

G^2 Surface Interpolation Over General Topology Curve Networks

Péter Salvi, Tamás Várady

Budapest University of Technology and Economics

Abstract

The basic idea of curve network-based design is to construct smoothly connected surface patches, that interpolate boundaries and cross-derivatives extracted from the curve network. While the majority of applications demands only tangent plane (G^1) continuity between the adjacent patches, curvature continuous connections (G^2) may also be required. Examples include special curve network configurations with supplemented internal edges, “master-slave” curvature constraints, and general topology surface approximations over meshes.

The first step is to assign optimal surface curvatures to the nodes of the curve network; we discuss different optimization procedures for various types of nodes. Then interpolant surfaces called parabolic ribbons are created along the patch boundaries, which carry first and second derivative constraints. Our construction guarantees that the neighboring ribbons, and thus the respective transfinite patches, will be G^2 continuous. We extend Gregory’s multi-sided surface scheme in order to handle parabolic ribbons, involving the blending functions, and a new sweep-line parameterization. A few simple examples conclude the paper.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—curvenet-based design, transfinite surfaces, Gregory patches, G^2 continuity

1. Introduction

In curve network-based design, surface models are directly defined by a collection of free-form curves, arranged into a single 3D network with general topology. Curves may come from (i) sketch input, (ii) feature curves extracted from orthogonal views, (iii) curves traced on triangular meshes or (iv) direct 3D editing. Once the curves are defined, all the surfaces are generated automatically. This calls for a representation based on geometric information extracted solely from the boundaries. Transfinite surface interpolation is a natural choice, as it does not require a grid of control points to define the interior shape, and all n boundaries are handled uniformly, unlike in the case of trimmed quadrilateral surfaces. The ability to interactively edit prescribed boundaries and cross-derivatives is also an advantage in contrast to recursive subdivision schemes.

The first step of surface generation is to compute cross-directional data, such as tangent planes and curvatures that are shared by adjacent patches. Then interpolant surfaces, called *ribbons*, are generated, that carry first or second-

degree cross-derivative constraints to be eventually interpolated by the transfinite surfaces.

The majority of multi-sided transfinite surfaces are defined over convex domains, combining only linear ribbon surfaces and enabling G^1 continuity between the adjacent patches. At the same time, there are several practical design situations, where this approach is not sufficient, and higher degree continuity is required.

(i) It often occurs that additional curves need to be inserted into the curve network to make it suitable for applying patches with convex domains. The supplemented curves must be compatible with the already defined ribbons, and it is particularly important to produce seamless transitions along these curves. Examples include handling curve configurations with concave angles, or connecting disjoint loops with prescribed slopes (see Figure 9 later in Section 5).

(ii) Another important situation is when a designer wants to create a G^2 connection based on two existing G^1 patches. He may want to retain one surface (the *master*), and modify the adjacent patch (the *slave*) accordingly, see Figure 8, or may prefer an averaged target curvature, see Figure 7.




$n = 2$	
$n = 3$	
$n > 3$	

Figure 1: Node configurations by # of independent curves.

(iii) A third example is when we have a general topology curve network defined over a mesh and would like to obtain a good approximation of the interior data points (see Figure 10). Clearly, if we extract not only the normal vectors, but also curvature information from the underlying mesh, approximation will be more accurate.

In this paper our interest is to generate smooth, curvature continuous transfinite surface patches over a curve network using parabolic ribbons. This is a complex topic, and most papers in the literature only deal with one of its specific facets. Twist compatibility of bi-parametric surface patches around vertices was studied, amongst others, in [Sel81, Pet91], the construction and the properties of curve networks were investigated in [VH96, Her96]. The necessary conditions to embed a curve network into a G^2 surface were recently analyzed in [HPS12]. Methods to create fair curve networks were suggested in [MS94]. Curvature continuous extensions of the Gregory patch were proposed in [GH89, HM99]. Recent developments in transfinite surface interpolation also include two new multi-sided patch representations proposed in [SVR14].

In this paper, we try to give a comprehensive view of the subject, examining general topology curve networks, surface curvatures at vertices, twist compatibility, ribbon generation, and transfinite surface representations, as well as some practical applications.

The rest of the paper is organized as follows. In Section 2 we discuss how to assign optimal surface curvatures to the vertices of the network, that make G^2 interpolation possible. Then, in Section 3, methods to compute linear and parabolic interpolants are discussed. In Section 4, we introduce an extended scheme of Gregory patches, using a new sweep-line parameterization, that merges parabolic ribbons and thus enables curvature continuous connections. A few examples illustrate our surface scheme in Section 5.

2. Surface Curvatures for G^2 networks

Prior to generating our surface model, we need to ensure that the defining network is G^2 . We need to assign optimal surface curvatures to the vertices, and in special cases slightly adjust some curves to meet the G^2 conditions.

In this section we will focus on how to determine opti-

mal surface curvatures in different cases. Assume we have n (at least C^2 continuous) curves meeting at a vertex; some of them terminates there, others pass through it. Here n denotes the number of “independent” curves, which is not necessarily equal to the valence of the vertex. For example, the valence of an X-node is four, but it represents only two independent curves (Figure 1).

We assume that the curves share a common tangent plane. This has a local coordinate system; the tangent of the i th curve spans an angle ω_i with the local x -axis. Each curve is also characterized by its normal curvature, denoted by k_i . We want to determine an optimal surface curvature, that is defined by two unknown principal curvatures κ_1 and κ_2 , and an unknown orientation of the first principal direction, defined by the angle $\lambda = \angle(e_1, x)$ to the x axis. Assume that these quantities are known, then the normal curvature at angle ω is defined by Euler’s equation

$$\kappa(\omega) = \kappa_1 \cos^2(\omega - \lambda) + \kappa_2 \sin^2(\omega - \lambda). \quad (1)$$

Our goal is to compute an optimal triplet of $(\kappa_1, \kappa_2, \lambda)$, that minimizes the given normal curvature deviations in least squares sense, i.e.,

$$\sum_{i=1}^n (\kappa(\omega_i) - k_i)^2 = \min. \quad (2)$$

We will follow the approach suggested in [VH96], where the above non-linear system of equations is transformed into a linear system using another triplet of unknowns (W_1, W_2, W_{12}) , where

$$W_1 = \kappa_1 \cos^2 \lambda + \kappa_2 \sin^2 \lambda, \quad (3)$$

$$W_2 = \kappa_1 \sin^2 \lambda + \kappa_2 \cos^2 \lambda, \quad (4)$$

$$W_{12} = (\kappa_1 - \kappa_2) \sin \lambda \cos \lambda. \quad (5)$$

Using these quantities the curvature constraints are

$$\kappa(\omega_i) = W_1 \cos^2 \omega_i + W_2 \sin^2 \omega_i + W_{12} 2 \sin \omega_i \cos \omega_i. \quad (6)$$

After computing (W_1, W_2, W_{12}) , the optimal surface curvature $(\kappa_1, \kappa_2, \lambda)$ can be easily determined. We will further analyze the above general approach for various n values. In this paper we have extended the basic algorithm of [VH96] to meet the requirements of G^2 curvenet-based design, notably cases (i) and (iv).

(i) *Underdetermined case.* Corner nodes, T-nodes and X-nodes are very frequently encountered in free-form curve networks (Figure 1, $n = 2$). In this case, we have two constraints and three unknowns, thus the system is underdetermined and an additional constraint is needed. We propose minimizing the squared sum of the principal curvature radii, which attempts to avoid flat surface areas and equalize the two values. The optimization

$$\kappa(\omega_i) = k_i, \quad i \in \{1, 2\} \quad (7)$$

$$\frac{1}{\kappa_1^2} + \frac{1}{\kappa_2^2} = \min. \quad (8)$$

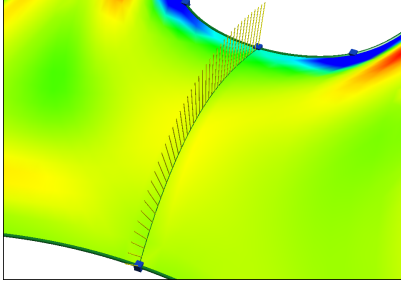


Figure 2: Normal fence and mean curvature map.

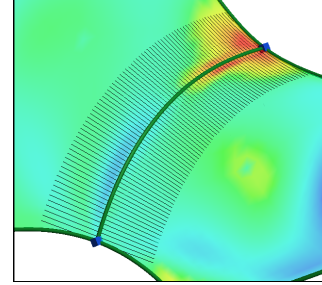


Figure 3: Normal curvature arcs and Gauss curvature map.

leads to a third-degree equation — details can be found in Appendix A. There are two exceptions: at umbilical points $k_1 = k_2$, so we set κ_1 and κ_2 accordingly, and choose λ arbitrarily. When one of the curvatures, say k_2 , is zero (e.g. at the side of a cylinder), the larger principal curvature κ_1 is uniquely determined, substituting ω_1 into Euler's equation.

(ii) *Solving the system.* Typical edge configurations for $n = 3$ — Y-node, T-node with two legs, etc. — are depicted in Figure 1. We have three constraints and three unknowns, so we can solve the system, and the surface curvature matches the three prescribed normal curvatures. The special situations of (i) need to be treated here, as well. (An alternative method was proposed in [VH96] to avoid hyperbolic surface curvatures, when possible.)

(iii) *Minimizing the system.* For $n > 3$, we perform the above least-squares minimization, but the result surface curvature only approximates the prescribed normal curvatures. As we wish to have an exact match, the curves involved may need to be modified. Details of this algorithm are discussed in [VH96], as well. Our practical experience shows that the majority of vertex nodes will fall into cases (i) or (ii), and curve tweaking occurs only in a few cases. Even then, the changes are relatively small, hardly noticeable visually.

(iv) *Constrained minimization.* There is a special case of $n > 3$, when we wish to optimize surface curvature, while one particular curvature constraint $\kappa(\omega_0) = k_0$ must be retained. We minimize the expression $\sum (\kappa(\omega_i) - k_i)^2$ using the Lagrange multiplier method. Details can be found in Appendix A. We will apply this calculation for averaging surface curvatures in the next section.

3. Computing Ribbons

As discussed earlier, cross-derivatives need to be extracted solely from the curve network. Once two ribbons on the opposite sides of a common boundary are set G^1 or G^2 continuous, the corresponding transfinite surfaces will inherit this property. Two ribbons are G^1 (tangent plane) continuous, if their first cross-derivatives are perpendicular to a common sweep of normal vectors (called *normal fence*) along the

common boundary. Figure 2 shows an example, where the fence is rendered as a series of yellow lines.

When we require G^2 (curvature) continuity between two surfaces, we apply the Linkage Curve Theorem [PW92]:

Two surfaces tangent along a C^1 -smooth linkage curve are curvature continuous, if and only if at every point of the linkage curve, their normal curvature agrees for an arbitrary direction other than the tangent of the linkage curve.

This means that if we have a particular directional sweep along the common boundary, and the normal curvatures of the two ribbons in this direction are always the same, then G^2 continuity is satisfied. Figure 3 shows an example with common normal curvatures rendered as circular arcs. (The matching colors of the Gauss map show G^2 continuity.)

In the rest of this section, we will investigate how to determine normals and curvatures from a curve network.

3.1. Preliminaries

A ribbon is a bi-parametric surface $R(s, d)$; s is the *side parameter*, and d is the *distance parameter*. The side parameter runs in the interval $[0, 1]$ along the boundary curve. The distance parameter is defined in the cross direction; it is zero on the boundary and increases as we move away from it.

For a given n -sided patch, there is a loop of curves, $P_i(s_i)$, and we need to create the corresponding ribbons $R_i(s_i, d_i)$, $i \in [1 \dots n]$. The parameters (s_i, d_i) can also be regarded as functions that map values from a common polygonal domain (see Section 4.3). Note also that the indexing is circular with 1 coming after n and vice versa, and we have $P_{i-1}(1) = P_i(0)$ for all i .

Given a boundary curve $P_i(s_i)$ and the corresponding cross-derivative $T_i(s_i)$, a linear ribbon can be written as

$$R_i(s_i, d_i) = P_i(s_i) + d_i T_i(s_i). \quad (9)$$

The equation of a parabolic ribbons is given as

$$R_i(s_i, d_i) = P_i(s_i) + d_i T_i(s_i) + \frac{1}{2} d_i^2 C_i(s_i), \quad (10)$$

where $C_i(s_i)$ denotes the second cross-derivative.

3.2. Normal Fence, Linear Ribbons and Compatibility

At vertices of the network, all curve tangents are in a common plane. For each boundary $P_i(s_i)$, we create a normal fence $N_i(s_i)$ that interpolates the normals at the related corners, and minimizes its rotation along the boundary; this is the well-known rotation-minimizing frame or RMF. An exact (closed form) solution of the underlying differential equation cannot be determined, but approximations can be computed via a sequence of discrete points [WJZL08].

The cross-derivatives of the ribbons can be defined as

$$\begin{aligned} T_i(s_i) &:= \frac{\partial}{\partial d_i} R_i(s_i, 0) \\ &= \alpha(s_i) D_i(s_i) + \beta(s_i) \frac{\partial}{\partial s_i} R_i(s_i, 0), \end{aligned} \quad (11)$$

where $\alpha(s_i)$ and $\beta(s_i)$ are scalar functions, and $D_i(s_i)$ represents a direction vector function, that is perpendicular to $N_i(s_i)$ everywhere. One trivial choice for D_i is

$$D_i(s_i) = N_i(s_i) \times \frac{\partial}{\partial s_i} R_i(s_i, 0), \quad (12)$$

but other definitions are also possible. $D_i(s_i)$ represents a common vector function assigned to the boundary, so G^1 continuity is obviously satisfied.

The scalar functions must satisfy end conditions at the corner points ($s_i = 0$ and $s_i = 1$). There are further degrees of freedom to define these in order to optimize the shape of the patch. For example, cross-derivatives at the middle of the boundary ($s_i = 0.5$) can be prescribed for manual shape editing. Alternatively, these can be optimized by fairing algorithms, which is subject of ongoing research.

The next issue is how to define ribbons that are *compatible* with the prescribed surface curvatures of the network. It is well-known from differential geometry [Pet91] that once the ingoing curves match a common surface curvature, there always exists a set of compatible twist vectors (mixed partial derivatives) at this corner. The “height” of the twist vectors is unambiguously defined by the first and second derivatives of the curves and the Gaussian curvature (K), thus the twist vector of a compatible ribbon R must satisfy the following condition at the corner:

$$\frac{\partial^2}{\partial s \partial d} R \cdot N = \left(\frac{\partial^2}{\partial s^2} R \cdot N \right) \left(\frac{\partial^2}{\partial d^2} R \cdot N \right) - K \left\| \frac{\partial}{\partial s} R \times \frac{\partial}{\partial d} R \right\|^2. \quad (13)$$

This also indirectly determines $D' \cdot N$, since the normal component of the twist vectors must also satisfy

$$\frac{\partial^2}{\partial s \partial d} R \cdot N = \alpha D' \cdot N + \beta P'' \cdot N. \quad (14)$$

An essential consequence of the above equation is that for a curvature compatible normal fence $N' \cdot N$ is also constrained, and instead of the RMF, we have to compute a more general normal fence, that approximates a smoothly varying se-

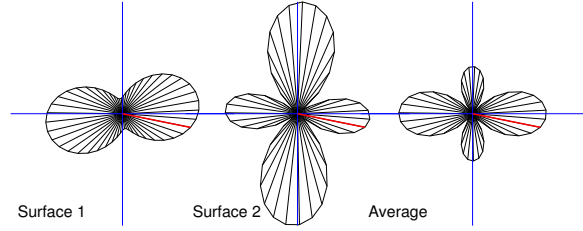


Figure 4: Curvature averaging, visualized in the tangent plane. The red line shows the constrained direction.

quence of normal vectors *and* satisfies the derivative constraints at the two ends, as well, e.g. by constrained fitting.

3.3. Normal Curvatures Along the Edges

We constrain the opposite curvatures to be identical, which ensures curvature continuity by the Linkage Curve Theorem. It would be nice to naturally blend surface curvatures of $(\kappa_1, \kappa_2, \lambda)$ from one endpoint to the other, which would define a “rotation minimizing curvature” function, but this seems to be a difficult problem and no such algorithm is known to the authors. Instead we discuss two important practical cases.

(i) The simplest method is to enforce a master-slave relationship, when we preserve the curvature function of the master patch, and enforce this to the slave to achieve a G^2 connection.

(ii) It is more challenging to ensure G^2 continuity by averaging. Assuming we have already generated two G^1 patches, we wish to compute a surface curvature at every point, such that it (i) interpolates the normal curvature of the common boundary, and (ii) approximates the surface curvatures of the two adjacent patches. This problem can be solved using the constrained surface curvature method of the previous section. At a given point, let ω_0 be the tangent direction of the boundary curve and k_0 its curvature. We take several curvature “measurements” in further ω_i directions from both surfaces, denoted by k_i^A and k_i^B , and define a target curvature as the mean of these two values, i.e., $k_i = (k_i^A + k_i^B)/2$. Then we minimize $\sum (\kappa(\omega_i) - k_i)^2$ using the constraint $\kappa(\omega_0) = k_0$. This method is visually demonstrated in Figure 4, where normal curvatures are drawn using the Euler equation.

3.4. Parabolic Arcs and Parabolic Ribbons

After the surface curvature along the boundary is determined, we need to compute appropriate parabolic arcs in the local sweeping planes defined by $T_i(s_i)$ and $N_i(s_i)$. The normal curvature $\kappa(s_i)$ in the direction of $T_i(s_i)$ is calculated using Euler’s formula. We still have some freedom in determining the exact arcs, as only the tangent direction and the curvature is constrained.

For ease of computation, let us transform the parabolic arc of R_i at a fixed \hat{s}_i into a local coordinate system, where the first point is the origin, and the tangent of the arc is the local x -axis. Then at a given boundary point, the equation of the parabola can be written as a quadratic Bézier curve of three control points:

$$R_i(\hat{s}_i, d_i) = B_0^2(d_i) \cdot (0, 0) + B_1^2(d_i) \cdot (l(\hat{s}_i)/2, 0) + B_2^2(d_i) \cdot (l(\hat{s}_i), h(\hat{s}_i)). \quad (15)$$

Assuming that the width of the parabolic ribbon $l(\hat{s}_i)$ is the same as the corresponding linear one, $h(\hat{s}_i)$ can be expressed by means of the prescribed curvature as $h(\hat{s}_i) = \frac{1}{2} \kappa(\hat{s}_i) l^2(\hat{s}_i)$, which defines the second and third control points of the parabolic arc. As a result, parabolic ribbons can either be directly evaluated by the above formula or represented as an approximating B-spline surface.

4. G^2 Gregory Surfaces

At this point we assume that the ribbons have already been determined. Various transfinite surface schemes exist to interpolate ribbons, see the review in [VRS11]. In this paper we elaborate an extended version of the classical Gregory patch. Its equation is

$$S(u, v) = \sum_{i=1}^n I_{i,i-1}(s_i(u, v), s_{i-1}(u, v)) B_{i,i-1}(u, v), \quad (16)$$

where $I_{i,i-1}$ is a corner interpolant between boundary curves i and $i-1$, and $B_{i,i-1}$ denotes a blending function of the related corner. The parameter function $s_i(u, v)$ is associated with the i th side of the polygonal domain; we omit its arguments for brevity. Let us analyze these constituents below.

4.1. Corner Interpolants

Corner interpolants are formulated by two adjacent side-based (linear or parabolic) ribbons. It is well-known from the classical theory of Boolean-sum surfaces, that in order to cancel out unwanted terms coming from two side interpolants, a correction patch $Q_{i,i-1}$ needs to be subtracted (see also Section 4.4):

$$I_{i,i-1}(s_i, s_{i-1}) = R_{i-1}(s_{i-1}, s_i) + R_i(s_i, 1 - s_{i-1}) - Q_{i,i-1}(s_i, s_{i-1}). \quad (17)$$

If we also consider $I_{i+1,i}(s_{i+1}, s_i)$, it is apparent that R_i occurs with two different parameterizations in the surface equation, which we will denote by $R_i^-(s_i, d_i = 1 - s_{i-1})$, and $R_i^+(s_i, d_i = s_{i+1})$.

4.2. Blending Functions

Transfinite surface schemes generally combine individual interpolants by special blending functions, that ensure required interpolation properties. For Gregory patches, *corner blends* are used, that yield 1 at a given corner, then gradually

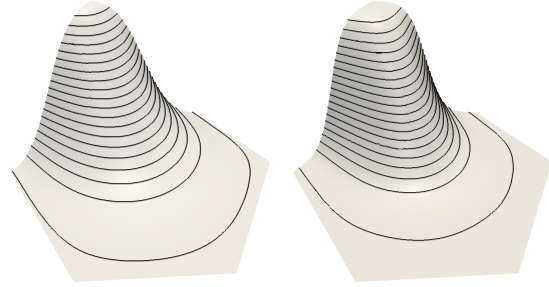


Figure 5: Blend functions with $m = 2$ (left) and $m = 3$ (right).

vanish on the adjacent sides, and become zero as we reach the remaining sides of the polygonal domain.

For each (u, v) point in the domain, we determine an n -tuple of distance values $\Delta_i(u, v)$. The distance Δ_i is zero on the i th side and increases monotonously as we move towards the center of the domain. These values may represent any distance measure with the above properties; for example, we may use distances on sweeping lines, or perpendicular distances dropped to the respective sides of the polygon.

Let $D_{i_1 \dots i_k} = \prod_{j \notin \{i_1 \dots i_k\}} \Delta_j^m$, then the corner blend is defined as

$$B_{i,i-1}(u, v) = \frac{D_{i,i-1}}{\sum_j D_{j,j-1}} \left(= \frac{1/(\Delta_i \Delta_{i-1})^m}{\sum_j 1/(\Delta_j \Delta_{j-1})^m} \right). \quad (18)$$

This function satisfies our requirements — it yields 1 at the $(i-1, i)$ corner, ensures a “gradual” $1 \rightarrow 0$ transition along sides $i-1$ and i as we move away from the corner, and then vanishes on the remaining sides. Note, that the expression in the brackets shows an efficient way to compute blends in the interior of the domain, however, the main formula must be used in the vicinity of polygon sides, where some $\Delta_i \leq \epsilon$, to avoid singularities.

The exponent m controls the variation of the ribbons; compare the two images in Figure 5. This ensures that the resulting surface retains the $(m-1)$ -th derivatives of the ribbons. We use $m = 2$ for linear ribbon patches, and $m = 3$ for parabolic ones, to achieve G^1 and G^2 continuity, respectively.

4.3. Parameterization

The goal of parameterization is to determine local parameters for the corner interpolants from a given (u, v) point. In this paper — for simplicity’s sake — we use regular polygonal domains. Previous research [VRS11] has shown that for extreme boundary configurations, i.e., very uneven side lengths or sudden curvature changes, the use of irregular polygons can improve surface quality.

Let us determine the parameter s_i for the i th side. Traditionally, Gregory patches are parameterized by radial sweeping lines [CG84] (Figure 6a), connecting the domain point

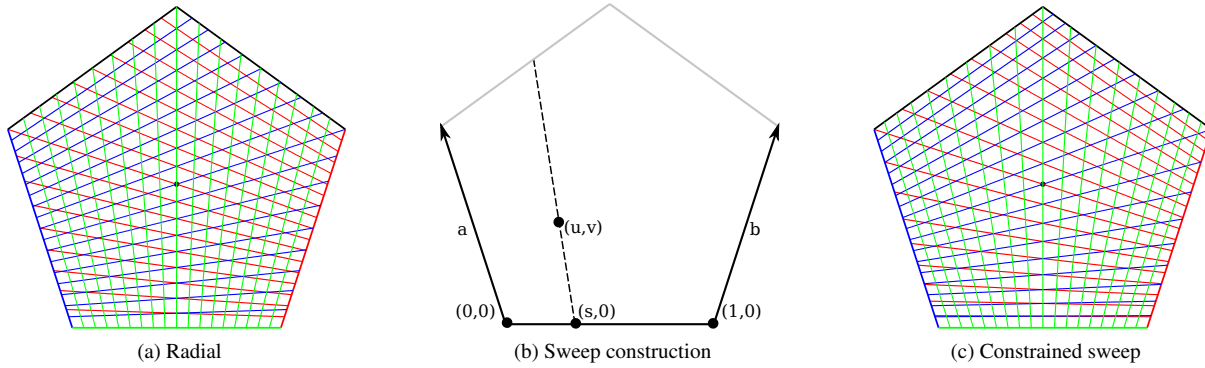


Figure 6: Parameterizations.

in question to the intersection of the extended polygon sides $i-1$ and $i+1$. This is suitable for G^1 continuous surfaces, but further differential properties are required for G^2 continuity. As discussed earlier, ribbon R_i has two parameterizations. It is necessary that these be identical at every point along the boundary, so the parametric speed of s_{i-1} and s_{i+1} should be the same:

$$-\frac{\partial s_{i-1}}{\partial w} = \frac{\partial s_{i+1}}{\partial w} \left(= \frac{\partial d_i}{\partial w} \right), \quad (19)$$

where $\frac{\partial}{\partial w}$ denotes differentiation along an arbitrary parametric direction in (u, v) , i.e., w is equal either to u or v , or any combination of them. While ribbons R_i^- and R_i^+ have different parameters in the interior of the domain, we refer to these parameterizations uniformly by d_i , when the above condition applies on side i .

This condition can be satisfied, if we create sweeping lines by a special construction using Hermite polynomials. Without loss of generality, let the base edge be a segment from $(0,0)$ to $(1,0)$, a and b edge vectors associated with sides $i-1$ and $i+1$, respectively, and (u, v) the point to be mapped, see Figure 6b. Then we can construct the equation

$$(u, v) = (s, 0) + d[a \cdot H(s) + b \cdot H(1-s)], \quad (20)$$

where $H(s) = 2s^3 - 3s^2 + 1$ is the third-degree Hermite polynomial[†], and s and d are unknown. This leads to a fourth-degree equation in s , but that does not pose any difficulty for real-time computation, as efficient algorithms exist for solving higher degree polynomial equations [PTVF92], and values for a given resolution can also be cached. The result is depicted in Figure 6c. Here $I_{i,i-1}$ is parametrized by the green and blue sweep lines, $I_{i+1,i}$ by the red and green; at the bottom the blue sweep lines of side $i-1$ and the red sweep lines of side $i+1$ are identical in a differential sense.

[†] Using fifth-degree Hermite polynomials instead also ensures $-\frac{\partial^2 s_{i-1}}{\partial w^2} = \frac{\partial^2 s_{i+1}}{\partial w^2} \left(= \frac{\partial^2 d_i}{\partial w^2} \right)$, resulting in C^2 interpolation.

4.4. Correction Terms

Now we return to corner interpolants and investigate the partial derivatives of two — linear or parabolic — ribbons, meeting at a common corner point. We need a single interpolant surface, but the partial derivatives are not necessarily identical, which poses the well-known problem of *twist incompatibility*. Let us introduce the notation $t_i = 1 - s_{i-1}$. We would need

$$\begin{aligned} \frac{\partial^{p+q}}{\partial s_i^p \partial t_i^q} R_i(0,0) &= \frac{\partial^{p+q}}{\partial t_i^q \partial s_i^p} R_{i-1}(1,0) \\ &=: W_{p,q} \quad p, q \in \{0, 1, 2\}. \end{aligned} \quad (21)$$

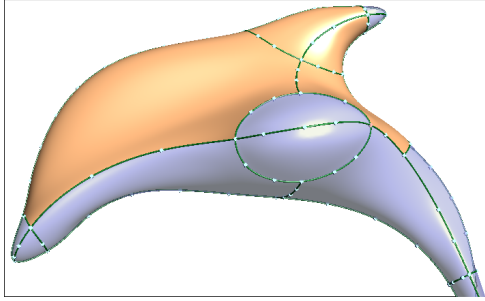
The curve network satisfies this equation for $p = q = 0$, as the boundaries meet at a fixed corner point. It is also natural to require that the equation holds for $p = 0, q = 1$ and $p = 1, q = 0$, constraining the first cross-derivatives of the ribbons to the tangents of the neighboring curves. When some partial derivatives of the ribbons are not compatible, we need to apply Gregory's rational twists. These replace the constant vectors $W_{p,q}$ by rational expressions combining the two parametric variables (see below). The correction patch for G^2 corner interpolants is defined as

$$\begin{aligned} Q_{i,i-1}(s_i, t_i) &= P_i(0) + s_i W_{1,0} + t_i W_{0,1} + s_i t_i W_{1,1} \\ &\quad + \frac{1}{2} s_i^2 W_{2,0} + \frac{1}{2} t_i^2 W_{0,2} + \frac{1}{2} s_i^2 t_i W_{2,1} \\ &\quad + \frac{1}{2} s_i t_i^2 W_{1,2} + \frac{1}{4} s_i^2 t_i^2 W_{2,2}, \end{aligned} \quad (22)$$

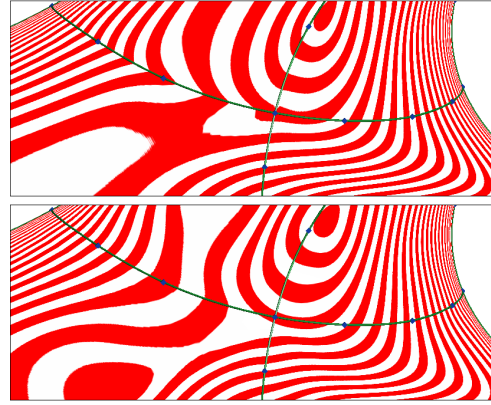
where each W is a rational function of s_i and t_i . Their computation is fairly straightforward, generalizing the classical Gregory twists [Wor84]. As an illustration, we show one such term, the remaining ones are similar:

$$W_{1,2}(s_i, t_i) = \frac{s_i^2 \frac{\partial^2}{\partial t_i^2} T_{i-1}(1) + t_i \frac{\partial}{\partial s_i} C_i(0)}{s_i^2 + t_i}. \quad (23)$$

Substituting 0 for either s_i or t_i eliminates one of the conflicting partial derivatives where needed.



(a) Dolphin model (curve network courtesy of C. Grimm).



(b) Isophotes — G^1 (top) and G^2 (bottom).

Figure 7: Continuity between Gregory patches of a complex model.

For G^1 curve networks, the twist vectors are often not set compatible, and Gregory terms must be used. In our G^2 case, the curves match a common surface curvature at each node, so the compatibility constraints of $W_{2,0}$, $W_{0,2}$ for the curves and $W_{1,1}$ for the twists are satisfied. However, rational corrections need to be applied for the remaining terms of $W_{2,1}$, $W_{1,2}$ and $W_{2,2}$.

5. Examples

Surface models in this paper were generated by a prototype system written in C++. Table 1 demonstrates computation times for G^1 vs. G^2 representations, evaluating a test object shown in Fig. 7a. It consists of two 2-sided, four 3-sided, five 4-sided, two 5-sided and two 6-sided patches; and the network contains 10 B-spline curves. While normal fence generation is really fast, parabolic ribbons require significantly more computation, as these are based on sampling and curvature estimations using the initial G^1 patches. Once parabolic ribbons are ready, the actual evaluation of surface points increases only to a minor extent (by a factor of 1.5).

The close-up view in Figure 7b shows an X-node with linear vs. parabolic ribbons; the target curvatures were computed by the averaging method in Section 3.3. It can be seen, that with parabolic ribbons the isophotes change smoothly across the shared boundaries, showing G^2 continuity.

In the rest of this section, we present some applications of parabolic ribbons.

5.1. Master-Slave Constructions

In computer-aided design, surfaces are typically defined by a well-defined hierarchy; for example, fillets smoothly join

	Ribbon computation	Surface model evaluation
G^1	0.02s	2.6s
G^2	5.1s	4.0s

Table 1: Average running times on a 2.8GHz CPU. The surface model was evaluated at 4000 sample points.

large primary surfaces, vertex blends connect several fillets, and so on. In the context of transfinite surfaces, this means that curvature information needs to be propagated along some boundary curves. We wish to retain the curvature of *master* surfaces, and enforce curvature constraints to ad-

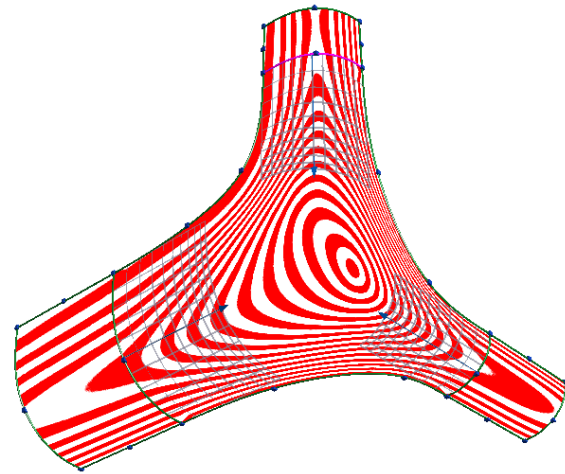


Figure 8: Vertex blend with three master surfaces.

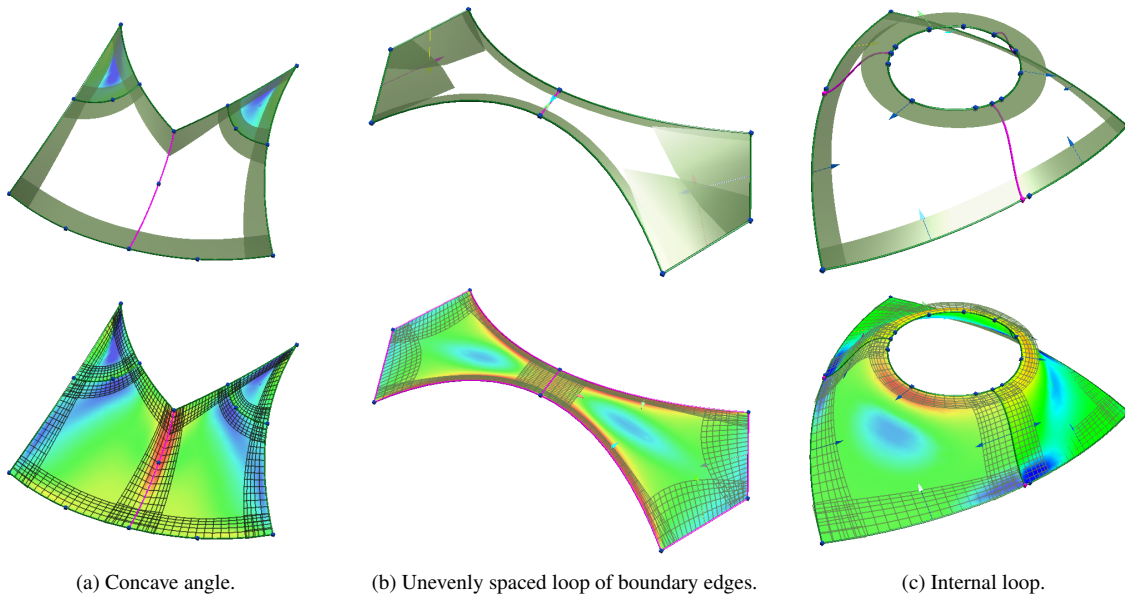


Figure 9: Improved surfacing using connection curves.

jacent *slave* surfaces. This can be accomplished by extracting cross curvatures from the master, and generating slaves using G^2 transfinite patches. Such an example is shown in Figure 8, where a setback vertex blend was created, satisfying curvature continuity joining three fillets, i.e., we have three master surfaces, and one slave in the middle.

5.2. Difficult Curve Network Configurations

While most of the time G^1 transfinite surface interpolation is sufficient to produce a collection of smooth convex patches, there are certain cases, where we need to insert artificial *connection curves* into the network in order to handle complex configurations or avoid shape artifacts, e.g.:

- curve loops with a concave angle,
- unevenly spaced loop of boundary edges,
- connecting disjoint loops (holes).

Examples are shown in Figure 9. In these cases, first we insert connection curves, and create G^1 patches. After taking the average of the curvatures, we compute parabolic ribbons and regenerate the patches, now with curvature continuity. As a result, seamlines within these composite patches cannot be noticed visually. In practical design applications it is assumed that the composite patches are automatically generated, and connection curves remain hidden from the users.

5.3. Mesh Approximation

Creating a highly compressed representation for a mesh is an important task, that can be accomplished by general topol-

ogy surfaces. First, a network of curves is drawn on the mesh, which also defines a multi-sided patch structure. By deducing boundary curves and cross-derivatives from the mesh, transfinite surfaces are created that nicely approximate the data points of the entire model. Using locally estimated normal vectors, we can create normal fences and linear ribbons for G^1 patches. Estimating local curvatures along the boundaries, as well, makes it possible to create parabolic ribbons and G^2 patches, which will produce smoother and more accurate surface models (see Figure 10).

6. Conclusion and Future Work

We have discussed an approach for G^2 transfinite surface interpolation, by (i) assigning curvatures to the nodes and edges of a general topology curve network, (ii) creating parabolic ribbons, and (iii) interpolating these with a G^2 extension of Gregory patches, based on corner interpolants and a special sweepline parameterization. Continuity issues and the computation of linear and parabolic ribbons were explained in details. A few applications where G^2 ribbons are needed have also been presented.

It should be noted that G^2 surface patches are more sensitive to the quality of the defining network than their G^1 counterparts; so applying fairing methods for curves and ribbons remains a challenging subject for future research. Another area of interest is to determine optimal transfinite patches for approximating triangular meshes. We also plan to apply GPU-s to speed up computations.

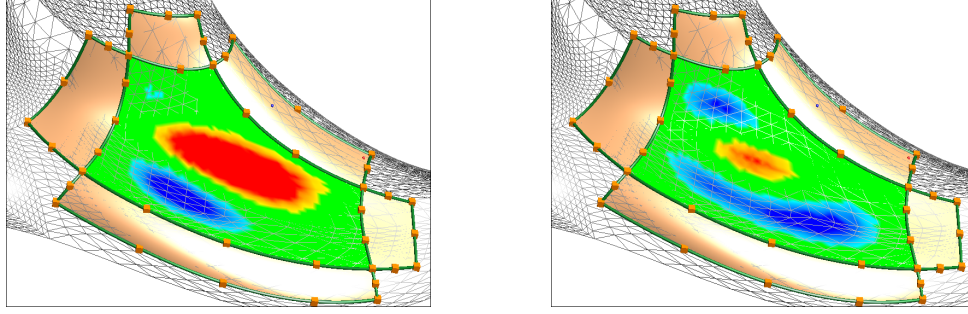


Figure 10: Deviation from the mesh using linear (left) and parabolic (right) ribbons. Relative deviations with respect to the bounding box are $(-0.38\%, +0.45\%)$ for the G^1 , and $(-0.22\%, +0.11\%)$ for the G^2 case.

Acknowledgements

This work was supported by the Hungarian Scientific Research Fund (OTKA, No. 101845). The pictures in this paper were generated by the Sketches system developed by ShapEx Ltd., Budapest. The contribution of György Karikó to develop this prototype system is highly appreciated.

References

- [CG84] CHARROT P., GREGORY J. A.: A pentagonal surface patch for computer aided geometric design. *Computer Aided Geometric Design* 1, 1 (1984), 87–94.
- [GH89] GREGORY J. A., HAHN J. M.: A C^2 polygonal surface patch. *Computer Aided Geometric Design* 6, 1 (1989), 69–75.
- [Her96] HERMANN T.: G^2 interpolation of free form curve networks by biquintic Gregory patches. *Computer Aided Geometric Design* 13, 9 (1996), 873–893.
- [HM99] HALL R., MULLINEUX G.: Continuity between Gregory-like patches. *Computer Aided Geometric Design* 16, 3 (1999), 197–216.
- [HPS12] HERMANN T., PETERS J., STROTMAN T.: Curve networks compatible with G^2 surfacing. *Computer Aided Geometric Design* 29, 5 (2012), 219–230.
- [MS94] MORETON H. P., SEQUIN C. H.: Minimum variation curves and surfaces for computer-aided geometric design. In *Designing Fair Curves and Surfaces*, Sapidis N. S., (Ed.). SIAM, 1994, pp. 123–159.
- [Pet91] PETERS J.: Smooth interpolation of a mesh of curves. *Constructive Approximation* 7, 1 (1991), 221–246.
- [PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, 1992.
- [PW92] PEGNA J., WOLTER F.-E.: Geometrical criteria to guarantee curvature continuity of blend surfaces. *Journal of Mechanical Design* 114, 1 (1992), 201–210.
- [Sel81] SELESNICK S. A.: Local invariants and twist vectors in computer-aided geometric design. *Computer Graphics and Image Processing* 17, 2 (1981), 145–160.
- [SVR14] SALVI P., VÁRADY T., ROCKWOOD A.: Ribbon-based transfinite surfaces. *Computer Aided Geometric Design* (2014).
- [VH96] VÁRADY T., HERMANN T.: Best fit surface curvature at vertices of topologically irregular curve networks. In *Mathematics of Surfaces VI* (1996), Clarendon Press, pp. 411–427.

[VRS11] VÁRADY T., ROCKWOOD A., SALVI P.: Transfinite surface interpolation over irregular n-sided domains. *Computer Aided Design* 43, 11 (2011), 1330–1340.

[WJZL08] WANG W., JÜTTLER B., ZHENG D., LIU Y.: Computation of rotation minimizing frames. *Transactions on Graphics* 27, 1 (2008), 2.

[Wor84] WORSEY A.: A modified C2 Coons' patch. *Computer Aided Geometric Design* 1, 4 (1984), 357–360.

Appendix A: Surface Curvature Formulae

The details of computing surface curvatures for cases (i) and (iv) in Section 2 are given, as follows.

Underdetermined Case

We write W_1 and W_2 in terms of W_{12} as

$$W_i = a_i W_{12} + b_i, \quad i \in \{1, 2\},$$

using

$$\begin{aligned} a_1 &= 2(s_2 c_2 s_1^2 - s_1 c_1 s_2^2)/d, & b_1 &= (k_1 s_2^2 - k_2 s_1^2)/d, \\ a_2 &= 2(s_1 c_1 c_2^2 - s_2 c_2 c_1^2)/d, & b_2 &= (k_2 c_1^2 - k_1 c_2^2)/d, \end{aligned}$$

where $d = s_2^2 c_1^2 - s_1^2 c_2^2$, $s_i = \sin \omega_i$, and $c_i = \cos \omega_i$. Applying

$$\begin{aligned} 1/\kappa_1^2 + 1/\kappa_2^2 &= \frac{\kappa_1^2 + \kappa_2^2}{\kappa_1^2 \kappa_2^2} = \frac{W_1^2 + W_2^2 + 2W_{12}^2}{[W_1 W_2 - W_{12}^2]^2} \\ &= f(W_1, W_2, W_{12}) = \hat{f}(W_{12}), \end{aligned}$$

leads to the third-degree equation

$$\hat{f}'(W_{12}) = c_3 W_{12}^3 + c_2 W_{12}^2 + c_1 W_{12} + c_0 = 0,$$

where the coefficients c_i are

$$\begin{aligned} c_3 &= 2((a_1 - a_2)^2 - a_1 a_2(a_1^2 + a_2^2) + 2), \\ c_2 &= 6(1 - a_1 a_2)(a_1 b_1 + a_2 b_2), \\ c_1 &= 2(2(b_1^2 + b_2^2 + b_1 b_2) - 3a_1 a_2(b_1^2 + b_2^2)), \\ c_0 &= -2(a_1 b_2^3 + a_2 b_1^3). \end{aligned}$$

Solving the equation gives W_{12} , and from that W_1 and W_2 can be determined. Using the equations

$$\begin{aligned} \kappa_{1,2} &= (W_1 + W_2 \pm \sqrt{D})/2, \\ D &= W_1^2 + W_2^2 - 2W_1 W_2 + 4W_{12}^2, \end{aligned}$$

we can compute the principal curvatures; we get the angle λ from

$$\sin^2 \lambda = \frac{W_1 - \kappa_1}{\kappa_2 - \kappa_1}.$$

Note, that there is a slight ambiguity here — we need the negative root when $W_{12} < 0$, and the positive otherwise.

Constrained Case

In the same vein as above, we write the minimization as

$$\sum_i \left(W_1 c_i^2 + W_2 s_i^2 + 2W_{12} s_i c_i - k_i \right)^2 \rightarrow \min,$$

and augment it with a Λ multiplier:

$$f(W_1, W_2, W_{12}, \Lambda) = \sum_i \left(W_1 c_i^2 + W_2 s_i^2 + 2W_{12} s_i c_i - k_i \right)^2 + \Lambda \left(W_1 c_0^2 + W_2 s_0^2 + 2W_{12} s_0 c_0 - k_0 \right).$$

We need to solve the equation system $\frac{\partial}{\partial k} f = 0$, $\frac{\partial}{\partial W_1} f = 0$, $\frac{\partial}{\partial W_2} f = 0$, $\frac{\partial}{\partial W_{12}} f = 0$, leading to

$$\begin{bmatrix} c_0^2 & s_0^2 & s_0 c_0 & 0 \\ 2c_i^4 & 2s_i^4 c_i^2 & 4s_i c_i^3 & c_0^2 \\ 2s_i^2 c_i^2 & 2s_i^4 & 4s_i^3 c_i & s_0^2 \\ 2s_i c_i^3 & 2s_i^3 c_i & 4s_i^2 c_i^2 & s_0 c_0 \end{bmatrix} \cdot \begin{bmatrix} W_1 \\ W_2 \\ W_{12} \\ \Lambda \end{bmatrix} = \begin{bmatrix} k_0 \\ 2c_i^2 k_i \\ 2s_i^2 k_i \\ 2s_i c_i k_i \end{bmatrix},$$

with an implicit sum for all terms with an i index. Note, that if the ω_i directions are fixed, we can speed up the computation by precomputing the LU decomposition of this matrix.

Appendix B: Proof of Continuity

We provide here the outline of a proof that extended Gregory patches match the parabolic ribbons in a G^2 sense. To save space, arguments of some functions are omitted, when this cannot cause any ambiguity. From here onwards, we evaluate points on the i th side, where $\Delta_i = 0$, $s_{i-1} = 1$ and $s_{i+1} = 0$.

The reproduction of the cross-derivatives on side i depends on $I_{i,i-1}$ and $I_{i+1,i}$. These, per definition, interpolate the R_i^- and R_i^+ ribbons, respectively, i.e.,

$$\begin{aligned} I_{i,i-1}(s_i, 1) &= R_i(s_i, 0) = I_{i+1,i}(0, s_i), \\ \frac{\partial}{\partial w} I_{i,i-1}(s_i, 1) &= \frac{\partial}{\partial w} R_i(s_i, 0) = \frac{\partial}{\partial w} I_{i+1,i}(0, s_i), \\ \frac{\partial^2}{\partial w^2} I_{i,i-1}(s_i, 1) &= \frac{\partial^2}{\partial w^2} R_i^-(s_i, 0), \\ \frac{\partial^2}{\partial w^2} R_i^+(s_i, 0) &= \frac{\partial^2}{\partial w^2} I_{i+1,i}(0, s_i), \end{aligned}$$

where R_i is used to signify when R_i^- and R_i^+ are the same. The direction w is equal to u or v , or an arbitrary direction in the domain, as before. Recall our parameterization requirement from Section 4.3,

$$-\frac{\partial s_{i-1}}{\partial w} = \frac{\partial s_{i+1}}{\partial w} \left(= \frac{\partial d_i}{\partial w} \right),$$

that ensures the equality of first derivatives, since

$$\frac{\partial}{\partial s_{i-1}} R_i^- \frac{\partial s_{i-1}}{\partial w} = \frac{\partial}{\partial s_{i+1}} R_i^+ \frac{\partial s_{i+1}}{\partial w}.$$

The second derivatives, on the other hand, are different.

We will use two important properties of the blending functions related to the i th boundary, which we list here without proof:

$$B_{i,i-1} + B_{i+1,i} = 1, \quad (24)$$

$$\frac{\partial^k}{\partial w^k} B_{j,j-1} = 0, \quad j \notin \{i, i+1\}, \quad k \in \{1, 2\} \quad (25)$$

From the above it also follows that

$$\frac{\partial^k}{\partial w^k} (B_{j,j-1} + B_{j+1,j}) = 0, \quad j \notin \{i-1, i+1\}, \quad k \in \{1, 2\} \quad (26)$$

In the following subsections we prove that the surface reproduces the parabolic ribbon in C^0 , C^1 , and G^2 sense.

C^0 continuity. This is trivially true, using property (24):

$$\begin{aligned} S &= R_i^- B_{i,i-1} + R_i^+ B_{i+1,i} \\ &= R_i(s_i, 0)(B_{i,i-1} + B_{i+1,i}) = R_i(s_i, 0). \end{aligned}$$

C^1 continuity. Half of the equation vanishes due to (25), leaving

$$\begin{aligned} \frac{\partial}{\partial w} S &= \frac{\partial}{\partial w} R_i^- B_{i,i-1} + \frac{\partial}{\partial w} R_i^+ B_{i+1,i} \\ &+ \left[R_i^- \frac{\partial}{\partial w} B_{i,i-1} + R_i^+ \frac{\partial}{\partial w} B_{i+1,i} \right]. \end{aligned}$$

Since $R_i^- = R_i^+$, we can use property (26) to eliminate the last term:

$$\frac{\partial}{\partial w} S = \frac{\partial}{\partial w} R_i(s_i, 0)(B_{i+1,i} - B_{i,i-1}) = \frac{\partial}{\partial w} R_i(s_i, 0).$$

G^2 continuity. Similarly, we can eliminate a large part of the equation using (25) again, after which the following remains:

$$\begin{aligned} \frac{\partial^2}{\partial w^2} S &= 2 \left[\frac{\partial}{\partial w} R_i^- \frac{\partial}{\partial w} B_{i,i-1} + \frac{\partial}{\partial w} R_i^+ \frac{\partial}{\partial w} B_{i+1,i} \right] \\ &+ \frac{\partial^2}{\partial w^2} R_i^- B_{i,i-1} + \frac{\partial^2}{\partial w^2} R_i^+ B_{i+1,i} \\ &+ \left[R_i^- \frac{\partial^2}{\partial w^2} B_{i,i-1} + R_i^+ \frac{\partial^2}{\partial w^2} B_{i+1,i} \right]. \end{aligned}$$

The first and last terms vanish due to (26). Substituting the derivatives of R_i^- and R_i^+ into the equation, we get

$$\begin{aligned} \frac{\partial^2}{\partial w^2} S &= P_i'(s_i) \frac{\partial^2 s_i}{\partial w^2} + T_i(s_i) \left(\frac{\partial^2 s_{i+1}}{\partial w^2} B_{i+1,i} + \frac{\partial^2 s_{i-1}}{\partial w^2} B_{i,i-1} \right) \\ &+ P_i''(s_i) \left(\frac{\partial s_i}{\partial w} \right)^2 + 2T_i'(s_i) \frac{\partial s_i}{\partial w} \frac{\partial d_i}{\partial w} + C_i(s_i) \left(\frac{\partial d_i}{\partial w} \right)^2, \end{aligned}$$

using the equivalent $\frac{\partial d_i}{\partial w}$ instead of $-\frac{\partial s_{i-1}}{\partial w}$ and $\frac{\partial s_{i+1}}{\partial w}$. The first and second derivatives by d_i are much simpler:

$$\begin{aligned} \frac{\partial}{\partial d_i} S &= \frac{\partial}{\partial d_i} R_i(s_i, 0) = T_i(s_i), \\ \frac{\partial^2}{\partial d_i^2} S &= C_i(s_i) + \xi_i \cdot T_i(s_i), \end{aligned}$$

where ξ_i is an appropriate constant. Consequently, the curvature in this direction is

$$\kappa = \frac{T_i(s_i) \times (C_i(s_i) + \xi_i \cdot T_i(s_i))}{\|T_i(s_i)\|^3} = \frac{T_i(s_i) \times C_i(s_i)}{\|T_i(s_i)\|^3},$$

which is the same as the curvature of the ribbon R_i , thus this proves G^2 continuity via the Linkage Curve Theorem.