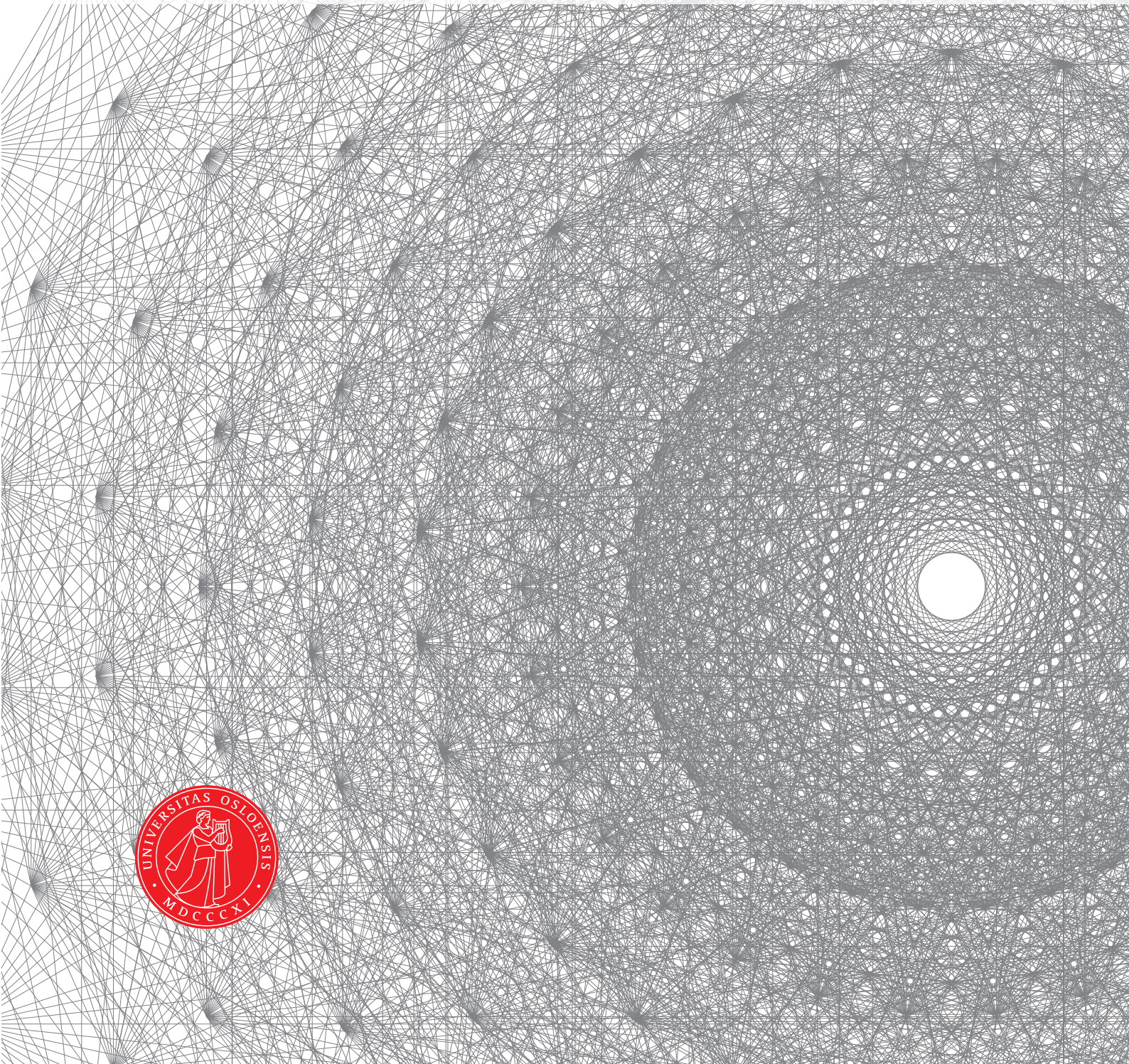
UiO : Department of Mathematics
University of Oslo

Transfinite surface interpolation over irregular *n*-sided domains

Carl Emil Kåshagen

Master's Thesis, Autumn 2017



This master's thesis is submitted under the master's programme *Computational Science and Engineering*, with programme option *Computational Science*, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Abstract

Parametric representations of surfaces in Computer-Aided Geometric Design (CAGD) are often based on connected patches with rectangular parameter domains. Given a loop of four space curves and normal-derivative curves, we want to find a parametric surface that interpolates the boundary data in a C^1-continuous way. The Coons patch developed by Steven A. Coons in the 1960s[4], is a well known technique for constructing such surfaces. However, the need to construct patches with non-rectangular domains can often occur within a rectangular patch framework[3]. In a recent paper by Várady, Rockwood & Salvi[22], three different methods which generalizes the original Coons patch to match n boundary curves using irregular n-sided domains, were presented. Another transfinite interpolation method called cubic mean value interpolation based on mean value coordinates was introduced by Floater & Schulz[10]. The purpose of this thesis is to review and compare these methods. All the methods were successfully implemented in MATLAB®. We discuss the pros and cons of the different constructions, and provide several numerical examples to compare the shape qualities and computational efficiency.

Contents

Chapter 1

Introduction

The construction of a function over a planar domain that matches a given function on the boundary, is often referred to as transfinite interpolation. This is because, in contrast to classical interpolation schemes where a finite number of distinct points are interpolated, a transfinite interpolation scheme match an infinity number of points on the boundary.

Transfinite interpolation has applications in both geometric modelling and in finite element methods[6]. We will focus on constructing patches interpolating a loop of 3D boundary curves. The boundary curves often comes from a network of curves representing a complex free-form object. The majority of the patches in a model are often four-sided, but general n-sided patches can also occur in a rectangular patch framework. See for example figure 1.1. Since we don't have any interior data points in this situation, it is difficult to apply traditional surfacing techniques. One solution is to only use

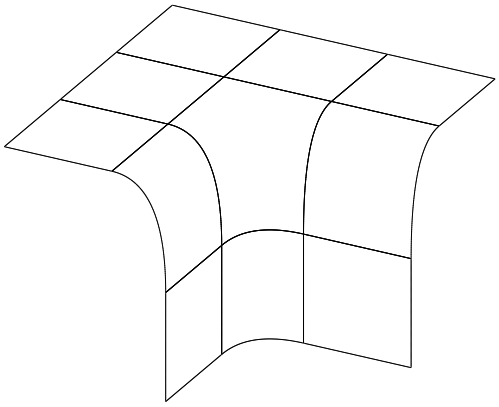


Figure 1.1: A 5-sided hole in a network of 4-sided surfaces.

quadrilaterals, and when an n-sided patch is needed it is stitched together

by multiple four-sided patches[21]. The problem with this approach is that it's difficult to find optimal splitting curves, and they have to be recomputed every time we modify the boundary curves. Another approach is to use recursive subdivision (see for example [15]). The Coons patch[4] is a well known solution for four-sided curve networks. The surface is obtained by "blending" the boundary curves together using appropriate blending functions.

In a recent paper by Várady, Rockwood & Salvi[22] three n-sided generalizations of the original Coons patch was presented; two side-based and one corner-based interpolant. The side-based schemes blend together side interpolants which are four-sided biparametric surfaces that interpolate one side each. These side interpolants are also called *ribbons*, and contains both positional and tangential boundary data. The corner based interpolant blends together biparametric surfaces which interpolate two consecutive sides. Since the corner-interpolants can be defined in terms of side-interpolants, we will sometimes refer to all these methods as *ribbon-based* methods.

Another transfinite interpolation scheme called cubic mean value (CMV) interpolation was introduced by Floater and Schulz in [10], and is an extension of mean value interpolation[6]. Mean value interpolation are based on mean value coordinates, which is a type of generalized barycentric coordinates (GBC's).

In contrast to the generalized Coons patches, the CMV interpolant can not be expressed in a single equation, but is rather defined pointwise. Computing one point on the surface involves solving boundary integrals which in general requires numerical integration. However, in [2] it was shown that the boundary integrals have simple closed form solutions in the case in which the domain is a polygon and the boundary data is piecewise quadratic.

The thesis is structured as follows. Chapter 2 gives an introduction to some variations of the original Coons patch. We also shortly discuss some related theory, including Gordon surfaces and a triangular version of the Coons patch. In Chapter 3 we review the three ribbon based methods together with different parametrization schemes, and prove their continuity properties. We also consider three different methods for constructing appropriate domain polygons given a loop of boundary functions. At the end of the chapter we give some numerical examples and discuss the results. In Chapter 4 we give a short introduction to generalized barycentric coordinates. Two types of GBC's are considered, namely Wachspress coordinates and mean value coordinates. Mean value and cubic mean value interpolation are reviewed in Chapter 5. We provide numerical examples and compare them to the ribbon-based patches. In Chapter 6 we derive closed form solution of the boundary integrals needed to compute the CMV interpolant in the case in which the boundary functions are piecewise quadratic polynomials. We also give a few

examples. A brief summary and discussion concludes the thesis in Chapter 7.

Chapter 2

Introduction to Coons patches

The Coons Patch was developed in the 1960s by Steven A. Coons who worked in the car industry for Ford[4]. The Coons patches are still widely used in both computer aided geometric design and in the finite element method.

2.1 Rectangular Coons surfaces

2.1.1 Bilinearly Blended Coons Patch

Let $\boldsymbol{F}(u,v) = [x(u,v), y(u,v), z(u,v)]$ be a given vector valued function defined on the unit square. We want to construct a surface that interpolates the boundary curves $\boldsymbol{F}(0,v)$, $\boldsymbol{F}(1,v)$, $\boldsymbol{F}(u,0)$ and $\boldsymbol{F}(u,1)$. By first linearly interpolating between the curves $\boldsymbol{F}(0,v)$ and $\boldsymbol{F}(1,v)$, and then between $\boldsymbol{F}(u,0)$ and $\boldsymbol{F}(u,1)$, we obtain two *ruled* surfaces:

$$\begin{aligned}\boldsymbol{c}_1(u,v) &= (1-u)\boldsymbol{F}(0,v) + u\boldsymbol{F}(1,v).\\ \boldsymbol{c}_2(u,v) &= (1-v)\boldsymbol{F}(u,0) + v\boldsymbol{F}(u,1).\end{aligned}$$

By adding together $\boldsymbol{c}_1(u,v)$ and $\boldsymbol{c}_2(u,v)$, and subtracting the correction patch

$$\boldsymbol{c}_3(u,v) = (1-u)(1-v)\boldsymbol{F}(0,0)+(1-u)v\boldsymbol{F}(0,1)+u(1-v)\boldsymbol{F}(1,0)+uv\boldsymbol{F}(1,1), \tag{2.1}$$

we obtain the surface

$$\boldsymbol{S}(u,v) = \boldsymbol{c}_1(u,v) + \boldsymbol{c}_2(u,v) - \boldsymbol{c}_3(u,v). \tag{2.2}$$

Definition 2.1 *The surface defined by $\boldsymbol{S}(u,v)$ in equation 2.2 is called the bi-linearly blended Coons patch[7].*

It is easy to check that this surface interpolates the boundary curves. If we let $u = 0$ in equation 2.2, we get

$$\begin{aligned}\boldsymbol{S}(0,v) &= \boldsymbol{F}(0,v) + (1-v)\boldsymbol{F}(0,0) + v\boldsymbol{F}(0,1) - (1-v)\boldsymbol{F}(0,0) - v\boldsymbol{F}(0,1) \\ &= \boldsymbol{F}(0,v).\end{aligned}$$

Similarly it can be shown that it also interpolates the three other curves. Neighbouring patches with a common boundary curve will therefore be joined with C^0 continuity. However, the cross boundary derivatives are not in general continuous on the boundary, so the patch will not be C^1 continuous. An example is shown in figure 2.1a, where two bilinearly blended Coons patches share one common boundary function. The two patches are defined over

(a) (b)

Figure 2.1: Coons patches: (a) Bilinearly blended; (b) Partially bicubically blended

$u \in [0,2]$, $v \in [0,1]$, and share the common boundary function $\boldsymbol{F}(1,v)$. Even though the boundary functions $v = 0$ and $v = 1$ are differentiable, we clearly see that the cross boundary derivative is discontinuous along $u = 1$. The reason for this lies in the fact that the cross boundary derivatives on one boundary depends on data not belonging to that boundary. For example, we have

$$\begin{aligned}\boldsymbol{S}_u(1,v) = &-\boldsymbol{F}(0,v) + \boldsymbol{F}(1,v) + (1-v)\boldsymbol{F}_u(1,0) + v\boldsymbol{F}_u(1,1) \\ &+ (1-v)\boldsymbol{F}(0,0) + v\boldsymbol{F}(0,1) - (1-v)\boldsymbol{F}(1,0) - v\boldsymbol{F}(1,1).\end{aligned}$$

Note that in the process of constructing the bilinearly blended Coons patch, we added two surfaces as an intermediate step. Adding two surfaces in this way is actually illegal since the sum of two surfaces would depend on

the choice of coordinate system. A cleaner way to derive the formula 2.2 can be achieved by the use of operators. Define the operators $\mathcal{P}_1$ and $\mathcal{P}_2$ on a function of two variables by

$$\mathcal{P}_1\boldsymbol{F}(u,v) = (1-u)\boldsymbol{F}(0,v) + u\boldsymbol{F}(1,v),$$
$$\mathcal{P}_2\boldsymbol{F}(u,v) = (1-v)\boldsymbol{F}(u,0) + v\boldsymbol{F}(u,1).$$

These operators have the property of being *idempotent*, meaning that when they are applied to its own output, the result is unchanged, e.i. $\mathcal{P}_i\mathcal{P}_i = \mathcal{P}_i$. Operators with this property are also called *projectors*. We want to express the bilinearly blended Coons patch in terms of these projectors. The surface $\mathcal{P}_1\boldsymbol{F}$ interpolates the two boundaries $u = 0$ and $u = 1$, and has the "error" $\boldsymbol{F} - \mathcal{P}_1\boldsymbol{F}$. If we add a correction surface to $\mathcal{P}_1\boldsymbol{F}$ that interpolates this error surface on the boundaries, we will get a surface that interpolate to all four boundaries. This correction surface can be obtained by applying $\mathcal{P}_2$ to the error surface. The resulting surface is

$$\mathcal{P}\boldsymbol{F} = \mathcal{P}_1\boldsymbol{F} + \mathcal{P}_2(\boldsymbol{F} - \mathcal{P}_1\boldsymbol{F}).$$

The operator $\mathcal{P}$ can then be written as

$$\mathcal{P} = \mathcal{P}_1 + \mathcal{P}_2 - \mathcal{P}_1\mathcal{P}_2.$$

This way of obtaining $\mathcal{P}$ from $\mathcal{P}_1$ and $\mathcal{P}_2$ is called the *boolean sum* and is often written as

$$\mathcal{P} = \mathcal{P}_1 \oplus \mathcal{P}_2.$$

The term $\mathcal{P}_1\mathcal{P}_2\boldsymbol{F}$ is the bilinear interpolant to the patch corners, same as the correction patch $\boldsymbol{c}_3$. Therefore, the surface $\mathcal{P}\boldsymbol{F}$ is the same as 2.2.

It can also be shown that $\mathcal{P}$ is a projector. Since the operators $\mathcal{P}_1$ and $\mathcal{P}_2$ commute, we have

$$\begin{aligned}\mathcal{P}\mathcal{P} &= (\mathcal{P}_1 + \mathcal{P}_2 - \mathcal{P}_1\mathcal{P}_2)(\mathcal{P}_1 + \mathcal{P}_2 - \mathcal{P}_1\mathcal{P}_2)\\ &= \mathcal{P}_1\mathcal{P}_1 + 2\mathcal{P}_1\mathcal{P}_2 - 2\mathcal{P}_1\mathcal{P}_1\mathcal{P}_2 + \mathcal{P}_2\mathcal{P}_2 - 2\mathcal{P}_1\mathcal{P}_2\mathcal{P}_2 + \mathcal{P}_1\mathcal{P}_1\mathcal{P}_2\mathcal{P}_2\\ &= \mathcal{P}_1 + \mathcal{P}_2 - \mathcal{P}_1\mathcal{P}_2 = \mathcal{P}.\end{aligned}$$

2.1.2 Partially bicubically blended Coons patch

The functions $u, 1-u, v$ and $1-v$ used in equation 2.2 are called *blending functions* because they are used to "blend" two boundary functions together. Other choices of blending functions will produce different surfaces.

We can write equation 2.2 more generally by introducing the blending functions $f_1(u)$, $f_2(u)$, $g_1(v)$ and $g_2(v)$:

$$\begin{aligned}\boldsymbol{S}(u,v) = &\begin{bmatrix} f_1(u) & f_2(u) \end{bmatrix} \begin{bmatrix} \boldsymbol{F}(0,v) \\ \boldsymbol{F}(1,v) \end{bmatrix} + \begin{bmatrix} \boldsymbol{F}(u,0) & \boldsymbol{F}(u,1) \end{bmatrix} \begin{bmatrix} g_1(v) \\ g_2(v) \end{bmatrix} \\ &- \begin{bmatrix} f_1(u) & f_2(u) \end{bmatrix} \begin{bmatrix} \boldsymbol{F}(0,0) & \boldsymbol{F}(0,1) \\ \boldsymbol{F}(1,0) & \boldsymbol{F}(1,1) \end{bmatrix} \begin{bmatrix} g_1(v) \\ g_2(v) \end{bmatrix}\end{aligned} \tag{2.3}$$

To make sure that the boundaries are interpolated, we need two restrictions on f_i and g_i. They must satisfy $f_1(0) = g_1(0) = 1$ and $f_1(1) = g_1(1) = 0$, and each pair must sum to one, i.e. $f_1(u) + f_2(u) \equiv 1$ and $g_1(u) + g_2(u) \equiv 1$.

To construct a C^1 continuous patch, we need to separate the cross boundary derivatives from data belonging to the opposite boundary. As we will see, this can be achieved by choosing blending functions that have zero slopes at the end points, i.e. $f_i'(0) = f_i'(1) = g_i'(0) = g_i'(1) = 0$, for $i = 1, 2$. One choice is to use cubic Hermite basis functions. The four cubic Hermite basis functions are defined as follows

$$\begin{aligned}\alpha_0(u) &= 2u^3 - 3u^2 + 1 \\ \alpha_1(u) &= -2u^3 + 3u^2 \\ \beta_0(u) &= u^3 - 2u^2 + u \\ \beta_1(u) &= u^3 - u^2\end{aligned} \tag{2.4}$$

Definition 2.2 *The partially bicubically blended Coons patch[7] is defined by equation 2.3, where*

$$f_1 = g_1 = \alpha_0, \qquad f_2 = g_2 = \alpha_1.$$

This patch will join adjacent patches with C^1 continuity, but this is only guaranteed if the they all are of the same type. Thus, assume $\boldsymbol{S}(u,v)$ is part of a network of partially bicubically blended Coons patches. In the same way as for the bi-linear Coons patch it's easy to check that it interpolates the boundary functions. To show C^1 continuity, we have to show that adjacent patches have the same partial derivatives $\boldsymbol{S}_u(u,v)$ and $\boldsymbol{S}_v(u,v)$ for all points (u,v) on their common boundary. If we assume that (u,v) is a point on the boundary of the unit square, and that the boundary functions are

differentiable, we have that

$$\begin{aligned}\boldsymbol{S}_u(u,v) &= \begin{bmatrix}\alpha_0'(u) & \alpha_1'(u)\end{bmatrix}\begin{bmatrix}\boldsymbol{F}(0,v)\\ \boldsymbol{F}(1,v)\end{bmatrix} + \begin{bmatrix}\boldsymbol{F}_u(u,0) & \boldsymbol{F}_u(u,1)\end{bmatrix}\begin{bmatrix}\alpha_0(v)\\ \alpha_1(v)\end{bmatrix}\\ &\quad - \begin{bmatrix}\alpha_0'(u) & \alpha_1'(u)\end{bmatrix}\begin{bmatrix}\boldsymbol{F}(0,0) & \boldsymbol{F}(0,1)\\ \boldsymbol{F}(1,0) & \boldsymbol{F}(1,1)\end{bmatrix}\begin{bmatrix}\alpha_0(v)\\ \alpha_1(v)\end{bmatrix}\\ &= \begin{cases}\boldsymbol{F}_u(u,0) & \text{for } v=0\\ \boldsymbol{F}_u(u,1) & \text{for } v=1\\ \alpha_0(v)\boldsymbol{F}_u(0,0)+\alpha_1(v)\boldsymbol{F}_u(0,1) & \text{for } u=0\\ \alpha_0(v)\boldsymbol{F}_u(1,0)+\alpha_1(v)\boldsymbol{F}_u(1,1) & \text{for } u=1\end{cases},\end{aligned}$$

where we have used that $\alpha_i'(0) = \alpha_i'(1) = 0$ for $i = 0, 1$. Clearly, when $v = 0$ or $v = 1$, we have that $\boldsymbol{S}_u(u,v) = \boldsymbol{F}_u(u,v)$. We also see that the two other boundaries, $u = 0$ and $u = 1$, doesn't any longer depend on data belonging to other boundaries than themselves. By symmetry a similar result holds for $\boldsymbol{S}_v(u,v)$. Thus, a neighbouring patch $\boldsymbol{G}$ will have the same partial derivatives on the common edge.

Figure 2.1 compares the partially bicubically blended Coons patch with the bilinearly blended Coons patch. We have used the same boundary data, and the cross boundary derivatives are set to zero. We clearly see that the surface is smoother across the common boundary curve, confirming C^1 continuity.

2.1.3 Bicubically blended Coons Patch

One problem about the partially bicubically blended Coons patch is that flat spots often occurs at the corners[7]. This is because the *twist vectors* $\boldsymbol{S}_{uv}(u,v) = \boldsymbol{0}$ for $u, v \in \{0,1\}$. The reason for this stems from the fact that we are only using blending functions with zero derivatives at the endpoints. This problem can be fixed by modifying the partially bicubically blended Coons patch using all the cubic Hermite basis functions in 2.4. In addition to the function values, we will also need to specify the cross boundary derivatives

$$\boldsymbol{F}_v(u,0), \qquad \boldsymbol{F}_u(1,v), \qquad \boldsymbol{F}_v(u,1), \qquad \boldsymbol{F}_u(0,v).$$

Definition 2.3 *Let*

$$U = \begin{bmatrix}\alpha_0(u) & \beta_0(u) & \alpha_1(u) & \beta_1(u)\end{bmatrix},$$

$$V = \begin{bmatrix} \alpha_0(v) & \beta_0(v) & \alpha_1(v) & \beta_1(v) \end{bmatrix},$$

$$S^u = \begin{bmatrix} \boldsymbol{F}(u,0) & \boldsymbol{F}_v(u,0) & \boldsymbol{F}(u,1) & \boldsymbol{F}_v(u,1) \end{bmatrix},$$

$$S^v = \begin{bmatrix} \boldsymbol{F}(0,v) & \boldsymbol{F}_u(0,v) & \boldsymbol{F}(1,v) & \boldsymbol{F}_u(1,v) \end{bmatrix}$$

and

$$S^{uv} = \begin{bmatrix} \boldsymbol{F}(0,0) & \boldsymbol{F}_u(0,0) & \boldsymbol{F}(1,0) & \boldsymbol{F}_u(1,0) \\ \boldsymbol{F}_v(0,0) & \boldsymbol{F}_{uv}(0,0) & \boldsymbol{F}_v(1,0) & \boldsymbol{F}_{uv}(1,0) \\ \boldsymbol{F}(0,1) & \boldsymbol{F}_u(0,1) & \boldsymbol{F}(1,1) & \boldsymbol{F}_u(1,1) \\ \boldsymbol{F}_v(0,1) & \boldsymbol{F}_{uv}(0,1) & \boldsymbol{F}_v(1,1) & \boldsymbol{F}_{uv}(1,1) \end{bmatrix}.$$

The bicubically blended Coons patch is then defined as[7]

$$\boldsymbol{S}(u,v) = V(S^u)^T + S^v U^T - V S^{uv} U^T. \tag{2.5}$$

To see that the patch really interpolates the boundaries, first note that

$$\begin{aligned} S^{uv} &= \begin{bmatrix} (S^u(0))^T & (\frac{\partial S^u}{\partial u}(0))^T & (S^u(1))^T & (\frac{\partial S^u}{\partial u}(1))^T \end{bmatrix} \\ &= \begin{bmatrix} (S^v(0))^T & (\frac{\partial S^v}{\partial v}(0))^T & (S^v(1))^T & (\frac{\partial S^v}{\partial v}(1))^T \end{bmatrix}^T. \end{aligned}$$

On the edge $u = 0$, we get

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix},$$

so

$$S(0,v) = V(S^u(0))^T + \boldsymbol{F}(0,v) - V(S^u(0)) = \boldsymbol{F}(0,v).$$

The same result can be shown for the remaining boundaries in a similar way. We can also show that the patch interpolates the first order derivatives along the boundaries, and the mixed derivatives at the corners. Since

$$U'(0) = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix},$$

we get

$$S_u(0,v) = V(\frac{\partial S^u(0)}{\partial u})^T + \boldsymbol{F}_u(0,v) - V(\frac{\partial S^u(0)}{\partial u})^T = \boldsymbol{F}_u(0,v).$$

In a similar way we can also show that the other cross boundary derivatives are interpolated. Now consider the corner point $(u,v) = (0,0)$. Since

$$U'(0) = V'(0) = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix},$$

we get

$$S_{uv}(0,0) = \boldsymbol{F}_{vu}(0,0) + \boldsymbol{F}_{uv}(0,0) - \boldsymbol{F}_{uv}(0,0) = \boldsymbol{F}_{uv}(0,0).$$

The proof for the remaining corners are similar.

While the partially bicubically blended patch connects to adjacent patches with C^1 continuity only if it's part of a network of patches of the same type, the bicubically blended patch connects with C^1 continuity to more general surfaces.

The bicubically blended Coons patch is a generalization of the partially bicubically blended Coons patch since if we set $\beta_0 = \beta_1 = 0$ we get the same patch as in definition 2.2.

Note that this method requires that we also specify the mixed partial derivatives at the corners. Thus, we have gotten rid of the problem with flat spots at the corner as we had with the partially bicubically blended Coons patch.

2.1.4 Twist Compatibility Problem

The bicubically blended Coons patch suffers from a compatibility problem as a result of the twist terms $\boldsymbol{F}_{uv}(0,0), \boldsymbol{F}_{uv}(1,0), \boldsymbol{F}_{uv}(0,1)$ and $\boldsymbol{F}_{uv}(1,1)$ in equation 2.5. We know from calculus that if $\boldsymbol{F}$ is twice continuously differentiable, the order of differentiation is interchangeable, i.e. we can set $\boldsymbol{F}_{vu} = \boldsymbol{F}_{uv}$. However, in our situation this is not always the case. The twists at the corners can be obtain in two different ways by differentiating the cross boundary derivative functions. The twist at $\boldsymbol{F}(0,0)$ can be obtain by differentiating $\boldsymbol{F}_v(u,0)$ with respect to u:

$$\boldsymbol{F}_{vu}(0,0) = \lim_{u \to 0} \frac{\partial}{\partial u} \boldsymbol{F}_v(u,0),$$

or by differentiating $\boldsymbol{F}_u(0,v)$ with respect to v:

$$\boldsymbol{F}_{uv}(0,0) = \lim_{v \to 0} \frac{\partial}{\partial v} \boldsymbol{F}_u(0,v).$$

In the case in which $\boldsymbol{F}_{vu}(0,0) \neq \boldsymbol{F}_{uv}(0,0)$ we have a dilemma: By choosing either one of these twists, the obtained surface will not interpolate to the given data.

There are two ways to deal with this problem. We can try to change the given data so the incompabilities disappear, or we can use a method called *Gregory's square*[7]. In this method the constant twist vectors are replaced

by the variable twist vectors

$$\boldsymbol{F}_{uv}(0,0) = \frac{u\frac{\partial}{\partial v}\boldsymbol{F}_u(0,0) + v\frac{\partial}{\partial u}\boldsymbol{F}_v(0,0)}{u+v}$$
$$\boldsymbol{F}_{uv}(1,0) = \frac{(1-u)\frac{\partial}{\partial v}\boldsymbol{F}_u(1,0) + v\frac{\partial}{\partial u}\boldsymbol{F}_v(1,0)}{1-u+v}$$
$$\boldsymbol{F}_{uv}(0,1) = \frac{-u\frac{\partial}{\partial v}\boldsymbol{F}_u(0,1) + (v-1)\frac{\partial}{\partial u}\boldsymbol{F}_v(0,1)}{-u+v-1}$$
$$\boldsymbol{F}_{uv}(1,1) = \frac{(u-1)\frac{\partial}{\partial v}\boldsymbol{F}_u(1,1) + (v-1)\frac{\partial}{\partial u}\boldsymbol{F}_v(1,1)}{u-1+v-1}$$

These terms are convex combinations of the derivatives of the functions $\boldsymbol{F}_v(u,0), \boldsymbol{F}_u(1,v), \boldsymbol{F}_v(u,1), \boldsymbol{F}_u(0,v)$. The mixed partial derivatives now depend on the direction in which we approach the corner. Approaching $\boldsymbol{F}(0,0)$ along the edge $v=0$ we get $\boldsymbol{F}_{uv}(0,0) = \frac{\partial}{\partial v}\boldsymbol{F}_u(0,0)$, and by approaching the same corner along $u=0$ we get $\boldsymbol{F}_{uv}(0,0) = \frac{\partial}{\partial u}\boldsymbol{F}_v(0,0)$.

2.2 Piecewise Coons and Gordon surfaces

When constructing a three-dimensional geometric model, designers often creates a network of curves which are automatically interpolated by a surface. Assume that the given a network of differentiable curves forming rectangular patches. In order to interpolate each region by bicubically blended Coons patches, we need to extract the cross boundary derivatives and the twist vectors from the given curve network. We start with the twist vectors $\boldsymbol{F}_{uv}(u_i, v_j)$.

2.2.1 Estimating twist vectors

There are several methods for estimating the twist vectors. The simplest method is to just set all them to zero. However, in some cases this can lead to a surface with "flat spots", and is therefore generally not a good idea.

A better method is the so called *Adini's twist*[7]. The idea is to first find the bilinearly blended Coons patch that interpolates the boundary functions, and then calculate the corner twists of this interpolant, and use them as our estimation. Unfortunately, these twist vectors will not guarantee that we get a globally C^1 surface. Instead, we find the bilinearly blended Coons patch that interpolates the outer boundary functions of four adjacent patches, as shown in figure 2.2. This surface has a well defined twist at the parameter value corresponding to the common corner of the four patches. Using this twist as our input twist for the bicubically blended Coons patch,

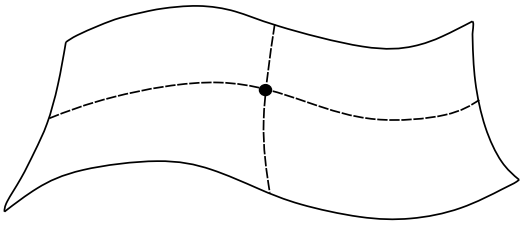


Figure 2.2: The outer boundary curves of four adjacent patches defines a bilinerly blended Coons patch

we are guaranteed a globally C^1 surface. Let the domains be denoted by $[u_i, u_{i+1}] \times [v_j, v_{j+1}]$. It can then be shown that Adini's twist is given by the following formula:

$$\begin{aligned}\boldsymbol{F}_{uv}(u_i, u_j) = &\frac{\boldsymbol{F}_v(u_{i+1}, v_j) - \boldsymbol{F}_v(u_{i-1}, v_j)}{u_{i+1} - u_{i-1}} + \frac{\boldsymbol{F}_u(u_i, v_{j+1}) - \boldsymbol{F}_u(u_i, v_{j-1})}{v_{j+1} - v_{j-1}} \\ &- \frac{\boldsymbol{F}(u_{i+1}, v_{j+1}) - \boldsymbol{F}(u_{i-1}, v_{j+1}) - \boldsymbol{F}(u_{i+1}, v_{j-1}) + \boldsymbol{F}(u_{i-1}, v_{j-1})}{(u_{i+1} - u_{i-1})(v_{j+1} - v_{j-1})},\end{aligned}$$

Another method called *Bessel's twist*[7] estimates the twist at $\boldsymbol{F}(u_i, v_j)$ by first finding the biquadratic interpolant to the nine points $\boldsymbol{F}(u_{i+r}, v_{j+s})$, $s, r \in \{-1, 0, 1\}$, and then use the twist at (u_i, v_j). Since the twist of a biquadratic surface is bilinear, Bessel's twist is the bilinear interpolant to the twists of the four patches formed by the nine points. These twists are given by

$$\boldsymbol{q}_{i,j} = \frac{\Delta^{1,1}\boldsymbol{F}(u_i, v_j)}{\Delta_i \Delta_j},$$

where $\Delta^{1,1}$ is the forward difference operator

$$\Delta^{1,1}\boldsymbol{F}(u_i, v_j) = \boldsymbol{F}(u_{i+1}, v_{j+1}) - \boldsymbol{F}(u_{i+1}, v_j) - \boldsymbol{F}(u_i, v_{j+1}) + \boldsymbol{F}(u_i, v_j),$$

and $\Delta_i = u_{i+1} - u_i$ and $\Delta_j = v_{j+1} - v_j$. Bessel's twist can then be written as

$$\boldsymbol{F}(u_i, v_j) = \begin{bmatrix} 1 - A_i & A_i \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_{i-1,j-1} & \boldsymbol{q}_{i-1,j} \\ \boldsymbol{q}_{i,j-1} & \boldsymbol{q}_{i,j} \end{bmatrix} \begin{bmatrix} 1 - B_j \\ B_j \end{bmatrix},$$

where

$$A_i = \frac{\Delta_{i-1}}{u_{i+1} - u_{i-1}}, \qquad B_j = \frac{\Delta_{j-1}}{v_{j+1} - v_{j-1}}.$$

2.2.2 Estimating tangent ribbons

After we have found the twist vectors we can construct the cross boundary derivatives in the following way. Consider the edge $u = 0$. We want to find $\boldsymbol{F}_v(u, 0)$. The values of $\boldsymbol{F}_v(u, 0)$ in the points $(0, 0)$ and $(1, 0)$ can be found by differentiating the adjacent boundary functions. The twist vectors $\boldsymbol{F}_{uv}(0, 0)$ and $\boldsymbol{F}_{uv}(1, 0)$ gives us the derivatives with respect to u in the same points. Thus we have all the information we need to apply univariate cubic Hermite interpolation. We get

$$\boldsymbol{F}_v(u, 0) = \boldsymbol{F}_v(0, 0)\alpha_0(u) + \boldsymbol{F}_{uv}(0, 0)\beta_0(u) + \boldsymbol{F}_v(1, 0)\alpha_1(u) + \boldsymbol{F}_{uv}(1, 0)\beta_1(u).$$

The other cross boundary functions can be found in a completely analogues way.

2.2.3 Gordon surfaces

Gordon Surfaces[7] were developed by W.Gordon in the late 1960s. Instead of interpolating just four boundary curves, this surface can interpolate a whole network of curves. These curves will be the isoparametric curves of the constructed surface $\boldsymbol{G}$, hence they will be denoted by $\boldsymbol{G}(u_i, v)$, $i = 0, \ldots, m$ and $\boldsymbol{G}(u, v_j)$, $j = 0, \ldots, n$. The curves intersect at the points $\boldsymbol{G}(u_i, v_j)$, $i = 0, \ldots, m$, $j = 0, \ldots, n$.

The construction of a Gordon surface is much like the Coons patch. First, we find a surface $\boldsymbol{G}_1$ that interpolates to the curves $\boldsymbol{G}(u_i, v)$, and another surface $\boldsymbol{G}_2$ that interpolates to the curves $\boldsymbol{G}(u, v_j)$. Then, we add these surfaces together and subtract a surface $\boldsymbol{G}_{12}$.

We start by finding the surface $\boldsymbol{G}_1$. By using a set of blending functions L_i^m which satisfies

$$L_i^m(u_k) = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases},$$

we can define $\boldsymbol{G}_1$ as

$$\boldsymbol{G}_1(u, v) = \sum_{i=0}^{m} \boldsymbol{G}(u_i, v) L_i^m(u).$$

In other words, we take a univariate interpolation scheme and apply it to the curves $\boldsymbol{G}(u_i, v)$. We can for example use the Lagrange polynomials:

$$L_i^m(t) \coloneqq \prod_{\substack{j=0 \\ j \neq i}}^{m} \frac{t - t_j}{t_i - t_j}.$$

The second surface, $\boldsymbol{G}_2$, is defined analogously:

$$\boldsymbol{G}_2(u,v) = \sum_{j=0}^{n} \boldsymbol{G}(u,v_j) L_j^n(v).$$

The last building block, $\boldsymbol{G}_{12}$, is the interpolating tensor product surface

$$\boldsymbol{G}_{12}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \boldsymbol{G}(u_i,v_j) L_i^m(u) L_j^n(v).$$

The Gordon surface can now be defined as

$$\boldsymbol{G}(u,v) = \boldsymbol{G}_1(u,v) + \boldsymbol{G}_2(u,v) - \boldsymbol{G}_{12}(u,v).$$

2.3 Triangular Coons patches

As figure 2.3 illustrates, triangular patches can occur within a rectangular patch system. Several approaches exists for creating a triangular version of

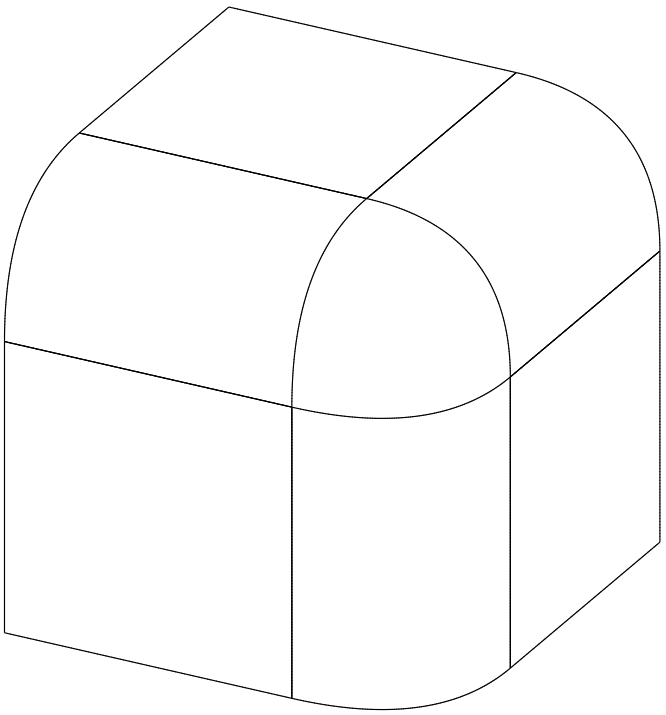


Figure 2.3: A 3-sided hole in a network of 4-sided surfaces

the Coons patch. Coons suggested in [4] to simply let one of the boundary

curves of a rectangular patch be degenerate, and appear as a single point. However, this method is unsymmetrical and restrictions must be imposed on the tangency conditions[12].

Another approach was developed by Barnhill, Birkhoff and Gordon[1]. Let T be a domain triangle with vertices $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$, and let the three given boundary curves be denoted by $\boldsymbol{F}(0, v, w), \boldsymbol{F}(u, 0, w), \boldsymbol{F}(u, v, 0)$. The parameters (u, v, w) are barycentric coordinates with respect to the domain triangle T. In other words, for every point $\boldsymbol{x} \in T$, there exists non-negative numbers u, v, w satisfying

$$\boldsymbol{x} = u\boldsymbol{p}_1 + v\boldsymbol{p}_2 + w\boldsymbol{p}_3$$

and

$$u + v + w = 1.$$

The barycentric coordinates are investigated in more depth in section 1. Now we define three ruled surfaces, each of which interpolates to two boundary functions:

$$\begin{aligned}
\mathcal{P}_1\boldsymbol{F}(u, v, w) &= (1 - r)\boldsymbol{F}(u, 0, w) + r\boldsymbol{F}(u, v, 0) \\
\mathcal{P}_2\boldsymbol{F}(u, v, w) &= (1 - s)\boldsymbol{F}(u, v, 0) + s\boldsymbol{F}(0, v, w) \\
\mathcal{P}_3\boldsymbol{F}(u, v, w) &= (1 - t)\boldsymbol{F}(u, 0, w) + t\boldsymbol{F}(0, v, w),
\end{aligned}$$

where $r = \frac{v}{v+w}$, $s = \frac{w}{w+u}$, $t = \frac{v}{v+u}$. By defining $\mathcal{P}$ as a convex combination of these, we obtain a surface which interpolates all three boundary functions:

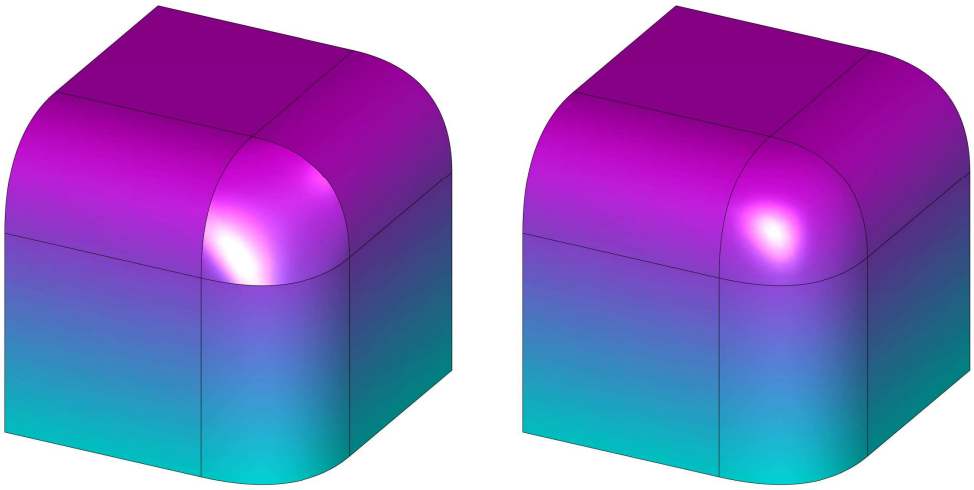
$$\mathcal{P}\boldsymbol{F}(u, v, w) = u\mathcal{P}_1\boldsymbol{F}(u, v, w) + v\mathcal{P}_2\boldsymbol{F}(u, v, w) + w\mathcal{P}_3\boldsymbol{F}(u, v, w). \qquad (2.6)$$

To check that the boundaries are interpolated, first consider a point on the edge $u = 0$. We have that

$$\begin{aligned}
\mathcal{P}\boldsymbol{F}(0, v, w) &= 0 + v\mathcal{P}_2\boldsymbol{F}(0, v, w) + w\mathcal{P}_3\boldsymbol{F}(0, v, w) \\
&= v\boldsymbol{F}(0, v, w) + w\boldsymbol{F}(0, v, w) = \boldsymbol{F}(0, v, w),
\end{aligned}$$

where the last equality follows from the property $u+v+w = 1$. The proof for the two remaining edges are completely analogous. This scheme is sometimes called *trilinear blending*, since the operators $\mathcal{P}_i$ interpolates linearly in r, s, and t. An example is shown in figure 2.4a where the interpolant is used to fill a three-sided "hole" in a rectangular network of biquadratic surfaces.

To construct a C^1 interpolant, we also need the directional derivatives along the boundaries. Let $\boldsymbol{T}_i$ denote the derivative of $\boldsymbol{F}$ taken in the direction



(a) Trilinear blending (b) Tricubic blending

Figure 2.4: Triangular Coons patches

parallel to edge i:

$$\begin{aligned}\boldsymbol{T}_1(u,v,w) &= (v+w)\boldsymbol{F}_{\boldsymbol{d}_1}(u,v,w),\\ \boldsymbol{T}_2(u,v,w) &= (u+w)\boldsymbol{F}_{\boldsymbol{d}_2}(u,v,w),\\ \boldsymbol{T}_3(u,v,w) &= (u+v)\boldsymbol{F}_{\boldsymbol{d}_3}(u,v,w),\end{aligned}$$

where $\boldsymbol{d}_1 = \boldsymbol{p}_2 - \boldsymbol{p}_3, \boldsymbol{d}_2 = \boldsymbol{p}_3 - \boldsymbol{p}_1, \boldsymbol{d}_3 = \boldsymbol{p}_2 - \boldsymbol{p}_1$. The factors $(v+w)$ etc. are necessarily because cubic Hermite interpolation is sensitive to interval lengths. Now we can define $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ to be the cubic Hermite interpolants in r, s, t:

$$\begin{aligned}\mathcal{P}_1\boldsymbol{F}(u,v,w) &= \alpha_0(r)\boldsymbol{F}(u,0,w) + \beta_0(r)\boldsymbol{T}_1(u,0,w)\\ &\quad + \beta_1(r)\boldsymbol{T}_1(u,v,0) + \alpha_1(r)\boldsymbol{F}(u,v,0),\\ \mathcal{P}_2\boldsymbol{F}(u,v,w) &= \alpha_0(s)\boldsymbol{F}(u,v,0) + \beta_0(s)\boldsymbol{T}_2(u,v,0)\\ &\quad + \beta_1(s)\boldsymbol{T}_2(0,v,w) + \alpha_1(s)\boldsymbol{F}(0,v,w),\\ \mathcal{P}_3\boldsymbol{F}(u,v,w) &= \alpha_0(t)\boldsymbol{F}(u,0,w) + \beta_0(t)\boldsymbol{T}_3(u,0,w)\\ &\quad + \beta_1(t)\boldsymbol{T}_3(0,v,w) + \alpha_1(t)\boldsymbol{F}(0,v,w),\end{aligned}$$

Again, we can define the surface by a convex combination of these operators as in equation 2.6. This scheme is sometimes called *tricubic blending*. In [1] it was proven that the patch is C^1.

An example is shown in figure 2.4b, where we have used the same boundary data as in the previous example. The six cross boundary derivative

functions were taken from the adjacent patches, and scaled such that they match the derivatives of the three boundary functions at the corners. Since the scheme interpolates these cross boundary derivatives, they lie in the tangent plane of the adjacent patch. Therefore, the patch joins the neighbouring patches with G^1 continuity.

Other triangular versions of the Coons patch exists, see for example [7] and [12].

Chapter 3

Generalized Coons patches

In the paper by Várady, Rockwood & Salvi[22] three different methods are presented which generalizes the Coons patch for n-sided domains. These methods defines a surface as a convex combination of different side-interpolants. The patch is obtained by adding the side-interpolants multiplied by some blending functions. We will assume that Ω is a convex polygonal domain in the (u, v) parameter plane with vertices $\boldsymbol{p}_i$ ordered counter clockwise. We will use cyclic indices, i.e. $\boldsymbol{p}_{n+1} = \boldsymbol{p}_1$ and $\boldsymbol{p}_{-1} = \boldsymbol{p}_n$ and so on. This kind of indexing will be used throughout the rest of this thesis. The edge from $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$, will be denoted by Γ_i. Each vertex $\boldsymbol{p}_i$ will be associated with a local parametrization (s_i, r_i), where $s_i = s_i(u, v)$ satisfies $s_i(u, v) \in [0, 1]$ for $(u, v) \in \Gamma_i$, and $s_i(\boldsymbol{p}_i) = 0$, $s_i(\boldsymbol{p}_{i+1}) = 1$. The parameter r_i is defined as $r_i = 1 - s_{i-1}$. Also let $d_i = d_i(u, v)$ be a distance parameter, which represents some distance measure from Γ_i. See figure 3.1. We require that $d_i(u, v) = 0$

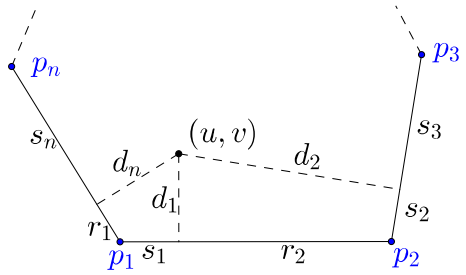


Figure 3.1: Local coordinates

for $(u, v) \in \Gamma_i$, and that it grows monotonously as we move away from Γ_i.

To describe the generalized methods we will use some different notation from what we used in the classical case in chapter 2. The positional and tangential boundary constraints will be denoted by $\boldsymbol{P}_i(s_i)$, $\boldsymbol{T}_i(s_i)$ respectively. The cross boundary derivative functions $\boldsymbol{T}_i$ are defined as

$$\boldsymbol{T}_i(s_i) = \alpha(s_i)\boldsymbol{N}_i(s_i) \times \boldsymbol{P}_i'(s_i) + \beta(s_i)\boldsymbol{P}_i'(s_i),$$

where α and β are scalar functions, and $\boldsymbol{N}_i(s_i)$ is a normal vector function (also called the *normal fence*) corresponding to side i, which interpolates the normals at the corners $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$. Thus, If two adjacent patches which interpolate the boundary constraints have the same normal fence, they will join with G^1 continuity, i.e. all the directional derivatives of both patches in a point $\boldsymbol{P}_i(s_i)$ will lie in the tangent plane spanned by $\boldsymbol{N}_i(s_i) \times \boldsymbol{P}'(s_i)$ and $\boldsymbol{P}'(s_i)$. The scalar functions $\alpha(s_i)$ and $\beta(s_i)$ represent two degrees of freedom, which can be used to give a greater "fullness" to the patch interior. From now on we will assume that we are given the cross boundary derivatives $\boldsymbol{T}_i$, and that they satisfy $\boldsymbol{T}_i(0) = \frac{\partial}{\partial r_i}\boldsymbol{P}_{i-1}(1)$ and $\boldsymbol{T}_i(1) = \frac{\partial}{\partial s_{i+1}}\boldsymbol{P}_{i+1}(0)$ for all i. See figure 3.2. We will also need the twist vectors $\boldsymbol{W}_i(0)$ at the

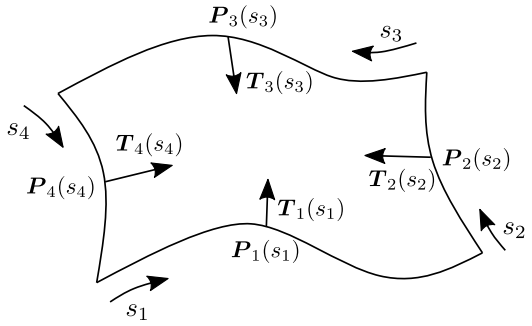


Figure 3.2: Rectangular Coons patch with circular indexing.

corners. These can be derived from the the cross-derivative functions $\boldsymbol{T}_i$ by calculating $\boldsymbol{W}_i(0) = \frac{\partial}{\partial s_i}\boldsymbol{T}_i(0)$ for all i. We will assume that the functions $\boldsymbol{T}_i$ are twist-compatible, i.e. $\frac{\partial}{\partial s_i}\boldsymbol{T}_i(0) = \frac{\partial}{\partial r_i}\boldsymbol{T}_{i-1}(1)$ for all i. Otherwise we need to apply rational twists similar to the ones in Gregory's square in section 1.4 [11].

3.1 Local parameterization schemes

The choice of parameterization scheme is crucial to the shape and continuity properties of the patches. From a given point (u, v) in the domain, we want to

compute distance parameters d_i and local side parameters s_i corresponding to each of the n sides of the domain polygon.

We will consider two kinds of parametrizations schemes: *simple parametrization* and *constrained parametrization*[21]. In simple parametrizations we have the following constraints for a point on side Γ_k:

$$\begin{aligned} d_k &= 0 \qquad s_k \in [0,1] \qquad s_{k-1} = 1 \\ s_{k+1} &= 0 \qquad d_{k-1} = s_k \qquad d_{k+1} = 1 - s_k. \end{aligned} \tag{3.1}$$

In constrained parametrizations we have in addition to 3.1 the constraints:

$$\frac{\partial d_{k-1}}{\partial u} = \frac{\partial s_k}{\partial u} \qquad \frac{\partial d_{k+1}}{\partial u} = -\frac{\partial s_k}{\partial u} \qquad \frac{\partial d_{k-1}}{\partial v} = \frac{\partial s_k}{\partial v} \qquad \frac{\partial d_{k+1}}{\partial v} = -\frac{\partial s_k}{\partial v} \tag{3.2}$$

In other words, for simple parametrization we require that parametrizations corresponding to adjacent sides are identical in a positional sense, while for constrained parametrization we also require that they are identical in a differential sense. Figure 3.3 illustrates this by showing constant parameter lines for s_i, d_{i-1} and d_{i+1}. As we will prove in the following sections, the

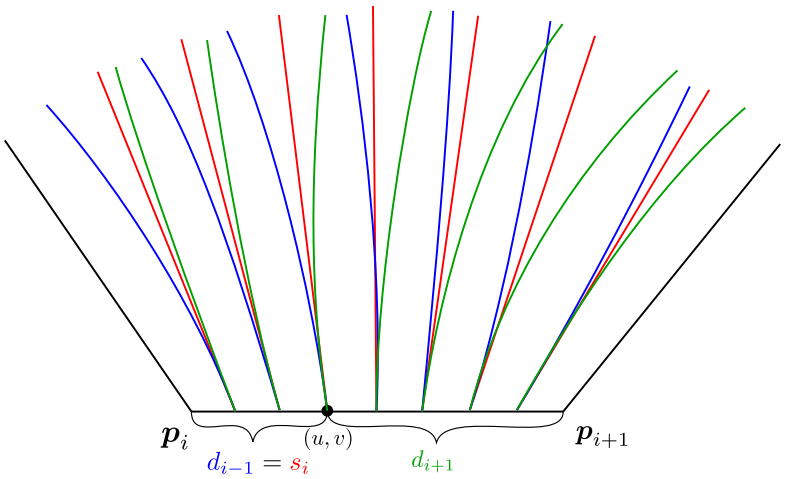


Figure 3.3: Constrained parametrization.

Direct generalization patch requires constrained parametrization to be G^1 continuous, while only simple parametrization is required for the two other patches.

3.1.1 Simple parametrizations

One way to compute the distance parameter is to simply take the length of the perpendicular line from (u, v) to each side. This is not, however, a good

way to define the side parameters s_i as they become negative if the projection of (u, v) falls beyond side i.

Another way to define the distance parameters is by *chord-based coordinates*[22]:

$$d_i = ||(u, v) - \boldsymbol{p}_i|| + ||(u, v) - \boldsymbol{p}_{i+1}|| - ||\boldsymbol{p}_{i+1} - \boldsymbol{p}_i||.$$

A simple method to determine the side parameters from the distance parameters is to let $s_i = d_{i-1}/(d_{i-1} + d_{i+1})$. These pairs of local coordinates will satisfy all the requirements in 3.1, but only if the sides of the domain polygon are of equal length. Another disadvantage of using this method is that s_i is not necessarily linear on side i. As a consequence, it's not guaranteed that the parametric midpoint, i.e. the point on side i where $s_i = 0.5$, will be equal to the actual midpoint of side i.

Radial line sweep

In the paper [3] by Charrot & Gregory, *radial distance functions* were used to find d_i and s_i for regular pentagonal domains. These also work for n-sided convex irregular domains, as long as the domain polygon doesn't have any edge where its adjacent edges are parallel, e.g. a rectangle. To compute (s_i, d_i), first compute the intersection point $\boldsymbol{c}_i$ of the extensions of side Γ_{i-1} and Γ_{i+1}, as shown in figure 3.4. Then the line connecting (u, v) and $\boldsymbol{c}_i$

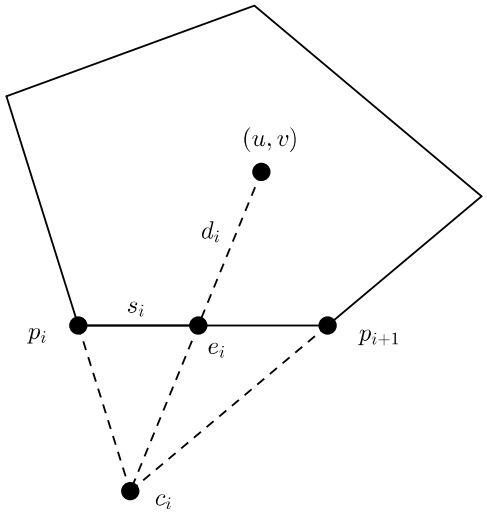


Figure 3.4: radial parametrization

intersects with side Γ_i in a point $\boldsymbol{e}_i$. Now we can define d_i and s_i as

$$d_i = \frac{||(u,v) - \boldsymbol{e}_i||}{||\boldsymbol{p}_{i+1} - \boldsymbol{p}_i||}$$

and

$$s_i = \frac{||\boldsymbol{e}_i - \boldsymbol{p}_i||}{||\boldsymbol{p}_{i+1} - \boldsymbol{p}_i||}.$$

If $(u,v) \in \Gamma_k$, then $(u,v) = \boldsymbol{e}_k$, $\boldsymbol{e}_{k-1} = \boldsymbol{p}_k$ and $\boldsymbol{e}_{k+1} = \boldsymbol{p}_{k+1}$. It follows that $d_k = 0$, $s_k \in [0,1]$, $s_{k-1} = 1$ and $s_{k+1} = 0$. However, the last requirements for simple parametrization, $d_{k-1} = s_k$ and $d_{k+1} = 1 - s_k$, is only satisfied if all the edges of the domain are of equal length.

Central line sweep

One desirable property of the parametrization schemes is that the constant parameter lines $s_i = \frac{1}{2}$ should go through the centre of the domain. This property is not always satisfied by the methods above, and this may lead to undesirable artefacts. Várady suggest a method called *central line sweep*[22] which deals with this problem. Let $\boldsymbol{c} = (c^u, c^v)$ be the center point of the domain calculated as a weighted average of the chord lengths[1]

$$\boldsymbol{c} = \frac{0.5 \sum_i \boldsymbol{p}_i (l_{i-1} + l_i)}{\sum_i l_i} \tag{3.3}$$

Other kinds of center points can also be used such as the average of all the corner points, but the above works better in practice[22]. Our goal is to find a parametrizing function $\boldsymbol{r}(s,d)$ such that the parameter line $\boldsymbol{r}(0.5,d)$ always include the center point $\boldsymbol{c}$. For simplicity's sake, we assume that $\boldsymbol{p}_1$ is placed at the origin, and $\boldsymbol{p}_2$ is on the positive u-axis. Now we define the parametrization function of side 1 as the linear-by-quadratic map

$$\boldsymbol{r}(s,d) = \boldsymbol{p}_2 s + [\boldsymbol{w}_1 (1-s)^2 + 2\boldsymbol{w}_{12}(1-s)s + \boldsymbol{w}_2 s^2]d, \tag{3.4}$$

where $\boldsymbol{w}_1 = \boldsymbol{p}_n - \boldsymbol{p}_1$ and $\boldsymbol{w}_2 = \boldsymbol{p}_3 - \boldsymbol{p}_2$. See figure 3.5. Now, we need to determine the vector $\boldsymbol{w}_{12} = (w_{12}^u, w_{12}^v)$. To simplify calculations, we set $w_{12}^v = 0.5(w_1^v + w_2^v)$. From the equation $c^v = r^v(0.5, d_c) = \frac{1}{4}(w_1^v + 2w_{12}^v + w_2^v)d_c$, we get that $d_c = \frac{2c^v}{w_1^v + w_2^v}$. Now we can solve the other coordinate equation to find w_{12}^u:

$$c^u = r^u(0.5, d_c) = \frac{1}{2}p_2^u + \frac{1}{4}(w_1^u + 2w_{12}^u + w_2^u)d_c.$$

[1] In the paper by Várady there is an error in their formula for $\boldsymbol{c}$. The denominator should be $\sum_i \boldsymbol{p}_i l_i$ instead of $\sum_i l_i$.

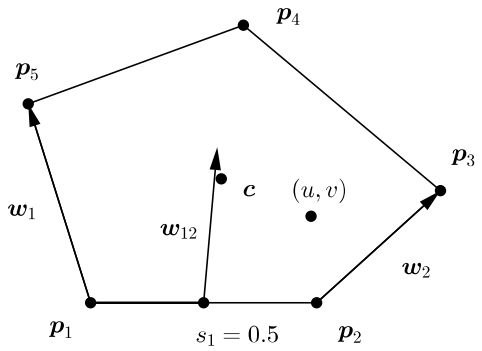


Figure 3.5: Central line sweep parametrization

Given a domain point (u, v), we first use 3.4 to find two expressions for d_1:

$$d_1 = \frac{u - p_2^u s_1}{w_1^u(1-s_1)^2 + 2w_{12}^u(1-s_1)s_1 + w_2^u s_1^2} = \frac{v}{w_1^v(1-s_1) + w_2^v s_1}. \qquad (3.5)$$

Now we can solve this quadratic equation to find s_1 and d_1. The remaining parameters s_i and d_i can be found by first translating the points $\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$, $\boldsymbol{p}_{i+2}$ and $\boldsymbol{c}$ with the transformation $\boldsymbol{f}(\boldsymbol{x}) = R(\boldsymbol{x} - \boldsymbol{p}_i)$, where R is the rotation matrix

$$\begin{bmatrix} n_i^v & -n_i^u \\ n_i^u & n_i^v \end{bmatrix}$$

and $\boldsymbol{n}_i$ is the normal vector of side Γ_i pointing inwards.

It is also possible to force the $d = \frac{1}{2}$ constant parameter line to go through the center point by reparameterizing $(s, d) \to (s, \hat{d})$, where

$$\hat{d} = [(1-s)^2 + 2\frac{1-d_c}{d_c}(1-s)s + s^2]d.$$

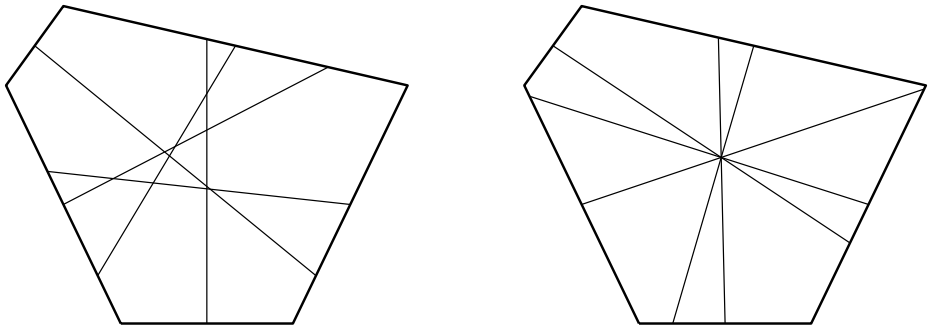
Another way to define the distance parameter $d(u, v)$ is to take the distance from (u, v) to the footpoint of the sweep line[18]:

$$d^f = ||(u, v) - (sp_2^u, 0)||.$$

As before, we can force the central distance parameter line to go through the centre point by reparametrizing d^f:

$$\tilde{d}(u, v) = [(1-s)^2\frac{1}{||\boldsymbol{w}_1||} + s^2\frac{1}{||\boldsymbol{w}_2||} + (1-s)s(\frac{2}{||\boldsymbol{c} - (p_2^u, 0)||} - \frac{1}{||\boldsymbol{w}_1||} - \frac{1}{\boldsymbol{w}_2})].$$

The need to solve the quadratic equation 3.5 makes the central line sweep more time consuming than the radial distance method, but in return, the central line sweep parametrization makes sure that the constant parameter lines $s_i = \frac{1}{2}$ always go through the centre. Figure 3.6 shows the constant parameter lines $s_i = \frac{1}{2}$ for the central line sweep and for the radial distance functions.



(a) (b)

Figure 3.6: Halving lines: (a) radial line sweep; (b) central line sweep.

Barycentric parametrization

Generalized barycentric coordinates, which are introduced in chapter 4, can also be used to parametrize the domain[19]. We can for example use the Wachspress coordinates ϕ_i defined in equation 4.9, which have the partition of unity property $\sum_{i=1}^{n} \phi_i(u,v) = 1$, and the Lagrange property $\phi_i(\boldsymbol{p}_i) = \delta_{i,j}$, where $\delta_{i,j}$ is the Kronecker delta function $\delta_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$. The Wachspress coordinates are also linear on the edges of the domain. The side and distance parameters can be defined as

$$s_i = \frac{\phi_{i+1}}{\phi_i + \phi_{i+1}}, \qquad d_i = 1 - \phi_i - \phi_{i+1}.$$

To prove that these parameters satisfies the requirements for simple parametrization, first assume (u,v) is a point on side k. Since ϕ_k and ϕ_{k+1} are linear along side k, and have the Lagrange property, we get $\phi_k + \phi_{k+1} = 1$, and consequently $s_k = \phi_{k+1}$. It follows that $d_k(u,v) = 0$ and $s_k \in [0,1]$. Since $\phi_{k-1} = \phi_{k+2} = 0$, we get $s_{k-1} = 1$ and $s_{k+1} = 0$. We also get $d_{k-1} =$

$1 - \phi_{k-1} - \phi_k = 1 - 0 - (1 - s_k) = s_k$, and $d_{k+1} = 1 - \phi_{k+1} - \phi_{k+1} = 1 - s_k$. Thus, all the requirements in 3.1 are satisfied.

3.1.2 Constrained parametrizations

The above parametrization methods does not satisfy all the requirements for constrained parametrization. A method that does, called *interconnected parametrization*, was proposed in [21]. First we take an arbitrary function $s_i(u, v)$ that is 0 on side Γ_{i-1}, 1 on side Γ_{i+1}, and takes values in $[0, 1]$ in the rest of the domain. We can for examples use the radial line sweeps or the central line sweeps since these functions satisfies $s_i \in [0, 1]$, $s_{i-1} = 1$ and $s_{i+1} = 0$ for a point on Γ_i. Now we let $\alpha(t) : [0, 1] \to [0, 1]$ be a blending function satisfying $\alpha(0) = 1$, $\alpha(1) = \alpha'(0) = \alpha'(1) = 0$. We can for example use the Hermite basis function $\alpha_0(t)$ defined in 2.4. The distance parameter d_i can be defined as:

$$d_i(u, v) = (1 - s_{i-1}(u, v))\alpha(s_i) + s_{i+1}(u, v)\alpha(1 - s_i).$$

Now we will show that the parametrization satisfies the constraints 3.1 and 3.2. Let (u, v) be a point on side Γ_i. Since $s_{i-1} = 1$ and $s_{i+1} = 0$, we have $d_i = 0$. It also follows from the properties of $\alpha(t)$ that

$$d_{i-1} = (1 - s_{i-2})\alpha(s_{i-1}) + s_i\alpha(1 - s_{i-1}) = s_i$$

and

$$\frac{\partial d_{i-1}}{\partial u} = \frac{\partial}{\partial u}(1 - s_{i-2})\alpha(s_{i-1}) + \frac{\partial}{\partial u}s_i\alpha(1 - s_{i-1}) = \frac{\partial s_i}{\partial u}.$$

By the same reasoning we also have that $\frac{\partial d_{i-1}}{\partial v} = \frac{\partial s_i}{\partial v}$. Similarly

$$d_{i+1} = (1 - s_i)\alpha(s_{i+1}) + s_{i+2}\alpha(1 - s_{i+1}) = 1 - s_i$$

and

$$\frac{\partial d_{i+1}}{\partial u} = \frac{\partial}{\partial u}(1 - s_i)\alpha(s_{i+1}) + \frac{\partial}{\partial u}s_{i+2}\alpha(1 - s_{i+1}) = -\frac{\partial s_i}{\partial u}.$$

Again, the same reasoning works for the v-derivative.

Other constrained parametrization schemes exists, for example biquadratic- and parabolic parametrization[18, 22].

3.2 Blending functions

When we construct the generalized Coons patches we will be adding products $\boldsymbol{S}_i(u, v) = \boldsymbol{R}_i(u, v)\alpha_i(d_i)$ where $\alpha_i(d_i)$ is a blending function, and $\boldsymbol{R}_i(u, v)$

interpolates both the positional data $\boldsymbol{P}_i$ and the tangential data $\boldsymbol{T}_i$ corresponding to the i-th side of the domain. To see what properties the blending functions should have, we evaluate $\boldsymbol{S}_i$ at a point on side i as a function of d_i: $\boldsymbol{S}_i(d_i) = \boldsymbol{R}_i(d_i)\alpha_i(d_i)$. Since $d_i(u,v) = 0$ on side i we see that in order to interpolate the positional data $\boldsymbol{P}_i$, we should have $\alpha_i(0) = 1$. If we differentiate $\boldsymbol{S}_i(d_i)$, we get $\frac{\partial \boldsymbol{S}_i}{\partial d_i} = \boldsymbol{R}_i'(d_i)\alpha_i(d_i) + \boldsymbol{R}_i(d_i)\alpha_i'(d_i)$. With $d_i = 0$, we get $\frac{\partial \boldsymbol{S}_i}{\partial d_i}(0) = \boldsymbol{T}_i\alpha_i(0) + \boldsymbol{P}_i\alpha_i'(0)$, and we see that $\alpha_i'(0)$ must be 0 in order to interpolate the tangential data.

We will make use of three different blending functions: *side blending*, *corner blending* and a *special side blending*.

3.2.1 Corner blending

Let

$$D^n_{i_1,i_2,\ldots,i_m} := \prod_{i \neq i_1,i_2,\ldots,i_m}^{n} d_i^2.$$

The corner blending function defined by

$$\kappa_i(d_1,\ldots,d_n) := \frac{D^n_{i-1,i}}{\sum_{j=1}^n D^n_{j-1,j}} \tag{3.6}$$

is equal to 1 at corner i and decreases to 0 along side i and $i-1$. At the other sides it is equal to 0. An example using perpendicular distance functions on a regular pentagon is shown in figure 3.7a. For example, with $n = 5$ and $i = 1$, we get

$$\kappa_1(d_1,d_2,d_3,d_4,d_5) = \frac{d_2^2 d_3^2 d_4^2}{d_2^2 d_3^2 d_4^2 + d_3^2 d_4^2 d_5^2 + d_4^2 d_5^2 d_1^2 + d_5^2 d_1^2 d_2^2 + d_1^2 d_2^2 d_3^2}.$$

If $d_1 = d_5 = 0$, then $\kappa_1 = 1$; if $d_2 = 0$, $d_3 = 0$ or $d_4 = 0$, then $\kappa_1 = 0$; and if $d_1 = 0$ and $d_5 = d_2$ we get $\kappa_1 = \frac{1}{2}$. Note that from the definition of κ_i it follows that they have the partition of unity property, i.e. $\sum_{i=1}^n \kappa_i = 1$.

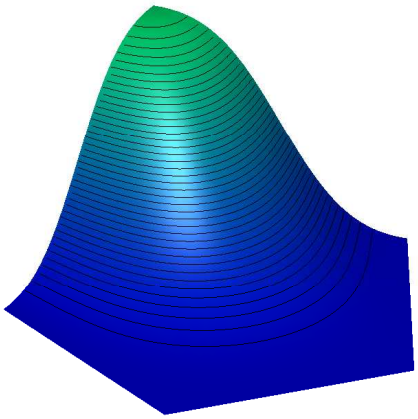
We will now prove the following lemma about the partial derivatives of κ_i at the boundaries of the domain.

Lemma 3.1 *Let κ_i be given by 3.6. Then*

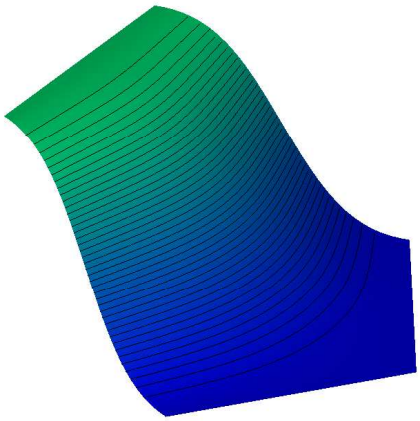
$$\frac{\partial}{\partial d_j}\kappa_i(d_1,\ldots,d_j = 0,\ldots,d_n) = 0, \qquad \forall j, \tag{3.7}$$

and

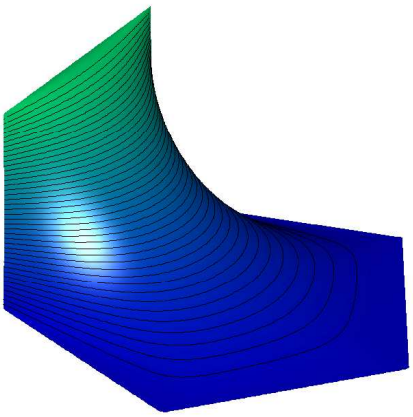
$$\frac{\partial}{\partial d_k}\kappa_i(d_1,\ldots,d_j = 0,\ldots,d_n) = 0, \qquad j \notin \{i-1,i\},\ \forall k. \tag{3.8}$$



(a) Corner blending



(b) Side blending



(c) Special side blending

Figure 3.7: Blending functions

Proof. We will follow the proof in [18]. We start by writing $\frac{\partial}{\partial d_j}\kappa_i(d_1, \ldots, d_j = 0, \ldots, d_n)$ as the limit

$$\lim_{\Delta \to 0} \frac{\kappa_i(d_1, \ldots, d_j = \Delta, \ldots, d_n) - \kappa_i(d_1, \ldots, d_j = 0, \ldots, d_n)}{\Delta}. \tag{3.9}$$

Now assume that $j \notin \{i-1, i\}$. Since the right hand side of the numerator

becomes 0, we get

$$\lim_{\Delta\to 0} \frac{\Delta^2 D^n_{i-1,i,j}}{\Delta(\Delta^2 \sum_{k\notin\{j,j+1\}} D^n_{k-1,k,j} + \sum_{k\in\{j,j+1\}} D^n_{k-1,k})}$$
$$= \lim_{\Delta\to 0} \frac{\Delta D^n_{i-1,i,j}}{\Delta^2 \sum_{k\notin\{j,j+1\}} D^n_{k-1,k,j} + \sum_{k\in\{j,j+1\}} D^n_{k-1,k}}$$
$$= \frac{0}{\sum_{k\in\{j,j+1\}} D^n_{k-1,k}} = 0.$$

Now assume that $j \in \{i-1, i\}$. To shorten the expressions we introduce some new notation. Let $\alpha = D^n_{i-1,i}$, $\beta = \sum_{k\notin\{j,j+1\}} D^n_{k-1,k,j}$ and $\gamma = D^n_{j-1,j} + D^n_{j,j+1}$. We can now write expression 3.9 as

$$\lim_{\Delta\to 0} \frac{1}{\Delta}\left(\frac{\alpha}{\Delta^2\beta+\gamma} - \frac{\alpha}{\gamma}\right) = \lim_{\Delta\to 0} \frac{1}{\Delta}\left(\frac{\alpha\gamma - (\Delta^2\beta+\gamma)\alpha}{\Delta^2\beta\gamma+\gamma^2}\right)$$
$$= \lim_{\Delta\to 0} -\frac{\Delta\alpha\beta}{\Delta^2\beta\gamma+\gamma^2} = 0.$$

Thus we have proved that $\frac{\partial}{\partial d_j}\kappa_i(d_1,\ldots,d_j=0,\ldots,d_n) = 0$, $\forall j$.

For the second part of the lemma, consider the limit

$$\lim_{\Delta\to 0} \frac{\kappa_i(d_1,\ldots,d_j=0,\ldots,d_k+\Delta,\ldots,d_n) - \kappa_i(d_1,\ldots,d_j=0,\ldots,d_k,\ldots,d_n)}{\Delta}.$$

Since $D^n_{i-1,i} = 0$ when $d_j = 0$ and $j \neq i-1, i$, this equals

$$\lim_{\Delta\to 0} \frac{0}{\Delta((d_k+\Delta)^2(\sum_{l\notin\{j,j+1\}} D^n_{l-1,l,j}) + \sum_{l\in\{j,j+1\}} D^n_{l-1,l})}$$
$$+ \frac{0}{\Delta(\sum_{l=1}^n D^n_{l-1,l,j})} = 0.$$

■

As we will see in the next sections, these properties are necessary for some of the patches to interpolate tangential constraints at the boundaries.

As pointed out in [22], it is also possible to use generalized barycentric coordinates with squared terms to construct a corner blending function with the same properties as above. We can for example use the mean value coordinates or the Wachspress coordinates introduced in chapter 4.

3.2.2 Side blending

The side blending functions are defined as

$$\lambda_i(d_1,\ldots,d_n) = \frac{D^n_{i-1,i} + D^n_{i,i+1}}{\sum_{j=1}^n D^n_{j-1,j}}. \tag{3.10}$$

This function is equal to 1 at side i, and decreases to 0 along side $i-1$ and side $i+1$. At the remaining sides it is zero. See figure 3.7b. For example, if $n=4$ and $i=2$ we get

$$\lambda_2(d_1,d_2,d_3,d_4)=\frac{d_3^2d_4^2+d_4^2d_1^2}{d_2^2d_3^2+d_3^2d_4^2+d_1^2d_4^2+d_1^2d_2^2}.$$

If $d_2=0$, then $\lambda_2=1$; if $d_4=0$, then $\lambda_2=0$; and if $d_1=0$ and $d_2=d_4$, then $\lambda_2=\frac{1}{2}$.

Lemma 3.2 *Let λ_i be given by 3.10. Then*

$$\frac{\partial}{\partial d_j}\lambda_i(d_1,\ldots,d_j=0,\ldots,d_n)=0,\qquad \forall j,$$

and

$$\frac{\partial}{\partial d_k}\lambda_i(d_1,\ldots,d_j=0,\ldots,d_n)=0,\qquad j\notin\{i-1,i+1\},\forall k.$$

Proof. Since $\lambda_i(d_1,\ldots,d_n)=\kappa_i(d_1,\ldots,d_n)+\kappa_{i+1}(d_1,\ldots,d_n)$, the first part of the lemma follows from Lemma 3.1. It also follows that

$$\frac{\partial}{\partial d_k}\lambda_i(d_1,\ldots,d_j=0,\ldots,d_n)=0,\qquad j\notin\{i-1,i,i+1\},\forall k.$$

Since $\lambda_i(d_1,\ldots,d_i=0,\ldots,d_n)=1$ and $\frac{\partial}{\partial d_i}\lambda_i(d_1,\ldots,d_i=0,\ldots,d_n)=0$, we get that $\frac{\partial}{\partial d_k}\lambda_i(d_1,\ldots,d_i=0,\ldots,d_n)=0,\ \forall k$. All in all, we get that

$$\frac{\partial}{\partial d_k}\lambda_i(d_1,\ldots,d_j=0,\ldots,d_n)=0,\qquad j\notin\{i-1,i+1\},\forall k.$$

■

3.2.3 Special side blending

The special side blending function

$$\mu_i(d_1,\ldots,d_n)=\frac{D_i^n}{\sum_{j=1}^n D_j^n}$$

is equal to 1 at side i and 0 at the others. Thus there will be a discontinuity at corner i and $i-1$, as can be seen in figure 3.7c. For example we have

$\mu_2(\epsilon, 0, d_3, \ldots, d_n) = 1$, while $\mu_2(0, \epsilon, d_3, \ldots, d_n) = 0$. However, when two adjacent blending functions are added together, the singularity vanishes:

$$\begin{aligned}&\lim_{d_2 \to 0} \mu_1(0, d_2, d_3, \ldots, d_n) + \mu_2(0, d_2, d_3, \ldots, d_n) \\ &= \lim_{d_1 \to 0} \mu_1(d_1, 0, d_3, \ldots, d_n) + \mu_2(d_1, 0, d_3, \ldots, d_n) = 1.\end{aligned}$$

The blending functions μ_i therefore have the partition of unity property for all points in the domain.

As an example, consider the case were $n = 4$ and $i = 2$. We get

$$\mu_2(d_1, \ldots, d_n) = \frac{d_1^2 d_3^2 d_4^2}{d_2^2 d_3^2 d_4^2 + d_1^2 d_3^2 d_4^2 + d_1^2 d_2^2 d_4^2 + d_1^2 d_2^2 d_3^2}.$$

If $d_2 = 0$, then $\mu_2 = 1$. If either $d_1 = 0$, $d_3 = 0$, or $d_4 = 0$, then $\mu_2 = 0$.

We will now prove that all the partial derivatives of μ_i on the boundary except the two corners $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ are zero.

Lemma 3.3 *Assume that (u, v) is a point on $\Gamma_j \setminus \{\boldsymbol{p}_j, \boldsymbol{p}_{j+1}\}$. Then*

$$\frac{\partial}{\partial d_k} \mu_i(d_1, \ldots, d_n) = 0 \quad \textit{for all } i, j.$$

Proof. Since $(u, v) \in \Gamma_j \setminus \{\boldsymbol{p}_j, \boldsymbol{p}_{j+1}\}$, we have that $d_j = 0$, $d_{j-1}, d_{j+1} \neq 0$. We start by proving that $\frac{\partial}{\partial d_j} \mu_i(d_1, \ldots, d_n) = 0$ for all i. This can be done in a similar way as we did in the case of the corner blending function. First, we write $\frac{\partial \mu_i}{\partial d_j}(d_1, \ldots, d_j = 0, \ldots, d_n)$ as the limit

$$\lim_{\Delta \to 0} \frac{\mu_i(d_1, \ldots, d_j = \Delta, \ldots, d_n) - \mu_i(d_1, \ldots, d_j = 0, \ldots, d_n)}{\Delta}. \tag{3.11}$$

Assume that $j \neq i$. The above expression now becomes

$$\lim_{\Delta \to 0} \frac{\Delta^2 D_{i,j}^n}{\Delta(\Delta^2 \sum_{k \neq j} D_{k,j}^n + D_j^n)} = \lim_{\Delta \to 0} \frac{\Delta D_{i,j}}{\Delta^2 \sum_{k \neq j} D_{k,j}^n + D_j^n} = 0.$$

Now assume that $j = i$, and $d_{i-1}, d_{i+1} \neq 0$. Expression 3.11 becomes

$$\begin{aligned}&\lim_{\Delta \to 0} \frac{D_i^n}{\Delta(\Delta^2 \sum_{k \neq i} D_{k,i}^n + D_i^n)} - \frac{D_i^n}{\Delta D_i^n} = \lim_{\Delta \to 0} \frac{D_i^n - \Delta^2 \sum_{k \neq i} D_{k,i}^n - D_i^n}{\Delta(\Delta^2 \sum_{k \neq i} D_{k,i}^n + D_i^n)} \\ &= \lim_{\Delta \to 0} \frac{-\Delta \sum_{k \neq i} D_{k,i}^n}{\Delta^2 \sum_{k \neq i} D_{k,i}^n + D_i^n} = 0.\end{aligned}$$

Since μ_i is constant on each side, the directional derivative of μ_i along side j is also zero. It follows that all the partial derivatives are zero, i.e., $\frac{\partial}{\partial d_k} \mu_i(d_1, \ldots, d_n) = 0$ for all k, i. ■

3.2.4 Alternative formulas

By rewriting the formulas for the blending functions, they can be calculated more computationally efficient. In [23] this is done for the special side blending. This can also be done with the two other blending functions. The corner blending function can be rewritten as follows:

$$\kappa_i(d_1,\dots,d_n) = \frac{D^n_{i-1,i}}{\sum_{j=1}^n D^n_{j-1,j}} = \frac{D^n(d_{i-1}^{-2}d_i^{-2})}{D^n\sum_{j=1}^n d_{j-1}^{-2}d_j^{-2}} = \frac{d_{i-1}^{-2}d_i^{-2}}{\sum_{j=1}^n d_{j-1}^{-2}d_j^{-2}}. \quad (3.12)$$

In the same way, the formulas for the side-blending and special side-blending function can also be written as

$$\lambda_i(d_1,\dots,d_n) = \frac{d_i^{-2}(d_{i-1}^{-2}+d_{i+1}^{-2})}{\sum_{j=1}^n d_{j-1}^{-2}d_j^{-2}} \quad (3.13)$$

and

$$\mu_i(d_1,\dots,d_n) = \frac{d_i^{-2}}{\sum_{j=1}^n d_j^{-2}}. \quad (3.14)$$

While the previous formulas required $\mathcal{O}(n^2)$ operations to be calculated, the new ones only requires $\mathcal{O}(n)$ operations. A disadvantage of the new expressions is that they become singular on some of the sides of the domain. In this case we have to use the original formulas. When evaluating the patches on the sides, this problem can also be solved by just using the value of the given boundary curves $\boldsymbol{P_i}$.

3.3 Direct generalization of Coons patch

To make a n-sided generalization of the Coons patch, we start by reformulating the bicubically blended Coons patch defined in equation 2.5. By collecting the constant vector quantities belonging to each corner in separate terms, and using our new notation, 2.5 can be rewritten as

$$\begin{aligned}\boldsymbol{S}(u,v) = &\sum_{i=1}^4 \begin{bmatrix}\alpha_0(r_i) & \beta_0(r_i)\end{bmatrix}\begin{bmatrix}\boldsymbol{P}_i(s_i)\\ \boldsymbol{T}_i(s_i)\end{bmatrix}\\ &-\sum_{i=1}^4 \begin{bmatrix}\alpha_0(r_i) & \beta_0(r_i)\end{bmatrix}\begin{bmatrix}\boldsymbol{P}_i(0) & \boldsymbol{T}_i^*(0)\\ \boldsymbol{T}_i(0) & \boldsymbol{W}_i(0)\end{bmatrix}\begin{bmatrix}\alpha_0(s_i)\\ \beta_0(s_i)\end{bmatrix},\end{aligned} \quad (3.15)$$

where $\boldsymbol{T}_i^*$ is defined as $\boldsymbol{T}_i^*(r_i) = \boldsymbol{T}_{i-1}(s_{i-1})$. In this case the cross boundary derivative functions $\boldsymbol{T}_i(s_i)$ are taken in the direction perpendicular to side i

pointing inwards, towards the domain such that we have

$$\begin{aligned}
&\boldsymbol{P}_1(s_1) = \boldsymbol{F}(u,0), \quad \boldsymbol{P}_2(s_2) = \boldsymbol{F}(1,v), \quad \boldsymbol{P}_3(s_3) = \boldsymbol{F}(u,1),\\
&\boldsymbol{P}_4(s_4) = \boldsymbol{F}(0,v), \quad \boldsymbol{T}_1(s_1) = \boldsymbol{F}_v(u,0), \quad \boldsymbol{T}_2(s_2) = -\boldsymbol{F}_u(1,v),\\
&\boldsymbol{T}_3(s_3) = -\boldsymbol{F}_v(u,1), \quad \boldsymbol{T}_4(s_4) = \boldsymbol{F}_u(0,v), \quad \boldsymbol{W}_1(0) = \boldsymbol{F}_{uv}(0,0),\\
&\boldsymbol{W}_2(0) = -\boldsymbol{F}_{uv}(1,0), \quad \boldsymbol{W}_3(0) = \boldsymbol{F}_{uv}(1,1), \quad \boldsymbol{W}_4(0) = -\boldsymbol{F}_{uv}(0,1).
\end{aligned}$$

Now we introduce the side-interpolants

$$\boldsymbol{R}_i(s_i, d_i) = \boldsymbol{P}_i(s_i) + d_i \boldsymbol{T}_i(s_i),$$

also called *ribbons*. Each ribbon is a ruled surface interpolating both the positional and tangential data corresponding to one side. These are used to make a ribbon based patch

$$\boldsymbol{S}(u,v) = \sum_{i=1}^{4} \boldsymbol{R}_i(s_i, d_i)\alpha_0(d_i) - \sum_{i=1}^{4} \boldsymbol{Q}_i(s_i, r_i)\alpha(0)(d_i)\alpha_0(s_i), \tag{3.16}$$

where the correction terms $\boldsymbol{Q}_i$ are defined as

$$\boldsymbol{Q}_i(s_i, r_i) = \boldsymbol{P}_i(0) + r_i \boldsymbol{T}_i(0) + s_i \boldsymbol{T}_i^*(0) + r_i s_i \boldsymbol{W}_i(0).$$

In the case the domain is four-sided, a natural choice for the distance parameter is $d_i = r_i$. In this case, the ribbon-based patch 3.16 is not identical to the original Coons Patch 3.15, but it can be shown that it still interpolates the boundary constraints. In fact, the only difference is that the blending functions $\beta_0(u)$ and $\beta_1(u)$ has been replaced by $u\alpha_0(u)$ and $(1-u)\alpha_1(u)$ respectively.

To generalize this to n-sided domains, we just run the indices in the sums up to n, and replace the blending functions by λ_i and κ_i. The *Direct generalization of Coons patches* is then given by

$$\boldsymbol{S}(u,v) = \sum_{i=1}^{n} \boldsymbol{R}_i(s_i, d_i)\lambda_i(d_1, \ldots, d_n) - \sum_{i=1}^{n} \boldsymbol{Q}_i(s_i, r_i)\kappa_i(d_1, \ldots, d_n) \tag{3.17}$$

A patch $\boldsymbol{S}(u,v)$ that interpolates the boundary curves, and has the same partial derivatives on the boundaries as the given ribbons, will be called C^1. If the partial derivatives lie in the same tangent plane as the ribbons, the patch will be called G^1. The given ribbons of a 5-sided patch is shown in figure 3.8.

In the paper by Várady[22] it was not proven that the patch satisfy the boundary constraints, but in a newer paper [21] by the same authors, it was proven that it does if we impose some additional constraints on the parametrizations.

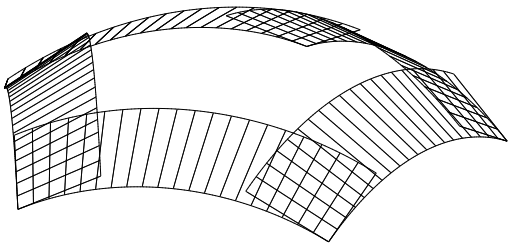


Figure 3.8: Ribbons of a 5-sided patch.

Theorem 3.4 *Assume that for any point $(u, v) \in \Gamma_k$ the parameters d_k and s_k satisfy the constraints in equation 3.1 and 3.2. Then the patch $\boldsymbol{S}(u, v)$ given by 3.17 is C^1.*

Proof. Let $(u, v) \in \Gamma_k$. To simplify the notation we will drop the arguments of the interpolants and the blending functions, i.e. $\boldsymbol{R}_i = \boldsymbol{R}_i(s_i, d_i)$, $\boldsymbol{Q}_i = \boldsymbol{Q}(s_i, r_i)$, $\kappa_i = \kappa_i(d_1, \ldots, d_n)$, $\lambda_i = \lambda_i(d_1, \ldots, d_n)$. To show C^0 continuity we have to show that $\boldsymbol{S}(u, v) = \boldsymbol{P}_k(s_k)$. Since $d_k = 0$ we have that $\kappa_i = 0$ for $i \neq k, k+1$, and $\kappa_k + \kappa_{k+1} = 1$. It also follows that

$$\lambda_i = \begin{cases} 0 & \text{for } i \neq k-1, k, k+1 \\ 1 & \text{for } i = k \\ \kappa_{k+1} & \text{for } i = k+1 \\ \kappa_k & \text{for } i = k-1 \end{cases} . \tag{3.18}$$

We then have that

$$\begin{aligned} \boldsymbol{S}(u, v) = \boldsymbol{R}_k &+ \kappa_k(\boldsymbol{R}_{k-1} - \boldsymbol{Q}_k) \\ &+ \kappa_{k+1}(\boldsymbol{R}_{k+1} - \boldsymbol{Q}_{k+1}) \end{aligned}$$

Using the constraints in 3.1, we get that

$$\begin{aligned} \boldsymbol{R}_{k-1} - \boldsymbol{Q}_k &= \Big(\boldsymbol{P}_{k-1}(s_{k-1}) + d_{k-1}\boldsymbol{T}_{k-1}(s_{k-1})\Big) \\ &- \Big(\boldsymbol{P}_k(0) + r_k\boldsymbol{T}_k(0) + s_k\boldsymbol{T}^*_k(0) + r_k s_k \boldsymbol{W}_k(0)\Big) \\ &= \Big(\boldsymbol{P}_{k-1}(1) + s_k\boldsymbol{T}_{k-1}(1)\Big) \\ &- \Big(\boldsymbol{P}_k(0) + s_k\boldsymbol{T}_{k-1}(1)\Big) = 0, \end{aligned}$$

and

$$\begin{aligned}\boldsymbol{R}_{k+1} - \boldsymbol{Q}_{k+1} &= \Big(\boldsymbol{P}_{k+1}(s_{k+1}) + d_{k+1}\boldsymbol{T}_{k+1}(s_{k+1})\Big) \\ &\quad - \Big(\boldsymbol{P}_{k+1}(0) + r_{k+1}\boldsymbol{T}_{k+1}(0) + s_{k+1}\boldsymbol{T}^*_{k+1}(0) + r_{k+1}s_{k+1}\boldsymbol{W}_{k+1}(0)\Big) \\ &= \Big(\boldsymbol{P}_{k+1}(0) + r_{k+1}\boldsymbol{T}_{k+1}(0)\Big) \\ &\quad - \Big(\boldsymbol{P}_{k+1}(0) + r_{k+1}\boldsymbol{T}_{k+1}(0)\Big) = 0.\end{aligned}$$

This leaves

$$\boldsymbol{S}(u,v) = \boldsymbol{R}_k(s_k, d_k) = \boldsymbol{P}_k(s_k) + d_k\boldsymbol{T}_k(s_k) = \boldsymbol{P}_k(s_k).$$

To prove C^1 continuity, we will show that $\frac{\partial}{\partial u}\boldsymbol{S}(u,v) = \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)\frac{\partial s_k}{\partial u} + \boldsymbol{T}_k(s_k)\frac{\partial d_k}{\partial u}$ and $\frac{\partial}{\partial v}\boldsymbol{S}(u,v) = \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)\frac{\partial s_k}{\partial v} + \boldsymbol{T}_k(s_k)\frac{\partial d_k}{\partial v}$. The partial derivative of $\boldsymbol{S}$ with respect to u can be written as

$$\begin{aligned}&\frac{\partial}{\partial u}\boldsymbol{S}(u,v) = \\ &\sum_{i=1}^{n}\left(\Big(\frac{\partial}{\partial s_i}\boldsymbol{R}_i\frac{\partial s_i}{\partial u} + \frac{\partial}{\partial d_i}\boldsymbol{R}_i\frac{\partial d_i}{\partial u}\Big)\lambda_i + \boldsymbol{R}_i\Big(\frac{\partial \lambda_i}{\partial d_1}\frac{\partial d_1}{\partial u} + \cdots + \frac{\partial \lambda_i}{\partial d_n}\frac{\partial d_n}{\partial u}\Big)\right) \\ &-\sum_{i=1}^{n}\left(\Big(\frac{\partial}{\partial s_i}\boldsymbol{Q}_i\frac{\partial s_i}{\partial u} + \frac{\partial}{\partial r_i}\boldsymbol{Q}_i\frac{\partial r_i}{\partial u}\Big)\kappa_i + \boldsymbol{Q}_i\Big(\frac{\partial \kappa_i}{\partial d_1}\frac{\partial d_1}{\partial u} + \cdots + \frac{\partial \kappa_i}{\partial d_n}\frac{\partial d_n}{\partial u}\Big)\right)\end{aligned}$$

By Lemma 3.1 and 3.2, all the partial derivatives of κ_i and λ_i vanish, except $\frac{\partial \kappa_k}{\partial d_j}$, $\frac{\partial \kappa_{k+1}}{\partial d_j}$, $\frac{\partial \lambda_{k-1}}{\partial d_j}$ and $\frac{\partial \lambda_{k+1}}{\partial d_j}$ for $j \neq k$. Also most of the blending functions vanish, except κ_k, κ_{k+1}, λ_{k-1}, λ_k and λ_{k+1}. Thus, by using equation 3.18 and the constraints 3.2, the above equation can be reduced to

$$\begin{aligned}\frac{\partial}{\partial u}\boldsymbol{S}(u,v) &= \Big(\frac{\partial}{\partial s_{k-1}}\boldsymbol{R}_{k-1} + \frac{\partial}{\partial r_k}\boldsymbol{Q}_k\Big)\frac{\partial s_{k-1}}{\partial u}\kappa_k \\ &\quad + \Big(\frac{\partial}{\partial d_{k-1}}\boldsymbol{R}_{k-1} - \frac{\partial}{\partial s_k}\boldsymbol{Q}_k\Big)\frac{\partial d_{k-1}}{\partial u}\kappa_k \\ &\quad + \Big(\frac{\partial}{\partial s_{k+1}}\boldsymbol{R}_{k+1} - \frac{\partial}{\partial s_{k+1}}\boldsymbol{Q}_{k+1}\Big)\frac{\partial s_{k+1}}{\partial u}\kappa_{k+1} \\ &\quad + \Big(\frac{\partial}{\partial d_{k+1}}\boldsymbol{R}_{k+1} - \frac{\partial}{\partial r_{k+1}}\boldsymbol{Q}_{k+1}\Big)\frac{\partial d_{k+1}}{\partial u}\kappa_{k+1} \\ &\quad + \frac{\partial}{\partial s_k}\boldsymbol{R}_k\frac{\partial s_k}{\partial u} + \frac{\partial}{\partial d_k}\boldsymbol{R}_k\frac{\partial d_k}{\partial u} \\ &\quad + \Big(\boldsymbol{R}_{k-1} - \boldsymbol{Q}_k\Big)\frac{\partial \kappa_k}{\partial u} + \Big(\boldsymbol{R}_{k+1} - \boldsymbol{Q}_{k+1}\Big)\frac{\partial \kappa_{k+1}}{\partial u}.\end{aligned}$$

Let's investigate this expression part by part. Using the constraints in 3.1, we get

$$\begin{aligned}\frac{\partial}{\partial s_{k-1}}\boldsymbol{R}_{k-1}+\frac{\partial}{\partial r_k}\boldsymbol{Q}_k &= \Big(\frac{\partial}{\partial s_{k-1}}\boldsymbol{P}_{k-1}(s_{k-1})+d_{k-1}\frac{\partial}{\partial s_{k-1}}\boldsymbol{T}_{k-1}(s_{k-1})\Big)\\ &+\Big(\boldsymbol{T}_k(0)+s_k\boldsymbol{W}_k(0)\Big)\\ &=\Big(\frac{\partial}{\partial s_{k-1}}\boldsymbol{P}_{k-1}(1)+s_k\frac{\partial}{\partial s_{k-1}}\boldsymbol{T}_{k-1}(1)\Big)\\ &+\Big(\boldsymbol{T}_k(0)+s_k\boldsymbol{W}_k(0)\Big)\\ &=\Big(-\boldsymbol{T}_k(0)-s_k\boldsymbol{W}_k(0)\Big)+\Big(\boldsymbol{T}_k(0)+s_k\boldsymbol{W}_k(0)\Big)=0,\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial d_{k-1}}\boldsymbol{R}_{k-1}-\frac{\partial}{\partial s_k}\boldsymbol{Q}_k &= \boldsymbol{T}_{k-1}(s_{k-1})-\Big(\boldsymbol{T}^*_k(0)+r_k\boldsymbol{W}_k(0)\Big)\\ &=\boldsymbol{T}_{k-1}(1)-\boldsymbol{T}_{k-1}(1)=0,\end{aligned}$$

$$\begin{aligned}\frac{\partial\boldsymbol{R}_{k+1}}{\partial s_{k+1}}-\frac{\partial}{\partial s_{k+1}}\boldsymbol{Q}_{k+1} &= \Big(\frac{\partial}{\partial s_{k+1}}\boldsymbol{P}_{k+1}(s_{k+1})+d_{k+1}\frac{\partial}{\partial s_{k+1}}\boldsymbol{T}_{i+1}(s_{k+1})\Big)\\ &-\Big(\boldsymbol{T}^*_{k+1}(0)+r_{k+1}\boldsymbol{W}_{k+1}(0)\Big)\\ &=\Big(\frac{\partial}{\partial s_{k+1}}\boldsymbol{P}_{k+1}(0)+r_{k+1}\frac{\partial}{\partial s_{k+1}}\boldsymbol{T}_{i+1}(0)\Big)\\ &-\Big(\boldsymbol{T}_k(1)+r_{k+1}\boldsymbol{W}_{k+1}(0)\Big)\\ &=\Big(\boldsymbol{T}_k(1)+r_{k+1}\boldsymbol{W}_{k+1}(0)\Big)\\ &-\Big(\boldsymbol{T}_k(1)+r_{k+1}\boldsymbol{W}_{k+1}(0)\Big)=0\end{aligned}$$

and

$$\begin{aligned}\frac{\partial}{\partial d_{k+1}}\boldsymbol{R}_{k+1}-\frac{\partial}{\partial r_{k+1}}\boldsymbol{Q}_{k+1} &= \boldsymbol{T}_{k+1}(s_{k+1})-\Big(\boldsymbol{T}_{k+1}(0)+s_{k+1}\boldsymbol{W}_{k+1}(0)\Big)\\ &=\boldsymbol{T}_{k+1}(0)-\boldsymbol{T}_{k+1}(0)=0.\end{aligned}$$

Since we have already showed that

$$\boldsymbol{R}_{k-1}-\boldsymbol{Q}_k=0,\qquad \boldsymbol{R}_{k+1}-\boldsymbol{Q}_{k+1}=0,$$

we are left with

$$\begin{aligned}\frac{\partial}{\partial u}\boldsymbol{S}(u,v) &= \frac{\partial}{\partial s_k}\boldsymbol{R}_k\frac{\partial s_k}{\partial u} + \frac{\partial}{\partial d_k}\boldsymbol{R}_k\frac{\partial d_k}{\partial u} \\ &= \Big(\frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k) + d_k\frac{\partial}{\partial s_k}\boldsymbol{T}_k(s_k)\Big)\frac{\partial s_k}{\partial u} + \boldsymbol{T}_k(s_k)\frac{\partial d_k}{\partial u} \\ &= \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)\frac{\partial s_k}{\partial u} + \boldsymbol{T}_k(s_k)\frac{\partial d_k}{\partial u}.\end{aligned}$$

In a completely analogous way it can be shown that

$$\frac{\partial}{\partial v}\boldsymbol{S}(u,v) = \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)\frac{\partial s_k}{\partial v} + \boldsymbol{T}_k(s_k)\frac{\partial d_k}{\partial v}.$$

■

3.4 Corner-Based Patch

This method is an extension of a pentagonal surface patch developed by Charrot and Gregory[3], and is therefore also sometimes called Gregory patch[21]. The surface patch is now constructed as a convex combination of *two-sided corner interpolants* which are made up by two ribbons minus a single corner correction term:

$$\begin{aligned}\boldsymbol{C}_i(s_i,r_i) &= \boldsymbol{R}_{i-1}(s_{i-1},s_i) + \boldsymbol{R}_i(s_i,r_i) - \boldsymbol{Q}_i(s_i,r_i) \\ &= \boldsymbol{P}_i^*(r_i) + \boldsymbol{P}_i(s_i) + s_i\boldsymbol{T}_i^*(r_i) + r_i\boldsymbol{T}_i(s_i) - \boldsymbol{Q}_i(s_i,r_i)\end{aligned}, \tag{3.19}$$

where we define $\boldsymbol{P}_i^*$ in the same way as $\boldsymbol{T}_i^*$, i.e. $\boldsymbol{P}_i^*(r_i) = \boldsymbol{P}_{i-1}(s_{i-1})$.[2] The corner-based patch is then defined as

$$\boldsymbol{S}(u,v) = \sum_{i=1}^{n}\boldsymbol{C}_i(s_i,r_i)\kappa_i(d_1,\ldots,d_n) \tag{3.20}$$

Theorem 3.5 *Assume that for any point $(u,v) \in \Gamma_k$ the parameters d_k and s_k satisfy the constraints in equation 3.1. Then the patch $\boldsymbol{S}(u,v)$ given by 3.20 is G^1 continuous.*

Proof. To simplify the notation, we will again drop the arguments of the corner blending functions and the corner interpolants, i.e. $\boldsymbol{C}_i = \boldsymbol{C}_i(s_i,r_i)$,

[2] In the paper by Várady[22], there is an error in the definition of $\boldsymbol{C}_i$. Instead of $\boldsymbol{P}_i^*(r_i)$ and $\boldsymbol{T}_i^*(r_i)$, they write $\boldsymbol{P}_{i-1}^*(r_i)$ and $\boldsymbol{T}_{i-1}^*(r_i)$ respectively, which doesn't lead to the desired surface with the current definition of $\boldsymbol{T}_i^*$ and $\boldsymbol{P}_i^*$.

$\boldsymbol{Q}_i = \boldsymbol{Q}(s_i, r_i)$, $\kappa_i = \kappa_i(d_1, \ldots, d_n)$. Let $(u, v) \in \Gamma_k$. First, we check that $\boldsymbol{S}(u, v) = \boldsymbol{P}_k(s_k)$. We have that

$$\boldsymbol{C}_k = \boldsymbol{P}_k^*(0) + \boldsymbol{P}_k(s_k) + s_k\boldsymbol{T}_k^*(0) - (\boldsymbol{P}_k(0) + s_k\boldsymbol{T}_k^*(0)) = \boldsymbol{P}_k(s_k)$$

and

$$\begin{aligned}\boldsymbol{C}_{k+1} &= \boldsymbol{P}_{k+1}^*(r_{k+1}) + \boldsymbol{P}_{k+1}(0) + r_{k+1}\boldsymbol{T}_{k+1}(0)\\ &\quad - (\boldsymbol{P}_{k+1}(0) + r_{k+1}\boldsymbol{T}_{k+1}(0))\\ &= \boldsymbol{P}_{k+1}^*(r_{k+1}) = \boldsymbol{P}_k(s_k).\end{aligned}$$

Since $\kappa_i = 0$ for $i \neq k, k+1$, and $\kappa_k + \kappa_{k+1} = 1$, it follows that

$$\boldsymbol{S}(u, v) = \sum_{i=1}^{n} \boldsymbol{C}_i(s_i, r_i)\kappa_i = \boldsymbol{P}_k(s_k).$$

To show G^1 continuity, we will show that $\frac{\partial}{\partial u}\boldsymbol{S}(u, v)$ and $\frac{\partial}{\partial v}\boldsymbol{S}(u, v)$ are linear combinations of $\frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)$ and $\boldsymbol{T}_k(s_k)$. Since, by Lemma 3.1, $\frac{\partial \kappa_i}{\partial d_j} = 0$ for all j and $i \neq k, k+1$, and $\kappa_i = 0$ for $i \neq k, k+1$, we get that

$$\begin{aligned}\frac{\partial}{\partial u}\boldsymbol{S}(u, v) &= \sum_{i=1}^{n}\Big(\frac{\partial}{\partial s_i}\boldsymbol{C}_i\frac{\partial s_i}{\partial u} + \frac{\partial}{\partial r_i}\boldsymbol{C}_i\frac{\partial r_i}{\partial u}\Big)\kappa_i + \boldsymbol{C}_i\Big(\frac{\partial \kappa_i}{\partial d_1}\frac{\partial d_1}{\partial u}, \ldots, \frac{\partial \kappa_i}{\partial d_n}\frac{\partial d_n}{\partial u}\Big)\\ &= \Big(\frac{\partial}{\partial s_k}\boldsymbol{C}_k\frac{\partial s_k}{\partial u} + \frac{\partial}{\partial r_k}\boldsymbol{C}_k\frac{\partial r_k}{\partial u}\Big)\kappa_k\\ &+ \Big(\frac{\partial}{\partial s_{k+1}}\boldsymbol{C}_{k+1}\frac{\partial s_{k+1}}{\partial u} + \frac{\partial}{\partial r_{k+1}}\boldsymbol{C}_{k+1}\frac{\partial r_{k+1}}{\partial u}\Big)\kappa_{k+1}\\ &+ \boldsymbol{C}_k\frac{\partial \kappa_k}{\partial u} + \boldsymbol{C}_{k+1}\frac{\partial \kappa_{k+1}}{\partial u}.\end{aligned}$$

Let's investigate this expression part by part. Using the constraints for simple parametrization in 3.1, we get

$$\begin{aligned}\frac{\partial}{\partial s_k}\boldsymbol{C}_k &= \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k) + \boldsymbol{T}_{k-1}(s_{k-1}) + r_k\frac{\partial}{\partial s_k}\boldsymbol{T}_k(s_k) - \frac{\partial}{\partial s_k}\boldsymbol{Q}_k\\ &= \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k) + \boldsymbol{T}_{k-1}(1) - \boldsymbol{T}_{k-1}(1) = \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k),\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial r_k}\boldsymbol{C}_k &= -\frac{\partial}{\partial s_{k-1}}\boldsymbol{P}_{k-1}(s_{k-1}) - s_k\frac{\partial}{\partial s_{k-1}}\boldsymbol{T}_{k-1}(s_{k-1}) + \boldsymbol{T}_k(s_k) - \frac{\partial}{\partial r_k}\boldsymbol{Q}_k\\ &= \boldsymbol{T}_k(0) + s_k\boldsymbol{W}_k(0) + \boldsymbol{T}_k(s_k) - (\boldsymbol{T}_k(0) + s_k\boldsymbol{W}_k(0)) = \boldsymbol{T}_k(s_k),\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial s_{k+1}}\boldsymbol{C}_{k+1} &= \frac{\partial}{\partial s_{k+1}}\boldsymbol{P}_{k+1}(s_{k+1}) + \boldsymbol{T}_k(s_k) + r_{k+1}\frac{\partial}{\partial s_{k+1}}\boldsymbol{T}_{k+1}(s_{k+1}) \\ &\quad - \frac{\partial}{\partial s_{k+1}}\boldsymbol{Q}_{k+1} \\ &= \boldsymbol{T}_k(1) + \boldsymbol{T}_k(s_k) + r_{k+1}\boldsymbol{W}_{k+1}(0) - (\boldsymbol{T}_{k+1}(1) + r_{k+1}\boldsymbol{W}_{k+1}(0)) \\ &= \boldsymbol{T}_k(s_k),\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial r_{k+1}}\boldsymbol{C}_{k+1} &= -\frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k) - s_{k+1}\frac{\partial}{\partial s_k}\boldsymbol{T}_k(s_k) + \boldsymbol{T}_{k+1}(s_{k+1}) - \frac{\partial}{\partial r_{k+1}}\boldsymbol{Q}_{k+1} \\ &= -\frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k) + \boldsymbol{T}_{k+1}(0) - \boldsymbol{T}_{k+1}(0) = -\frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k).\end{aligned}$$

We showed above that

$$\boldsymbol{C}_k = \boldsymbol{C}_{k+1} = \boldsymbol{P}_k(s_k).$$

Since $\frac{\partial \kappa_k}{\partial d_k} = \frac{\partial \kappa_{k+1}}{\partial d_k} = 0$, and $\kappa_k + \kappa_{k+1} = 1$ for all points on Γ_k, we have $\frac{\partial \kappa_k}{\partial u} + \frac{\partial \kappa_{k+1}}{\partial u} = 0$. It follows that

$$\boldsymbol{C}_k\frac{\partial \kappa_k}{\partial u} + \boldsymbol{C}_{k+1}\frac{\partial \kappa_{k+1}}{\partial u} = \boldsymbol{P}_k(s_k)\Big(\frac{\partial \kappa_k}{\partial u} + \frac{\partial \kappa_{k+1}}{\partial u}\Big) = 0.$$

We are then left with

$$\frac{\partial}{\partial u}\boldsymbol{S}(u,v) = \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)\frac{\partial s_k}{\partial u}(\kappa_k + \kappa_{k+1}) + \boldsymbol{T}_k(s_k)\Big(\kappa_k\frac{\partial r_k}{\partial u} + \kappa_{k+1}\frac{\partial s_{k+1}}{\partial u}\Big) \tag{3.21}$$

$$= \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)\frac{\partial s_k}{\partial u} + \boldsymbol{T}_k(s_k)\Big(\kappa_k\frac{\partial r_k}{\partial u} + \kappa_{k+1}\frac{\partial s_{k+1}}{\partial u}\Big). \tag{3.22}$$

In the same way a similar result can be shown for $\frac{\partial}{\partial v}\boldsymbol{S}(u,v)$. Thus, both $\frac{\partial}{\partial u}\boldsymbol{S}(u,v)$ and $\frac{\partial}{\partial v}\boldsymbol{S}(u,v)$ lie in the tangent plane spanned by $\frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)$ and $\boldsymbol{T}_k(s_k)$. ■

Note that if $\frac{\partial r_k}{\partial u} = \frac{\partial s_{k+1}}{\partial u} = \frac{\partial d_k}{\partial u}$ and $\frac{\partial r_k}{\partial v} = \frac{\partial s_{k+1}}{\partial v} = \frac{\partial d_k}{\partial v}$ whenever $d_k = 0$, it follows from equation 3.21 that we get C^1 continuity.

Also note that in the proof above, we haven't used the last requirements for simple parametrization, $d_{k-1} = s_k$ and $d_{k+1} = 1 - s_k$. Thus, we can also use the radial line sweep parametrization method.

Even though the direct generalization and the corner based patch seems quite different, a closer look reveals that the only difference is the parametrization of the ribbons[20]. To see that this is the case, note that the direct

generalization of Coons patch can be rewritten as

$$\boldsymbol{S}_{\mathrm{GC}}(u,v) = \sum_{i=1}^{n} \boldsymbol{R}_i(s_i,d_i)\kappa_i + \boldsymbol{R}_i(s_i,d_i)\kappa_{i+1} - \sum_{i=1}^{n} \boldsymbol{Q}_i(s_i,r_i)\kappa_i,$$

while the corner-based patch can be written as

$$\begin{aligned}\boldsymbol{S}_{\mathrm{CB}}(u,v) &= \sum_{i=1}^{n} [\boldsymbol{R}_{i-1}(s_{i-1},s_i) + \boldsymbol{R}_i(s_i,r_i) - \boldsymbol{Q}_i(s_i,r_i)]\kappa_i \\ &= \sum_{i=1}^{n} \boldsymbol{R}_i(s_i,r_i)\kappa_i + \boldsymbol{R}_i(s_i,s_{i+1})\kappa_{i+1} - \sum_{i=1}^{n} \boldsymbol{Q}_i(s_i,r_i)\kappa_i.\end{aligned}$$

3.5 Side-Based Patch

A third method, which combines the ribbons $\boldsymbol{R}_i$ without use of correction terms, was introduced by Kato[14]. The correction terms is avoided by using the special-side blending functions $\mu_1,\ldots,\mu_n$, where each μ_i is equal to 1 on side i, and 0 on the remaining sides. The surface is defined by

$$\boldsymbol{S}(u,v) = \sum_{i=1}^{n} \boldsymbol{R}_i(s_i,d_i)\mu_i(d_1,\ldots,d_n). \tag{3.23}$$

Theorem 3.6 *Assume that the parametrizations d_k and s_k satisfy the constraints 3.1 for any point $(u,v) \in \Gamma_k$ for all k. Then the patch $S(u,v)$ given by 3.23 is C^1.*

Proof. Let $(u,v) \in \Gamma_k \setminus \{\boldsymbol{p}_k, \boldsymbol{p}_{k+1}\}$. As before, we will drop the arguments of the side-interpolants and the blending functions, i.e., $\boldsymbol{R}_i = \boldsymbol{R}_i(s_i,d_i)$, $\mu_i = \mu_i(d_1,\ldots,d_n)$. First we check that $\boldsymbol{S}(u,v) = \boldsymbol{P}_k(s_k)$. Since $\mu_i = 0$ for $i \neq k$, and $\mu_k = 1$, we have that

$$\boldsymbol{S}(u,v) = \boldsymbol{R}_k(s_k,d_k) = \boldsymbol{P}_k(s_k) + d_k\boldsymbol{T}_k(s_k) = \boldsymbol{P}_k(s_k).$$

Now we will show that the patch interpolates the boundary derivatives. We have that

$$\frac{\partial}{\partial u}\boldsymbol{S}(u,v) = \sum_{i=1}^{n} \Big(\frac{\partial}{\partial s_i}\boldsymbol{R}_i\frac{\partial s_i}{\partial u} + \frac{\partial}{\partial d_i}\boldsymbol{R}_i\frac{\partial d_i}{\partial u}\Big)\mu_i + \boldsymbol{R}_i\frac{\partial \mu_i}{\partial u}.$$

It follows from Lemma 3.3 that $\frac{\partial \mu_i}{\partial u} = 0$, and since $\mu_i = 0$ for $i \neq k$, and $\mu_k = 1$, we get

$$\begin{aligned}\frac{\partial}{\partial u}\boldsymbol{S}(u,v) &= \frac{\partial}{\partial s_k}\boldsymbol{R}_k\frac{\partial s_k}{\partial u} + \frac{\partial}{\partial d_k}\boldsymbol{R}_k\frac{\partial d_k}{\partial u} \\ &= \frac{\partial}{\partial s_k}\boldsymbol{P}_k(s_k)\frac{\partial s_k}{\partial u} + \boldsymbol{T}_k(s_k)\frac{\partial d_k}{\partial u}.\end{aligned}$$

In the same way, this can also be shown for $\frac{\partial}{\partial v}\boldsymbol{S}(u,v)$, and we thus have C^1 continuity. ■

Note that the side based interpolation scheme does not require compatible twist vectors since the rational blending functions will average incompatible twists.

3.6 Domain Construction

The shape of the domain polygon can have huge influence on the shape of the patch[22]. To get the best result, we want evenly spaced constant parameter lines in the domain to map to evenly spaced curves on the 3D surface. To achieve this, the domain polygon should mimic the shape of the 3D boundary curves in the best possible way.

Let L_i denote the arc length of $\boldsymbol{P}_i$, and let ϕ_i be the angle between $\boldsymbol{P}_i'(0)$ and $\boldsymbol{P}_{i-1}'(1)$. In order to map the 3D boundary curves on to the (u,v) plane with minimal distortion, we want to determine the corner points $\boldsymbol{p}_i = (u_i, v_i)$ that minimize the function

$$\sum (l_i - c_{length} L_i)^2 + \sum (\alpha_i - c_{angle}\phi_i)^2, \tag{3.24}$$

where l_i is the length of side i of the domain polygon, α_i is the angle between side $i-1$ and i, and c_{length} and c_{angle} are some chosen constants. In [22], three different methods to construct a convex domain polygon, given a loop of 3D-curves, are presented.

(i) In the first and simplest method, the corner points $\boldsymbol{p}_i$ are placed on the perimeter of the unit circle, such that the central angles between two consecutive points are proportional to L_i. The first point $\boldsymbol{p}_1$ is placed at $(1,0)$, while for $i \geq 2$, $\boldsymbol{p}_i$ is placed with a central angle

$$\beta_i = 2\pi \frac{\sum_{k=1}^{i-1} L_k}{\sum_{k=1}^{n} L_k}$$

with the positive u-axis as shown in figure 3.9.

(ii) The second method places the corner points on a circle, such that the side lengths l_i are equal to L_i. We start with a circle with a sufficiently large radius R, and then place the corner points on the circle such that the distance from $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$ is equal to L_i. We start with the points corresponding to the longest arc, which we will call L_1. Let $\boldsymbol{p}_1 = (R, 0)$. Then we start decreasing the radius such that the points slides towards the other end. See figure 3.10a. Our goal is to get the last point, denoted by $\boldsymbol{p}^*$, to be equal to $\boldsymbol{p}_1$. However, if the diameter $2R$ becomes equal to L_1 before the loop is closed, we have to

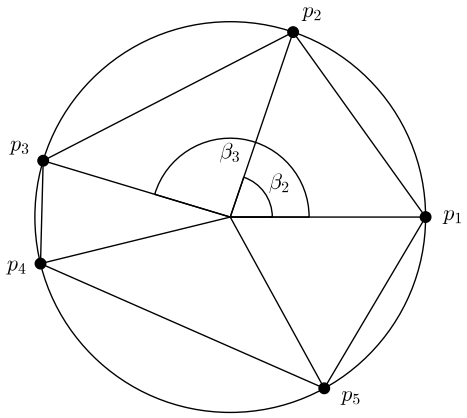


Figure 3.9: Polygon domain (i).

start increasing the radius again. See figure 3.10b. In this case the center of the circle will be outside of the polygon. If $L_1 \leq 2R$ when the loop is closed, the chords will satisfy $\sum_{i=1}^{n} \arcsin \frac{L_i}{2R} = \pi$. Otherwise, if $L_1 > 2R$ when the loop is closed, the chords will satisfy $\sum_{i=2}^{n} \arcsin \frac{L_i}{2R} = \arcsin \frac{L_1}{2R}$.[3] For the existence of such a polygon, it's sufficient that $L_1 < \sum_{i=2}^{n} L_i$.

Using this method, the first sum in 3.24 will always be equal to zero with $c_{\text{length}} = 1$. The second sum, on the other hand, can become large. This method is also difficult to implement compared to the other methods.

(iii) While the previous methods only consider the length of the boundary curves to minimize equation 3.24, we now also consider the angles ϕ_i. We start by normalizing these angles such that the sum of the angles is equal to the sum of the interior angles of a n-sided polygon, i.e. $\alpha_i = c_{angle}\phi_i$, where $c_{angle} = \frac{(n-2)\pi}{\sum_{j=1}^{n} \phi_j}$. Then, the polygon is constructed by first placing the chord lengths L_i with the angles α_i between them, as shown in figure 3.11a. This will most likely yield an open polyline. We will denote the end point of the last chord by $\boldsymbol{p}^*$, and the vector going from $\boldsymbol{p}^*$ to $\boldsymbol{p}_1$ by $\boldsymbol{e}$. Then, we move each point $\boldsymbol{p}_i$ by adding the vector $\frac{i-1}{n}\boldsymbol{e}$. The point $\boldsymbol{p}^*$ is translated to $\boldsymbol{p}^* + \boldsymbol{e} = \boldsymbol{p}_1$. See figure 3.11b. We end up with a closed polygon where both the angles and the side lengths are somewhat distorted compared to the polyline in the previous step.

[3] In the paper by Várady [22] there's an error in these equations. Instead of arcsin, they use arccos.

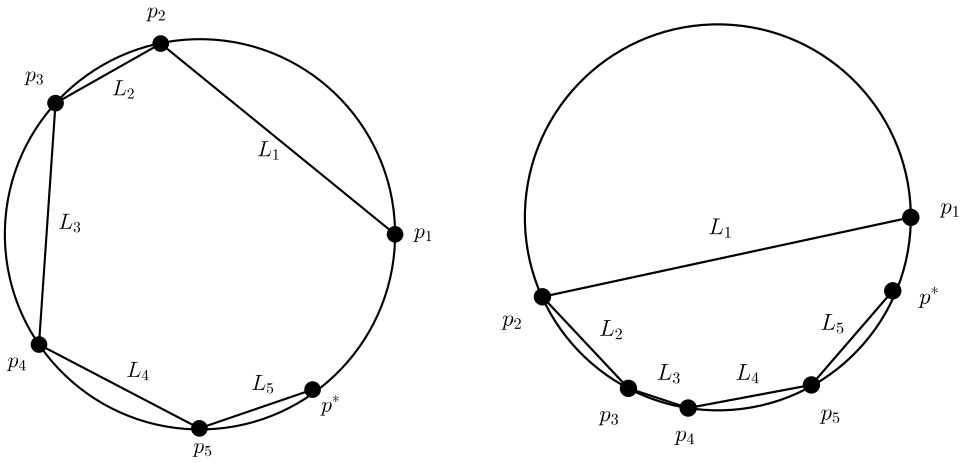


Figure 3.10: Polygon domain (ii).

3.7 Implementation

The direct generalization, the corner based and the side-based method were implemented in MATLAB®. Given a set of domain vertices $\boldsymbol{p}_i$, boundary curves $\boldsymbol{P}_i$ and cross boundary derivatives $\boldsymbol{T}_i$, we start by triangulate the domain using Delaunay triangulation. For each node in the triangular mesh, we first calculate the side- and distance parameters s_i and d_i using an appropriate parametrization method. Then we can compute the blending functions and evaluate the surface. When this is done for every node in the triangular mesh, we can plot the surface using the obtained surface points and the triangulation of the domain.

In the side-based method we need to take some extra care. Since the special side blending functions are singular at two corners, we have to check if each domain point $\boldsymbol{x}$ is at a corner. If $\boldsymbol{x}$ is at corner i, we can then substitute in the value of $\boldsymbol{P}_i(0)$ as our surface point.

3.8 Examples and discussion

In this section we will show some numerical examples of the three patches introduced above. We will use the abbreviation GC for the direct generalization of Coons patch, CB for the corner based patch, and SB for the side based patch. Since the different interpolation schemes depend on both the

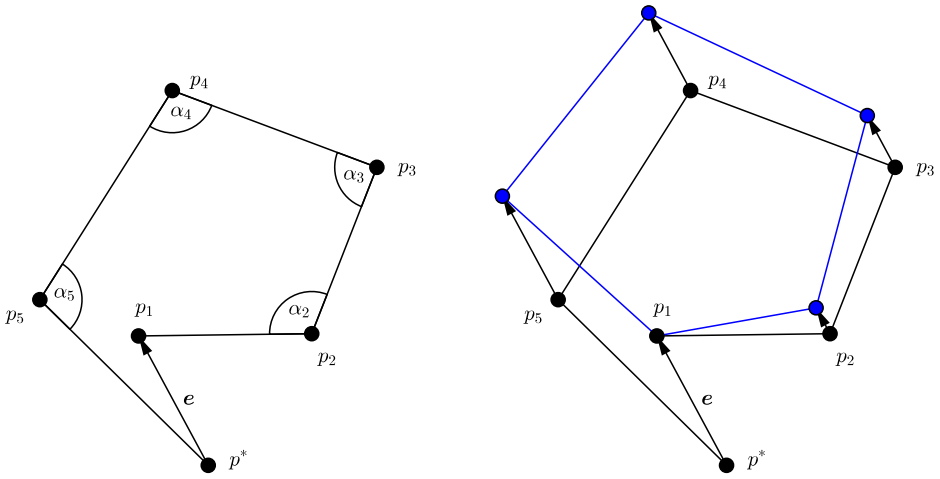


Figure 3.11: Polygon domain (iii).

choice of domain polygon and parametrization scheme, the number of possible patch constructions is to large to compare them all. In the following examples we have used the central line sweep parametrization for the CB and SB patches, and interconnected parametrization for the GC patch if not stated otherwise.

Example 3.7 *An interesting property to investigate is how well the surfaces reproduce regular shapes, such as parts of a sphere. In this example, the patches were constructed from three boundary curves and cross-tangent functions obtained from an octant of a sphere with radius* 1*. Figure 3.12 shows the original octant and the three different approximations. It's hard to visually distinguish them from each other, but numerical comparisons of the radius values shows that side based patch gives the best approximation. Table 3.1 shows the minimum, maximum, average and standard deviation of the radius values for all three schemes.*

Example 3.8 *An example with a 5-sided patch used to fill a hole formed by rectangular patches is shown in figure 3.13. The boundary data was taken straight from the neighbouring patches which are Bézier surfaces of degree* $(2, 1)$ *and* $(1, 1)$*. The globally surfaces are then* G^1 *continuous. We chose to let the domain polygon* Ω *to be a regular pentagon with vertices on the unit circle. The choice of magnitude of the cross boundary derivative functions* $\boldsymbol{T}_i$

	Average	Min	Max	Std. deviation
Direct generalization(GC)	1.0184	1.0000	1.0556	0.01670
Corner-based interpolants(CB)	1.0155	1.0000	1.0555	0.01587
Side-based interpolants(SB)	1.0055	1.0000	1.0106	0.00389

Table 3.1: Comparison of radius values.

give us an extra degree of freedom which can be used to modify the surface. In this example, the SB patch seems to be more sensitive to change in the magnitude of the cross boundary derivative functions than the CB and GC patch, which were little affected. The reason for this is that $\boldsymbol{T}_i$ *is constant along each side, such that the correction surfaces used in the GC and the CB patch somewhat cancel out the cross boundary derivatives. In this example we scaled the vector valued functions* $\boldsymbol{T}_i$ *such that they have length* 1*. As can be seen, all three methods performs well in this example, and it's hard to favour one method over the others.*

Example 3.9 *A common problem solved by n-sided patches is the corner molding problem like the one from section 3. In this example we have used the same boundary data as we did in the example from section 3 for the triangular Coons patch. Figure 3.14 shows the result. As can be seen, all three methods give good results.*

Example 3.10 *The computational efficiency of the three patches is mainly determined by evaluation of ribbons and correction surfaces, and computation of blending functions and local side and distance parameters. Among these, the evaluation of the ribbons is the most time consuming part. In table 3.2 the average evaluation time per surface point are shown for different values of n. The methods were implemented in MATLAB, and run on a 3.50GHz pro-*

	n=3	n=4	n=5	n=6	n=7	n=8
Direct Generalization(GC)	0.17	0.21	0.28	0.32	0.37	0.41
Corner-based interpolants(CB)	0.21	0.22	0.29	0.36	0.41	0.46
Side-based interpolants(SB)	0.16	0.16	0.21	0.27	0.31	0.34

Table 3.2: Average evaluation time per surface point in milliseconds on a 3.50GHz processor.

cessor. The averages were calculated from a surface with ca 1000 points. The GC patch was evaluated by using the interconnected parametrization, while the others used central line sweep. We used regular domain polygons with vertices on the unit circle, and the boundary data was obtained from a sphere.

We see that the SB patch is the fastest, closely followed by the GC patch. The main reason for this is that the SB patch only requires n evaluations of ribbons and doesn't need any correction terms. The GC patch requires n evaluations of both ribbons and correction terms, while the CB patch requires evaluations of $2n$ *ribbons and n correction terms, and is therefore the slowest method, even though the GC patch requires a more complex parametrization method.*

Another desirable property of a n-sided interpolant, is that if we are editing the patch by inserting a new small edge, the parts of the patch not directly adjacent to the new side should remain unchanged. Experiments in [21] showed that in respect to this property the GC and CB patch are reasonable good, while the SB patch is somewhat weaker.

In conclusion, it's hard to say which method is best. In most of our examples it's hard to visually distinguish the different patches. In some cases the SB patch seems to work best, while in other cases the CB patch is better. The GC patch combines in some sense the two other methods, and will therefore merge the shape features of the two other patches. Unfortunately, the GC patch also requires a more complex parametrization, and it's a bit more time consuming to compute than the SB patch.

In [21] another ribbon based patch called *composite ribbon patch* was proposed. Instead of using linear ribbons, the patch is made up by a combination of doubly curved ribbons. The authors compared the method with the direct generalization of Coons patch, and showed that if the linear ribbons strongly deviates from the n-sided surface due to high curvatures, using curved ribbons will produce more predictable interior shapes. It was also shown that the method has roughly the same computational complexity as the GC patch.

As the patches in this chapter are defined as a blend of the boundaries, we have little control over the centre of the surface. As mention, one way to affect the "fullness" of the patch is to adjust the width of the ribbons. This can be done by introducing a scaling factor $w_i(s_i)$. The ribbons can then be redefined as

$$\boldsymbol{R}_i(s_i, d_i) = \boldsymbol{P}_i(s_i) + d_i w_i(s_i) \boldsymbol{T}_i(s_i).$$

If the ribbons are too wide, we can get sudden curvature changes in the middle of the patch. If the ribbons are too narrow, their influence will be too weak, and can create sudden curvature changes near the boundaries.

Another method used to give further control over the interior of the patch was proposed in [23]. If we in addition to the ribbons are given an auxiliary point $\boldsymbol{P}_i$ together with a normal vector $\boldsymbol{N}_i$, we can define a circular ribbon

as

$$\boldsymbol{R}_i(s_i, d_i) = \boldsymbol{P}_i + d_i w_i \boldsymbol{T}_i(s_i),$$

where $\boldsymbol{T}_i$ is a rotating line perpendicular to $\boldsymbol{N}_i$, and $s_i \in [0, 2\pi]$ is an angular parameter. The other parameter d_i represent the distance from the image of $\boldsymbol{P}_i$ in the domain. The surface can then be obtained by combining the ribbons using the same kind of blending functions as before, but including $n + k$ distance values, where k is the number of auxiliary points.

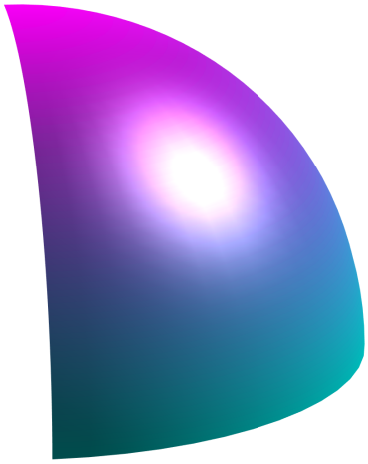
In a similar way it's also possible to force the surface to match an auxiliary curve $\boldsymbol{P}_i$. Its corresponding ribbons is given as

$$\boldsymbol{R}_i(s_i, d_i) = \boldsymbol{P}_i(s_i) \pm d_i w_i(s_i) \boldsymbol{T}_i(s_i).$$

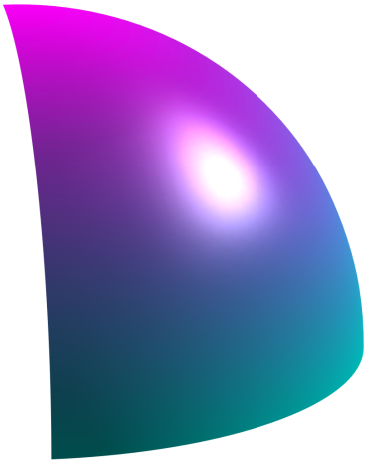
It's also possible to construct one-sided and two-sided patches[22]. A two-sided patch can be created by using a domain bounded by two parabolic arcs, and using simple line sweeps by a quadratic function to compute the distance parameters. We can then use the following blending function to combine the ribbons:

$$K_i(d_1, d_2) = \frac{d_j^2}{d_1^2 + d_2^2}, \quad i, j \in \{1, 2\},\ i \neq j.$$

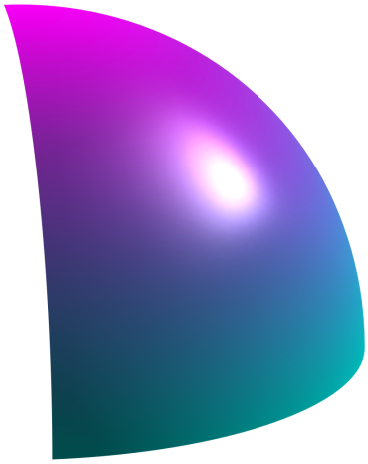
To construct a one-sided patch, we assume that we are given a closed curve in the plane together with a ribbon without local self-intersections. We can use a circle as the domain with radial sweep lines. If we also are given an auxiliary point in 3D with a normal vector, we can define an additional annular ribbon as above in the middle of the patch. We can then apply the same blending function as above to combine the two ribbons to create the surface.



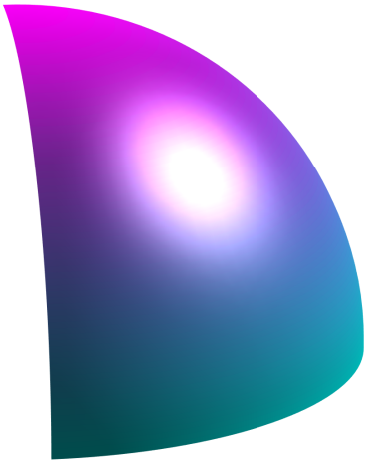
(a) Original sphere



(b) Generalized Coons patch (GC).

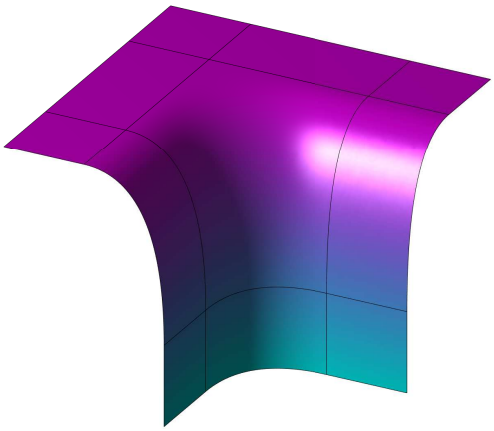


(c) Corner-based patch (CB).

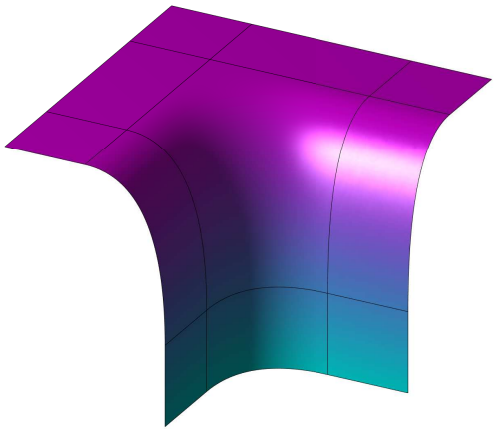


(d) Side-based patch (SB).

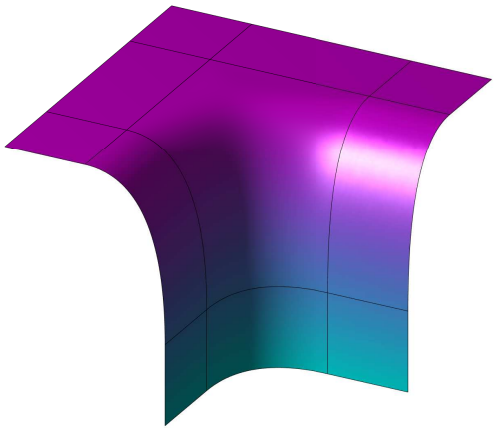
Figure 3.12: Sphere reproduction.



(a) Direct generalization of Coons patch.

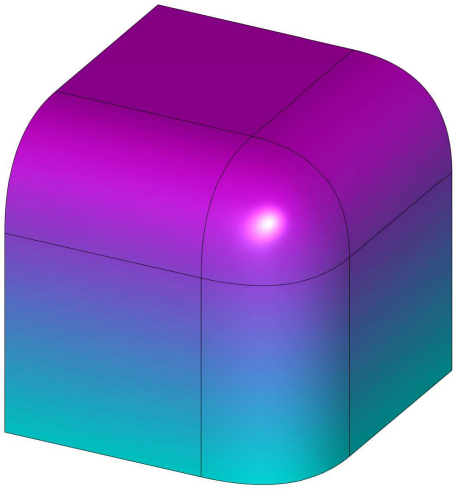


(b) Corner based patch.

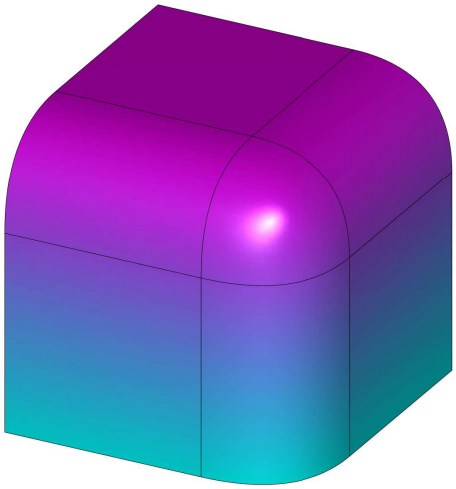


(c) Side based patch.

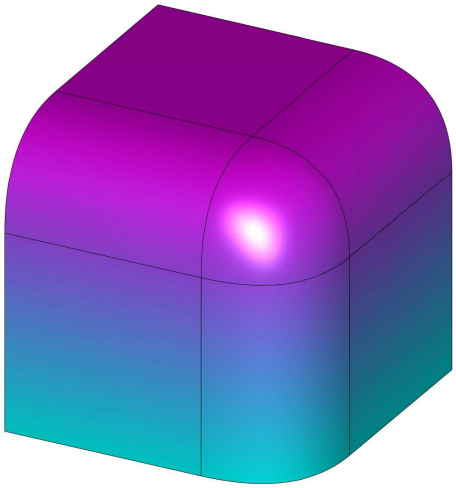
Figure 3.13: 5-sided patch



(a) Direct generalization of Coons patch



(b) Corner-based patch.



(c) Side-based patch

Figure 3.14: 3-sided patches

Chapter 4

Generalized barycentric coordinates

Barycentric coordinates have many applications in computer graphics, including curve, surface and image deformation, ray-tracing and parametrization of triangular meshes. In Chapter 3 we also saw that they can be used to parameterize the ribbon interpolants and constructing blending functions. As we will see in the next chapter, they can also be used for transfinite surface interpolation. While the triangular barycentric coordinates have been known for centuries, generalizations to n-sided polygons haven't been known before more recent years[8].

4.1 Triangular barycentric coordinates

Let $T \in \mathbb{R}^2$ be a non-degenerate triangle with vertices $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ in a counterclockwise ordering. The *barycentric coordinates* of a point $\boldsymbol{x} \in T$ is the unique numbers $\phi_1, \phi_2, \phi_3 \geq 0$, such that

$$\boldsymbol{x} = \phi_1 \boldsymbol{p}_1 + \phi_2 \boldsymbol{p}_2 + \phi_3 \boldsymbol{p}_3, \tag{4.1}$$

and

$$\phi_1 + \phi_2 + \phi_3 = 1. \tag{4.2}$$

To see that the coordinates are unique, first observe that equation 4.1 and 4.2 together can be written as a linear system of equations

$$\begin{pmatrix} 1 & 1 & 1 \\ p_1^u & p_2^u & p_3^u \\ p_1^v & p_2^v & p_3^v \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix} = \begin{pmatrix} 1 \\ x^u \\ x^v \end{pmatrix}, \tag{4.3}$$

where $\boldsymbol{p}_i = (p_i^u, p_i^v)$ for all i, and $\boldsymbol{x} = (x^u, x^v)$. Since the area of triangle T is given by

$$A(T) = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_1^u & p_2^u & p_3^u \\ p_1^v & p_2^v & p_3^v \end{vmatrix},$$

and by assumption is greater than zero, the matrix in 4.3 is non-singular, and therefore has a unique solution. Using Cramer's rule to solve the linear system 4.3, we get that

$$\phi_i = \frac{A(T_i)}{A(T)}, \qquad i = 1, 2, 3, \tag{4.4}$$

where T_i is the triangle with vertices $\boldsymbol{x}$ and $\boldsymbol{p}_j$ where $j \in \{1, 2, 3\} \setminus i$. See figure 1.

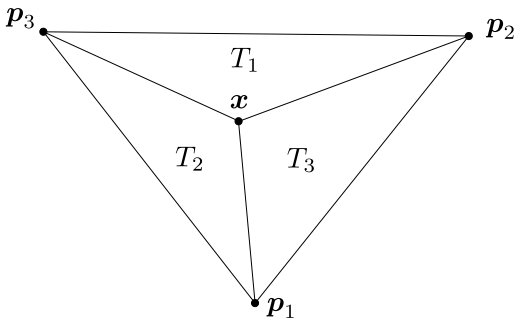


Figure 4.1: Subtriangles formed by $\boldsymbol{x}$

4.2 Barycentric coordinates on polygons

Suppose $\Omega \subset \mathbb{R}^2$ is a convex polygon with vertices $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$, $n \geq 3$, in a counter-clockwise ordering.

Definition 4.1 *The generalized barycentric coordinates (GBC's) of a point $\boldsymbol{x}$ w.r.t. Ω is a set of functions $\phi_i : \Omega \to \mathbb{R}^2$ which for all $\boldsymbol{x} \in \Omega$ satisfy the following properties:*

$$\phi_i(\boldsymbol{x}) \geq 0, \qquad i = 1, \ldots, n, \tag{4.5}$$

$$\sum_{i=1}^{n} \phi_i(\boldsymbol{x}) = 1 \tag{4.6}$$

and

$$\sum_{i=1}^{n} \phi_i(\boldsymbol{x}) \boldsymbol{p}_i = \boldsymbol{x}. \tag{4.7}$$

For $n = 3$, the functions ϕ_i are the unique triangular coordinates given by 4.4, while for $n \geq 4$, the functions $\phi_1(\boldsymbol{x}), \ldots, \phi_n(\boldsymbol{x})$ that satisfy the properties in definition 4.1 can not be uniquely determined.

Given a function $f\colon \Omega \to \mathbb{R}$, the GBC's can be used to interpolate the function values at the corners of Ω by using the formula

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} \phi_i(\boldsymbol{x}) f(\boldsymbol{p}_i), \qquad \boldsymbol{x} \in \Omega. \tag{4.8}$$

The barycentric interpolant has linear precision, i.e., if f is a linear polynomial, then $g = f$. To see this, first assume

$$f(\boldsymbol{x}) = \boldsymbol{a} \cdot \boldsymbol{x} + b.$$

Then, by using the barycentric properties 4.6 and 4.7, we get

$$\begin{aligned} g(\boldsymbol{x}) &= \sum_{i=1}^{n} \phi_i(\boldsymbol{x})(\boldsymbol{a} \cdot \boldsymbol{p}_i + b) = \boldsymbol{a} \cdot \sum_{i=1}^{n} \phi_i(\boldsymbol{x}) \boldsymbol{p}_i + b \sum_{i=1}^{n} \phi_i(\boldsymbol{x}) \\ &= \boldsymbol{a} \cdot \boldsymbol{x} + b = f(\boldsymbol{x}). \end{aligned}$$

The coordinate functions ϕ_i are also linear on each side of the polygon Ω and have the Lagrange property

$$\phi_i(\boldsymbol{p}_j) = \delta_{i,j},$$

where

$$\delta_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}.$$

This can be shown as follows[9]. Since the barycentric interpolant has linear precision, and the triangle area $A(\boldsymbol{x}, \boldsymbol{p}_j, \boldsymbol{p}_{j+1})$ is linear in $\boldsymbol{x}$, we get

$$\sum_{i=1}^{n} \phi_i(\boldsymbol{x}) A(\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_{j+1}) = A(\boldsymbol{x}, \boldsymbol{p}_j, \boldsymbol{p}_{j+1}).$$

If $\boldsymbol{x}$ is a point on the edge $[\boldsymbol{p}_j, \boldsymbol{p}_{j+1}]$, we get $A(\boldsymbol{x}, \boldsymbol{p}_j, \boldsymbol{p}_{j+1}) = 0$, and consequently

$$\sum_{i \neq j, j+1} \phi_i(\boldsymbol{x}) A(\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_{j+1}) = 0.$$

Since Ω is convex, $A(\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_{j+1}) > 0$ for $i \neq j, j+1$. By property 4.5, $\phi_i(\boldsymbol{x}) \geq 0$, it follows that $\phi_i(\boldsymbol{x}) = 0$ for all $i \neq j, j+1$. It then follows from property 4.7 that

$$\phi_j(\boldsymbol{x})\boldsymbol{p}_j + \phi_{j+1}(\boldsymbol{x})\boldsymbol{p}_{j+1} = \boldsymbol{x}.$$

By property 4.6 we have $\phi_j(\boldsymbol{x}) = 1 - \phi_{j+1}(\boldsymbol{x})$, so

$$\phi_j(\boldsymbol{x})(\boldsymbol{p}_j - \boldsymbol{p}_{j+1}) - \boldsymbol{p}_{j+1} = \boldsymbol{x}.$$

Thus, all functions $\phi_i(\boldsymbol{x})$ are linear on the edges, and it's easy to see that they also satisfies the Lagrange property.

We will consider two types of GBC's; Wachspress coordinates and Mean value coordinates.

4.3 Wachspress coordinates

The Wachspress coordinates introduced by Wachspress[24] can be defined by[17]

$$\phi_i(\boldsymbol{x}) = \frac{w_i(\boldsymbol{x})}{\sum_{j=1}^{n} w_j(\boldsymbol{x})}, \tag{4.9}$$

where

$$w_i(\boldsymbol{x}) = B_i \prod_{j \neq i-1, i} A_j(\boldsymbol{x}),$$

and

$$A_j(\boldsymbol{x}) = A(\boldsymbol{x}, \boldsymbol{p}_j, \boldsymbol{p}_{j+1}), \qquad B_i = A(\boldsymbol{p}_{i-1}, \boldsymbol{p}_i, \boldsymbol{p}_{i+1})$$

The function $A(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})$ denotes the signed area of the triangle with vertices $\boldsymbol{p}, \boldsymbol{q}$ and $\boldsymbol{r}$.

An advantage of the Wachspress coordinates is that they are rational functions. This makes them efficient to compute since they have no square roots etc. Since the triangle areas $A_j(\boldsymbol{x})$ are linear in $\boldsymbol{x}$, the numerator in 4.9 must be a polynomial of degree $n-2$, and the denominator must be of degree at most $n-2$. In fact, it can be shown that the denominator is of degree $n-3$. As we will see, this follows from the barycentric properties.

An alternative way of expressing the Wachspress coordinates was given in [17]. By dividing each w_i by $\prod_{j=1}^{n} A_j$, which does not change ϕ_i, we get

$$\phi_i(\boldsymbol{x}) = \frac{\tilde{w}_i(\boldsymbol{x})}{\sum_{j=1}^{n} \tilde{w}_j(\boldsymbol{x})}, \tag{4.10}$$

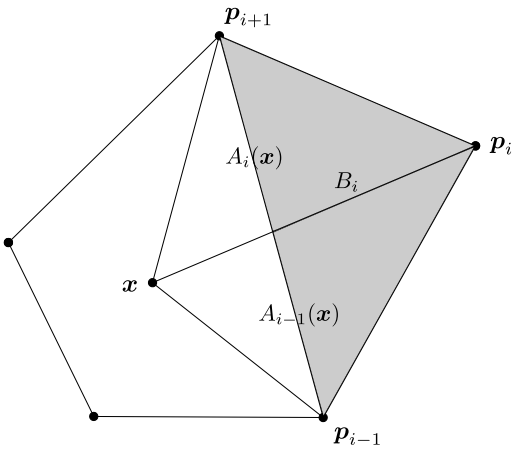


Figure 4.2: Triangular areas.

where

$$\tilde{w}_i(\boldsymbol{x}) = \frac{B_i}{A_{i-1}(\boldsymbol{x})A_i(\boldsymbol{x})}.$$

The triangular areas are shown in figure 4.2. Since $\tilde{w}_i$ only depends on the vertices $\boldsymbol{p}_{i-1}, \boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$, this local expression requires less computation when n is large. However, it's not valid on the boundary of Ω, as $A_{i-1}(\boldsymbol{x})$ and $A_i(\boldsymbol{x})$ becomes zero. To avoid numerical problems when evaluating the coordinates near the boundary, one can simply switch to the "global" form in 4.9.

Theorem 4.2 *The Wachspress coordinates defined by 4.10 satisfy all the requirements for GBC's given in definition 4.1.*

Proof. We will follow the proof by Meyer [17]. Since we assume Ω is a convex polygon, the signed triangular areas $A_i(\boldsymbol{x})$ are always positive, and it follows that $\phi_i(\boldsymbol{x})$ also is positive for $\boldsymbol{x}$ strictly inside the polygon. The second property property follows directly from 4.10. To show the third property, first note that

$$\begin{aligned} & \sum_{i=1}^{n} \phi_i(\boldsymbol{x})\boldsymbol{p}_i = \boldsymbol{x} \\ \Leftrightarrow \quad & \frac{1}{\sum_{j=1}^{n} w_j} \sum_{i=1}^{n} w_i \boldsymbol{p}_i = \boldsymbol{x} \\ \Leftrightarrow \quad & \sum_{i=1}^{n} w_i(\boldsymbol{x})(\boldsymbol{p}_i - \boldsymbol{x}) = 0 \end{aligned}$$

Using 4.4 and 4.1 we can express $\boldsymbol{x}$ as the triangular barycentric coordinates with respect to the triangle $T = (\boldsymbol{p}_{i-1}, \boldsymbol{p}_i, \boldsymbol{p}_{i+1})$. This equation holds even if

$\boldsymbol{x}$ is outside triangle T.

$$\begin{aligned}\boldsymbol{x} &= \frac{A_i(\boldsymbol{x})}{B_i}\boldsymbol{p}_{i-1} + \frac{A_{i-1}(\boldsymbol{x})}{B_i}\boldsymbol{p}_{i+1} + \frac{A(\boldsymbol{p}_{i-1}, \boldsymbol{p}_{i+1}, \boldsymbol{x})}{B_i}\boldsymbol{p}_i \\ &= \frac{A_i(\boldsymbol{x})}{B_i}\boldsymbol{p}_{i-1} + \frac{A_{i-1}(\boldsymbol{x})}{B_i}\boldsymbol{p}_{i+1} + \frac{B_i - A_i(\boldsymbol{x}) - A_{i-1}(\boldsymbol{x})}{B_i}\boldsymbol{p}_i.\end{aligned}$$

Since $\boldsymbol{x}$ is strictly inside the polygon, none of the triangular areas becomes zero, so the previous equation can be rearranged as:

$$\frac{B_i}{A_{i-1}(\boldsymbol{x})A_i(\boldsymbol{x})}(\boldsymbol{p}_i - \boldsymbol{x}) = \frac{1}{A_{i-1}(\boldsymbol{x})}(\boldsymbol{p}_i - \boldsymbol{p}_{i-1}) - \frac{1}{A_i(\boldsymbol{x})}(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)$$

By summing both sides of this equation over $i = 1, \ldots, n$, we get

$$\sum_{i=1}^{n} \tilde{w}_i(\boldsymbol{x})(\boldsymbol{p}_i - \boldsymbol{x}) = \sum_{i=0}^{n-1} \frac{(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)}{A_i(\boldsymbol{x})} - \sum_{i=1}^{n} \frac{(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)}{A_i(\boldsymbol{x})} = 0,$$

which proves the last property. ■

As we showed above, the last barycentric property is equivalent to

$$\sum_{i=1}^{n} w_i(\boldsymbol{x})(\boldsymbol{p}_i - \boldsymbol{x}) = 0,$$

so

$$\sum_{i=1}^{n} w_i(\boldsymbol{x})\boldsymbol{p}_i = \boldsymbol{x}\sum_{i=1}^{n} w_i(\boldsymbol{x}).$$

Since the left hand side of this equation is a polynomial of degree $n-2$, the sum $\sum_{i=1}^{n} w_i(\boldsymbol{x})$, i.e. the denominator of equation 4.9, must be a polynomial of degree $n-3$.

4.4 Mean value coordinates

Although the Mean value (MV) coordinates works on arbitrary domains, we will in this section only consider the case in which Ω is a convex polygon. The mean value coordinates are defined as

$$\phi_i(\boldsymbol{x}) = \frac{w_i(\boldsymbol{x})}{\sum_{j=1}^{n} w_j(\boldsymbol{x})}, \tag{4.11}$$

where

$$w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{||\boldsymbol{p}_i - \boldsymbol{x}||}, \tag{4.12}$$

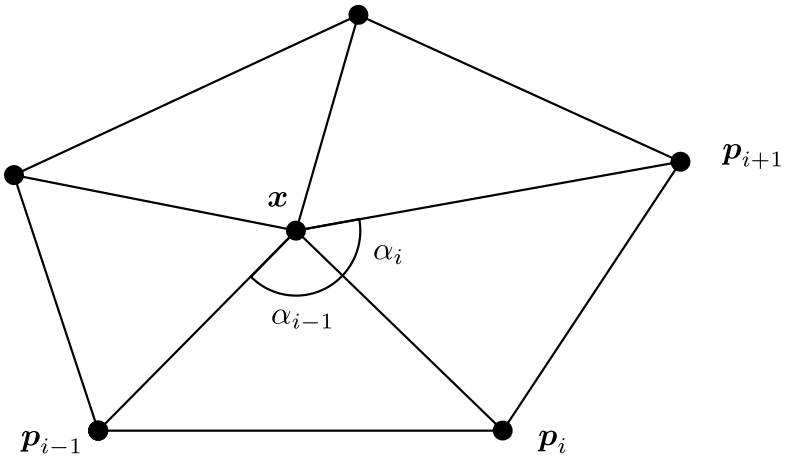


Figure 4.3: Angles in MV formula

where α_i is the angle between the vectors $(\boldsymbol{p}_i - \boldsymbol{x})$ and $(\boldsymbol{p}_{i+1} - \boldsymbol{x})$ as shown in figure 4. Note that, using the notation $\boldsymbol{e}_i = \frac{\boldsymbol{p}_i - \boldsymbol{x}}{||\boldsymbol{p}_i - \boldsymbol{x}||}$, and the planar cross product $\boldsymbol{a} \times \boldsymbol{b} = \det([\boldsymbol{a}, \boldsymbol{b}])$, the tangents functions in the formula can be evaluated by using that

$$\sin(\alpha_i) = \boldsymbol{e}_i \times \boldsymbol{e}_{i+1}, \qquad \cos(\alpha_i) = \boldsymbol{e}_i \cdot \boldsymbol{e}_{i+1},$$

and for instance the formula

$$\tan(\alpha/2) = \frac{\sin(\alpha)}{1 + \cos(\alpha)}.$$

We will now show that the MV coordinates satisfy the properties for GBC's in definition 4.1. Assume $\boldsymbol{x}$ is in the interior of Ω, i.e., $\boldsymbol{x} \in \mathrm{Int}(\Omega)$. The functions $\phi_i(\boldsymbol{x})$ are obviously non-negative since the w_i's are non-negative. The second property also follows straight from the definition of ϕ_i. As we saw in the previous section, to prove the third property, it's enough to prove the following theorem.

Theorem 4.3 *Let w_i be defined by 4.12. Then*

$$\sum_{i=1}^{n} w_i(\boldsymbol{x})(\boldsymbol{p}_i - \boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \mathit{Int}(\Omega) \tag{4.13}$$

Proof (see [8]). Let $\boldsymbol{x} \in \mathrm{Int}(\Omega)$ be fixed. Equation 4.13 can be written as

$$\begin{aligned}\sum_{i=1}^{n} w_i(\boldsymbol{x})(\boldsymbol{p}_i - \boldsymbol{x}) &= \sum_{i=1}^{n} (\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2))\boldsymbol{e}_i \\ &= \sum_{i=1}^{n} \tan(\alpha_i/2)(\boldsymbol{e}_i + \boldsymbol{e}_{i+1})\end{aligned}$$

Now, express $\boldsymbol{e}_i$ in polar coordinates: $\boldsymbol{e}_i = (\cos\theta_i, \sin\theta_i)$, and note that $\alpha_i = \theta_{i+1} - \theta_i$ for all i. We now have

$$\begin{aligned}\tan(\alpha_i/2)(\boldsymbol{e}_i + \boldsymbol{e}_{i+1}) &= \tan(\frac{\theta_{i+1} - \theta_i}{2})(\cos\theta_i + \cos\theta_{i+1}, \sin\theta_i + \sin\theta_{i+1}) \\ &= (\sin\theta_{i+1} - \sin\theta_i, \cos\theta_{i+1} - \cos\theta_i),\end{aligned}$$

where the last equality can be shown by trigonometry. The sum of this expression over $i = 1, \ldots, n$ equals to zero, which completes the proof. ■

The MV coordinates can also be applied to non-convex polygons[8]. The only difference is that the angles α_i must be treated as signed angles, i.e., we let α_i have the same sign as $\boldsymbol{e}_i \times \boldsymbol{e}_{i+1}$. Even though some of the w_i's may become negative, the coordinates are still well-defined, since the sum $\sum_{i=1}^{n} w_i(\boldsymbol{x})$ remains positive for any $\boldsymbol{x}$ in $\mathrm{Int}(\Omega)$.

4.4.1 Alternative formula

Like the Wachspress coordinates, the Mean value coordinates also have a global form in which $\phi_i(\boldsymbol{x})$ is well-defined on the whole polygon Ω including the boundary $\partial\Omega$ [8]. However, the global form involves more square roots and requires more computation that the local form. Let $\boldsymbol{d}_i = \boldsymbol{p}_i - \boldsymbol{x}$ and $r_i = ||\boldsymbol{d}_i||$, for $i = 1, \ldots, n$. The global form is then given by

$$\phi_i(\boldsymbol{x}) = \frac{\hat{w}_i(\boldsymbol{x})}{\sum_{j=1}^{n} \hat{w}_j(\boldsymbol{x})},$$

where

$$\hat{w}_i(\boldsymbol{x}) = (r_{i-1}r_{i+1} - \boldsymbol{d}_{i-1} \cdot \boldsymbol{d}_{i+1})^{\frac{1}{2}} \prod_{j \neq i-1, i} (r_j r_{j+1} + \boldsymbol{d}_j \cdot \boldsymbol{d}_{j+1})^{\frac{1}{2}}.$$

Chapter 5

Pointwise radial minimization

The *Pointwise radial minimization* method was introduced by Floater and Schulz[10]. This method is a generalization of mean value interpolation[6], which can be viewed as the pointwise minimization of a radial energy function involving first order derivatives of a radially linear function[10]. To match boundary derivatives up to order k, we instead minimize over $(k+1)$-st order derivatives of polynomials of degree $(2k+1)$. The pointwise radial minimization method can be applied to non-convex domains, and can match boundary derivatives of arbitrary order. However, in this chapter we will assume that the domain Ω is convex, and only consider interpolation of the boundary functions and its first order derivatives. In this case the method is sometimes called *cubic mean value* (CMV) interpolation since it has cubic precision[16].

We start by reviewing mean value interpolation which uses the mean value coordinates from Chapter 4.

5.1 Mean value interpolation

Let $\Omega \subset \mathbb{R}^2$ be a bounded, open and convex domain, and assume $f\colon \partial\Omega \to \mathbb{R}$ is a continuous function defined on the boundary. Let $\boldsymbol{x}$ be a point in Ω, and define $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})$ to be the point where the boundary $\partial\Omega$ intersects with the ray $\{\boldsymbol{x} + r\boldsymbol{v} : r \geq 0\}$ as shown in figure 5.1. This point is unique as long as Ω is convex. Let $\rho(\boldsymbol{x}, \boldsymbol{v})$ denote the euclidean distance from $\boldsymbol{x}$ to $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})$. The mean value interpolant can be defined as follows[6]: Assume $\boldsymbol{x} \in \Omega$ is fixed. Let $G\colon \overline{\Omega} \to \mathbb{R}$ be the function that for all $\boldsymbol{y} \in \partial\Omega$ satisfies $G(\boldsymbol{y}) = f(\boldsymbol{y})$, is linear along the line segment $[\boldsymbol{x}, \boldsymbol{y}]$, and satisfy the mean value property

$$G(\boldsymbol{x}) = \frac{1}{2\pi r} \int_\Gamma G(\boldsymbol{z}) d\boldsymbol{z}, \tag{5.1}$$

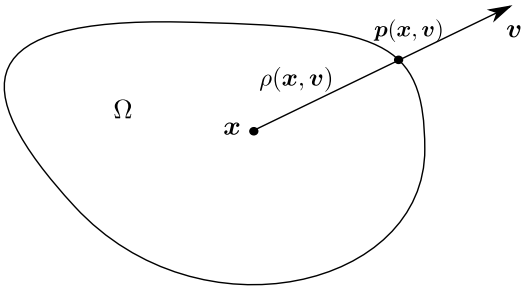


Figure 5.1: Definition of $\boldsymbol{p}(\boldsymbol{x},\boldsymbol{v})$

where Γ is a circle in Ω with centre in $\boldsymbol{x}$ and radius r. The mean value interpolant g is then defined in a pointwise fashion by setting $g(\boldsymbol{x}) = G(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$. To find $G(\boldsymbol{x})$, and thus $g(\boldsymbol{x})$, we first note that 5.1 can be written as

$$g(\boldsymbol{x}) = G(\boldsymbol{x}) = \frac{1}{2\pi}\int_0^{2\pi} G(\boldsymbol{x} + r(\cos(\theta), \sin(\theta)))d\theta, \tag{5.2}$$

where $(\cos(\theta), \sin(\theta)) = \boldsymbol{v}$. Since $G(\boldsymbol{x} + r\boldsymbol{v})$ is linear in r, we have

$$G(\boldsymbol{x} + r\boldsymbol{v}) = \frac{\rho(\boldsymbol{x},\boldsymbol{v}) - r}{\rho(\boldsymbol{x},\boldsymbol{v})} g(\boldsymbol{x}) + \frac{r}{\rho(\boldsymbol{x},\boldsymbol{v})} f(\boldsymbol{p}(\boldsymbol{x},\boldsymbol{v}))$$

Thus, we can write 5.2 as

$$\frac{1}{2\pi}\int_0^{2\pi} \frac{r(f(\boldsymbol{p}(\boldsymbol{x},\theta)) - g(\boldsymbol{x}))}{\rho(\boldsymbol{x},\theta)} d\theta = 0.$$

Solving this for $g(\boldsymbol{x})$ gives the unique solution

$$g(\boldsymbol{x}) = \int_0^{2\pi} \frac{f(\boldsymbol{p}(\boldsymbol{x},\theta))}{\rho(\boldsymbol{x},\theta)} d\theta \frac{1}{\phi(\boldsymbol{x})}, \tag{5.3}$$

where

$$\phi(\boldsymbol{x}) = \int_0^{2\pi} \frac{1}{\rho(\boldsymbol{x},\theta)} d\theta.$$

Note that even though the function G satisfies the mean value property 5.1, it does not mean that the interpolant $g(\boldsymbol{x})$ itself satisfies this property. It can also be shown that the interpolant $g(\boldsymbol{x})$ has linear precision. To see this, assume f is a linear function

$$f(\boldsymbol{x}) = \boldsymbol{a} \cdot \boldsymbol{x} + b.$$

Since $G = f$ satisfies the mean value property for all $\boldsymbol{x} \in \Omega$, we get $g(\boldsymbol{x}) = f(\boldsymbol{x})$. A proof that g interpolates the boundary function f can be found in [6].

5.2 Mean value interpolation on convex polygons

Assume Ω is a convex polygon. We will show that if f is linear along each edge then g is the barycentric interpolant to f using the mean value coordinates with respect to Ω.

Lemma 5.1 *Let $e = [\boldsymbol{p}_0, \boldsymbol{p}_1]$ be a line segment, and $f\colon e \to \mathbb{R}$ be a linear function. Let $\boldsymbol{x}$ be a point in the open half-plane on the left side of the vector $\boldsymbol{p}_1 - \boldsymbol{p}_0$. Let $\theta_0 < \theta_1$ be the two angles such that $\boldsymbol{p}_i - \boldsymbol{x} = \rho_i(\cos\theta_i, \sin\theta_i)$ for $i = 0, 1$, where $\rho_i = ||\boldsymbol{p}_i - \boldsymbol{x}||$. Then*

$$\int_{\theta_0}^{\theta_1} \frac{f(\boldsymbol{p}(\boldsymbol{x},\theta))}{\rho(\boldsymbol{x},\theta)} d\theta = \left(\frac{f(\boldsymbol{p}_0)}{\rho_0} + \frac{f(\boldsymbol{p}_1)}{\rho_1}\right) \tan\left(\frac{\theta_1 - \theta_0}{2}\right). \tag{5.4}$$

Proof (see [6]). Let $\boldsymbol{p} = \boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})$. Since f is linear

$$f(\boldsymbol{p}) = af(\boldsymbol{p}_0) + bf(\boldsymbol{p}_1), \tag{5.5}$$

where

$$a = \frac{||\boldsymbol{p} - \boldsymbol{p}_1||}{||\boldsymbol{p}_1 - \boldsymbol{p}_0||}, \qquad b = \frac{||\boldsymbol{p} - \boldsymbol{p}_0||}{||\boldsymbol{p}_1 - \boldsymbol{p}_0||}.$$

The scalars a and b can also be written as

$$a = \frac{A_1}{A}, \qquad b = \frac{A_0}{A}, \tag{5.6}$$

where A, A_0 and A_1 are the triangle areas

$$A = A(\boldsymbol{p}_0, \boldsymbol{x}, \boldsymbol{p}_1), \qquad A_0 = A(\boldsymbol{p}_0, \boldsymbol{x}, \boldsymbol{p}), \qquad A_1 = A(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{p}_1).$$

To see this, first let h denote the height of triangle $(\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{x})$ relative to the edge $(\boldsymbol{p}_0, \boldsymbol{p}_1)$. See figure 5.2. Then we have that

$$A = \frac{h||\boldsymbol{p}_1 - \boldsymbol{p}_0||}{2}, \qquad A_0 = \frac{h||\boldsymbol{p}_0 - \boldsymbol{p}||}{2}, \qquad A_1 = \frac{h||\boldsymbol{p}_1 - \boldsymbol{p}||}{2},$$

which shows 5.6. Using the sine rule for triangle areas, we can rewrite a and b further as

$$a = \frac{A_1}{A} = \frac{\sin(\theta - \theta_1)\rho}{\sin(\theta_1 - \theta_0)\rho_0}, \qquad b = \frac{A_0}{A} = \frac{\sin(\theta - \theta_0)\rho}{\sin(\theta_1 - \theta_0)\rho_1}.$$

Substituting this into equation 5.5, dividing by $\rho(\boldsymbol{x}, \boldsymbol{v})$, and integrating with respect to θ gives the result. ∎

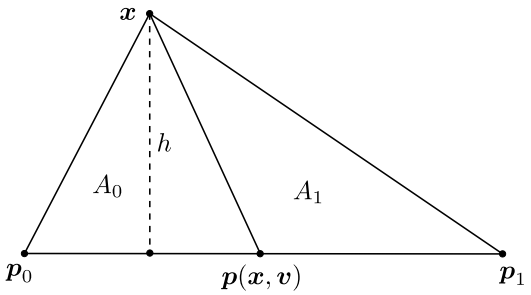


Figure 5.2: Triangle areas.

If we let $f \equiv 1$, it follows from the lemma that

$$\int_{\theta_0}^{\theta_1} \frac{1}{\rho(\boldsymbol{x}, \theta)} d\theta = \Big(\frac{1}{\rho_0} + \frac{1}{\rho_1}\Big) \tan\Big(\frac{\theta_1 - \theta_0}{2}\Big). \tag{5.7}$$

Using this together with 5.4, the mean value interpolant 5.3 reduces to

$$\begin{aligned} g(\boldsymbol{x}) &= \sum_{i=1}^{n} \int_{\theta_i}^{\theta_{i+1}} \frac{f(\boldsymbol{p}(\boldsymbol{x}, \theta))}{\rho(\boldsymbol{x}, \theta)} d\theta \frac{1}{\phi(\boldsymbol{x})} \\ &= \sum_{i=1}^{n} \Big(\frac{f(\boldsymbol{p}_i)}{\rho_0} + \frac{f(\boldsymbol{p}_{i+1})}{\rho_{i+1}}\Big) \tan\Big(\frac{\theta_{i+1} - \theta_i}{2}\Big) \frac{1}{\phi(\boldsymbol{x})} \\ &= \sum_{i=1}^{n} f(\boldsymbol{p}_i) \Big(\frac{f(\boldsymbol{p}_i)}{\rho_0} + \frac{f(\boldsymbol{p}_{i+1})}{\rho_{i+1}}\Big) \tan\Big(\frac{\theta_{i+1} - \theta_i}{2}\Big) \frac{1}{\phi(\boldsymbol{x})}, \end{aligned}$$

where

$$\begin{aligned} \phi(\boldsymbol{x}) &= \sum_{i=1}^{n} \Big(\frac{1}{\rho_i} + \frac{1}{\rho_{i+1}}\Big) \tan\Big(\frac{\theta_{i+1} - \theta_i}{2}\Big) \\ &= \sum_{i=1}^{n} \frac{\tan((\theta_{i+1} - \theta_i)/2) + \tan((\theta_i - \theta_{i-1})/2)}{\rho_i}. \end{aligned}$$

Letting $\alpha_i = \theta_{i+1} - \theta_i$, we see that g is exactly the same as the barycentric interpolant 4.8 with the mean value coordinates defined in equation 4.11 and 4.12.

5.3 Cubic mean value interpolation

Let $\Omega \subset \mathbb{R}^2$ be a bounded, open and convex domain, and assume $F \colon \partial\Omega \to \mathbb{R}$ is a given function defined on the boundary with continuous first order

derivatives. As before , let $\boldsymbol{x}$ be a point in the domain Ω, and let $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})$ to be the point where the boundary $\partial\Omega$ intersects with the ray $\{\boldsymbol{x} + r\boldsymbol{v} : r \geq 0\}$ as shown in figure 5.1. Let $S^1_{x,3}$ denote the linear space of all functions $G\colon \Omega \to \mathbb{R}$ that are C^1 continuous at $\boldsymbol{x}$ and which for all $\theta \in [0, 2\pi]$ are cubic polynomials on the line segment $[\boldsymbol{x}, \boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})]$, where $\boldsymbol{v} = (\cos\theta, \sin\theta)$. It was shown in [10] that there exists a unique function $G_* \in S^1_{x,3}$ that minimizes the local "energy"

$$E(G) \coloneqq \int_0^{2\pi} \int_0^{\rho} (D^2_{\boldsymbol{v}} G(\boldsymbol{x} + r\boldsymbol{v}))^2 dr d\theta, \tag{5.8}$$

where ρ is the Euclidean distance from $\boldsymbol{x}$ to $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})$, subject to the constraints

$$D^k_{\boldsymbol{v}} G(\boldsymbol{p}) = D^k_{\boldsymbol{v}} F(\boldsymbol{p}), \quad k = 0, 1, \quad \theta \in [0, 2\pi]. \tag{5.9}$$

By solving this problem for all $x \in \Omega$ we can define a function $g : \Omega \to \mathbb{R}$ pointwise by setting $g(\boldsymbol{x}) = G_*(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$. Note that although G_* is a polynomial, the obtained function g will not in general be a polynomial itself, since $G(\boldsymbol{x})$ can be a different function for each $\boldsymbol{x}$. It was proven in [16], that $g(\boldsymbol{x})$ interpolates F and its first order derivatives in the case in which Ω is a polygon, but it still remains to be proved in the general case. However numerical examples in [10] strongly suggest that it is C^1 also in the general case. It was shown in [10] that the CMV interpolant has cubic precision, i.e. if F is a cubic polynomial, then $g = F$.

We will now show how to find the minimizer G_*. In [8] it was shown that G_* satisfy the equations

$$\int_0^{2\pi} D^2_{\boldsymbol{v}} G_*(\boldsymbol{x}) \boldsymbol{v} d\theta = \boldsymbol{0} \tag{5.10}$$

and

$$\int_0^{2\pi} D^3_{\boldsymbol{v}} G_*(\boldsymbol{x}) d\theta = \boldsymbol{0}. \tag{5.11}$$

By the assumption that G_* is a cubic polynomial on the line segments $[\boldsymbol{x}, \boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})]$, and by the constraints 5.9, G_* is uniquely determine by

$$a \coloneqq G_*(\boldsymbol{x}) \tag{5.12}$$

and

$$\boldsymbol{b} \coloneqq \nabla G_*(\boldsymbol{x}). \tag{5.13}$$

To find a and $\boldsymbol{b}$ we use that the equations 5.10 and 5.11 can be expressed as the linear system[8]

$$M \begin{pmatrix} a \\ \boldsymbol{b} \end{pmatrix} = \boldsymbol{c}, \tag{5.14}$$

where

$$M = \begin{bmatrix} 6\int \sigma^3 & 3\int \sigma^2 \boldsymbol{v}^T \\ 3\int \sigma^2 \boldsymbol{v} & 2\int \sigma \boldsymbol{v}\boldsymbol{v}^T \end{bmatrix},$$

$$\boldsymbol{c} = \begin{bmatrix} 3\int \sigma^2(2\sigma F(\boldsymbol{p}) - D_{\boldsymbol{v}}F(\boldsymbol{p})) \\ \int \sigma(3\sigma F(\boldsymbol{p}) - D_{\boldsymbol{v}}F(\boldsymbol{p}))\boldsymbol{v} \end{bmatrix},$$

and $\sigma = 1/\rho$.

Note that we don't need the gradient $\nabla G_*(\boldsymbol{x})$ to define the surface g, but our experiments show that it's a good approximation to the gradient of g.

5.4 Implementation

The CMV interpolation method was implemented in MATLAB®, and used to generate n-sided parametric patches $\boldsymbol{S}\colon \Omega \to \mathbb{R}^3$. In order to compare the method with the ribbon based patches, we assume that the domain Ω is a n-sided polygon with vertices $\boldsymbol{p}_i$, and that we are given n boundary functions $\boldsymbol{P}_i(s_i)$, $s_i \in [0,1]$ and cross boundary derivative functions $\boldsymbol{T}(s_i)$. Let $\boldsymbol{d}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$ and $\boldsymbol{e}_i = \frac{\boldsymbol{d}_i}{||\boldsymbol{d}_i||}$. The functions $\boldsymbol{T}_i$ will be matched by $\boldsymbol{S}$ such that $D_{\boldsymbol{n}_i(s_i)}\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{T}_i(s_i)$ for $\boldsymbol{x} = s_i\boldsymbol{p}_{i+1} + (1-s_i)\boldsymbol{p}_i$. We assume that the vector $\boldsymbol{n}_i(s_i)$ can be obtained by linearly interpolating between $-\boldsymbol{e}_{i-1}$ and $\boldsymbol{e}_{i+1}$ and then normalizing, i.e.

$$\boldsymbol{n}_i(s_i) = \frac{s_i\boldsymbol{e}_{i+1} + (1-s_i)(-1)\boldsymbol{e}_{i-1}}{||s_i\boldsymbol{e}_{i+1} + (1-s_i)(-1)\boldsymbol{e}_{i-1}||}$$

As before, we start by triangulate the domain polygon using Delaunay triangulation. For every point $\boldsymbol{x}$ in the triangular mesh, we then need to find a 3D surface point $\boldsymbol{S}(\boldsymbol{x})$. For each $\boldsymbol{x}$, we first check if it's on any of the sides of Ω. If $\boldsymbol{x}$ is on side k, then we simply set $\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{P}_k(s_k)$, where $s_k = \frac{||\boldsymbol{x}-\boldsymbol{p}_k||}{||\boldsymbol{d}_k||}$. To find out if $\boldsymbol{x}$ is on an edge, we check for every $i = 1, \ldots, n$ if the vectors $(\boldsymbol{x} - \boldsymbol{p}_i)$ and $\boldsymbol{d}_i$ are collinear. This can be done by calculating the cross product $(\boldsymbol{x} - \boldsymbol{p}_i) \times \boldsymbol{d}_i$. If this is equal to zero, the two vectors are collinear, and $\boldsymbol{x}$ must be on side i.

If $\boldsymbol{x}$ is not on the boundary of Ω, we use Gauss quadrature to approximate the integrals in 5.14 (see for example [5]). Using N-point Gauss quadrature rule with domain $[0, 2\pi]$, we get N points θ_i and weights w_i, such that

$$\int_0^{2\pi} f(\theta)d\theta \approx \sum_{i=1}^{N} w_i f(\theta_i), \tag{5.15}$$

for some function $f\colon [0, 2\pi] \to \mathbb{R}$. This quadrature rule is exact for polynomials of degree $2N - 1$ or lower. For each angle θ_i, we therefore need to

find $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v}_i)$ and $\rho(\boldsymbol{x}, \boldsymbol{v}_i)$, where $\boldsymbol{v}_i = (\cos\theta_i, \sin\theta_i)$. To do this, we check for each edge $\boldsymbol{d}_j$ if $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v}_i)$ is on $\boldsymbol{d}_j$. This can be done by fist finding numbers t and s such that $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v}_i) = \boldsymbol{p}_j + t\boldsymbol{d}_j = \boldsymbol{x} + s\boldsymbol{v}_i$. Solving this linear equation using Cramer's rule, we get

$$s = \frac{\det([\boldsymbol{p}_j - \boldsymbol{x}, -\boldsymbol{d}_j])}{\det([\boldsymbol{v}_i, -\boldsymbol{d}_j])}, \qquad t = \frac{\det([\boldsymbol{v}_i, \boldsymbol{p}_j - \boldsymbol{x}])}{\det([\boldsymbol{v}_i, -\boldsymbol{d}_j])}.$$

If $s \geq 0$ and $t \in [0, 1]$ for $j = k$, then $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v}_i)$ must be on side k. We then have that $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v}_i) = \boldsymbol{p}_k + t\boldsymbol{d}_k$ and $\rho_i(\boldsymbol{x}, \boldsymbol{v}_i) = s$. To find $D_{\boldsymbol{v}_i} F(\boldsymbol{p})$, we first express $\boldsymbol{v}_i$ as a linear combination of the vectors $\boldsymbol{e}_k$ and $\boldsymbol{n}_k(s_k)$. This can be done by solving the linear system of equations

$$\begin{bmatrix} \boldsymbol{e}_k & \boldsymbol{n}_k(s_k) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \boldsymbol{v}_i.$$

Then we have

$$D_{\boldsymbol{v}_i} F(\boldsymbol{p}) = y_1 \frac{P_k'(s_k)}{||\boldsymbol{d}_k||} + y_2 T_k(s_k).$$

We can now apply formula 5.15 to compute the integrals in 5.14. Finally, using Cramer's rule to solve the linear system 5.14, we get

$$a = \frac{\det(\boldsymbol{c}, M_2, M_3)}{\det(M)},$$

where M_2 and M_3 are respectively the second and third column of matrix M. Note that we don't need to solve the linear system for vector $\boldsymbol{b}$, since we only need the value $G_*(\boldsymbol{x})$, not the gradient $\nabla G_*(\boldsymbol{x})$.

Also note that we need to solve the linear system 5.14 for each of the three coordinate of $\boldsymbol{S}(\boldsymbol{x})$, for different values of $\boldsymbol{c}$, but the same matrix M.

5.5 Examples

Example 5.2 (sphere reproduction) *In this example we have applied CMV interpolation to boundary data taken from an octant of a sphere with radius* 1 *as in example 3.7. The result is shown in figure 5.3. Table 5.1 compares the radius values with the ones from example 3.7. As we can see, the CMV interpolant is a fairly good approximation to the sphere, but it slightly underestimates the radius values, while the ribbon based methods overestimates the radius values.*

	Average	Min	Max	Std. deviation
Direct generalization(GC)	1.0184	1.0000	1.0556	0.01670
Corner-based interpolants(CB)	1.0155	1.0000	1.0555	0.01587
Side-based interpolants(SB)	1.0055	1.0000	1.0106	0.00389
Cubic mean value(CMV)	0.9822	0.9405	1.0000	0.01734

Table 5.1: Comparison of radius values

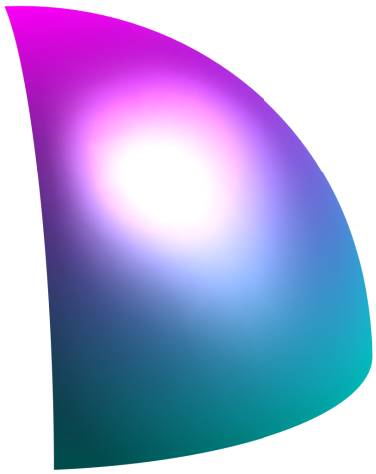
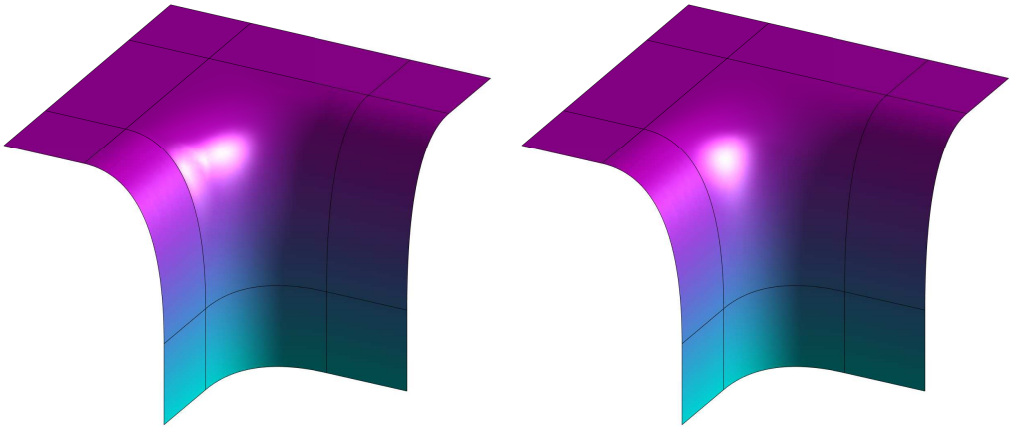


Figure 5.3: Sphere reproduction with cubic mean value interpolation

Example 5.3 (5-sided patch) *Figure 5.4a shows the CMV interpolation patch used to fill a 5-sided hole formed by bi-quadratic Bézier patches. We have used the same boundary data as in example 3.8. Unfortunately, some small artifacts can be seen in parts of the surface. In this case, the boundary integrals in 5.14 was calculated by 3-point Gauss quadrature rule. By raising the number of points used in the quadrature rule, the artifacts gets less noticeable. In this example, the artefacts did not completely disappear before the number of points in the quadrature rule was raised to 8, as shown in figure 5.4b. This has the disadvantage of making the computations of the surface heavier, as we will see in example 5.5.*

Example 5.4 (3-sided patch) *In this example we have used CMV interpolation to fill a 3-sided hole as in example 3.9. The vector valued boundary derivatives $D_{\boldsymbol{v}}\boldsymbol{f}$ was obtained from the adjacent patches, and scaled such that the interior of the patch looks as smooth as possible. Thus, the patch will match the surrounding surfaces with G^1 continuity. Figure 5.5 shows the*



(a) (b)

Figure 5.4: A 5-sided patch using CMV interpolation with 3-point(a) and 8-point(b) Gauss quadrature

result. In this example there were no visible artifacts using 3-points Gauss quadrature.

Example 5.5 (Computational efficiency) *The average evaluation times per surface point for the CMV interpolation method was calculated using the same boundary data and domain polygons as in example 3.10. Table 5.2 extends the table from 3 with the test results from using both 3-point and 8-point Gauss quadrature. We see that the evaluation time for all methods grows linearly with the number of boundary functions n. For the ribbon based methods this is because we for each point $\boldsymbol{x}$ need to evaluate ribbons, correction surfaces and blending functions corresponding to each edge of the domain polygon. When we evaluate the CMV interpolant the only calculations depending on n are done when we find out if $\boldsymbol{x}$ is on the boundary of Ω, and when we find the point $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{v})$. Since these operations are less time consuming, we see that the average evaluation time for the CMV interpolation scheme grows at a smaller rate than the other methods. For $n = 3$ we see that the CMV method using 3-point Gauss quadrature is slower than the ribbon based methods, but when $n = 6$ it catches up with the GC and CB method. On the other hand, if we use 8-point Gauss quadrature, it's still the slowest method for $n = 8$.*

Note that more effort could have been done to make the computations more efficient, for example with extensive caching.

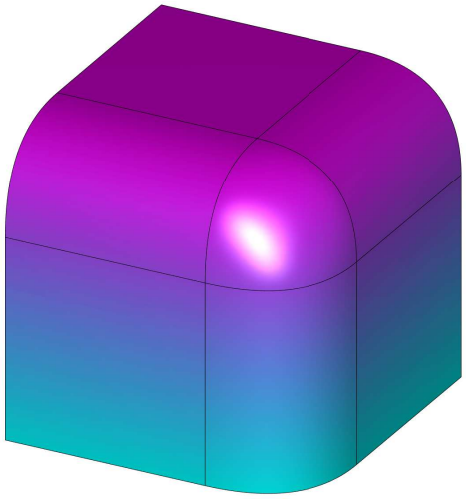


Figure 5.5: A 3-sided patch

	n=3	n=4	n=5	n=6	n=7	n=8
Direct Generalization	0.17	0.21	0.28	0.32	0.37	0.41
Corner-based interpolants	0.21	0.22	0.29	0.36	0.41	0.46
Side-based interpolants	0.16	0.16	0.21	0.27	0.31	0.34
CMV (3-point Gauss)	0.22	0.24	0.26	0.29	0.31	0.35
CMV (8-point Gauss)	0.49	0.55	0.61	0.69	0.74	0.80

Table 5.2: Average evaluation time per surface point in milliseconds on a 3.50GHz processor.

Chapter 6

Cubic mean value interpolation with piecewise quadratic boundary data

To avoid using numerical integration to calculate the boundary integrals in the linear system 5.14, we would like to find the analytic solutions. In [16], closed forms for these integrals were found in the case in which Ω is a polygon, g is cubic along each edge, and the normal derivatives of g are linear. However, these closed forms are complicated, and involves operations on complex numbers. Nor were we able to implement them. If we instead assume that g is piecewise quadratic along each edge, and the normal derivatives are piecewise linear, we will see that the boundary integrals have much simpler closed forms. These formulas were first derived in [2]. Note that since the closed forms derived by Li[16] assume that the normal derivatives are linear along each edge, it won't reproduce cubic functions, only quadratics. Thus we don't loose any degree of precision, compared to the method by Li, by assuming that the boundary functions are piecewise quadratic.

We start reviewing the formulas in [2] by introducing some notation and definitions. First define

$$(v_0, v_1, v_2) = (\rho^{-1}, \cos\theta, \sin\theta),$$

and observe that M and $\boldsymbol{c}$ in the linear system 5.14 can be written as

$$M = \begin{bmatrix} 6I_{00} & 3I_{01} & 3I_{02} \\ 3I_{10} & 2I_{11} & 2I_{12} \\ 3I_{20} & 2I_{21} & 2I_{22} \end{bmatrix}, \qquad \boldsymbol{c} = \begin{bmatrix} 6J_0 - 3K_0 \\ 3J_1 - K_1 \\ 3J_2 - K_2 \end{bmatrix}, \tag{6.1}$$

where

$$I_{jk} = \int_0^{2\pi} v_0 v_j v_k \, d\theta, \qquad j, k = 0, 1, 2,$$

and

$$J_j = \int_0^{2\pi} v_0^2 g(\boldsymbol{p}) v_j \, d\theta, \qquad K_j = \int_0^{2\pi} v_0 D_{\boldsymbol{v}} g(\boldsymbol{p}) v_j \, d\theta, \qquad j = 0, 1, 2.$$

For $i = 1, \ldots, n$, let $\boldsymbol{q}_i$ be the midpoint of side i of the domain polygon as shown in figure 6.1. The only data we need to construct the interpolant g

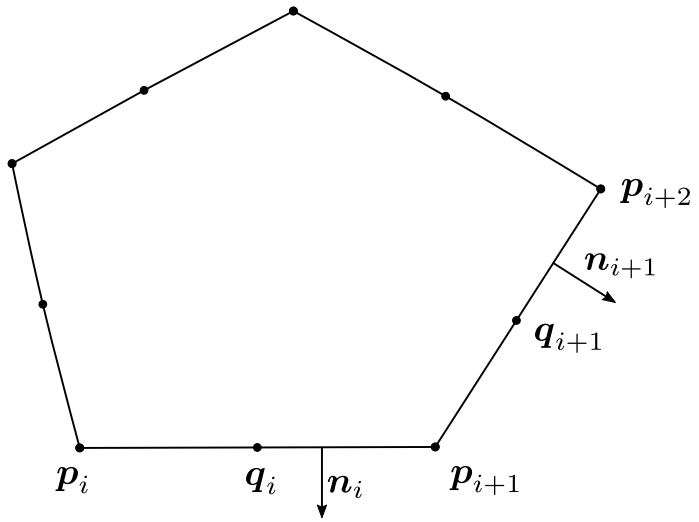


Figure 6.1: The domain polygon with midpoints $\boldsymbol{q}_i$.

are

$$f(\boldsymbol{p}_i),\ \nabla f(\boldsymbol{p}_i),\ D_{\boldsymbol{n}_i} f(\boldsymbol{q}_i), \qquad i = 1, \ldots, n, \tag{6.2}$$

where $\boldsymbol{n}_i$ now is the outward unit normal vector to edge i.

On each edge $[\boldsymbol{p}_i, \boldsymbol{p}_{i+1}]$, the boundary function f and the normal normal derivatives $D_{\boldsymbol{n}_i} g$ will be represented in respectively piecewise quadratic and piecewise linear Bernstein form (see for example [5]). Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be points on respectively the first and second half of this edge, e.i. let

$$\boldsymbol{p} = (1-t)\boldsymbol{p}_i + t\boldsymbol{q}_i, \quad \text{and} \quad \boldsymbol{q} = (1-u)\boldsymbol{q}_i + u\boldsymbol{p}_{i+1}, \qquad t, u \in [0, 1].$$

The Bernstein form of g is then

$$\begin{aligned} g(\boldsymbol{p}) &= (1-t)^2 c_0 + 2t(1-t)c_1 + t^2 c_2, \\ g(\boldsymbol{q}) &= (1-u)^2 c_2 + 2u(1-u)c_3 + u^2 c_4, \end{aligned}$$

where

$$c_0 = f(\boldsymbol{p}_i), \qquad c_1 = f(\boldsymbol{p}_i) + (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \cdot \nabla f(\boldsymbol{p}_i)/4, \qquad c_2 = (c_1 + c_3)/2,$$

$$c_3 = f(\boldsymbol{p}_{i+1}) - (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \cdot \nabla f(\boldsymbol{p}_{i+1})/4, \qquad c_4 = f(\boldsymbol{p}_{i+1}).$$

The Bernstein form of the normal derivative $D_{\boldsymbol{n}_i} g$ is

$$D_{\boldsymbol{n}_i} g(\boldsymbol{p}) = (1-t)d_0 + td_1,$$
$$D_{\boldsymbol{n}_i} g(\boldsymbol{q}) = (1-u)d_1 + ud_2,$$

where

$$d_0 = \boldsymbol{n}_i \cdot \nabla f(\boldsymbol{p}_i), \qquad d_1 = D_{\boldsymbol{n}_i} f(\boldsymbol{q}_i), \qquad d_2 = \boldsymbol{n}_i \cdot \nabla f(\boldsymbol{p}_{i+1}). \tag{6.3}$$

6.1 Computing the integrals

To find simple formulas for the integrals in 6.1, we keep $\boldsymbol{x} \in \Omega$ fixed, and split each integral into pieces, each corresponding to a half-edge of the polygon. Each half-edge will either be of the form $[\boldsymbol{p}_i, \boldsymbol{q}_i]$ or $[\boldsymbol{q}_i, \boldsymbol{p}_{i+1}]$. For simplicity's sake, we let the generic half-edge $[\boldsymbol{r}_0, \boldsymbol{r}_1]$ represent both kinds. See figure 6.2. The integrals can then be calculated as the sum of the integrals over

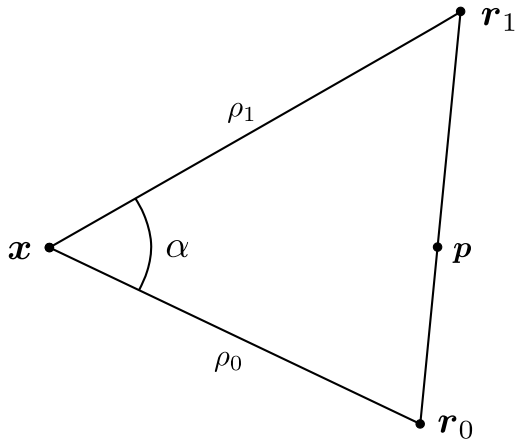


Figure 6.2: Notation for the generic half-edge $[\boldsymbol{r}_0, \boldsymbol{r}_1]$.

each half-edge. Since g is a quadratic polynomial on the half-edge $[\boldsymbol{r}_0, \boldsymbol{r}_1]$, we have

$$g(\boldsymbol{p}) = (1-t)^2 c_0 + 2t(1-t)c_1 + t^2 c_2, \tag{6.4}$$

where

$$\boldsymbol{p} = (1-t)\boldsymbol{r}_0 + t\boldsymbol{r}_1, \qquad t \in [0,1], \tag{6.5}$$

and c_0, c_1, c_2 are the Bézier coefficients. Similarly, since the outward unit normal derivative $D_{\boldsymbol{n}} g$ is linear, we have

$$D_{\boldsymbol{n}} g(\boldsymbol{p}) = (1-t)d_0 + td_1, \tag{6.6}$$

for some coefficients d_0 and d_1. Now, define $\rho_i = ||\boldsymbol{r}_i - \boldsymbol{x}||$, $i = 0, 1$. Let θ_0 and θ_1 be the angles such that

$$\boldsymbol{r}_i = \boldsymbol{x} + \rho_i \boldsymbol{v}_i, \qquad i = 0, 1, \tag{6.7}$$

where $\boldsymbol{v}_i = (\cos\theta_i, \sin\theta_i)$. Also, let

$$(v_{i,0}, v_{i,1}, v_{i,2}) = (\rho_i^{-1}, \cos\theta_i, \sin\theta_i), \qquad i = 0, 1. \tag{6.8}$$

We start by considering the elements of matrix M, since these doesn't depend on the function f, only on the geometry of Ω. Let $\alpha = \theta_1 - \theta_0$ be the interior angle as shown in figure 6.2. To simplify some expressions we will use the sets of multi-indices

$$\mathcal{P}_{r,n} = \{(i_1, \ldots, i_n) : i_1, \ldots, i_n \in \{0, 1\}, i_1 + \cdots + i_n = r\}, \tag{6.9}$$

for $n \geq 1$ and $0 \leq r \leq n$. In particular, for $r = 3$ we have

$$\begin{aligned} &\mathcal{P}_{0,3} = \{(0,0,0)\}, \quad \mathcal{P}_{1,3} = \{(1,0,0), (0,1,0), (0,0,1)\}, \\ &\mathcal{P}_{2,3} = \{(1,1,0), (1,0,1), (0,1,1)\}, \quad \mathcal{P}_{3,3} = \{(1,1,1)\}. \end{aligned}$$

Let

$$L_0 = L_3 = \frac{(1-\cos\alpha)^2(2+\cos\alpha)}{3\sin^3\alpha}, \qquad L_1 = L_2 = \frac{(1-\cos\alpha)^2}{3\sin^3\alpha}. \tag{6.10}$$

Theorem 6.1

$$\int_{\theta_0}^{\theta_1} v_0 v_j v_k \, d\theta = \sum_{r=0}^{3} L_r \sum_{(i_1,i_2,i_3)\in\mathcal{P}_{r,3}} v_{i_1,0} v_{i_2,j} v_{i_3,k}, \quad j, k = 0, 1, 2. \tag{6.11}$$

Proof (See [2]). Let

$$A = A(\boldsymbol{x}, \boldsymbol{r}_0, \boldsymbol{r}_1), \qquad A_0 = A(\boldsymbol{x}, \boldsymbol{r}_0, \boldsymbol{p}), \qquad A(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{r}_1).$$

The sine rule for triangular areas gives that

$$A = \frac{1}{2}\rho_0\rho_1\sin\alpha, \qquad A_0 = \frac{1}{2}\rho\rho_0\sin(\theta - \theta_0), \qquad A_1 = \frac{1}{2}\rho\rho_1\sin(\theta_1 - \theta).$$

It follows that

$$1 = \frac{A_0}{A} + \frac{A_1}{A} = \rho\frac{\sin(\theta - \theta_0)}{\rho_1\sin\alpha} + \rho\frac{\sin(\theta_1 - \theta)}{\rho_0\sin\alpha},$$

which again gives

$$\rho^{-1} = \lambda\rho_0^{-1} + \mu\rho_1^{-1} \tag{6.12}$$

where

$$\lambda = \frac{\sin(\theta_1 - \theta)}{\sin\alpha}, \qquad \mu = \frac{\sin(\theta - \theta_0)}{\sin\alpha}.$$

Now we will show that

$$\boldsymbol{v} = \lambda\boldsymbol{v}_0 + \mu\boldsymbol{v}_1. \tag{6.13}$$

Using the addition formula for sines,

$$\sin(a - b) = \sin a\cos b - \sin b\cos a, \tag{6.14}$$

we get that

$$\begin{aligned}
&\lambda\cos\theta + \mu\cos\theta \\
&\quad = \frac{\sin(\theta_1 - \theta)}{\sin\alpha}\cos\theta_0 + \frac{\sin(\theta - \theta_0)}{\sin\alpha}\cos\theta_1 \\
&\quad = \frac{(\sin\theta_1\cos\theta - \sin\theta\cos\theta_1)\cos\theta_0 + (\sin\theta\cos\theta_0 - \sin\theta_0\cos\theta)\cos\theta_1}{\sin\alpha} \\
&\quad = \frac{(\sin\theta_1\cos\theta_0 - \sin\theta_0\cos\theta_1)\cos\theta + (\cos\theta_0\cos\theta_1 - \cos\theta_0\cos\theta_1)\sin\theta}{\sin\alpha} \\
&\quad = \frac{\sin\alpha\cos\theta + 0}{\sin\alpha} = \cos\theta.
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
&\lambda\sin\theta + \mu\sin\theta \\
&\quad = \frac{(\sin\theta_1\cos\theta - \sin\theta\cos\theta_1)\sin\theta_0 + (\sin\theta\cos\theta_0 - \sin\theta_0\cos\theta)\sin\theta_1}{\sin\alpha} \\
&\quad = \frac{(\sin\theta_0\sin\theta_1 - \sin\theta_0\sin\theta_1)\cos\theta + (\sin\theta_1\cos\theta_0 - \sin\theta_0\cos\theta_1)\sin\theta}{\sin\alpha} \\
&\quad = \frac{0 + \sin\alpha\sin\theta}{\sin\alpha} = \sin\theta.
\end{aligned}$$

Thus we have

$$v_j = \lambda v_{0,j} + \mu v_{1,j}, \qquad j = 0, 1, 2. \tag{6.15}$$

Then

$$v_0 v_j v_k = \sum_{r=0}^{3} \mu^r \lambda^{3-r} \sum_{(i_1,i_2,i_3)\in\mathcal{P}_{r,3}} v_{i_1,0} v_{i_2,j} v_{i_3,k}, \qquad j, k = 0, 1, 2.$$

Integrating this over $\theta \in [\theta_0, \theta_1]$ proves the theorem if we can show that the integrals

$$L_r := \int_{\theta_0}^{\theta_1} \mu^r \lambda^{3-r}\, d\theta, \qquad r = 0, 1, 2, 3$$

equals the expressions in 6.10. To show this, first let $\beta = \theta - \theta_0$, and observe that

$$L_r = \frac{1}{\sin^3 \alpha} \widehat{L}_r,$$

where

$$\widehat{L}_r = \int_0^\alpha \sin^r \beta \sin^{3-r}(\alpha - \beta)\, d\beta.$$

By symmetry, we have that $\widehat{L}_0 = \widehat{L}_3$ and $\widehat{L}_1 = \widehat{L}_2$, so it's enough to find $\widehat{L}_2$ and $\widehat{L}_3$. Using the identity

$$\sin^2 \beta = 1 - \cos^2 \beta$$

and integration by parts we get

$$\begin{aligned}\widehat{L}_3 &= \int_0^\alpha \sin\beta - \sin\beta\cos^2\beta\, d\beta = \frac{1}{3}(2 - 3\cos\alpha + \cos^3\alpha) \\ &= \frac{1}{3}(1-\cos\alpha)^2(2+\cos\alpha).\end{aligned}$$

To find $\widehat{L}_2$, we first use the addition formula 6.14.

$$\begin{aligned}\widehat{L}_2 &= \int_0^\alpha \sin(\alpha-\beta)\sin^2\beta\, d\beta = \int_0^\alpha \sin\alpha\cos\beta\sin^2\beta - \cos\alpha\sin^3\beta\, d\beta \\ &= \sin\alpha \int_0^\alpha \cos\beta\sin^2\beta\, d\beta - \cos\alpha\widehat{L}_3.\end{aligned}$$

Using integration by parts we get that

$$\int_0^\alpha \cos\beta\sin^2\beta\, d\beta = \frac{1}{3}\sin^3\alpha,$$

so

$$\widehat{L}_2 = \frac{1}{3}\sin^4\alpha - \cos\alpha\widehat{L}_3.$$

Finally, using the identity $\sin^2\alpha = 1 - \cos^2\alpha$, this simplifies to

$$\widehat{L}_2 = \frac{1}{3}(1-\cos\alpha)^2.$$

■

To find the elements of $\boldsymbol{c}$ in 5.14 we need to find the integrals J_i and K_i. Let us first consider the integrals J_i which only depends on the values of g.

Theorem 6.2 *Let g be as in 6.4 , then*

$$\int_{\theta_0}^{\theta_1} v_0^2 g(\boldsymbol{p}) v_j = \sum_{r=0}^{3} L_r \sum_{(i_1,i_2,i_3)\in\mathcal{P}_{r,3}} c_{i_1+i_2} v_{i_1,0} v_{i_2,0} v_{i_3,j}, \quad j = 0,1,2.$$

Proof (See [2]). From the fact that $\boldsymbol{p} = \boldsymbol{x} + \rho\boldsymbol{v}$ and from equation 6.5, it follows that

$$\boldsymbol{x} + \rho\boldsymbol{v} = (1-t)\boldsymbol{r}_0 + t\boldsymbol{r}_1 \Leftrightarrow \rho\boldsymbol{v} = (1-t)\boldsymbol{v}_0\rho_0 + t\boldsymbol{v}_1\rho_1$$

Since

$$\boldsymbol{v} = \lambda\boldsymbol{v}_0 + \mu\boldsymbol{v}_1,$$

which we showed in the proof of theorem 6.1, and since $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ are linearly independent we must then have

$$\rho\lambda = (1-t)\rho_0 \qquad \text{and} \qquad \rho\mu = t\rho_1,$$

from which it follows that

$$1 - t = \rho\lambda\rho_0^{-1} \qquad \text{and} \qquad t = \rho\mu\rho_1^{-1}.$$

Substituting this in to 6.4 and multiplying by $v_0^2 = \rho^{-2}$, we find

$$v_0^2 g(\boldsymbol{p}) = \lambda^2\rho_0^{-2}c_0 + 2\mu\lambda\rho_0^{-1}\rho_1^{-1}c_1 + \mu^2\rho_1^{-2}c_2.$$

By multiplying this by v_j for $j = 0, 1, 2$, and using the expansion 6.15, we get

$$\begin{aligned} v_0^2 g(\boldsymbol{p})v_j &= \lambda^3\rho_0^{-2}v_{0,j}c_0 + \mu\lambda^2(\rho_0^{-2}v_{1,j}c_0 + 2\rho_0^{-1}\rho_1^{-1}v_{0,j}c_1) \\ &\quad + \mu^2\lambda(2\rho_0^{-1}\rho_1^{-1}v_{1,j}c_1 + \rho_1^{-2}v_{0,j}c_2) + \mu^3\rho_1^{-2}v_{1,j}c_2 \\ &= \sum_{r=0}^{3}\mu^r\lambda^{3-r}\sum_{(i_1,i_2,i_3)\in\mathcal{P}_{r,3}} c_{i_1+i_2}v_{i_1,0}v_{i_2,0}v_{i_3,j}, \end{aligned}$$

for $j = 0, 1, 2$. The result follows by integrating this over $\theta \in [\theta_0, \theta_1]$. ■

Next, consider the integrals K_j. These depend both on the values of g and the directional derivative $D_{\boldsymbol{v}}g$. To find $D_{\boldsymbol{v}}g$, we first differentiate g in 6.4 along the half-edge $[\boldsymbol{r}_0, \boldsymbol{r}_1]$ to find $D_{\boldsymbol{e}}g$:

$$D_{\boldsymbol{e}}g(\boldsymbol{p}) = (1-t)\widehat{c}_0 + t\widehat{c}_1, \tag{6.16}$$

where $\boldsymbol{e}$ is the unit vector

$$\boldsymbol{e} = \frac{\boldsymbol{r}_1 - \boldsymbol{r}_0}{|\boldsymbol{r}_1 - \boldsymbol{r}_0|},$$

and

$$\widehat{c}_0 = \frac{2(c_1 - c_0)}{|\boldsymbol{r}_1 - \boldsymbol{r}_0|}, \qquad \widehat{c}_1 = \frac{2(c_2 - c_1)}{|\boldsymbol{r}_1 - \boldsymbol{r}_0|}.$$

Since $\boldsymbol{n}$ and $\boldsymbol{e}$ are orthogonal vectors, the directional derivative $D_{\boldsymbol{v}}g$ at $\boldsymbol{p}$ can be written as

$$D_{\boldsymbol{v}}g(\boldsymbol{p}) = (\boldsymbol{v}\cdot\boldsymbol{n})D_{\boldsymbol{n}}g(\boldsymbol{p}) + (\boldsymbol{v}\cdot\boldsymbol{e})D_{\boldsymbol{e}}g(\boldsymbol{p}). \tag{6.17}$$

Theorem 6.3 *Let g, $D_{\boldsymbol{n}}g$, and $D_{\boldsymbol{e}}g$ be given by 6.4, 6.6, and 6.16 respectively. Then*

$$\int_{\theta_0}^{\theta_1} v_0 D_{\boldsymbol{v}} g(\boldsymbol{p}) v_j \, d\theta = \sum_{r=0}^{3} L_r \sum_{(i_1,i_2,i_3)\in\mathcal{P}_{r,3}} (\boldsymbol{v}_{i_1} \cdot \widehat{\boldsymbol{d}}_{i_2}) v_{i_2,0} v_{i_3,j}, \quad j = 0,1,2,$$

where

$$\widehat{\boldsymbol{d}}_i = d_i \boldsymbol{n} + \widehat{c}_i \boldsymbol{e}, \qquad i = 0,1. \tag{6.18}$$

Proof (See [2]). In the proof of Theorem 6.2 we showed that

$$1 - t = \rho\lambda\rho_0^{-1} \qquad \text{and} \qquad t = \rho\mu\rho_1^{-1}.$$

Substituting this in equations 6.6 and 6.16, gives

$$\begin{aligned} D_{\boldsymbol{n}} g(\boldsymbol{p}) &= \rho(d_0\rho_0^{-1}\lambda + d_1\rho_1^{-1}\mu), \\ D_{\boldsymbol{e}} g(\boldsymbol{p}) &= \rho(\widehat{c}_0\rho_0^{-1}\lambda + \widehat{c}_1\rho_1^{-1}\mu). \end{aligned}$$

By using the expansion of $\boldsymbol{v}$ in 6.13 we get

$$\begin{aligned} \boldsymbol{v} \cdot \boldsymbol{n} &= \lambda(\boldsymbol{v}_0 \cdot \boldsymbol{n}) + \mu(\boldsymbol{v}_1 \cdot \boldsymbol{n}), \\ \boldsymbol{v} \cdot \boldsymbol{e} &= \lambda(\boldsymbol{v}_0 \cdot \boldsymbol{e}) + \mu(\boldsymbol{v}_1 \cdot \boldsymbol{e}). \end{aligned}$$

Substituting this into equation 6.17 and multiplying by $v_0 = \rho^{-1}$ gives

$$v_0 D_{\boldsymbol{v}} g(\boldsymbol{p}) = \sum_{r=0}^{2} \mu^r \lambda^{2-r} \sum_{(i_1,i_2)\in\mathcal{P}_{r,2}} (\boldsymbol{v}_{i_1} \cdot \widehat{\boldsymbol{d}}_{i_2}) v_{i_2,0}. \tag{6.19}$$

As in the proof of Theorem 6.2, we multiply this by v_j for $j = 0,1,2$, and use the expansion 6.15, which gives

$$v_0 D_{\boldsymbol{v}} g(\boldsymbol{p}) v_j = \sum_{r=0}^{3} \mu^r \lambda^{3-r} \sum_{(i_1,i_2,i_3)\in\mathcal{P}_{r,3}} (\boldsymbol{v}_{i_1} \cdot \widehat{\boldsymbol{d}}_{i_2}) v_{i_2,0} v_{i_3,j}, \quad j = 0,1,2.$$

Integrating this over $\theta \in [\theta_0, \theta_1]$ completes the proof. ■

6.2 Implementation

Assume that we are given a n-sided convex polygonal parameter domain Ω, and that $\boldsymbol{f}$ is piecewise quadratic on each edge of Ω. Given the function values $\boldsymbol{f}(\boldsymbol{p}_i)$, $\nabla\boldsymbol{f}(\boldsymbol{p}_i)$, $D_{\boldsymbol{n}_i}(\boldsymbol{q}_i)$ for $i = 1,\ldots,n$, we want to construct a patch $\boldsymbol{S}\colon \Omega \to \mathbb{R}^3$ that interpolate $\boldsymbol{f}$ on the boundary of Ω. As before, we start

by triangulate the domain polygon. Then we find the Bézier control points corresponding to each half-edge. For each point $\boldsymbol{x}$ in the triangulation we first find $\boldsymbol{v}_0$, $\boldsymbol{v}_1$ and σ for every half-edge $[\boldsymbol{r}_0, \boldsymbol{r}_1]$. Then we calculate $\cos\alpha$ and $\sin\alpha$ by using respectively the dot product and the planar cross product:

$$\cos\alpha = \boldsymbol{v}_0 \cdot \boldsymbol{v}_1, \qquad \sin\alpha = \boldsymbol{v}_0 \times \boldsymbol{v}_1.$$

Note that it's not necessary to compute the angle α. Now we can compute the values L_0,L_1,L_2,L_3 given in 6.10. Then we can compute the integrals given in Theorem 6.1, 6.2 and 6.3. By summing these values over all the half-edges we find the integrals in 6.1. Finally, we solve the linear system 5.14 using Cramer's rule.

6.3 Examples

In [16] the authors focused on applying the CMV interpolation method to shape- and image deformation problems. We will instead focus on constructing parametric patches with piecewise quadratic boundary curves.

Example 6.4 *In figure 6.3 we have plotted the interpolant to a quadratic polynomial on the unit square* $[-1,1] \times [-1,1]$. *As expected, the interpolant reproduces the quadratic function, confirming quadratic precision.*

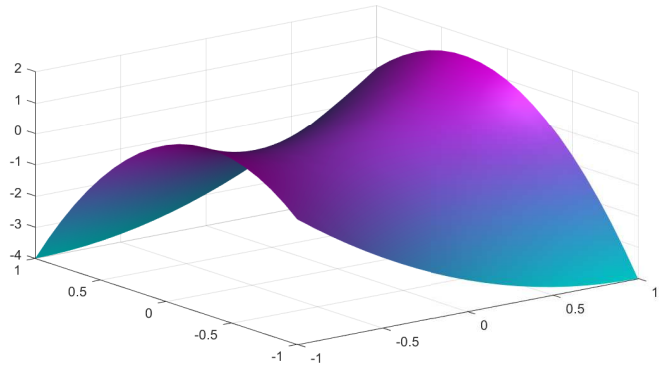


Figure 6.3: The interpolant to a quadratic polynomial

Example 6.5 (5-sided patch) *In example 5.3 we saw that numerical integration can lead to unwanted surface artifacts. In this example we have used the same boundary data* $\boldsymbol{P}_i$ *and* $\boldsymbol{T}_i$, *and parameter domain* Ω *as in example 5.3 to construct a 5-sided patch by using the closed form solution of the boundary integrals in the linear system 5.14.*

First we extracted the data $\boldsymbol{f}(\boldsymbol{p}_i)$, $\nabla \boldsymbol{f}(\boldsymbol{p}_i)$, $D_{\boldsymbol{n}_i}(\boldsymbol{q}_i)$ *for* $i = 1, \ldots, n$ *from the boundary functions* $\boldsymbol{P}_i$ *and the cross boundary derivative functions* $\boldsymbol{T}_i$ *as follows. We obviously have that* $\boldsymbol{f}(\boldsymbol{p}_i) = \boldsymbol{P}_i(0)$. *The gradient vector* $\nabla \boldsymbol{f}(\boldsymbol{p}_i)$ *was found by first finding the directional derivatives*

$$D_{\boldsymbol{e}_i}\boldsymbol{f}(\boldsymbol{p}_i) = \frac{\boldsymbol{P}_i'(0)}{||\boldsymbol{d}_i||} \qquad \textit{and} \qquad D_{\boldsymbol{e}_{i-1}}\boldsymbol{f}(\boldsymbol{p}_i) = \frac{\boldsymbol{P}_{i-1}'(1)}{||\boldsymbol{d}_{i-1}||},$$

where $\boldsymbol{d}_i = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$ *and* $\boldsymbol{e}_i = \frac{\boldsymbol{d}_i}{||\boldsymbol{d}_i||}$ *for all* i. *Then, by solving the linear system of equations*

$$\left[\boldsymbol{e}_i, \boldsymbol{e}_{i-1}\right] \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

we get

$$\frac{\partial \boldsymbol{f}}{\partial u}(\boldsymbol{p}_i) = aD_{\boldsymbol{e}_i}\boldsymbol{f}(\boldsymbol{p}_i) + bD_{\boldsymbol{e}_{i-1}}\boldsymbol{f}(\boldsymbol{p}_i).$$

In the same way, we found $\frac{\partial \boldsymbol{f}}{\partial v}(\boldsymbol{p}_i)$ *by first solving*

$$\left[\boldsymbol{e}_i, \boldsymbol{e}_{i-1}\right] \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

In this example the three normal derivatives

$$D_{\boldsymbol{n}_i}(\boldsymbol{p}_i), \quad D_{\boldsymbol{n}_i}(\boldsymbol{p}_{i+1}), \quad D_{\boldsymbol{n}_i}(\boldsymbol{q}_i).$$

was obtained from the given function $\boldsymbol{T}_i$. *By scaling this vector valued data, one can control the fullness of the interior of the patch. In this example all the normal derivatives were chosen to have length* 1 *which gave a good result. Figure 6.4 shows the result. As we can see, there are no visible artifacts.*

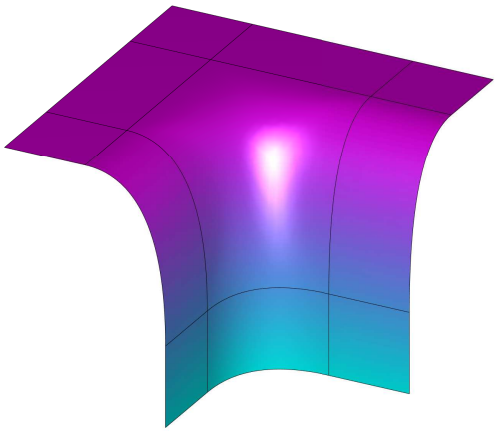


Figure 6.4: A 5-sided patch

Chapter 7

Summary and discussion

As we have seen, the ribbon based methods and the CMV method are quite different in construction. While each of the ribbon based patches can be defined by a single equation, the CMV interpolant is defined in a pointwise fashion. Both the ribbon based methods and the CMV method performed well in our examples, and it's hard to visually distinguish the surfaces from another. The CMV interpolant is always C^∞ smooth in the interior[10], while the ribbon based interpolants inherits the continuity of the boundary functions. The ribbon based surfaces connect to adjacent patches with either C^1 or G^1 continuity depending on the choice of parametrization scheme. The CMV interpolant also connects to adjacent surfaces with C^1 continuity (at least when the domain is a polygon), but can also be extended to match derivatives of arbitrary order.

Scaling the vector valued boundary derivatives can have huge influence over over the interior of the patch, and how to make good choices of scaling factors is a topic for future research. There also existes other ribbon based interpolants which gives more control over the interior of the patch, see for example [19, 23].

The CMV interpolant has the advantage that it can be applied to non convex and curved domains, while the ribbon based methods only works with convex polygonal domains. However, this is not a crucial property when constructing surface patches in 3D space, but is more important in for example shape and image deformation applications. Other applications include so called *guided subdivision* where the methods can be used to create a guide surface which is sampled at some interior points before subdivision is applied[13].

We showed that both the ribbon based patches and the CMV patch have roughly the same computational complexity when n is relatively low, but the CMV interpolant surpasses the other methods as n increases. However, we

saw that in some cases numerical quadrature can lead to unwanted artifacts, and we need to raise the number of points in the quadrature rule, which makes the method a bit slower. We also showed that we can avoid using numerical integration if the boundary functions are piecewise quadratic polynomials and the normal derivatives are piecewise linear along each edge of the domain.

Bibliography

[1] R. E. Barnhill, Garrett Birkhoff, and William J. Gordon. "Smooth interpolation in triangles". In: *Journal of Approximation Theory* 8.2 (1973), pp. 114–128.

[2] Rick Beatson, Michael S. Floater, and Carl Emil Kåshagen. "Hermite mean value interpolation on polygons". unpublished. 2017.

[3] Peter Charrot and John A. Gregory. "A pentagonal surface patch for computer aided geometric design". In: *Computer Aided Geometric Design* 1.1 (1984), pp. 87–94.

[4] Steven A. Coons. *Surfaces for computer-aided design of space forms*. DTIC Document, 1967.

[5] Germund Dahlquist and Ake Bjorck. *Numerical Methods in Scientific Computing: Volume 1*. SIAM, Jan. 1, 2008. 742 pp. ISBN: 978-0-89871-778-5.

[6] Christopher Dyken and Michael S. Floater. "Transfinite mean value interpolation". In: *Computer Aided Geometric Design* 26.1 (Jan. 2009), pp. 117–134. ISSN: 01678396.

[7] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. 4th ed. Academic Press, 1997. ISBN: 0-12-249054-1.

[8] Michael S. Floater. "Generalized barycentric coordinates and applications". In: *Acta Numerica* 24 (2015), pp. 161–214.

[9] Michael S. Floater. *Lecture notes in Generalized barycentric coordinates*. Aug. 2012.

[10] Michael S. Floater and Christian Schulz. "Pointwise radial minimization: Hermite interpolation on arbitrary domains". In: *Computer Graphics Forum*. Vol. 27. Wiley Online Library, 2008, pp. 1505–1512.

[11] John A. Gregory. "Smooth interpolation without twist constraints". In: *Computer Aided Geometric Design* (1974), pp. 71–87.

[12] John A. Gregory and Peter Charrot. “A C1 triangular interpolation patch for computer-aided geometric design”. In: *Computer Graphics and Image Processing* 13.1 (1980), pp. 80–87.

[13] Kestutis Karciauskas and Jörg Peters. *Guided subdivision*. 2005.

[14] Kiyokata Kato. “Generation of n-sided surface patches with holes”. In: *Computer-Aided Design* 23.10 (1991), pp. 676–683.

[15] Adi Levin. “Interpolating nets of curves by smooth subdivision surfaces”. In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1999, pp. 57–64.

[16] Xian-Ying Li, Tao Ju, and Shi-Min Hu. “Cubic mean value coordinates.” In: *ACM Trans. Graph.* 32.4 (2013), pp. 126–1.

[17] Mark Meyer et al. “Generalized barycentric coordinates on irregular polygons”. In: *Journal of graphics tools* 7.1 (2002), pp. 13–22.

[18] Péter Salvi. “Fair curves and surfaces”. PhD thesis. Ph. D. thesis, Eötvös Loránd University, Budapest, 2012.

[19] Péter Salvi, István Kovács, and Tamás Várady. “Computationally efficient transfinite patches with fullness control”. In: *Workshop on the Advances of Information Technology*. 2017, pp. 96–100.

[20] Péter Salvi and Tamás Várady. “Comparison of Two n-Patch Representations in Curve Network-Based Design”. In: *Proceedings of the 10th Conference of the Hungarian Association for Image Processing and Pattern Recognition*. 2015, pp. 612–624.

[21] Péter Salvi, Tamás Várady, and Alyn Rockwood. “Ribbon-based transfinite surfaces”. In: *Computer Aided Geometric Design* 31.9 (2014), pp. 613–630.

[22] Tamás Várady, Alyn Rockwood, and Péter Salvi. “Transfinite surface interpolation over irregular n-sided domains”. In: *Computer-Aided Design* 43.11 (2011), pp. 1330–1340.

[23] Tamás Várady, Péter Salvi, and Alyn Rockwood. “Transfinite surface interpolation with interior control”. In: *Graphical Models* 74.6 (2012), pp. 311–320.

[24] Wachspress. *A Rational Finite Element Basis*. Academic Press, Sept. 26, 1975. 348 pp. ISBN: 978-0-08-095623-7.