a^j, \ldots, a^d\right)}_{d+1 \text{ terms}}$$

stands for the polynomial

$$t \mapsto a^0(1-t)^d + \ldots + \binom{d}{j} a^j (1-t)^{d-j} t^j + \ldots + a^d t^d$$

of degree d. For example, raising the degree of the quadratic polynomial $(a^0, 2a^1, a^2)$ is expressed as $(1, 1)(a^0, 2a^1, a^2) = (a^0, 2a^1 + a^0, 2a^1 + a^2, a^2)$. If nothing else, this avoids writing fractions. Greek letters indicate scalar coefficients, capital letters denote the Bézier coefficients of the patches, and lower case letters mark the coefficients of difference polynomials. Superscripts number the coefficients of a polynomial and subscripts count the curves cyclically as they emanate from a mesh point.

The next section derives the constraints for oriented tangent plane continuity and gives an example of a match with three nonconstant, non degree-raised weight functions. The following section states the algorithm and its extensions; this is followed by a technical section that proves correctness. Subsequently, the problem of admissible meshes for the cubic setup is examined. Finally, examples of the implementation are given.

C^1 CONSTRAINTS

This section reviews the definition of a C^1 surface match and specializes it to patches in BB form. Consider a patch p parameterized by u and v and its neighbour patch, q, parameterized by u and w as illustrated by Figure 1.

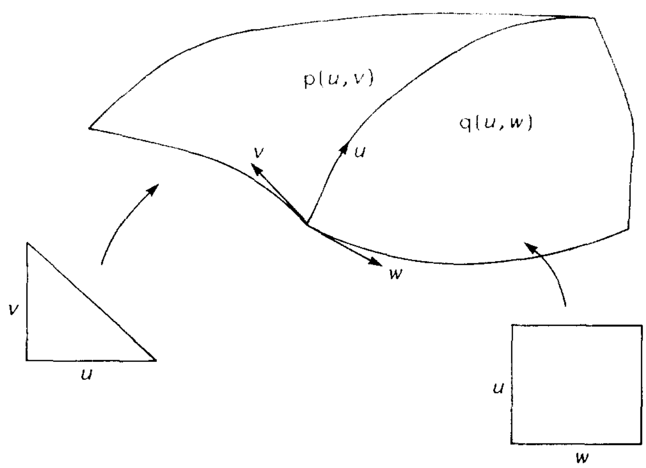

Figure 1. Parametrization of abutting patches

The following discussion applies to patches with an arbitrary number of sides, in particular to any combination of tensor product and total degree patches. Attention is focused on the common boundary where $v = w = 0$. Assuming that both patches are sufficiently smooth, subscripts denote partial derivatives along the boundary, i.e.

$$p_u := \frac{\partial}{\partial u} p(u, 0)$$

$$p_v := \frac{\partial}{\partial v} p(u, 0)$$

$$q_w := \frac{\partial}{\partial w} q(u, 0).$$

Then p and q are part of a C^1 surface if and only if the surface normal of p is well-defined and agrees with the normal of q at each point of the boundary:

$$\frac{p_v \times p_u}{\| p_v \times p_u \|} = \frac{p_u \times q_w}{\| p_u \times q_w \|} \text{ [matching normal]} \qquad (1)$$

$$p_v \times p_u \neq 0 \quad \text{[nonvanishing normal]} \qquad (2)$$

That is, for surfaces, C^1 means oriented tangent plane continuity. The orientation of the surface is captured by the ordering the vector products in equation (1). Control over orientation is necessary to produce proper smooth surfaces rather than razor-sharp edges. C^1-smoothness for surfaces differs from C^1-smoothness for (bivariate) functions, since the former characterizes the range of a collection of maps from the unit square or triangle to 3-space while the latter labels certain maps from the plane to 1-space. The following more convenient, but equivalent, characterization of the C^1 conditions for polynomial patches was developed by Peters[8]. Similar statements by Liu[12], Beeker[13] (p. 225), Degen[14] (p. 10), Piper[2] (p. 227) and Liu and Hoschek[15] (Theorem 1) ignore the orientation.

Lemma 1 [C^1 conditions]

A surface match between two smooth patches $p(u, v)$ and $q(u, w)$ across a common boundary parametrized by u is C^1 if and only if there exist scalar-valued functions λ, μ and ν of u such that, at each point of the boundary,

$$\lambda p_u = \mu p_v + \nu q_w \text{ [common tangent plane]} \quad \text{(E)}$$

$$p_v \times p_u \neq 0, \quad \mu\nu > 0 \quad \text{[regularity and proper orientation]} \quad \text{(I)}$$

Proof (1) and (2)⇒(E) and (I). By equation (2), p_u and p_v are linearly independent and hence $0 = (p_v \times p_u)q_w = \det(p_v, p_u, q_w)$ implies that q_w is a linear combination of p_u and p_v. Hence there exist scalar-valued functions λ, μ and ν such that expression (E) holds. Since, by equations (2) and (E),

$$0 < \lambda^2((p_v \times p_u)(p_v \times p_u)\frac{\| p_u \times q_w \|}{\| p_v \times p_u \|}$$
$$= ((p_v \times \lambda p_u)(\lambda p_u \times q_w) = ((p_v \times \nu q_w)(\mu p_v \times q_w)$$

also expression (I) holds.

Proof [(E) and (I)⇒(1) and (2)]. Conversely, taking the cross product of expression (E) with p_u, we find that $(p_v \times p_u)$ and $(p_u \times q_w)$ are collinear vectors. By expression (I) their orientation agrees. □

Expressions (E) and (I) were originally derived from the following alternative formulation of the C^1 constraints:

$$(p_v \times p_u)q_w = 0 \quad \text{[common tangent plane]} \quad \text{(E')}$$

$$((p_v \times p_u) \times p_u)q_w > 0 \quad \text{[regularity and proper orientation]} \quad \text{(I')}$$

When plots of α and β in $q_w = \alpha p_u + \beta p_v$ for matches produced by the scheme corresponding to expressions (E') and (I') (Peters[16]) defined a polynomial fit, Sabin[17] suggested using a quadratic divided by a quadratic Ansatz. This led to the constructions in Peters[8]. Such constructions with nonconstant λ, μ and ν are handily characterized by the triple (degree of λ, degree of μ, degree of ν). While it is harder to derive and verify matches with nonconstant weight functions, this paper demonstrates that the resulting algorithm can be very simple. The following example illustrates the use of nonconstant weights.

Example 1

Consider the following mesh data (see also Figure 2 for notation):

$$A_i^k = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad Q_{2i}^k = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \quad A_j^l = \begin{pmatrix} 6 \\ 1 \\ 1 \end{pmatrix}$$

first level of patch p

$$P^k = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad D_i^k = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} \quad D_j^l = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \quad P^l = \begin{pmatrix} 6 \\ 0 \\ 1 \end{pmatrix}$$

common boundary

$$A_{i+1}^k = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} \quad Q_{2i+1}^k = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \quad A_{j-1}^l = \begin{pmatrix} 5 \\ -1/2 \\ 1 \end{pmatrix}$$

first level of patch q

By exploiting the full freedom of reparametrization, it is possible to determine the coefficients Q, so that the cubic patches p and q join with oriented tangent plane continuity. The solution,

$$Q_{2i}^k = \begin{pmatrix} 23/6 \\ 13/12 \\ 0 \end{pmatrix} \quad Q_{2i+1}^k = \begin{pmatrix} 7/3 \\ -11/12 \\ 1 \end{pmatrix}$$

is checked by computing

$$p_u = 3\left\{\begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, 2\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}\right\}$$

$$p_v = 3\left\{\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, 2\begin{pmatrix} 11/16 \\ 13/12 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}\right\}$$

$$q_w = 3\left\{\begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, 2\begin{pmatrix} 1/3 \\ -11/12 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1/2 \\ 0 \end{pmatrix}\right\}$$

and showing that $\det(p_u, p_v, q_w) = 0$. The interpolant corresponds to expression (E) with

$$\lambda := \lambda(u) := (3/2, 1), \quad \mu := \mu(u) := (3/2, 1/2),$$
$$\nu := \nu(u) := (3/2, 1)$$

Note that none of the weight functions is a degree-raising polynomial, the cross boundary quadrilaterals or 'butterflies' are not all coplanar (in contrast to a function-C^1 match):

$$\det\left\{\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 11/6 \\ 13/12 \\ 0 \end{pmatrix}, \begin{pmatrix} 1/3 \\ -11/12 \\ 1 \end{pmatrix}\right\}$$
$$= \frac{1}{8} \neq 0$$

and

$$p_u = \frac{(3/2, 1/2)}{(3/2, 1)} p_v + q_w$$

SPLITTING AND AVERAGING ALGORITHM FOR SMOOTH MESH INTERPOLATION

Given a mesh of cubic curves, SPLAV determines the coefficients starting from the mesh curves and working its way inward towards the splitting point. That is (see Figure 2), with the coefficients labelled D and E given, the algorithm first determines the coefficients labelled

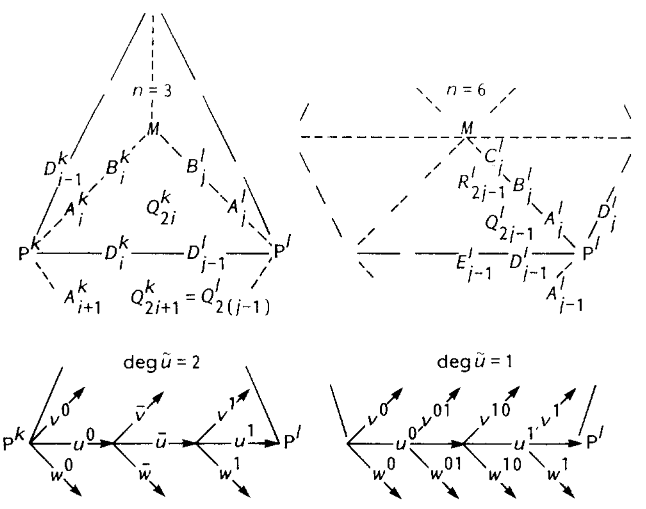

Figure 2. Notation for the cubic patches (left) and quartic patches (right): $\mathbf{N}^k$ *is the normal at* P^k*, and* $\tilde{u} := (u^0, \ldots, u^1) := p_u/d$

A, then those labelled Q, then those labelled B and finally those labelled M (plus C and R in the quartic case).

The following functions are called by the algorithm (* is the vector product, × the cross product):

- nbrs(k) returns the number of neighbours of point k
- nbr(k, i) returns the ith neighbour of point k
- sds(k, i) returns the number of sides of the ith facet attached to point k; sds(f) returns the number of sides of the facet f
- avg(p1, p2, p3, i) returns the vector p1 + (p2 + p3 − 2p1)/λ, where $\lambda := 2(1 - \cos(2\pi/i))$
- cs(v1, v2) returns the scalar (v1 * v2)/(v1 * v1) (almost, but not quite cos(v1, v2))
- ratio(v1, v2, v3, n) — returns the scalars $\gamma :=$ det(v3, v1, n)/det(v3 − v2, v1, n) and $\eta := \gamma$ det(v2, v3, n)/det(v1, v3, n) such that $\eta v1 = \gamma v2 + (1 - \gamma) v3$
- skip in a loop skips the statements below to continue at the loop head with the next iteration step

The Preprocessor precedes SPLAV. It determines the degree of the boundary curve (linear, quadratic or cubic) and ensures, in the case of a cubic boundary, that equation (3) is well-conditioned. If only a 'mesh' of points and no boundary curves are input, the Mesh Generator creates a default mesh. The following is the basic version of SPLAV, for three- and four-sided facets only. The Δ symbols are anchors for additional preprocessing and n-facet code. Assignments like $Q^k_{2i} = D^k_i + \bar{v}_i$ that follow from Figure 2 are omitted.

Algorithm SPLAV

Δ_1
for $k = 1$:points

  for $i = 1$:nbrs(k) [compute A^k_i]
    $A^k_i = \text{avg}(P^k, D^k_{i-1}, D^k, \text{sds}(k, i))$;

  for $i = 1$:nbrs(k) [compute Q^k_i]
    if $k <$ nbr(k, i) [match needs to be computed from one side only]
    skip;

  if $\tilde{u} == (u^0, u^1)$, [quadratic or linear boundary]
    $u^0_\perp = \mathbf{N}^0 \times u^0$; $u^1_\perp = \mathbf{N}^1 \times u^1$;
    $2\bar{v} = \text{cs}(u^0_\perp, v^0)u^1_\perp + \text{cs}(u^1_\perp, v^1)u^0_\perp + \text{cs}(u^0, v^0)u^1 + \text{cs}(u^1, v^1)u^0$;
    $2\bar{w} = \text{cs}(u^0_\perp, w^0)u^1_\perp + \text{cs}(u^1_\perp, w^1)u^0_\perp + \text{cs}(u^0, w^0)u^1 + \text{cs}(u^1, w^1)u^0$;
    Δ^2
  fi [quadratic or linear boundary]

  if $\tilde{u} == (u^0, 2\bar{u}, u^1)$, [cubic boundary]
    $[\gamma^0, \eta_0] = \text{ratio}(u^0, v^0, w^0, \mathbf{N}^0)$;
    $[\gamma^1, \eta_1] = \text{ratio}(u^1, v^1, w^1, \mathbf{N}^1)$;
    if $\gamma^0 == \gamma^1$, [preprocessing ensures $\eta^0 == \eta^1$]
      $\delta_v = \delta_w = \eta^0$,
    else $[\delta_v, \delta_w]$ = solution of

$$\begin{pmatrix} \gamma^0 & 1-\gamma^0 \\ \gamma^1 & 1-\gamma^1 \end{pmatrix} \begin{pmatrix} \delta_v \\ \delta_w \end{pmatrix} = \begin{pmatrix} \eta^0 \\ \eta^1 \end{pmatrix} \tag{3}$$

    fi

$$\bar{v} = \delta_v \bar{u} + \frac{1}{2}\left(\frac{1-\gamma^0}{1-\gamma^1}(v^1 - \delta_v u^1) + \frac{1-\gamma^1}{1-\gamma^0}(v^0 - \delta_v u^0)\right);$$

$$\bar{w} = \delta_w \bar{u} - \frac{1}{2}\left(\frac{\gamma^0}{1-\gamma^1}(v^1 - \delta_v u^1) + \frac{\gamma^1}{1-\gamma^0}(v^0 - \delta_v u^0)\right);$$

    Δ_3
  fi [cubic boundary]
 [The first layer consisting of the A^k_i and Q^k_i is determined]
 for $k = 1$:points
  for $i = 1$:nbrs(k)
   $B^k_i = \text{avg}(A^k_i, Q^k_{2i-1}, Q^k_{2i}, \text{sds}(k, i))$;

 for $f = 1$:facets [Determine the midpoint M]
  $M = \text{avg}(B^k_1, B^k_0, B^k_2, \text{sds}(f))$;
[Done]

Remarks

- The pseudocode includes symmetric statements that are more elegantly coded as subroutines or macros.
- The next section explains and proves the correctness of the construction.
- The problematic of equation (3) is discussed in the section on admissible data.

Before extending SPLAV to arbitrary n-sided facets and quartic patches (Δ_2 and Δ_3), the boundary generation and preprocessing are made precise. In many applications only the mesh points are known. Reasonable normals can be calculated by least squares, e.g. by a spherical fit. For a cubic curve mesh it is then sufficient to generate the tangent vectors at the data points. Since it is desirable to keep the degree of the curves low, the boundary generator checks whether the data at the endpoints admit a linear or a quadratic interpolant. The following functions are used: sgn(i) returns 1, if $i > 0$; 0, if $i = 0$ and -1 otherwise. proj($p1$, $n1$, $p2$, $n2$) returns the scalar

$$(n1 + (n1^*n2)n2)^*(p2 - p1)/(1 - (n1^*n2)^2).$$

Default Mesh Generator

```
for k = 1:points

  for i = 1:nbrs(k)
    l = nbr(j, i);
    σ^0 = sgn(N^k * (P^k − P^l);
    σ^1 = sgn(N^l*(P^l − P^k))

    if σ^0 == σ^1 == 0   [linear curve]
      u_i^0 = (P^k − P^l)/2;
      skip;
    fi [ũ = (u^0, u^1)]
    if σ^0σ^1 > = 0 [quadratic curve; simpler choices
      are possible; this choice minimizes the length]
      δ_0 = proj(P^k, N^k, P^l, N^l); δ_1 = proj(P^l, N^l, P^k, N^k);
      2u_i^0 = (P^l − P^k) − δ_0N^k − δ_1N^l;
      skip;
    fi [quadratic curve; ũ = (u^0, u^1)]
    [cubic curve: ũ = (u^0, 2ū, u^1)]
    u_i^0 = ((P^k − P^l) − (P^k − P^l)*N^kN^k)/3;
                  [projection into the tangent plane]
```

Remarks

- The 'correctness' section motivates the construction.

The Mesh Generator tries to establish expression (I) at the data points. Both the generator and the Preprocessor work under the hypothesis that the neighbours of a point can be ordered consistently by their projections into the tangent plane (Constraint 2). To be on the safe side, the Preprocessor checks this constraint and replaces bulging quadratic by taut cubic curves if necessary. If equation (2) is ill-conditioned, then additional points are introduced. This is cumbersome and usually indicates that the input mesh or normals at the data points are chosen poorly (the admissible data section). The Preprocessor uses the following constants: TOL1 = 0.1 – the contribution of the u^i terms in the cubic construction should not be disproportionate; TOL2 ≈ precision.

Preprocessor

```
Δ_1:
for k = 1:points
  for i = 1:nbrs(k) [enforce proper orientation]
    if det(D_i^k − P^k, D_{i−1}^k − P^k, N^k) ⩽ 0
      if ũ == (u^0, 2ū, u^1),
        error('Constraint 2 violated:
          not a projective ordering');
        skip;
      fi
      u_i^0 = ((P^k − P^l) − (P^k − P^l) * N^kN^k)/3;
        [make cubic]
        i = i − 1; [check again]
    fi

  for k = 1:points [enforce equation (3) at cubic
                    boundaries]
    for i = 1:nbrs(k)
      if ũ == (u^0, 2ū, u^1), [cubic boundary]
        [γ^0, η_0] = ratio(u^0, v^0, w^0, N^0);
        [γ^1, η_1] = ratio(u^1, v^1, w^1, N^1);
        if |γ^0 − γ^1| < TOL1
          if |γ^0 − γ^1| + |η^0 − η^1| < TOL2
                              [has a simple solution]
            skip;
          fi
          warning('Constraint 5 violated')
          subdivide boundary curve;
                        [use quartics alternatively]
          add midpoint as new point;
          connect midpoint to original data points;
        fi
      fi
  for m = 1:new points
      [connect new point to the rest of the mesh]
    k = nbr(m, 1); [Let m be the ith neighbour of k]
    [connect m to nbr(k, i − 1)]
    l = nbr(m, 2); [Let m be the jth neighbour of l]
    [connect m to nbr(l, j − 1)]
```

Remarks

- The next sections explain and prove the correctness of the construction.
- The values returned by the 'ratio' function are stored and hence need not be computed in SPLAV.
- If the data structures provide for it, the local use of quartic patches instead of adding points is reasonable (see the section on methods for quartic patches). SPLAV opts to introduce additional points so that the surface remains cubic if there are only three- and four-sided mesh facets.

The final piece of code extends the basic algorithm to arbitrary n-sided facets ($n > 4$) by local use of quartic patches. If the mesh curve is quadratic, degree raising is sufficient:

```
Δ_2:
if ṽ == (v^0, 3v^01, 3v^10, v^1)
                    [raise the degree of the solution]
  v^01 = (v^0 + 2v̄)/3; v^10 = (v^1 + 2v̄)/3;
[analogously for w̃]
```

Cubic boundary curves also pose no problem. If the abutting patches are both quartic, i.e. in the case of abutting n-sided facets, the following statements suffice.

```
Δ_3:
if ṽ == (v^0, 3v^01, 3v^10, v^1)
  s* = cs(u_⊥^0, v^0) + cs(u_⊥^1, v^1);
                  [many other choices are possible]
  3v^01 = cs(u_⊥^0, v^0)u_⊥^1 + s*u_⊥^0 + cs(u^1, v^1)u^0
        + 2cs(u^0, v^0)ū
```

$$3v^{10} = cs(u^1_-, v^1)u^0_- + s^*u^1_- + cs(u^0, v^0)u^1 + 2cs(u^1, v^1)\bar{u}$$
[analogously for $\tilde{w}$]

Finally, in the case of an isolated n-sided facet, i.e. one cubic and one quartic patch, the cubic patch can be completed by degree raising, while the quartic patch still allows for a C^1 transition.

Δ_3:
$\tilde{w} = w^0 + w^1$;
$\eta^0_* = \eta^0/3\gamma^0$; $\eta^1_* = \eta^1/3\gamma^1$;
$\delta^0 = (1-\gamma^0)/3\gamma^0$; $\delta^1 = 1-\gamma^1/3\gamma^1$;
$v^{01} = 2\eta^0_*\bar{u} + \eta^1_* u^0 - (2\delta^0\bar{w} + \delta^1 w^0)$
$v^{10} = 2\eta^1_*\bar{u} + \eta^0_* u^1 - (2\delta^1\bar{w} + \delta^0 w^1)$

The quartic interpolant is completed by computing the central coefficients M, R and C:

$$M = \frac{1}{sds(f)} \sum P^k + \frac{4}{3}(A^k_i - P^k) + \frac{4}{3}(B^k_i - A^k_i);$$

[average of cubic boundary]

$c = 2\cos(2\pi/n)$; $\lambda := 2 - c$;
$[R_1, \ldots, R_n]$ = coordinate solution of

$$\min \sum \left\| \frac{B_i + B_{i+1}}{2} - x_i \right\|^2$$

$$\begin{pmatrix} 1 & 1-c & 1-c & 1 & 0 & \ldots & 0 \\ 0 & 1 & 1-c & 1-c & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1-c & 1 & 0 & 0 & 0 & \ldots & 1-c \\ 1-c & 1-c & 1 & 0 & 0 & \ldots & 1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c(B_2 - cB_3 + B_4) + \lambda^2 M \\ c(B_3 - cB_4 + B_5) + \lambda^2 M \\ \vdots \\ c(B_1 - cB_2 + B_3) + \lambda^2 M \end{pmatrix} \quad (E_{34})$$

for $i = 1{:}n$, $C_i = \mathrm{avg}(B_i, R_i, R_{i-1})$;

Remarks

- The section on smoothness across splitting boundaries explains and proves the correctness of the construction.
- There are simpler choices than solving expression (E_{34}).

CORRECTNESS

This section motivates the algorithms of the last section and shows how SPLAV uses the theory developed in the constraints section. In particular, this section exhibits the scalar weight functions λ, μ and v such that expressions (E) and (I) hold for the assignments of SPLAV. We start with the boundary generator.

Boundary generation

To rule out various cases of 3D geometry that defy easy analysis and are usually not of interest in design and interpolation, the following constraint is assumed to hold.

Constraint 1 [slowly changing orientation]
If $\mathbf{N}^0$ and $\mathbf{N}^1$ are the normals at two neighbouring points P^k and P^l, connected by the straight line segment $L := P^l - P^k$ then $(\mathbf{N}^0 \times L) * (\mathbf{N}^1 \times L) > 0$. In other words, when projected into the plane spanned by $\mathbf{N}^0$ and L, $\mathbf{N}^1$ at P^l must lie on the same side of L as $\mathbf{N}^0$. To keep the boundaries taut and avoid adding points, the boundary generator strives to deliver a curve of minimal degree. If the 'sgn' function returns $\sigma^0 = = \sigma^1 = = 0$, then the straight line connecting the mesh points is perpendicular to the normals and hence the sought-for minimal interpolant. If the product is negative, then the curve must have a point of inflection (see Figure 3).

Hence, if the product of the 'sgn' functions is negative, the curve must be cubic. The projection of L into the tangent plane at each endpoint, i.e.

$$u^0 := \frac{1}{d}((P^l - P^k) - (P^l - P^k) * \mathbf{N}^k\mathbf{N}^k),$$

$$d := \text{degree of } \tilde{u} \quad (4)$$

is a good choice, since it reflects the local geometry and degenerates gracefully to the planar case. If the product is positive or zero, then a quadratic interpolant can be constructed. SPLAV chooses to minimize the length of the control polygon by choosing the solution of

$$\min\ (u^0 * u^0 + u^1 * u^1)$$
$$\mathbf{N}^k u^0 = 0 \quad (5)$$
$$\mathbf{N}^l u^1 = 0$$

Another option is to minimize the distance to the cubic interpolant. In either case a solution to the optimization problem in its dual form can be computed explicitly with the help of proj: $[\delta^0, \delta^1]$ is the dual solution. Deriving u^0 by forcing it into the plane spanned by the average of the normals and $P^l - P^k$ also yields good results:

$$a = ((\mathbf{N}^k + \mathbf{N}^l) \times (P^l - P^k)) \times \mathbf{N}^k;$$

$$u^0_i = \frac{2}{3} \frac{\mathbf{N}^k * (P^l - P^k)}{\mathbf{N}^k * a} a;$$

The main challenge of the mesh generation is to avoid cusps at the data points. For arbitrary combinations of points, normals and neighbour relationships this seems to be a hopeless task in three dimensions. Hence, the following constraint is useful.

Constraint 2 [projective ordering]
The neighbours of any point P^k are ordered clockwise

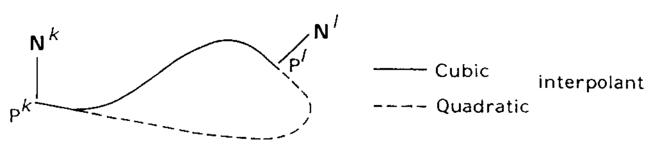

Figure 3. Local 'shape' and minimal degree of boundary curve

according to their projections into the tangent plane at P^k and the ordering is transitive.

Here transitivity means that if A orders its neighbours by projections as B, C and B as C, A, then C must order A, B. With the constraint, the correctness of the Mesh Generator can be established.

Lemma 2 [correct curve generation]
If Constraint 1 and 2 hold, then the curves generated by the Mesh Generator satisfy expressions (E) and (I) at the data points.

Proof. All three constructions of Δ_1, linear, quadratic and cubic, enforce coplanarity of the mesh curves at the data point. If the neighbours are projectively ordered, then

$$\det(D_i^k - P^k, D_{i-1}^k - P^k, N^k) > 0 \tag{6}$$

is (necessary and) sufficient to enforce expression (I) between D_i^k and D_{i-1}^k for the cubic and the linear construction. Quadratic curves that violate expression (6) are replaced by cubics. □

The following discussion assumes that Constraint 3 below holds.

Constraint 3
The mesh curves define a unique tangent plane at each data point and expression (6) holds for the tangent vectors.

Matches between two cubic patches

The cubic–cubic matches of this section are derived with the common-tangent-direction technique of Peters[8] (see also Degen[14]). Proposition 1 for quadratic boundaries, for example, uses $\tilde{c}(u) := (u_\perp^0, u_\perp^1)$ to be the tangent direction common to p_v and q_w, i.e. $p_v = \alpha^v p_u + \beta^v \tilde{c}$ and $q_w = \alpha^w p_u + \beta^w \tilde{c}$ and hence

$$\beta^w p_v - \beta^v q_w = (\beta^w \alpha^v - \beta^v \alpha^w) p_u$$

The proper orientation of these constructions can be established with Constraint 3 and the following mild restriction that prevents the clustering of all tangent vectors in one half plane. Curves generated by the Mesh Generator satisfy the constraint provided the projections of all neighbours do not fall into one half plane.

Constraint 4
The angle $(D_{i-1}^k P^k D_i^k)$ is in $(0, \pi)$.

The proof of regularity is based on the following two geometric observations that, due to Constraint 1, can be checked by looking at projections into the $\mathbf{N}^0 - L$-plane.

Observation 1 [quadratic curves]
If $\tilde{u} = (u^0, u^1)$ and Constraint 1 holds, then $\mathrm{sgn}(\mathbf{N}^0 * u^1) \neq \mathrm{sgn}(\mathbf{N}^1 * u^0)$.

Observation 2 [cubic curves]
If $\tilde{u} = (u^0, 2\bar{u}, u^1)$, $\mathrm{sgn}(\mathbf{N}^0 * (P^0 - P^1)) * \mathrm{sgn}(\mathbf{N}^1 * (P^1 - P^0)) < 0$ for the two endpoints and Constraint 1 holds, then $\mathrm{sgn}(\mathbf{N}^0 * u^1) == \mathrm{sgn}(\mathbf{N}^1 * u^0)$.

Since the (2, 1, 1) construction below needs only 'one-sided' information, it is also used for the global boundary of open surfaces.

Proposition 1 [(2,1,1) match for two cubic patches connected by a quadratic boundary]
If Constraint 4 and Constraint 3 hold, then SPLAV connects two cubic patches across a quadratic boundary curve with a (2,1,1) C^1 match.

Proof. Substitution shows that expression (E) holds for the assignments of the last section and

$$\mu := (\mu^0, \mu^1) := (-\mathrm{cs}(u_\perp^0, w^0), -\mathrm{cs}(u_\perp^1, w^1)),$$
$$v := (v^0, v^1) := (\mathrm{cs}(u_\perp^0, v^0), \mathrm{cs}(u_\perp^1, v^1)),$$
$$\lambda := (\lambda^0, 2\bar{\lambda}, \lambda^1) := (\mathrm{cs}(u^0, v^0), \mathrm{cs}(u^1, v^1))\mu + (\mathrm{cs}(u^0, w^0), \mathrm{cs}(u^1, w^1))v$$

That is, all four coefficients of

$$\mu\tilde{u} - \mu\tilde{v} - v\tilde{w}$$

vanish. Due to symmetry, only the first two have to be checked. The first term vanishes, since by Constraint 3,

$$v^0 = \mathrm{cs}(u^0, v^0)u^0 + \mathrm{cs}(u_\perp^0, v^0)u_\perp^0$$
$$w^0 = \mathrm{cs}(u^0, w^0)u^0 + \mathrm{cs}(u_\perp^0, w^0)u_\perp^0 \tag{7}$$

and hence

$$\mu^0 v^0 + v^0 w^0 = (\mu^0\,\mathrm{cs}(u^0, v^0) + v^0\,\mathrm{cs}(u^0, w^0))u^0 + (\mu^0 v^0 - v^0 \mu^0)u_\perp^0 = \lambda^0 u^0$$

Similarly, with equations (7) and more terms to cancel,

$$2\mu^0\bar{v} + \mu^1 v^0 + 2v^0\bar{w} + v^1 w^0 = \lambda^0 u^1 + 2\bar{\lambda} u^0$$

Constraint 4 and the averaging construction guarantee that v^i and w^i lie on the appropriate opposite sides of u^i ($i \in \{0, 1\}$ in the following) and hence $\mathrm{cs}(u_\perp^0, w^0) < 0$ and $\mathrm{cs}(u_\perp^0, v^0) > 0$. It also implies that $p_u \times p_v \neq 0$ at the endpoints. Assume that $(p_u \times p_v)(t_0) = 0$ for some $t_0 \in (0, 1)$, i.e. that $p_u(t_0)$ and $p_v(t_0)$ are collinear. Since $p_u(t_0) * (u^0 \times u^1) = 0$, $p_v(t_0) * (u^0 \times u^1) = -\alpha \mathbf{N}^0 u^1 + \beta \mathbf{N}^1 u^0$ for positive constants $\alpha := (1 - t_0)^2 / |v^0 * u^0| + (1 - t_0)t_0 / |u_\perp^0 * u^0|$ and β contradicts Observation 1. □

The case of a cubic boundary curve does not always have a solution. The next section analyses this in detail. For now the argument proceeds under the following sufficient constraint.

Constraint 5
If $[\gamma^0, \eta_0] = \mathrm{ratio}(u^0, v^0, w^0, \mathbf{N}^0)$ and $[\gamma^1, \eta_1] = \mathrm{ratio}(u^1, v^1, w^1, \mathbf{N}^1)$ and $\gamma^0 == \gamma^1$, then $\eta^0 == \eta^1$.

In terms of Figure 2, this means: if the line through $P^kD_i^k$ divides the segment $A_i^kA_{i+1}^k$ in the same ratio $\gamma^0:(1-\gamma^0)$ as $D_{i-1}^lP^l$ divides the segment $A_i^lA_{i-1}^l$, then the division ratios for $P^kD_i^k$ by $A_i^kA_{i+1}^k$ and for $D_{i-1}^lP^l$ by $A_i^lA_{i-1}^l$ must also agree.

Proposition 2 [(1,1,1) match for two cubic patches connected by a cubic boundary]
If Constraints 3, 4 and 5 hold and $\text{sgn}(\mathbf{N}^0 * (P^0 - P^1))\ \text{sgn}(\mathbf{N}^1 * (P^1 - P^0)) < 0$ for the two endpoints P^0 and P^1, then SPLAV connects two cubic patches across a cubic boundary curve with a (1,1,1) C^1 match.

Proof. Constraint 5 is necessary (and sufficient) for equation (3) to have a solution. Substitution then shows that expression (E) holds for the given assignments with

$$\mu := (\gamma^0, \gamma^1), \quad v := ((1-\gamma^0), (1-\gamma^1)),$$
$$\lambda := \delta_v \mu + \delta_w v$$

Due to symmetry, it is sufficient to check that the first two terms of $\lambda\tilde{u} - \mu\tilde{v} - v\tilde{w}$ vanish. Using equation (3) and the definition of the 'ratio' function, one checks

$$(\gamma^0\delta_v + (1-\gamma^0)\delta_w)u^0 = \eta^0 u^0 = \gamma^0 v^0 + (1-\gamma^0)\omega^0$$

Similarly,

$$2\delta_v\gamma^0\bar{u} + \delta_v\gamma^1 u^0 + 2\delta_w(1-\gamma^0)\bar{u} + \delta_w(1-\gamma^1)u^0$$
$$= 2\eta^0\bar{u} + \eta^1 u^0 = 2\gamma^0\bar{v} + \gamma^1 v^0 + 2(1-\gamma^0)\bar{w}$$
$$+ (1-\gamma^1)w^0$$

Just as in Proposition 1, Constraint 4 implies $\mu v > 0$. Since $\mu(p_u \times p_v) = -v(p_u \times q_w)$, this implies that $p_u \times p_v$ and $p_u \times q_w$ must vanish simultaneously. Also, by Constraints 1 and 4, $p_u \times p_v$ has to vanish an even number of times to twist back into the proper orientation. Hence assume that $p_u \times p_v = 0$ both at a and at b in (0, 1). Then there are constants α_i and β_i for $i \in \{a, b\}$ such that $p_v(i) = \alpha_i p_u(i)$ and $q_w(i) = \beta_i p_u(i)$ and therefore

$$(\gamma^0(p_v(i) - \alpha_i p_u(i)) + (1-\gamma^0)(q_w(i)$$
$$- \beta_i p_u(i)) * (u^0 \times \bar{u}) = 0$$

After cancelling terms that combine as or include u^0 and $\bar{u}$, all $(1-i)^2$ terms vanish and, dividing by i, one is left with

$$\left(\frac{1-i}{i}\frac{\gamma^0-\gamma^1}{1-\gamma^0} v^0 + \gamma^0 v^1 + (1-\gamma^0)w^1 - (\alpha_i + \beta_i)u^1\right)$$
$$*(u^0 \times \bar{u}) = 0 \tag{8_i^0}$$

The v terms are eliminated by subtracting the expressions for $i = b$ from that for $i = a$:

$$\left(\frac{\gamma^0-\gamma^1}{1-\gamma^0}\left(\frac{1-a}{a} - \frac{1-b}{b}\right)v^0 + \right.$$
$$\left. + (\alpha_b + \beta_b - \alpha_a - \beta_a)u^1\right)$$
$$*(u^0 \times \bar{u}) = 0 \tag{8^0}$$

By adding the symmetric statement to equation (8^0) with $u^0 \times \bar{u}$ replaced by $u^1 \times \bar{u}$, i.e.

$$\left(\frac{\gamma^1-\gamma^0}{1-\gamma^1}\left(\frac{a}{1-a} - \frac{b}{1-b}\right)v^1 + \right.$$
$$\left. + (\alpha_b + \beta_b - \alpha_a - \beta_a)u^0\right)$$
$$*(u^1 \times \bar{u}) = 0$$

the term involving $\alpha_b + \beta_b - \alpha_a - \beta_a$ is cancelled and the constraint reads

$$\frac{1-\gamma^1}{1-\gamma^0}\frac{(1-a)(1-b)}{ab}(v^0 \times u^0) * \bar{u} + (v^1 \times u^1) * \bar{u} = 0$$

Since the coefficient of the first term is positive, and $v^0 \times u^0$ and $v^1 \times u^1$ are negative multiples of $\mathbf{N}^0$ and $\mathbf{N}^1$, Observation 2 is violated. □

The proofs of regularity confirm the intuition that averaging transversal difference vectors with positive weights is orientation preserving.

Matches for quartic patches

As the next subsection will show, it may be hard, if not impossible, to construct locally a cubic splitting interpolant to n-sided asymmetric facets. Consequently, this section derives quartic–cubic and quartic–quartic matches corresponding to isolated and abutting n-sided facets ($n > 4$). The constructions do not require that the degree of the cubic patches be raised, so that the overall interpolant remains cubic with isolated quartic subregions.

If the boundary curve is quadratic, the degree raised solution of Proposition 1, i.e. $(1, 1)\lambda$, $(1, 1)\mu$ and $(1, 1)v$, will do. An analogous construction for the quartic–quartic match and $\tilde{u} = (u^0, 2\bar{u}, u^1)$ solves

$$(v^0, 3v^{01}, 3v^{10}, v^1) = (\text{cs}(u^0, v^0), \text{cs}(u^1, v^1))(u^0, 2\bar{u}, u^1)$$
$$+ (\text{cs}(u_\perp^0, v^0), \rho^v, \text{cs}(u_\perp^1, v^1))(u_\perp^0, u_\perp^1), \tag{9}$$

where the scalar ρ^v is only constrained by expression (I). Averaging or degree raising leads to the choice $\rho^v := \text{cs}(u^0, v^0) + \text{cs}(u_\perp^1, v^1)$. The equality constraints are then checked as in Proposition 1. In fact the second constraint of equation (9) can be read off directly:

$$3v^{01} = \text{cs}(u_\perp^0, v^0)u_\perp^1 + \rho^v u_\perp^0 + \text{cs}(u^1, v^1)u^0$$
$$+ 2\text{cs}(u^0, v^0)\bar{u}$$

The weight functions are

$$\mu := (-\text{cs}(u_\perp^0, w^0), -\rho^w, -\text{cs}(u_\perp^1, w^1)),$$

$$\nu := (\mathrm{cs}(u_\perp^0, v^0), \rho^v, \mathrm{cs}(u_\perp^1, v^1)),$$

$$\lambda := (\mathrm{cs}(u^0, v^0), \mathrm{cs}(u^1, v^1))\mu + (\mathrm{cs}(u^0, w^0), \mathrm{cs}(u^1, w^1))\nu$$

implying as in the earlier proofs that $\mu\nu > 0$.

At an isolated n-facet, the cubic patch is completed by degree raising, i.e. by $\bar{w} := w^0 + w^1$. One computes the quartic patch construction from

$$(\lambda^0, \lambda^1)(u^0, 2\bar{u}, u^1) = \mu(v^0, 3v^{01}, 3v^{10}, v^1) + (v^0, v^1)(w^0, 2\bar{w}, w^1)$$

Smoothness across splitting boundaries

The previous subsections established smoothness across the original mesh curves for given boundary coefficients D and A. This subsection proves the correctness of the construction for the interior (splitting) boundaries and suggests that cubics are not flexible enough to yield a smooth solution to asymmetric data if $n > 4$. The main reason for the breakdown of the cubic construction is that the coefficients labelled B no longer automatically end up in a common (tangent) plane. In that sense the constructions for $n = 3$ and $n = 4$ are fortuitous.

Formula 1 [(0,0,0) match for interior boundaries]
SPLAV connects interior patches by a (0,0,0) C^1 match. The construction has at least two (sign-restricted) shape parameters.

Proof. For the interior boundaries of an n-sided patch, averaging gives $\mu := \nu := 1$ and $\lambda := 2(1 - \cos(2\pi/n))$. This is motivated by the following relation that characterizes a regular n-sided domain with vertices p_i and centre m:

$$\frac{1}{2}(-(-m - p_i) + (p_{i-1} - p_i) + (-(m - p_i) + (p_{i+1} - p_i)) = \cos(2\pi/n)(-(m - p_i))$$

If $n = 3$, then $\mu = \nu = 1$ and $\lambda = 3$ and the splitting is analogous to the Clough–Tocher or centroidal split. The construction succeeds, since any three coefficients B lie in a plane and so does their average M. This corresponds to choosing the barycentric coordinates for the averaged point in terms of the previously established points as (1/3, 1/3, 1/3). Any other choice of positive barycentric coordinates per facet will also yield a C^1 match (Peters[8], Lemma 2.1.1). Hence the interpolant obtained from SPLAV can be deformed by adapting the function 'avg' for the three splitting edges.

If $n = 4$, then $\mu = \nu = 1$ and $\lambda = 0$. The construction succeeds, since the four coefficients B lie on the midpoints of the (not necessarily planar) quadrilateral spanned by the coefficients labelled Q. Thus, the straight lines connecting nonadjacent (opposing) coefficients B intersect in the ratio 1:1 and the intersection point, M lies in a well defined (tangent) plane. By changing the ratio for opposing mesh points simultaneously, i.e. by choosing barycentric coordinates $((1/2) - \varepsilon, (1/2) + \varepsilon, 0)$ and $(1/2) + \varepsilon, (1/2) - \varepsilon, 0)$ at the opposing mesh point, the construction allows for a two-parameter family of solutions.

If $n > 4$ and the data are symmetric, then, by definition, the coefficients B come to lie in a common plane and are distributed like the vertices of a regular n-sided figure. Again, the splitting point of such an n-sided figure can be chosen freely within the n-sided figure yielding two shape parameters restricted by a positivity requirement.

If $n > 4$ and the data are unsymmetric, then the coefficients B will, in general, not lie in a common plane. This motivates the quartic approach which results in $2n$ constraints for $2n + 1$ coefficients ($\lambda := 2 - 2\cos(2\pi/n)$):

$$(R_i - B_i) + (R_{i-1} - B_i) = \lambda(C_i - B_i) \qquad (E_i^3)$$

$$(C_{i+1} - C_i) + (C_{i-1} - C_i) = \lambda(M - C_i) \qquad (E_i^4)$$

The circulant E_3 corresponding to (each coordinate of) the constraints (E_i^3) and the coefficients R_i is rank deficient by 1 if n is even and of full rank otherwise. Similarly, Fourier analysis of the circulant E_4 corresponding to the (E_i^4) and the coefficients C reveals a rank deficiency by one for each $k \in \{1, \ldots, n\}$ such that

$$\cos(2\pi k/n) = \cos(2\pi/n)$$

Since the cosine function is symmetric, E_3 is rank deficient by one for n odd and by two for n even. That is, one may choose M freely and solve for the cordinates of the coefficients C and R relative to M. This shift makes explicit that the constraints (E_i^4) form a homogenous system and hence that the combined system of (E_i^3) and (E_i^4) in C and R is solvable and rank deficient by one in each coordinate. SPLAV solves the system by solving the $n \times n$ system

$$R_{i-2} + (1 - c)R_{i-1} + (1 - c)R_i + R_{i+1} = c(B_{i-1} - cB_i + B_{i+1}) + (2 - c)^2 M \qquad (E_i^{34})$$

with $c := 2\cos(2\pi/n)$ by least squares and setting $C_i = \mathrm{avg}(B_i, R_i, R_{i-1})$. If M is not input, the existing boundary curves are interpreted as cubics (degree raising) and M is chosen as an average of the third coefficients counted from the mesh points. Since E_{34} is still underdetermined, one can hope to find an averaging construction of the form $R_i = \Sigma \alpha_j B_j$ rather than solving the quadratic minimization problem:

$$\min \sum \| R_i - R_i^* \| \text{ subject to expression } (E_i^{34})$$

In the case of $n = 5$ and $n = 7$ (and presumably for all odd n) such a construction with symmetic weights indeed exists. For $n = 5$ the weights are $\alpha_i = \alpha_{i+1} = 0.16540$, $\alpha_{i-1} = \alpha_{i+2} = -0.25416$, $\alpha_{i+3} = 0.48654$. However, this construction is likely to overshoot and 'knot' around the splitting point. Choosing $R_i^* := (B_i + B_{i+1})/2$ and minimization leads to better results.

Remarks

- A (0, 0, 0) match is a parametric C^1 match. That is, the patches covering a facet can be thought of as one piecewise polynomial function.
- Quadratic boundaries may still allow for a cubic interpolant of n-sided facets with arbitrary data.

ADMISSIBLE DATA FOR CUBIC MESH INTERPOLATION VIA SPLITTING

This section analyses the peculiar gap between cubic boundary data that allow for the generic cubic interpolants generated by SPLAV and data that do not allow any cubic splitting interpolant at all. By analyzing Constraint 5 the section tries to characterize 'bad' data, since computational experience suggests that such data occurs too rarely to justify a quartic approach or more elaborate splittings in general. Problems only occur if the boundary curve is cubic. (This motivates efforts to keep the degree of the mesh curves minimal.) since Constraint 5 is sufficient to guarantee a solution, the analysis proceeds under the following assumption on the common boundary curve.

Assumption 1

The data satisfy Constraint 3, there is a point of inflection ($u^0 \times \bar{u} \neq 0$ and $u^1 \times \bar{u} \neq 0$) and, for $[\gamma^i, \eta^i] = \text{ratio}(u^i, v^i, w^i, \mathbf{N}^i)$, $\gamma^0 == \gamma^1$ and $\eta^0 \neq \eta^1$.

The following construction with $\lambda := (\eta^0, 2\rho, \eta^1)$, $\mu := (\gamma^0, 2\sigma, \gamma^1)$ and $\nu := ((1-\gamma^0), 2\tau, (1-\gamma^1))$ shows that Constraint 5 is not necessary. The (2,2,2) match is valid if Assumption 1 holds and, for $F := (u^0 - u^1, v^0 - v^1, w^0 - w^1)$, either rank $F = 3$, or rank $F = 2$ and $(P^1 - P^0) \in \text{span}(u^0, u^1)$, or rank $F = 1$ and $\bar{u} \in \text{span}(u^0 - u^1)$ (Peters[8], Lemma 2.2.3).

Example 2

$[\rho, \sigma, \tau]$ = solution of

$$(u^0 - u^1, v^0 - v^1, w^0 - w^1)(\rho, \sigma, \tau)^t = \bar{u}(\eta^0 - \eta^1)$$

$[\bar{v}, \bar{w}]$ = solution of

$$\gamma^0 \bar{v} + (1-\gamma^0)\bar{w} = -\sigma v^0 - \tau w^0 + \eta^0 \bar{u} + \rho u^0$$

$$\sigma \bar{v} + \tau \bar{w} = \rho \bar{u}$$

Instead of looking at higher, more complex C^1 matches, the analysis looks at $\det(p_u, p_v, q_w)$ directly. Motivated by Piper's example[2], Lemma 2.2.4 of Peters[8] characterizes necessary constraints as follows.

Lemma 3

Under Assumption 1, $\det(p_u, p_v, q_w) = 0$ is equivalent to

$$0 = \mathbf{N}^i x^i \qquad (10_1)$$

$$0 = y^i x^i \qquad (10_2)$$

$$0 = (1 - (\eta^1/\eta^0))\gamma^0 u^1 + \frac{1}{\eta^0}\det(x^0 v^1 w^0) + \frac{1}{\eta^1}\det(x^1 v^0 w^1) + 4\det(\bar{v}\bar{u}\bar{w}) \qquad (10_3)$$

where $i \in \{0, 1\}$, $x^i := (1-\gamma^0)\bar{w} + \gamma^0 \bar{v} - \eta^i \bar{u} \neq 0$, $y^i := v^i \times \bar{w} - w^i \times \bar{v}$ and $\mathbf{N}^i$ is the normal at i.

The following two examples exhibit categories of nonadmissible data, i.e. data that do not allow for a cubic C^1 match.

Example 3

The first category is characterized by $\mathbf{N}^0 = \mathbf{N}^1 := \mathbf{N}$, but $\mathbf{N}\bar{u} \neq 0$. This violates equation (10_1) since $\eta^0 \neq \eta^1$. Here is the specific instance from Piper[2]:

$$A_i^k = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad Q_{2i}^k = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \quad A_j^l = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$$

first level of patch p

$$P^k = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad D_i^k = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad D_j^l = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \quad P^l = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

common boundary

$$A_{i+1}^k = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} \quad Q_{2i+1}^k = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \quad A_{j-1}^l = \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix}$$

first level of patch q

□

If $\mathbf{N}^0 = \mathbf{N}^1 =: \mathbf{N}$ and $\mathbf{N}\bar{u} = 0$, then $\mathbf{N}\bar{v} = 0$ and $\mathbf{N}\bar{w} = 0$ with $\bar{v}$ and $\bar{w}$ on the appropriate opposite sides of the boundary curve guarantees a C^1 match. Hence, the analysis focuses on $\mathbf{N}^0 \neq \mathbf{N}^1$.

Example 4

A second category of nonadmissible data is characterized by

$$\mathbf{N}^0 \bar{u} = 0, \quad \eta^1 \mathbf{N}^1 \bar{u} \neq 0,$$

$$(\mathbf{N}^1 \times \mathbf{N}^0) \times v^1 = \delta(1-\gamma^0)\mathbf{N}^1,$$

$$(\mathbf{N}^1 \times \mathbf{N}^0) \times w^1 = -\delta\gamma^0 \mathbf{N}^1$$

The first equation implies $\mathbf{N}^1 x^1 = 0 = \mathbf{N}^0 x^1$, i.e. $x^1 = \text{const}\mathbf{N}^0 \times \mathbf{N}^1$ and the last two equations force a contradiction to the second equation:

$$0 = \frac{1}{\gamma\delta} x^1 y^1 = \frac{1}{\delta}(\mathbf{N}^0 \times \mathbf{N}^1)(v^1 \bar{w} - w^1 \bar{v})$$

$$= \mathbf{N}^1(\gamma^0 \bar{v} + (1-\gamma^0)\bar{w}) = \eta^1 \mathbf{N}^1 \bar{u}$$

A specific instance is

$$A_i^k = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad Q_{2i}^k = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \quad A_j^l = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$$

first level of patch p

$$P^k = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad D_i^k = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad D_j^l = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \quad P^l = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

common boundary

$$A^k_{i+1} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} \quad Q^k_{2i+1} = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix} \quad A^l_{j-1} = \begin{pmatrix} 3 \\ 1 \\ -1 \end{pmatrix}$$

first level of patch q

□

The first example can be viewed as a part of a staircase where the foot of the step is not a data point. The second features a twist of $\pi/2$ between the neighbouring tangent planes. Since all possibilities must be accounted for, SPLAV monitors the determinant, $\gamma^0 - \gamma^1$, of equation (3). If $|\gamma^0 - \gamma^1| < \mathrm{TOL1}$ (and $|\eta^0 - \eta^1| > \mathrm{TOL2}$) there are, offhand, three options:

- using quartic patches
- introducing an additional point
- perturbing the data

Perturbation is complicated and there are cases (see Figure 4) when symmetric data would have to be perturbed in a subtle way to achieve the desired effect. Also all changes involve more than one boundary[8]. If the data structures provide for it, a local switch to quartics and to the matches discussed earlier is a good solution. SPLAV opts for introducing additional points so that the surface remains cubic if there are only three- and four-sided mesh facets. Figure 5 below displays the different configurations produced by the preprocessor.

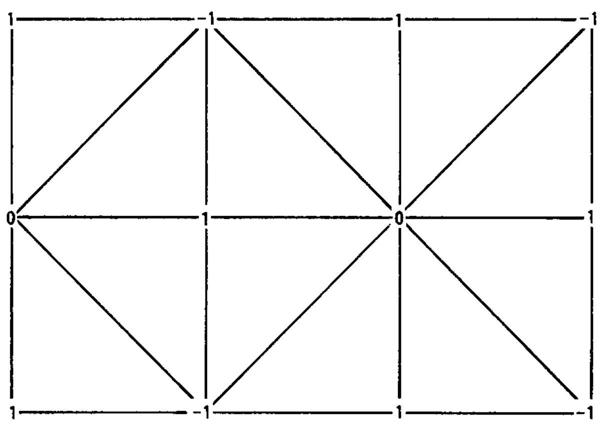

Figure 4. Symmetric data to be completed as in Example 3

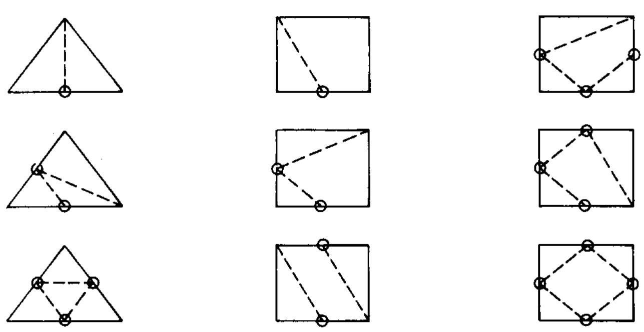

Figure 5. Circles denote added points, dashed lines denote new boundaries

Table 2. Approximating the unit sphere

Points	Volume	Maximum curvature
4	1.3358	1.347
8	1.2153	2.081
12	1.0807	1.005

Figure 6. Skewed dodecahedron with a 5-sided facet

Figure 7. Three semi-doughnuts meet at an angle of $2\pi/3$

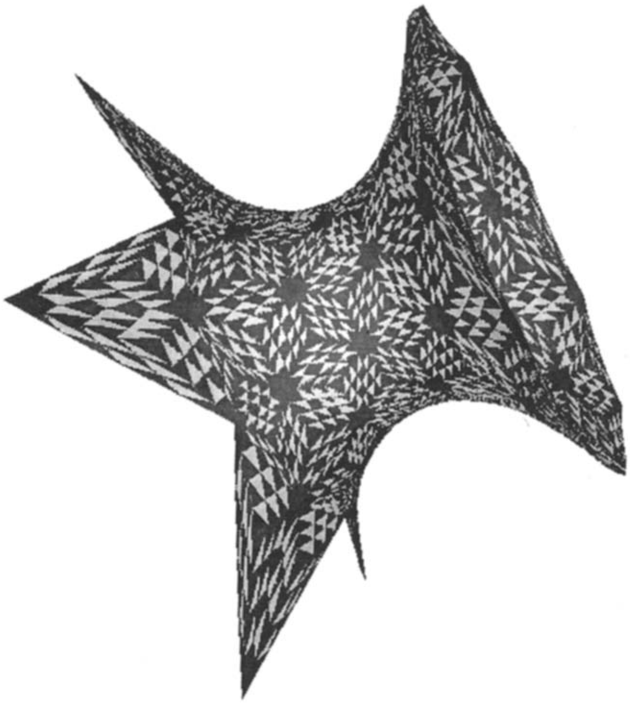

Figure 8. A fancy catenoid

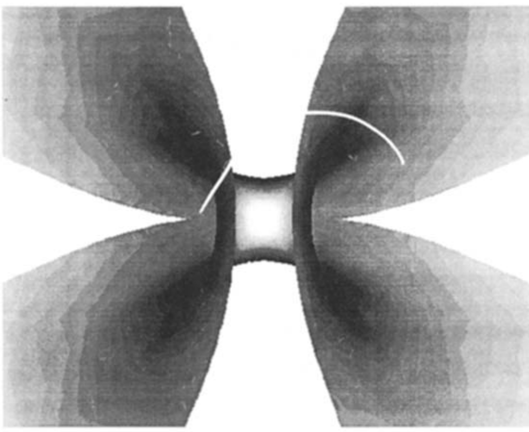

Figure 9. A slice of Enneper's minimal surface

EXAMPLES

This section shows some surfaces generated by SPLAV. Table 2 summarizes the results for approximating the unit sphere. The data points are evenly spaced and the volume is measured relative to the volume of the unit sphere.

Figure 6 shows a variation of the 12-point interpolant: the top point is removed to create a 5-sided facet and the neighbouring points are dragged up and down to destroy any symmetry of the facet boundary. The quartic patches can be recognized as the larger patches covering the northern hemisphere. The checker pattern highlights isoparametric lines and betrays the patch distribution. Figure 7 displays a surface of genus 2. The surface consists of three smoothly connected handles constructed over a mesh of 50 points. Figure 8 shows an open surface. The catenoid 'bangle' is reproduced from sampled data. Figure 9 gives some 'insight' into Enneper's minimal surface[18]. This surface is self-intersecting and hence only a slice is displayed.

ACKNOWLEDGEMENT

I thank Carl de Boor for his comments. This research was supported by Grant number NSF DMS-8701275.

REFERENCES

1 **Farin, G** 'Smooth interpolation to scattered 3D data' in **Barnhill, R F and Boehm, W (eds)** *Surfaces in CAGD* (1983)

2 **Piper, B** 'Visually smooth interpolation with triangular Bézier patches' in **Farin, G (ed.)** *Geometric Modeling* SIAM

3 **Shirman, L A and Séquin, C H** 'Local surface interpolation with Bézier patches' *Computer Aided Geometric Design* Vol 4 (1987) pp 279–295

4 **Jones, A K** 'Nonrectangular surface patches with curvature continuity' *Comput.-Aided Des.* Vol 20 No 6 (1988) pp 325–335

5 **Peters, J** 'Local smooth surface interpolation: a classification' *Comput. Aided Geom. Des.* (1989) (to appear)

6 **Nielson, G M** 'The side-vertex method for interpolation in triangles' *J. Approx. Theory* Vol 25 (1979) pp 318–336

7 **Charrot, P and Gregory, J A** 'A pentagonal surface patch for computer aided geometric design' *Comput. Aided Geom. Des.* Vol 1 (1984) pp 87–94

8 **Peters, J** 'Local piecewise cubic C^1 surface interpolants via splitting and averaging' *CMS Technical Report no. 89-11* (1988) (presented at the Oberwolfach meeting April 1989)

9 **Peters, J** 'Rectangulation algorithms: smooth surface interpolation with bicubics' *CMS Tech. Report No 90-1* University of Wisconsin (1989)

10 **Farin, G** 'Triangular Bernstein-Bézier patches' *Comput. Aided Geom. Des.* Vol 3 (1986)

11 **de Boor, C** 'B-form basics' in **Farin, G (ed.)** *Geometric Modeling* SIAM (1987)

12 **Liu, D** 'A geometric condition for smoothness between adjacent Bézier surface patches' *Acta Mathematicae Sinica* Vol 9 No 4 (1986)

13 **Beeker, E** 'Smoothing of shapes designed with free-form surfaces' *Comput.-Aided Des.* Vol 18 No 4 (1986) pp 224–232

14 **Degen, W** 'Explicit Continuity Conditions for adjacent Bézier Surface Patches' *Preprint No. 5 Math. Inst. B* Universität Stuttgart (1989)

15 **Liu, D and Hoschek, J** 'GC^1 continuity conditions between adjacent rectangular and triangular Bézier surface patches' *Comput.-Aided Des.* Vol 21 No 4 (1989) pp 194–200

16 **Peters, J** 'Local cubic and bicubic C^1 surface interpolation with linearly varying boundary normal' *Comput. Aided Geom. Des.* (1988)

17 **Sabin, M A** *Private communication* (1987)

18 **do Carmo,** *Differential Geometry of Curves and Surfaces* Prentice-Hall (1976)