Change of variables

Jörg Peters

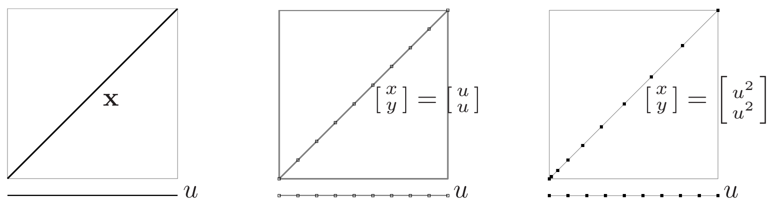
January 14, 2012

Changing variables is a useful tool that appears in many guises in computer graphics and geometric modeling. Changing the variables allows us to change the way we traverse a curve or surface, change their derivatives or place or interpret textures and other properties associated with them. In calculus, changing variables miraculously converts hard integration problems into simple standard ones and in algebra, seemingly hard root finding problems turn into an easy exercises. Even change of coordinates can be interpreted as a change of variables, albeit linear and therefore not our focus below.

Curves and Sampled curves For starters, let us look at how the choice of variables influences the speed at which we traverse a curve segment. For simplicity, we look at a curve segment that traverses the diagonal of the unit square (Fig. 1a). The diagonal, let us call it $\mathbf{x}_1(u)$, is the image of a unit domain interval, say $[0..1]$. That is the variable u is restricted to run from $u = 0$ to $u = 1$. In Fig. 1, this domain interval is placed just below the image curve; but of course domain and range really live in different spaces and the convention of elementary calculus, to combine domain and range into a 'graph', does not apply to computer graphics or geometric modeling: in graphics or geometric modeling, we only display the image curve or surface. Concretely, and formally, we define $\mathbf{x}_1$ as a map into the plane $\mathbb{R}^2$, as follows:

$$\mathbf{x}_1(u) := \begin{bmatrix} x \\ y \end{bmatrix}(u) := \begin{bmatrix} u \\ u \end{bmatrix}, \qquad u \in [0..1]. \tag{1}$$

Fig. 1b shows the image of $\mathbf{x}_1$ when we sample u at 10 equi-spaced points on the unit interval and mark each sample by a dot. Next we change the variable to $t := u^2$, i.e.



$\mathbf{x}$
u
$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u \\ u \end{bmatrix}$
u
$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u^2 \\ u^2 \end{bmatrix}$
u

(a) geometry (b) linear (c) non-linear

Figure 1: **Change of parameterization** (change of variables) does not change the exact continuous image – here the curve (a), but it does affect the kinematics, i.e. the speed of traversal (b) vs (c).

look at

$$\mathbf{x}_1(t) := \mathbf{x}_1(u^2) := \begin{bmatrix} u^2 \\ u^2 \end{bmatrix}, \qquad u \in [0..1]. \tag{2}$$

This yields still the same image diagonal, but Fig. 1c illustrates the different spacing of the images of the 10 equi-spaced points on the unit interval. We say, that the curve is traced out at a different speed and that the kinematic or motion representation of the curve is changed while the geometry of the curve is not.

Where speed matters In computer graphics we switch from the continuous exact mathematical formula to sampling the curve at specific locations of the domain, for example an evaluation grid. By convention the evaluated curve-points are connected by the simplest, namely linear pieces before rasterizing the image. Now the different speed leads to different samples, therefore to different linear approximation and ultimately to different rasterized geometry. For example, the circle approximation in Fig. 2b is clearly different from the one in Fig. 2c and both differ from the exact circle Fig. 2a. Both (b) and (c) are uniformly sampled, but the image of Fig. 2b also uniformly traces out the image as

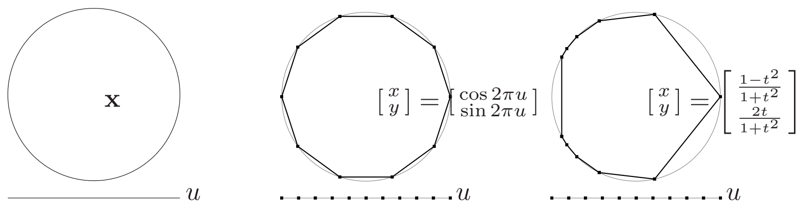
$$\mathbf{x}_2^\circ(u) := \begin{bmatrix} x \\ y \end{bmatrix}(u) := \begin{bmatrix} \cos 2\pi u \\ \sin 2\pi u \end{bmatrix}, \quad u \in [0..1], \tag{3}$$

since $\cos$ and $\sin$ are exactly the functions defined to translate the uniform motion on the circle to x, y coordinates. The second parameterization, shown in Fig. 2c,

$$\mathbf{x}_2(t) := \begin{bmatrix} \frac{1-t^2}{1+t^2} \\ \frac{2t}{1+t^2} \end{bmatrix}, \quad t \in (-\infty, \infty), \tag{4}$$

is the stereographic (rational) parameterization of the circle. As the samples get denser, both parameterizations $\mathbf{x}_2$ and $\mathbf{x}_2^\circ$ trace out arcs of the circle. This can be easily verified for $\mathbf{x}_2$ by checking that $x^2(t) + y^2(t) \equiv 1$, or by changing the variables as $t := \tan \frac{u}{2}$.

Arc-length We just saw for rendering that the choice of parameterization (variable) matters. Speed of traversal also matters in many other situations. Motion paths for virtual cameras or cutter paths for real-life milling machines require tight control on



$\mathbf{x}$
u
$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos 2\pi u \\ \sin 2\pi u \end{bmatrix}$
u
$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{1-t^2}{1+t^2} \\ \frac{2t}{1+t^2} \end{bmatrix}$
u
(a) exact geometry
(b) arclength uniform
(c) non-uniform

Figure 2: **Change of parameterization** (change of variables) does change the sampled geometry (b) vs (c). In (b) $u \in \{0, \frac{1}{10}, \ldots, 1\}$, in (c) $t := 8u - 4$.

the speed of the traversal. A key notion here is the (arc)length of the curve,

$$s = \int_a^b \sqrt{(x'(u))^2 + (y'(u))^2} du. \tag{5}$$

This is the length of the curve $\mathbf{x} := (x, y)$ as an ant would experience it if it started at $\mathbf{x}(a)$ and walked up to $\mathbf{x}(b)$. Here x' and y' are the first derivatives of the two coordinate functions, i.e. we assumed that $\mathbf{x}$ is differentiable. We can approximate the expression (5) by linear segments by sampling the curve as

$$s \approx \tilde{s} := \sum_{i=1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \tag{6}$$
$$x_j := x(a + j\frac{b-a}{n}),\ y_j := y(a + j\frac{b-a}{n}).$$

Famously, differentiability of $\mathbf{x}$ matters: An ever-finer staircase approximation to the diagonal $\mathbf{x}_1$, starting with the right angle $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ and continuing with $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1/2\\0\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1/2\\1/2\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1\\1/2\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, has always length 2, while the arclength of $\mathbf{x}_1$ from $u = 0$ to $u = 1$ is $\sqrt{2}$.

Tracing by arc-length. Generally, adjusting the step-length for motion paths to be uniform, requires numerical integration since the square root of a polynomial $(x'(u))^2 + (y'(u))^2$ is not usually a polynomial or has an otherwise simple formula. The subclass of polynomials that satisfies $(x'(u))^2 + (y'(u))^2 = f^2$, aptly called Pythagorean, has therefore been advertised for many applications [Far08]. However, one caveat applies. The subclass forms a non-linear space: taking the average of two Pythagorean parameterizations does not yield a Pythagorean parameterization.

Geometric continuity Where the resulting geometric shape is the overriding goal, as in geometric modeling, change of variables, or *re-parameterization*, offers flexibility to create richer classes of geometrically smooth curves formed from smoothly connected polynomial pieces. Such curves are often called splines and available in the openGL interface under the name gluNurbsCurve. We can take advantage of the invariance of the (mathematical, not sampled) curve pieces under a change of variables when connecting the pieces of a spline by allowing derivatives to agree only after a change of variables. This is natural as the example of the following two pieces of our diagonal shows:

$$\mathbf{x}_1^\ell(u) := \frac{1}{3}\mathbf{x}_1(u), \quad \mathbf{x}_1^r(u) := \frac{1}{3}\left[\begin{smallmatrix}1-u\\1-u\end{smallmatrix}\right] + \mathbf{x}_1(u), \qquad u \in [0..1]. \tag{7}$$

At their common end point $\frac{1}{3}\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, the pieces $\mathbf{x}_1^\ell$ and $\mathbf{x}_1^r$ match up with infinite smoothness (they are pieces of the same diagonal), even though their derivatives disagree:

$$(\mathbf{x}_1^\ell)'(1) = \frac{1}{3}\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right] \neq \frac{2}{3}\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right] = (\mathbf{x}_1^r)'(0). \tag{8}$$

But after doubling the speed of traversal of the left piece, $\mathbf{x}_1^\ell$ then $(\mathbf{x}_1^\ell(2u))'$ at $u = 1$ has the matching derivative $\frac{2}{3}\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$. In general, given a map f defined over the interval $[a, b]$ and a map g over $[c, d]$ so that $f(b) = g(c)$, we say that f and g join G^k, or with *kth order geometric continuity* if there exists a reparameterization $\mathbf{r} : \mathbb{R} \to \mathbb{R}$ to that with ∂^k denoting the kth derivative

$$(\partial^\kappa f)(b) = (\partial^\kappa(g(\mathbf{r})))(b), \quad \kappa = 1..k, \quad \mathbf{r}(b) = c. \tag{9}$$

In example (8) above, $k = 1$ and $\mathbf{r}(u) := 2u$. Indeed, for $k = 1$, (9) simplifies by the chain rule to

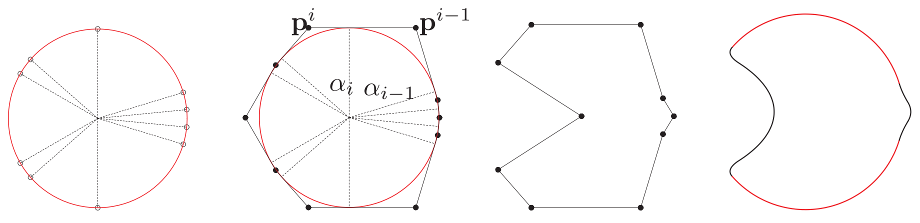
$$f'(b) = (\partial^1 f)(b) = (\partial^1 g(\mathbf{r}))(b) = \mathbf{r}'(b)g'(\mathbf{r}(b)) = \beta g'(c), \tag{10}$$

i.e. the tangent directions $f'(b)$ and $g'(c)$ have to be parallel and the derivatives are stretched by a factor β to make them agree. Of course we have to be careful not to flip the orientation and end up with a cusp. So we require $\beta > 0$. Indeed, we have to be even more careful and require that derivatives of f and g do not vanish at the common point.

$$\mathbf{x}_3^\ell(u) := \left[\begin{smallmatrix}-1\\1\end{smallmatrix}\right](1-u)^2, \quad \mathbf{x}_3^r(u) := \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]u^2 \tag{11}$$

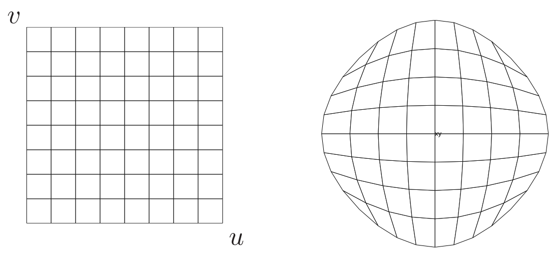
have identical derivatives at their common point $(0, 0)$ corresponding to $u = 0$ but clearly, the two line segments do not smoothly connect. Such examples explain why differential geometry texts typically start by requiring *regularity* of the curves involved where regularity means that the first derivatives are not zero at the point of interrogation. Of course, non-regular curves may still join smoothly, e.g. $\left[\begin{smallmatrix}-1\\-1\end{smallmatrix}\right](1-u)^2$ and $\mathbf{x}_3^r(u)$, but we just cannot base the analysis on the first derivatives.

Geometric continuity We can take advantage of the invariance of the curve under a change of variables when joining curve pieces into a spline. The construction of cubic C^2 curves that include conic sections, such as the circle (see Fig. 3), would just not be possible without geometric continuity.



(a) designer (b) algorithm (c) designer (d) algorithm

Figure 3: Design work-flow for modeling with conic pieces (from [KP11]). (a) The designer partitions the circle to isolate sections that will be modified; (b) the algorithm creates the control net of points $\mathbf{p}^i$ connected by line segments; (c) the designer modifies the net; (d) the algorithm creates the curve so that (the red) portions remain on the circle.



(a) domain (b) image

Figure 4: A bi-quadratic map in two variables: domain (a) to image (b).

Maps in two variables We now look at re-parameterizations in two variables, i.e. maps from the plane to the plane: $\mathbf{r} : \mathbb{R}^2 \to \mathbb{R}^2$. These maps deform flat regions as illustrated in Fig. 4. Changing continuously from the image of the identity map id to the image of $\mathbf{r}$ is also called 'morphing'. The example in Fig. 4 was obtained as a polynomial map in Bernstein-Bézier form in two variables:

$$\mathbf{r} := \sum_{i=0}^{d}\sum_{j=0}^{d} \mathbf{c}_{ij} B_i(u) B_j(v) \quad B_k(t) := \binom{d}{k}(1-t)^{d-k}t^k, \quad u, v \in [0..1]. \tag{12}$$

This mapping is called glMap2 in openGL and is a generalization from a single variable by 'tensoring' which, for our purposes, means that for every v fixed, we get a curve of type $\sum_{i=0}^{d} \tilde{\mathbf{c}}_i B_i(u)$ in u and vice versa. Specifically, the map $\mathbf{r}$ in Fig. 4 has $d = 2$ and the coefficients

$$\mathbf{c} := [\mathbf{c}_{ij}]_{i=0,1,2;j=0,1,2} := \begin{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 2 \end{bmatrix} & \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \begin{bmatrix} -2 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 2 \\ 0 \end{bmatrix} \\ \begin{bmatrix} -1 \\ -1 \end{bmatrix} & \begin{bmatrix} 0 \\ -2 \end{bmatrix} & \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \tag{13}$$

Iso-para-geo maps Maps $\mathbb{R}^2 \to \mathbb{R}^2$ can be used to describe a region Ω of interest as the image of a standard region $\Box$, here the unit square. In the finite element literature, the region of interest can be a *physical domain*. A physical domain is a possible very irregular region on which a function is to be computed. For example, we might compute pressure on the region in Fig. 4b with pre-image the unit square $\Box = [0..1]^2$. The point is that both pressure and the shape of the region can be represented a maps with $\Box$ as the domain. This approach is the *iso-parametric method*. More recently, to emphasize that Ω is not described as a union of triangles but as the image of a collection of genuinely non-linear maps, the approach is also called the isogeometric method. The underlying principle (change of two variables) is available in standard OpenGL in the context of texture mapping as we will see next.

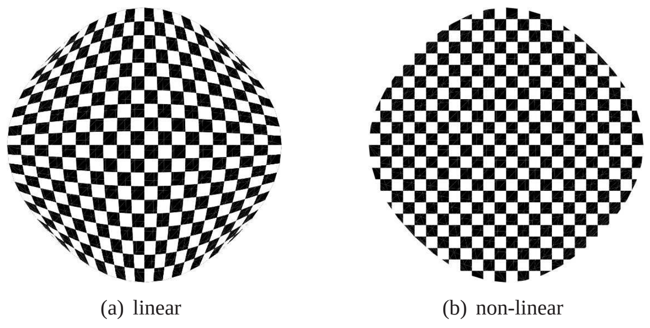


Figure 5: Two different bi-quadratic mappings of a texture. (a) preserves the number of black and white regions, (b) preserves angles and relative size.

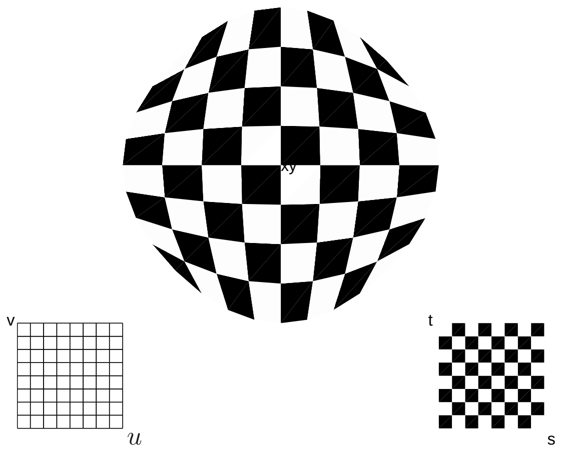


Figure 6: Texture mapping: domain □ in (u, v), physical domain in (x, y) and black-white valued function in (s, t) mapped to the physical domain.

Texture mapping In computer graphics, a change of variables can affect how an image is attached to a surface. The two images in Fig. 5 are both checkerboard textures attached to $\mathbf{r}$. While Fig. 5b preserves the right angles and the size of the black and white mini-squares by cutting or trimming the checker board as with a cookie cutter, Fig. 5a is the result of squeezing all available mini-squares into the image and thereby distorting both angles and size. The distortion is the result of attaching the texture in Fig. 5a with texture coordinates that are the identity map, a far too common choice in openGL sample programs. By contrast, the texture in Fig. 5b uses $\mathbf{r}$ as the texture map. A sample program is available at XXYY. The use of splines in two variables as texture coordinate functions is neglected due to the prevalence of facetted, triangulated models. There the change of variables is called 'setting texture coordinates' and corresponds to defining a piecewise linear spline that is defined by just specifying the (s,t) texture coordinates of the vertices of the triangulation. The texture mapping mechanism used by OpenGL is illustrated in Fig. 6: a texture coordinate map maps u, v parameters to the checkerboard texture in its s, t parameters and the corresponding (here black or white, in general RGB) color values RGB$(s(u,v), t(u,v))$ are attached to a spline patch at $\mathbf{x}(u,v)$.

So, if we let glMap2 (12) determine a region that plays the role of the physical domain and think of the texture as the function on this physical domain, we have exactly the setting of the non-linear iso-parametric or iso-geometric method. And if we do the same by setting texture coordinates just for the vertices of a triangulation, we are in the piecewise linear setting of the iso-parametric method.

Smoothly joining patches Analogous to the 1-variable case, change of two variables increases flexibility when modeling and smoothly glueing together curved surface pieces, called *patches* into smooth compound objects. With $\mathbf{r} : \mathbb{R}^2 \to \mathbb{R}^2$, of some sufficiently smooth maps f and g and ∂^k denoting the kth partial derivative f and g joining along a curve $f(B) = g(C)$ join with k order continuity G^k if (cf. [Pet02])

$$(\partial^\kappa f)(b) = (\partial^\kappa (g(\mathbf{r})))(c), \quad \kappa = 1..k. \tag{14}$$

In fact, in two variables, this change of variables is necessary if we want to model arbitrary smooth objects without awkward spots called singularities. For example, an object that is a smooth deformation of the sphere must have a singularity (or two) called pole(s) where the unique parameterization breaks down. Such singularities have to appear in ever increasing numbers once we attach more than one 'handle' to create a smooth object with several holes (see Fig.). Only deformations of the torus, an object with one handle, have a simple, periodic parameterization over a rectangular domain. *Subdivision surfaces* [DKT98, PR08] have many singularities, called extraordinary points and can thereby avoid change of variables. (Although, when computing with them near extraordinary points, one should still switch to local coordinates in terms of their so-called characteristic spline or map.)

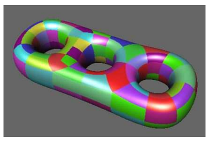


Figure 8: Object with 3 handles.

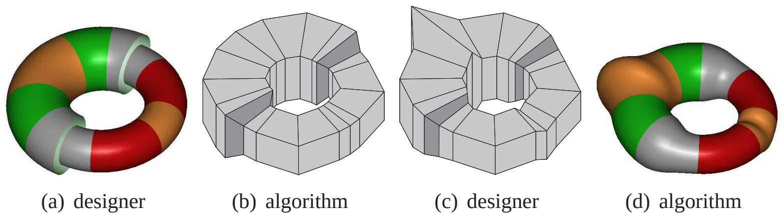


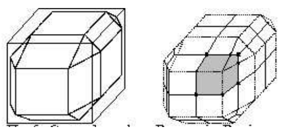
Figure 7: Surface design work-flow (from [KP11]). (a) designer marks regions of the two torus halves. The regions will serve as either transitions (gray), for later modification (orange) or remain on the torus (green, red). The modification regions are partitioned into three equal sub-sectors, transition regions into two. (b) The algorithm creates the control net. (c) The designer modifies the net. (d) The algorithm creates the surface. A sample program is available XXYY.

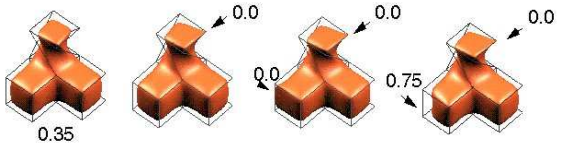
If we do change variables, this corresponds, in the language of differential geometry, to changing between the charts of an atlas. Just as in one variable such change of variables allows for more flexible surface constructions and spline surfaces of lower polynomial degree. Fig. 7 shows the surface analogue of Fig. 3 which allows inclusion of torus or sphere pieces etc. into a smooth compound surface.

Spline surfaces and semi-smooth creases Change of variables in the form of geometric continuity and change of variables for texture mapping come together in an construction of semi-smooth creases that is a precursor of the famous Pixar rules for semi-smooth creases in subdivision [DKT98]. While the subdivision rules average a curve subdivision scheme with a surface subdivision scheme, surface splines [Pet95] modeled different amounts of curvature when blending patches by allocating more or less space for rounding via so-called *blend ratios* that pull the surface towards its control cage, Fig. 9b. (A third alternative, prevalent in the CAD industry, is to trim the patches, i.e. to restrict their domain, and then fit a fillet, e.g. a 'rolling ball blend', to fill the resulting gap.) One potential drawback of the approach via blend ratios is that as the transition becomes sharper, the image of the domain is squeezed into ever less space. For example, our checker texture in Fig. 5 would be squeezed differently according to how much space is given at either end. But, by choosing the texture coordinates intelligently as in Fig. 5b one can trade off between preserving right angles or equal size and the deformed texture.

References

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of the ACM Conference on Computer Graphics (SIGGRAPH-98)*, pages 85–94, New York, July 19–24 1998. ACM Press.


(a) biquadratic control points via corner cutting



(b) local blend ratios determine curvature

Figure 9: (a) Cutting corners and edges to obtain the 9 coefficients per patch (13). (b) Surfaces obtained by locally varying the parameterization.

[Far08] Rida Farouki. *Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable*. Geometry and Computing 1. Springer Verlag, 2008.

[KP11] K. Karčiauskas and J. Peters. Modeling with rational biquadratic splines. In C. Bajaj, M-S. Kim, and S. Hahmann, editors, *SIAM GD/SPM conference*, pages 1–10, 2011.

[Pet95] J. Peters. C^1-surface splines. *SIAM J. Numer. Anal.*, 32(2):645–666, 1995.

[Pet02] J. Peters. Geometric continuity. In *Handbook of Computer Aided Geometric Design*, pages 193–229. Elsevier, 2002.

[PR08] J. Peters and U. Reif. *Subdivision Surfaces*, volume 3 of *Geometry and Computing*. Springer-Verlag, New York, 2008.