Nonlinear Subdivision Through Nonlinear Averaging

S. Schaefer, E. Vouga, R. Goldman

We investigate a general class of nonlinear subdivision algorithms for functions of a real or complex variable built from linear subdivision algorithms by replacing binary linear averages such as the arithmetic mean by binary nonlinear averages such as the geometric mean. Using our method, we can easily create stationary subdivision schemes for Gaussian functions, spiral curves, and circles with uniform parametrizations. More generally, we show that stationary subdivision schemes for $e^{p(x)}$, $cos(p(x))$ and $sin(p(x))$ for any polynomial or piecewise polynomial $p(x)$ can be generated using only addition, subtraction, multiplication, and square roots. The smoothness of our nonlinear subdivision schemes is inherited from the smoothness of the original linear subdivision schemes and the differentiability of the corresponding nonlinear averaging rules. While our results are quite general, our proofs are elementary, based mainly on the observation that generic nonlinear averaging rules on a pair of real or complex numbers can be constructed by conjugating linear averaging rules with locally invertible nonlinear maps. In a forthcoming paper we show that every continuous nonlinear averaging rule on a pair of real or complex numbers can be constructed by conjugating a linear averaging rule with an associated continuous locally invertible nonlinear map. Thus the averaging rules considered in this paper are actually the general case. As an application we show how to apply our nonlinear subdivision algorithms to intersect some common transcendental functions.

1. Introduction

Subdivision is a standard technique in Geometric Modeling for generating smooth shapes from coarse data. The de Casteljau algorithm for Bezier curves [5], the Lane-Riesenfeld algorithm for uniform B-splines [12], the four-point scheme for smooth interpolatory curves [7], [6], the Catmull-Clark algorithm for rectangular meshes [3], and the Loop algorithm for triangular meshes [13] are all examples of well-known subdivision procedures for generating smooth freeform curves and surfaces from piecewise linear functions. However, the only common functions that these subdivision schemes are capable of reproducing are polynomial and piecewise polynomial functions. The goal of this paper is to explore a new class of nonlinear subdivision schemes based on nonlinear averages and show how these nonlinear schemes can be used to extend subdivision to a much wider class of functions, including some common transcendental and piecewise transcendental functions.

A *stationary subdivision* scheme is a recursive technique defined by a set of rules encoded in a single operator S. Given a set of control points p^0, stationary subdivision generates

new sets of control points by setting

$$p^k = S(p^{k-1}) = S^k(p^0) \qquad k = 1, 2, \ldots$$

Here p^k denotes the control points produced after k levels of subdivision, and S^k denotes the operator S applied k times. When the subdivision rule S is well chosen, then connecting adjacent points at level k with straight lines produces a sequence of piecewise linear functions that converge in the limit to some smooth shape which we denote by

$$p^\infty = S^\infty(p^0).$$

If the subdivision rules change from level to level, then the subdivision scheme is said to be *nonstationary*. The techniques we shall develop in this paper are valid both for stationary and nonstationary subdivision schemes. Nevertheless, to simplify both the discussion and the notation, we shall restrict our attention here to stationary subdivision procedures.

Most of the standard stationary subdivision schemes that exist today are linear subdivision schemes. A *linear subdivision* scheme is a scheme where the rules encoded in the subdivision operator are defined by affine combinations of the control points. Typically, linear subdivision rules are represented locally by multiplying the control points p with a matrix L. Thus we write

$$p^{k+1} = S(p^k) = Lp^k.$$

When the columns of the matrix L are uniform shifts of a single column, then the subdivision scheme is said to be *uniform*. For uniform subdivision schemes, the smoothness of the limit functions can be determined by analyzing the eigenvalues and eigenvectors of the matrix L [17].

Nonlinear subdivision schemes have been introduced recently for three reasons: to eliminate artifacts, to preserve shape, and to generate smooth curves on manifolds [8]. We shall briefly review each of these applications in turn and then compare our method and motivation to the approaches and goals of previous authors.

Several nonlinear variants have been proposed for the classical linear four-point interpolatory subdivision scheme. The classical four-point scheme is known to generate well-behaved curves when the distances between adjacent control points do not vary too much. But when some edges of the control polygon are short and others are long, then artifacts that are not in the data such as self intersections and inflection points may appear in the curve. To eliminate these artifacts, Marinov *et al* [14] introduce a tension parameter into the four-point scheme and then adapt the tension parameter to the data. This approach leads to a nonlinear version of the four-point scheme. Another approach to nonlinear subdivision in the four-point scheme is provided by Sabin and Dogson [18], who introduce nonlinearity to reproduce circles and to reduce curvature variation for data off the circle.

Cohen *et al* [4] modify the four-point scheme by developing several alternative sets of masks and then choosing the mask for each new point depending on the nearby data. Again these rules for choosing the appropriate masks depending on the data introduce nonlinearity into the four-point scheme. In the functional setting, when the data is taken off a smooth function, this approach for choosing the masks leads to better estimates and fewer oscillations in the curves.

Floater and Micchelli [9] generate a family of nonlinear interpolatory subdivision schemes by introducing the harmonic mean to modify the classical four-point interpolatory subdivision algorithm in order to build an interpolatory subdivision procedure that preserves the convexity of the original data. Micchelli [15] also introduces a modification of the four-point scheme capable of reproducing any function of the form $p(x)e^{l(x)}$, where $p(x)$ is a polynomial and $l(x)$ is a linear function.

More recently Aspert *et al* [2] modify the four-point scheme in a nonlinear fashion in order to generate a subdivision scheme for curves that can be adapted readily to surfaces. Their scheme can reproduce circles, but they provide no general criteria for guaranteeing smoothness.

Finally, nonlinear subdivision schemes are typically required when the goal is to create smooth shapes from coarse data on curved manifolds [16], where affine combinations no longer apply. The smoothness of these nonlinear subdivision schemes on curved manifolds is generally a good deal more difficult to analyze than the smoothness of linear subdivision schemes in Euclidean space [21].

In this paper we are going to introduce a simple, systematic approach to generating nonlinear subdivision schemes by replacing the arithmetic mean (midpoint averaging) by simple nonlinear averaging rules such as the geometric mean in standard linear subdivision algorithms such as the Lane-Riesenfeld algorithm for uniform B-splines. Unlike other researchers whose goals are to use nonlinearity either to eliminate artifacts, preserve shape, or generate smooth curves on manifolds, our goal is to take advantage of nonlinearity to generate a much wider class of functions than is possible to generate with linear subdivision algorithms. Using our method, we will build simple stationary subdivision schemes for Gaussians, spirals, and circular arcs for which no known stationary linear subdivision schemes exist. We will also show that subdivision schemes for $e^{p(x)}, \cos(p(x)), \sin(p(x))$ for any polynomial or piecewise polynomial $p(x)$ can be generated using only addition, subtraction, multiplication, and square roots.

We begin in Section 2 with a brief review of three of the most popular linear subdivision schemes: the de Casteljau subdivision algorithm for Bezier curves, the Lane-Riesenfeld subdivision algorithm for uniform B-splines, and the four-point subdivision rule for generating smooth interpolatory curves. Each of these schemes is based either on repeated midpoint averaging or on repeated two point linear interpolation. In Section 3 we provide a systematic way to modify these subdivision algorithms as well as other linear subdivision schemes to generate an interesting new class of nonlinear subdivision procedures by altering the meaning of averaging or, more generally, of linear interpolation (weighted averages). We show that these nonlinear subdivision schemes based on nonlinear averages generate smooth functions, typically as smooth as the functions generated by the linear subdivision schemes from which they are derived. We also present an elementary, geometric interpretation of these nonlinear subdivision algorithms in terms of geodesic averaging, and we reinterpret the convex hull and variation diminishing properties in terms of these nonlinear geodesics. Using this geometric interpretation, we then go on to investigate intersection algorithms for transcendental functions based on these nonlinear subdivision algorithms and their corresponding nonlinear geodesics. We close in Section 4 with a short summary of our work and a few open questions for future research.

2. Linear Subdivision Algorithms: Midpoint Averaging and Linear Interpolation

The simplest bounded curve is a straight line segment, and the easiest subdivision algorithm is midpoint averaging. Two points P_0, P_1 determine a line segment P_0P_1. The midpoint M of the line segment P_0P_1 is located at the average of the two end points – that is,

$$M = \frac{P_0 + P_1}{2}. \tag{1}$$

Equation 1 is called *midpoint averaging*. Iterating midpoint averaging on the two line segments P_0M and MP_1 generates a dense set of points along the original line segment P_0P_1 (see Figure 4, top).

Midpoint averaging provides a simple subdivision algorithm for straight line segments. Repeated midpoint averaging is also at the heart of several classical subdivision algorithms for more complicated curves. The de Casteljau subdivision algorithm for Bezier curves and the Lane-Riesenfeld subdivision algorithm for uniform B-splines are both based on repeated midpoint averaging.

In the de Casteljau algorithm for Bezier curves, we start with a sequence of control points $P_0, \ldots, P_n$ for a Bezier curve of degree n. Using midpoint averaging, we then compute a triangular array of points P_j^k, $j = 0, \ldots, n-k$, $k = 0, \ldots, n$ by setting

$$\begin{aligned} P_j^0 &= P_j \qquad\qquad j = 0, \ldots, n \\ P_j^k &= \tfrac{P_j^{k-1} + P_{j+1}^{k-1}}{2} \quad j = 0, \ldots, n-k \quad k = 1, \ldots, n. \end{aligned} \tag{2}$$

The points $P_0^0, \ldots, P_0^n$ and $P_0^n, \ldots, P_n^0$ are control points for the left and right segments of the original Bezier curve. Iterating the de Casteljau subdivision algorithm on these two sets of control points generates a sequence of points that converge to a dense set of points along the Bezier curve for the original control points $P_0, \ldots, P_n$ (see Figure 1, top).

In the Lane-Riesenfeld subdivision algorithm for uniform B-splines, we are given a sequence of control points $P_0, \ldots, P_n$ for a B-spline curve of degree d with knots at the integers. We begin the subdivision algorithm by doubling the control points, and then use midpoint averaging to compute new control points in the following manner:

$$\begin{aligned} P_{2j}^0 &= P_j \\ P_{2j+1}^0 &= P_j \\ P_j^k &= \tfrac{P_j^{k-1} + P_{j+1}^{k-1}}{2}. \end{aligned} \tag{3}$$

The points $P_0^d, \ldots, P_{2n+1-d}^d$ are the control points for the original B-spline curve of degree d with knots at the half integers. Iterating the Lane-Riesenfeld subdivision algorithm generates a sequence of points that converge to a dense set of points along the B-spline curve for the original control points $P_0, \ldots, P_n$ (see Figure 2, top).

Midpoint averaging is a special case of linear interpolation or weighted averaging. In weighted averaging, instead of placing M at the midpoint of the line segment P_0P_1, we place M at the point along the line segment that splits the original segment into two segments whose lengths are in the ratio $\frac{\lambda}{1-\lambda}$. Thus

$$M = (1 - \lambda)P_0 + \lambda P_1. \tag{4}$$

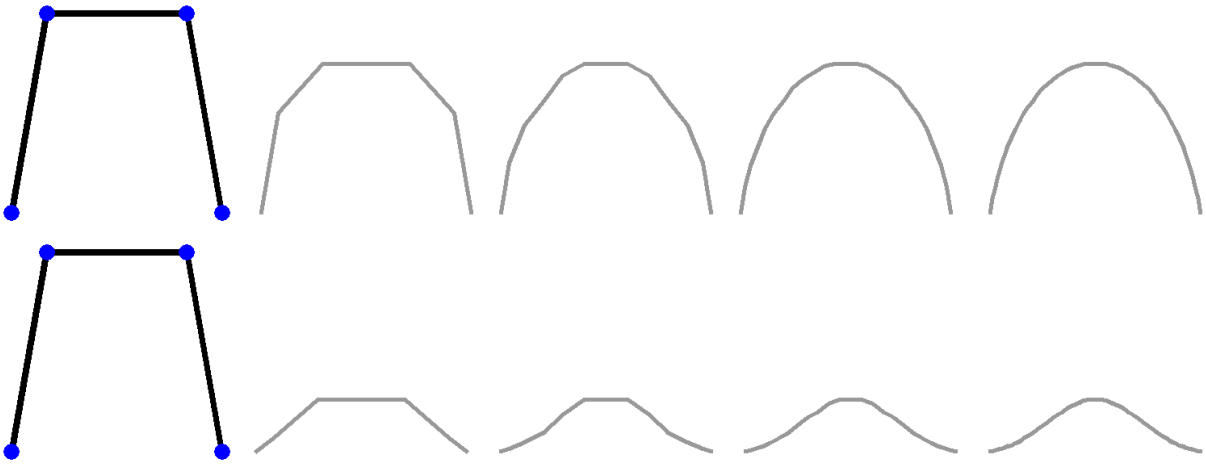

Figure 1. A polynomial function (Bezier curve) generated by the de Casteljau subdivision algorithm using the arithmetic mean (Top), and a function generated by the de Casteljau subdivision algorithm starting with the same data but using the geometric mean (Bottom). Here we illustrate the functional case.

If we replace midpoint averaging by linear interpolation at some fixed value $0 < \lambda < 1$ in the de Casteljau subdivision algorithm for Bezier curves, we still get a valid subdivision algorithm for Bezier curves. Moreover, every linear subdivision algorithm for curves can be factored into a sequence of steps using only two point linear interpolation, though this decomposition is not unique [8]. For example, one level of the well-known four-point subdivision scheme is typically written as

$$\begin{aligned} P_{2j}^{new} &= P_j^{old} \\ P_{2j+1}^{new} &= \frac{-1}{16}P_{j-1}^{old} + \frac{9}{16}P_j^{old} + \frac{9}{16}P_{j+1}^{old} + \frac{-1}{16}P_{j+2}^{old}. \end{aligned}$$

This rule can be factored into the following sequence of two point linear interpolations:

$$\begin{aligned} P_{2j}^0 &= \tfrac{3}{2}P_j - \tfrac{1}{2}P_{j+1} \\ P_{2j+1}^0 &= \tfrac{3}{2}P_{j+1} - \tfrac{1}{2}P_j \\ P_j^k &= \tfrac{P_j^{k-1}+P_{j+1}^{k-1}}{2} \quad k = 1, 2, 3 \end{aligned}$$

Iterating the four-point subdivision algorithm generates a sequence of points $P_0^3, \ldots, P_{2n-5}^3$ that interpolate the original control points and converge to a dense set of points along a smooth curve (see Figure 3, top).

3. Nonlinear Subdivision Algorithms from Nonlinear Averages

We are now going to provide a systematic way to modify linear subdivision algorithms based either on repeated midpoint averaging or on repeated two point linear interpolation

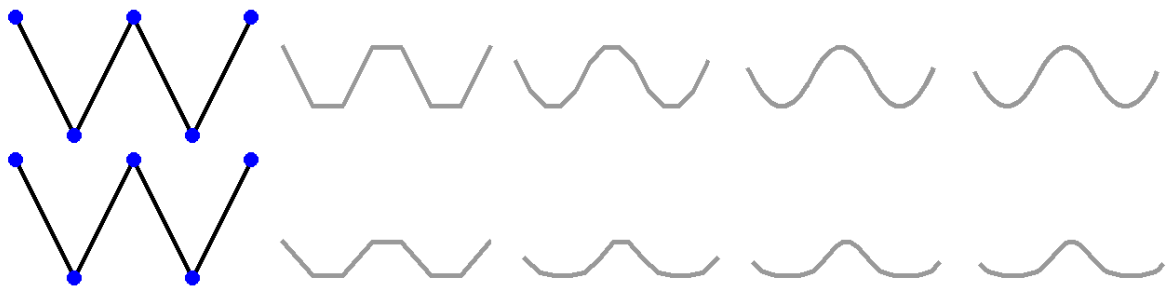

Figure 2. A spline function generated by the Lane-Riesenfeld subdivision algorithm using the arithmetic mean (Top), and a function generated by the Lane-Riesenfeld subdivision algorithm starting with the same data but using the geometric mean (Bottom). Again we illustrate the functional case.

to generate a new class of nonlinear subdivision procedures. We will begin with a purely algebraic approach: replacing the arithmetic mean by the geometric mean in standard subdivision algorithms based on midpoint averaging. We shall observe that, somewhat surprisingly, this new averaging strategy still generates smooth functions. Next we will show that subdivision based on the geometric mean actually arises quite naturally from a functional equation for exponentials. We will then invoke other functional equations to further generalize the meaning of midpoint averaging in order to develop some additional novel nonlinear subdivision schemes. To explain why these new nonlinear subdivision methods based on novel averaging techniques always generate smooth functions, we will study the effect of conjugating linear subdivision algorithms with locally invertible nonlinear maps. After explaining the basic theory underlying our general approach, we shall present some examples to flesh out the theory. We will also provide an equivalent geometric interpretation for our new nonlinear subdivision methods based on geodesic midpoint averaging or geodesic linear interpolation, where we change the geodesics but not the underlying geometry of Euclidean space.

3.1. The Algebraic Approach: Changing the Meaning of Averaging

Consider a sequence $c_0, \ldots, c_n$ of positive real numbers. We can graph these values in the xy-plane by interpreting these constants as y-coordinates over the integers $0, \ldots, n$ along the x-axis. Applying one level of the de Casteljau subdivision algorithm to this functional data, we generate a new sequence of real numbers, which we graph over the half integers. If we iterate this procedure, then in the limit we generate the polynomial

$$y = \sum_{k=0}^{n} c_k B_k^n(x),$$

where $B_0^n(x), \ldots, B_n^n(x)$ are the degree n univariate Bernstein basis functions for the interval $[0, n]$. If instead of the de Casteljau subdivision algorithm, we iterate d rounds of averaging in the Lane-Riesenfeld algorithm, then in the limit we would generate the

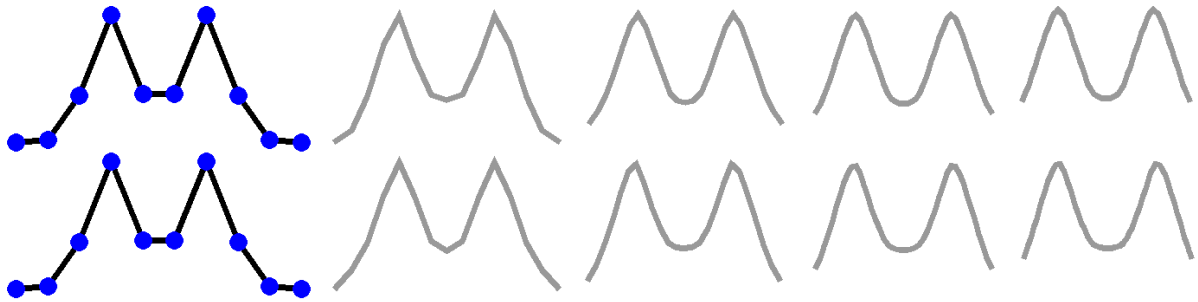

Figure 3. A smooth function generated by the classical four-point scheme using the linear interpolation formula $(1-\lambda)a+\lambda b$ (Top), and a function generated by the four-point scheme starting with the same data but using the analogue $a^{1-\lambda}b^{\lambda}$ of linear interpolation for the geometric mean (Bottom). As in Figures 1 and 2, we illustrate here the functional case.

spline

$$y=\sum_{k=0}^{2n+1-d} c_k N_k^d(x),$$

where $N_0^d(x),\ldots,N_{2n+1-d}^d(x)$ are the degree d uniform B-spline basis functions.

Both the de Casteljau subdivision algorithm for Bezier curves and the Lane-Riesenfeld algorithm for uniform B-splines are based on repeatedly averaging pairs of real numbers. But there is more than one way to compute the average of two real numbers. Suppose that instead of the arithmetic mean $\frac{(a+b)}{2}$, we were to use the geometric mean $\sqrt{ab}$. Starting with the same positive real numbers $c_0,\ldots,c_n$ and iterating either the de Casteljau subdivision algorithm or the Lane-Riesenfeld algorithm with the geometric mean in place of the arithmetic mean, we again generate a function $y=g(x)$. Moreover, we observe that the functions we generate with these subdivision algorithms using the geometric mean are smooth, at least as smooth as the corresponding functions generated with the standard versions of these algorithms using the arithmetic mean (see Figures 1 and 2).

We can also extend the four-point scheme by replacing the linear interpolation (weighted average) formula $(1-\lambda)a+\lambda b$ by the analogue $a^{1-\lambda}b^{\lambda}$ of linear interpolation for the geometric mean. Figure 3 illustrates that once again we get smooth functions.

These observations naturally raise the following questions:

- Do we always get smooth functions if we replace the arithmetic mean by the geometric mean in linear subdivision algorithms?
- Precisely what functions do we generate when we replace the arithmetic mean by the geometric mean in linear subdivision algorithms?
- Are there other averaging techniques that also always lead to smooth functions?
- What functions do these alternative averaging techniques generate?

The remainder of this section is devoted to answering these four questions.

3.2. Averaging Rules and Functional Equations

The study of linear subdivision starts with linear functions; to investigate nonlinear subdivision, we will begin with exponential functions. Any linear function in the plane can be expressed in the form $y = mx+b$. Therefore two parameters – m and b – determine a unique line; thus two points determine a unique line. Similarly, the exponential of a linear function in the plane is expressed in the form $y = e^{\lambda x+\mu}$. Therefore two parameters – λ and μ – determine a unique exponential; thus two points also determine a unique exponential!

To subdivide a segment of a straight line $L(x) = mx + b$, we fix two end points $L(x_0)$ and $L(x_1)$ and compute their midpoint

$$\begin{aligned} L\left(\tfrac{x_0+x_1}{2}\right) &= m\left(\tfrac{x_0+x_1}{2}\right) + b \\ &= \left(\tfrac{mx_0+b}{2} + \tfrac{mx_1+b}{2}\right) \\ &= \tfrac{L(x_0)+L(x_1)}{2}. \end{aligned}$$

Similarly, given an exponential function $F(x) = e^{\lambda x+\mu}$, we can fix two end points $F(x_0)$ and $F(x_1)$ of a segment along the exponential curve. To find the point on the exponential function midway along the x-axis between these two end points, we must compute

$$F\left(\frac{x_0 + x_1}{2}\right) = e^{\frac{\lambda(x_0+x_1)}{2}+\mu} = e^{\frac{\lambda x_0+\mu}{2}} e^{\frac{\lambda x_1+\mu}{2}} = \sqrt{F(x_0)F(x_1)}. \tag{5}$$

The right hand side of Equation 5 represents the geometric mean $GM(x_0, x_1)$ of $F(x_0)$ and $F(x_1)$. Iterating the geometric mean on the two exponential segments defined by the pairs of points $F(x_0), GM(x_0, x_1)$ and $GM(x_0, x_1), F(x_1)$ generates a dense set of points along the original exponential function $y = F(x) = e^{\lambda x+\mu}$ (see Figure 4, bottom). Thus to subdivide these exponential functions, we simply replace midpoint averaging by the square root of the product – that is, we replace the arithmetic mean by the geometric mean.

Now comes the crucial insight. We observed in Section 2 that repeated midpoint averaging is at the heart of several classical subdivision algorithms for freeform curves, including the de Casteljau subdivision algorithm for Bezier curves and the Lane-Riesenfeld subdivision algorithm for uniform B-splines. In light of what we have just observed concerning subdivision for exponential functions, it seems natural to ask: what would happen if we were to replace midpoint averaging by the geometric mean in each of these classical subdivision algorithms? That is, what kinds of functions would we generate if we were to replace the arithmetic mean by the geometric mean in Equations 2 and 3? Figures 1 and 2 show that if we alter these classical algorithms in this way, we appear to get smooth functions; in fact, Figure 1(bottom) is a Gaussian.

Might there be other averaging formulas which when repeated would also generates smooth functions? The functional equation

$$F\left(\frac{x_0 + x_1}{2}\right) = \sqrt{F(x_0)F(x_1)}$$

for exponentials suggests replacing the arithmetic mean $\frac{a+b}{2}$ by the geometric mean $\sqrt{ab}$ in standard subdivision algorithms. Instead of starting with an exponential function,

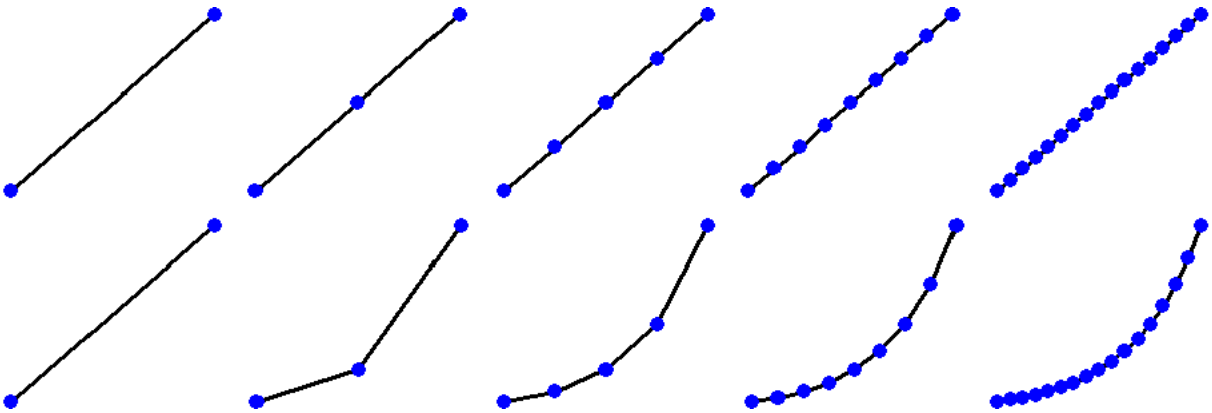

Figure 4. Iterating midpoint averaging generates a dense set of points along a line segment (Top). Iterating the geometric mean generates a dense set of points along an exponential function (Bottom).

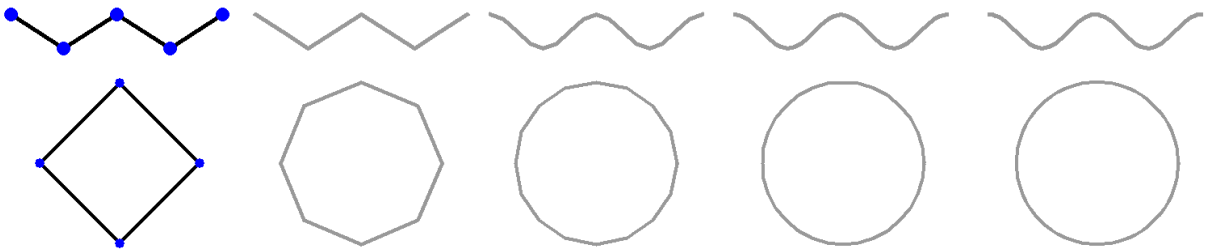

Figure 5. Graph of the cosine function created by replacing midpoint averaging in the de Casteljau subdivision algorithm for Bezier curves by the averaging rule in Equation 7 (Top). Subdivision of a circle using Bezier subdivision, where midpoint averaging for both the x and y coordinates is replaced by the averaging rule in Equation 7 (Bottom).

suppose we start with a trigonometric function. Let $F(x) = \cos(\lambda x + \mu)$. Then

$$F\left(\frac{x_0 + x_1}{2}\right) = \cos\left(\lambda\frac{x_0 + x_1}{2} + \mu\right) = \cos\left(\frac{\lambda x_0 + \ \mu}{2} + \frac{\lambda x_1 + \mu}{2}\right).$$

Using the summation and half angle formulas for cosine, we find that $F(x) = \cos(\lambda x + \mu)$ satisfies the functional equation:

$$F\left(\frac{x_0 + x_1}{2}\right) = \frac{\sqrt{(1 + F(x_0))(1 + F(x_1))} - \sqrt{(1 - F(x_0))(1 - F(x_1))}}{2}. \tag{6}$$

This formula suggests the averaging rule:

$$av(a, b) = \frac{\sqrt{(1 + a)(1 + b)} - \sqrt{(1 - a)(1 - b)}}{2}. \tag{7}$$

If $\lambda = 1$ and $\mu = 0$, then $F(x) = \cos(x)$; if $\lambda = 1$ and $\mu = -\frac{\pi}{2}$, then $F(x) = \cos(x - \frac{\pi}{2}) = \sin(x)$. Thus starting with the two values $\cos(x_0)$ and $\cos(x_1)$, where $0 < x_0, x_1 < \pi$, and iterating Equation 7 generates a dense set of points along a segment of the function $y = \cos(x)$. Similarly, starting with the two values $\sin(x_0)$ and $\sin(x_1)$, where $-\frac{\pi}{2} < x_0, x_1 < \frac{\pi}{2}$, and iterating Equation 7 generates a dense set of points along a segment of the function $y = \sin(x)$. Moreover, and once again this observation is the key point, if we replace midpoint averaging by Equation 7 either in the de Casteljau subdivision algorithm for Bezier curves or in the Lane-Riesenfeld algorithm for uniform B-splines, we still generate smooth functions (see Figure 5).

3.3. Conjugating Linear Subdivision Algorithms with Locally Invertible Nonlinear Functions

The examples we have just considered suggest that if we start with a subdivision algorithm based on repeated midpoint averaging that generates smooth curves and we replace midpoint averaging by a rule derived from a functional equation that for some smooth function $F(x)$ expresses $F\left(\frac{x_0+x_1}{2}\right)$ in terms of $F(x_0)$ and $F(x_1)$, then the resulting nonlinear subdivision algorithm will again generate smooth functions. We are now going to explain the basic theory behind this general result. The key idea turns out to be conjugation of linear subdivision algorithms and linear averages by locally invertible nonlinear maps.

3.3.1. Conjugation and Averaging

Conjugation, averaging, functional equations, and subdivision are all inextricably intertwined. Conjugation is related to averaging because we can generate new averaging rules by conjugating any averaging rule with a locally invertible map.

The mathematical concept of an average is quite general. No matter how we ultimately define the notion of average – arithmetic mean, geometric mean, or Equation 7 – we expect that the average of two values a, b should be symmetric in a and b and lie somewhere between a and b. That is, we expect

i. $av(a, b) = av(b, a)$

ii. $av(a, a) = a$

iii. $\min(a, b) < av(a, b) < \max(a, b)$ whenever $a \neq b$.

Suppose that F is a locally invertible univariate function. Since locally F^{-1} exists, the function F is locally 1-1; therefore F is locally monotonic. If we define

$$\widehat{av}(a, b) = F(av(F^{-1}(a), F^{-1}(b))),$$

then clearly

$$\widehat{av}(a, b) = \widehat{av}(b, a)$$

and

$$\widehat{av}(a, a) = a.$$

Moreover, since

$F^{-1}(a) < F^{-1}(b) \Rightarrow F^{-1}(a) < av(F^{-1}(a), F^{-1}(b)) < F^{-1}(b)$

then because F is monotonic either

$a < \widehat{av}(a, b) < b$

or

$b < \widehat{av}(a, b) < a.$

Thus in either case

$\min(a, b) < av(a, b) < \max(a, b)$ whenever $a \neq b$.

Setting

$F^{-1}(a, b) = (F^{-1}(a), F^{-1}(b))$

we arrive at the new averaging rule

$\widehat{av}(a, b) = F \circ av \circ F^{-1}(a, b).$

Thus conjugating an averaging rule with a locally invertible map generates another averaging rule. Such averaging rules are discussed in Hardy et al. [11].

Moreover, every continuous averaging rule $\widehat{av}(a, b)$ that satisfies the natural condition

iv. $\widehat{av}\left(\widehat{av}(a, b), \widehat{av}(c, d)\right) = \widehat{av}\left(\widehat{av}(a, c), \widehat{av}(b, d)\right)$

can be generated by conjugating the arithmetic mean with a continuous locally invertible map [10]. Condition *iv* simply asserts that the average of four numbers does not depend on how the numbers are paired. To find the map F corresponding to an averaging rule $\widehat{av}$, consider the subdivision scheme S that inserts the new value

$$S\left(\frac{a+b}{2}\right) = \widehat{av}(a, b)$$

between any two values a and b. If $\widehat{av}$ is a continuous function, then recursively applying the subdivision rule S generates a continuous limit function $S^{\infty}(a, b)$ [10]. Thus for any interval (a, b) and any initial values $(F(a), F(b))$, this subdivision scheme defines a 1-1 function $F(x)$ that satisfies the functional equation

$$F\left(\frac{x_0 + x_1}{2}\right) = \widehat{av}(F(x_0), F(x_1)).$$

Let av denote the arithmetic mean. Then

$F \circ av = \widehat{av} \circ F,$

so

$\widehat{av} = F \circ av \circ F^{-1}.$

Therefore, we conclude that:

- Every continuous averaging rule $\widehat{av}$ satisfying condition iv can be generated by conjugating the arithmetic mean with a continuous locally invertible function F.

- The invertible function F satisfies a functional equation defined by the averaging rule $\widehat{av}$.

- The invertible function F can be generated by a subdivision scheme S based on the averaging rule $\widehat{av}$.

Since we have seen that many subdivision algorithms are generated by repeated averaging, these observations motivate our interest in conjugating subdivision algorithms built from averaging rules with locally invertible maps.

3.3.2. Some Simple Properties of Conjugation

Before we proceed then with the study of subdivision, let us recall a few simple properties of conjugation. Suppose that Σ is a semigroup – a set with an associative binary operation which we shall denote by juxtaposition – and let G be a group acting on Σ. Fix an element g in the group G and for each element s in the set Σ, let $\widehat{s}$ denote conjugation of s by g – that is, let

$$\widehat{s} = g \cdot s \cdot g^{-1}$$

where $\cdot$ denotes the action of the group G on the set Σ. Then it is easy to show that conjugation satisfies following three properties.
Properties of Conjugation

C1. $\widehat{s_1 s_2} = \widehat{s}_1 \widehat{s}_2$

C2. $\widehat{s_1 \dots s_n} = \widehat{s}_1 \dots \widehat{s}_n$

C3. $\widehat{s^k} = \widehat{s}^k$

The first property follows directly from the definition of conjugation; the second and third properties follow easily by induction from the first property. We shall have occasion to use these three properties in the following section.

3.3.3. Conjugation and Subdivision

Now back to subdivision. Let p be a set of control points lying in $U \subseteq R^n$ and let $F : U \to R^n$. We shall denote by $F(p)$ the control points generated by applying F to each point in p.

Theorem 3.1: *Suppose that $F : U \to \widehat{U}$ is a continuous 1-1 function on an open set $U \subseteq R^n$. Let S be a subdivision algorithm that maps points in U to points in U and let $\widehat{S} = F \circ S \circ F^{-1}$. Then*

a. $\widehat{S}^\infty = F \circ S^\infty \circ F^{-1}$

b. $S^\infty(p^0) = p^\infty \Rightarrow \widehat{S}^\infty(F(p^0)) = F(p^\infty)$.

Proof: To establish part a, observe that by property C3

$\widehat{S}^k = \widehat{S^k} = F \circ S^k \circ F^{-1}$.

Therefore since F is a continuous function, in the limit

$\widehat{S}^\infty = F \circ S^\infty \circ F^{-1}$.

Moreover part b is a direct consequence of part a, since if $S^\infty(p^0) = p^\infty$, then by part a

$\widehat{S}^\infty(F(p^0)) = F \circ S^\infty(p^0) = F(p^\infty)$.

The condition that S maps points in U to points in U is necessary because the domain of F is U. Notice that this condition is automatically satisfied if U is convex and S uses only convex combinations of the control points. For our examples, all of the subdivision schemes we consider have this convexity property with the sole exception of the four-point interpolatory subdivision scheme.

Corollary 3.2: *Suppose that $F : U \to \widehat{U}$ is a continuous 1-1 function on an open set $U \subseteq R^n$. Let S be a subdivision algorithm that maps points in U to points in U and let $\widehat{S} = F \circ S \circ F^{-1}$. If the curves generated by S are C^k and if F is C^n, then the curves generated by the subdivision algorithm $\widehat{S}$ are at least C^m, where $m = min(k, n)$.*

Corollary 3.3: *Suppose that $F : U \to \widehat{U}$ is a continuous 1-1 function on an open set $U \subseteq R^n$. Let S be a subdivision algorithm that maps points in U to points in U and let $\widehat{S} = F \circ S \circ F^{-1}$. If S is an interpolatory subdivision algorithm, then $\widehat{S}$ is also an interpolatory subdivision algorithm.*

Theorem 3.1 and Corollary 3.2 tell us that if we conjugate a smooth subdivision algorithm S with a smooth map F, then we get another smooth subdivision algorithm $\widehat{S}$. Moreover, if we can generate the function $p(x)$ with the subdivision algorithm S, then we can generate the function $F(p(x))$ with the subdivision algorithm $\widehat{S} = F \circ S \circ F^{-1}$. Stated in this way, these results seem like nothing more than applying the function F to the results of the subdivision algorithm S. To generate the subdivision results in Sections 3.1 and 3.2 where we actually change the notion of averaging at each step of the subdivision algorithm, we need one additional observation.

Recall that both the de Casteljau subdivision algorithm for Bezier curves and the Lane-Riesenfeld subdivision algorithm for uniform B-splines consist of several rounds of midpoint averaging. In general, we say that a subdivision algorithm S has d rounds $S_1, \ldots, S_d$ if

$S(p) = S_d \circ \ldots \circ S_1(p)$

where $S_1, \ldots, S_d$ are themselves rules for generating new control points from old control points. We can now state the following key result about subdivision and conjugation.

Proposition 3.4: *Suppose that $F : U \to \widehat{U}$ is a continuous 1-1 function on an open set $U \subseteq R^n$. Let S be a subdivision algorithm that maps points in U to points in U and let $\widehat{S} = F \circ S \circ F^{-1}$. If S consists of d rounds $S_1, \ldots, S_d$ where each S_i maps points in U to points in U, then $\widehat{S}$ consists of d rounds $\widehat{S}_1 = F \circ S_1 \circ F^{-1}, \ldots, \widehat{S}_d = F \circ S_d \circ F^{-1}$.*

Proof: This result is just a restatement of property C2 for conjugation.

If a subdivision algorithm S consists of d rounds with the rules $S_1, \ldots, S_d$, then by Proposition 3.4 the subdivision algorithm $\widehat{S} = F \circ S \circ F^{-1}$ generated by conjugating S with a locally invertible function F consists of d rounds with the rules $\widehat{S}_1 = F \circ S_1 \circ F^{-1}, \ldots, \widehat{S}_d = F \circ S_d \circ F^{-1}$. In particular, if $S_1, \ldots, S_d$ are midpoint averaging rules, then

$$\widehat{S}_j(\widehat{p}) = F \circ S_j \circ F^{-1}(\widehat{p}) = F\left(\frac{F^{-1}(\widehat{p}_k) + F^{-1}(\widehat{p}_{k+1})}{2}\right), \tag{8}$$

Similarly, if $S_1, \ldots, S_d$ are linear interpolation rules, then

$$\widehat{S}_j(\widehat{p}) = F \circ S_j \circ F^{-1}(\widehat{p}) = F\left((1-\lambda)F^{-1}(\widehat{p}_k) + \lambda F^{-1}(\widehat{p}_{k+1})\right). \tag{9}$$

Now let $p = F^{-1}(\widehat{p})$. Then these rules are equivalent to

$$\widehat{S}_j(F(p)) = F\left(\frac{p_k + p_{k+1}}{2}\right) \tag{10}$$

$$\widehat{S}_j(F(p)) = F((1-\lambda)p_k + \lambda p_{k+1}). \tag{11}$$

Notice that the right hand side of Equation 10 resembles the left hand sides of Equations 5 and 6 – in particular, to subdivide F, we simply evaluate F midway between known values on F.

Equations 8–11 represent new averaging rules based on arbitrary invertible functions. These new rules are particularly interesting when F satisfies a simple functional equation of the form

$$F\left(\frac{p_k + p_{k+1}}{2}\right) = G(F(p_k), F(p_{k+1})), \tag{12}$$

or equivalently if $\widehat{p} = F(p)$,

$$F\left(\frac{F^{-1}(\widehat{p}_k) + F^{-1}(\widehat{p}_{k+1})}{2}\right) = G(\widehat{p}_k, \widehat{p}_{k+1}). \tag{13}$$

When G is simpler than F and F^{-1}, the expressions on the right hand sides of Equations 8–11 simplify, and we can replace a complicated expression involving F and F^{-1} by a simpler expression involving only G. Our averaging rule then reduces to

$$av(a, b) = G(a, b). \tag{14}$$

Now suppose F is a smooth function. Then by Theorem 3.1, if we replace midpoint averaging by the averaging rule in Equation 14 in a subdivision algorithm that generates smooth functions, the new subdivision algorithm based on this new averaging rule will also generate smooth functions. We shall now examine several examples of smooth functions F for which this simplification occurs.

3.4. Examples of Nonlinear Subdivision Algorithms Based on Nonlinear Averages

We begin by reviewing the examples we first encountered in section 3.2: exponentials and trigonometric functions.

Example 3.1: *Exponential Functions*

Let $F(x) = e^x$; then $F^{-1}(x) = log(x)$. Let S be a subdivision algorithm consisting of d rounds $S_1, \ldots, S_d$ of repeated midpoint averaging. If $\widehat{S} = F \circ S \circ F^{-1}$, then at each round $\widehat{S}_j$ of the subdivision algorithm $\widehat{S}$

$$\widehat{S}_j(\widehat{p}) = F \circ S \circ F^{-1}(\widehat{p}) = F \circ S_j(log(\widehat{p})) = e^{\frac{log(\widehat{p}_k)+log(\widehat{p}_{k+1})}{2}} = \sqrt{\widehat{p}_k \widehat{p}_{k+1}}.$$

Thus we conclude from Theorem 3.1 and Proposition 3.4 that replacing midpoint averaging by the square root of the product in the de Casteljau subdivision algorithm generates functions of the form $e^{p(x)}$, where $p(x)$ is a polynomial. Moreover to generate the function $e^{p(x)}$, we must start with the data $\widehat{p} = e^p$, where p are the Bezier coefficients of $p(x)$. Similarly, replacing midpoint averaging by the square root of the product in the Lane-Riesenfeld algorithm generates functions of the form $e^{s(x)}$, where $s(x)$ is a spline, and to generate the function $e^{s(x)}$, we must start with the data $\widehat{s} = e^s$, where s are the uniform B-spline coefficients for $s(x)$. Notice that, in general, we must be careful to choose all the values in $\widehat{p}$ to be positive so that both the log and the square root function are well-defined; however, this restriction is automatically enforced, since our data must be of the form $\widehat{p} = e^p$ and the values of e^p are always positive. Similarly, if we replace the midpoint averaging rules $S_1, \ldots, S_d$ by the linear interpolation (weighted averaging) rules $S'_1, \ldots, S'_d$, then at each round $\widehat{S}'_j$ of the subdivision algorithm for $\widehat{S}'$

$$\begin{aligned} \widehat{S}'_j(\widehat{p}) &= F \circ S'_j \circ F^{-1}(\widehat{p}) \\ &= F \circ S'_j(log(\widehat{p})) \\ &= e^{((1-\lambda)log(\widehat{p}_k)+\lambda log(\widehat{p}_{k+1}))} \\ &= \widehat{p_k}^{1-\lambda} \widehat{p_{k+1}}^{\lambda}. \end{aligned} \tag{15}$$

Replacing the linear interpolation steps in the four-point rule by the averaging rule on the right hand side of Equation 15 will generate smooth functions of the form $e^{g(x)}$, where $g(x)$ are functions generated by the four-point rule (see Figure 3).

Example 3.2: *Trigonometric Functions*

Let $F(x) = \cos(x)$; then $F^{-1}(x) = \cos^{-1}(x)$. Again let S be a subdivision algorithm consisting of d rounds $S_1, \ldots, S_d$ of repeated midpoint averaging. If $\widehat{S} = F \circ S \circ F^{-1}$, then at each round $\widehat{S}_j$ of the subdivision algorithm $\widehat{S}$

$$\begin{aligned} \widehat{S}_j(\widehat{p}) &= F \circ S_j \circ F^{-1}(\widehat{p}) \\ &= F \circ S_j(\cos^{-1}(\widehat{p})) \\ &= \cos\left(\frac{\cos^{-1}(\widehat{p}_k)+\cos^{-1}(\widehat{p}_{k+1})}{2}\right). \end{aligned}$$

Applying the summation and half angle formulas for cosine, we find that at each round

$$\begin{aligned} \widehat{S}_j(\widehat{p}) &= \cos\left(\frac{\cos^{-1}(\widehat{p}_k)+\cos^{-1}(\widehat{p}_{k+1})}{2}\right) \\ &= \frac{\sqrt{(1+\widehat{p}_k)(1+\widehat{p}_{k+1})}-\sqrt{(1-\widehat{p}_k)(1-\widehat{p}_{k+1})}}{2}, \end{aligned}$$

which is precisely the averaging rule in Equation 7. Thus we conclude from Theorem 3.1 and Proposition 3.4 that replacing midpoint averaging in the de Casteljau subdivision algorithm with the averaging rule provided by Equation 7 generates functions of the form $\cos(p(x))$, where $p(x)$ is a polynomial, and replacing midpoint averaging in the Lane-Riesenfeld algorithm with the averaging rule in Equation 7 generates functions of the form $\cos(s(x))$, where $s(x)$ is a spline. Note that, in general, here we must be careful to choose all the values in $\widehat{p}$ to lie in the interval $(-1, 1)$, where $\cos^{-1}(x)$ is well-defined. Similar results hold for the sine function, since as we observed in section 3.2, sine and cosine satisfy the same functional equation. Notice that if we consider control points in the xy-plane and we use the averaging rule for cosine on the first coordinate and the averaging rule for sine on the second coordinate, then we can apply these subdivision rules to generate evenly space points on the circle – see Figure 5 (bottom).

Example 3.3: *Averages Generated from Power Functions*

To construct a generic collection of averaging rules, let $F_p : (0, \infty) \to (0, \infty)$ be defined by

$$\begin{array}{ll} F_p(x) & = x^{\frac{1}{p}} \qquad p \neq 0 \\ & = e^x \qquad p = 0. \end{array}$$

Then the corresponding averaging rules are

$$A_p(a, b) = F_p\left(\frac{F_p^{-1}(a) + F_p^{-1}(a)}{2}\right).$$

Thus

$$\begin{array}{ll} A_p(a, b) & = \left(\frac{a^p + b^p}{2}\right)^{\frac{1}{p}} \qquad p \neq 0 \\ & = \sqrt{ab} \qquad p = 0. \end{array}$$

In particular,

$$\begin{array}{ll} A_{-1}(a, b) = \frac{2ab}{a+b} & \text{(harmonic mean)} \\ A_0(a, b) = \sqrt{ab} & \text{(geometric mean)} \\ A_1(a, b) = \frac{a+b}{2} & \text{(arithmetic mean).} \end{array}$$

This family of averages is investigated by Hardy et al. in [11]. Notice that by L'Hopital's rule

$$\begin{array}{l} \lim_{p \to 0} A_p(a, b) = A_0(a, b) \\ \lim_{p \to -\infty} A_p(a, b) = \min(a, b) \\ \lim_{p \to \infty} A_p(a, b) = \max(a, b). \end{array}$$

If $av = A_p$, then by Theorem 3.1 and Proposition 3.4, the output of the Lane-Risenfeld algorithm with d rounds of averaging on a sequence of positive real numbers $\{c_i\}$ is

$$\begin{array}{ll} \left(\sum_i c_i^p N_i^d(t)\right)^{\frac{1}{p}} & p \neq 0 \\ \\ e^{\sum_i \log(c_i) N_i^d(t)} & p = 0 \end{array}$$

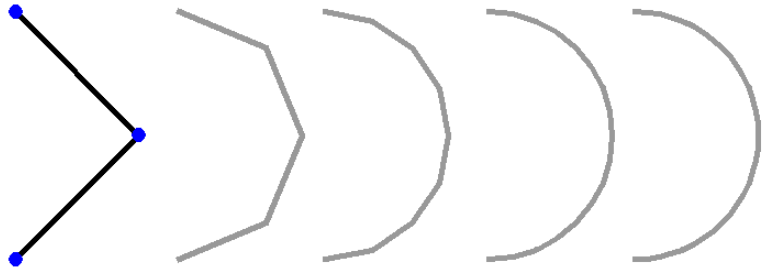

Figure 6. A circular arc reproduced by replacing midpoint averaging in the de Casteljau algorithm with the product of the complex square roots.

where $N_0^d(t), \ldots, N_{2n+1-d}^d(t)$ are the degree d uniform B-spline basis functions. Analogous results hold for any subdivision procedure, such as the de Casteljau subdivision algorithm for Bezier curves, when linear averaging is replaced by the averaging rule $av = A_p$.

We can also investigate smooth 1-1 functions $F : U \to \widehat{U}$, where $U, \widehat{U}$ are open convex subsets of R^2. One way to generate interesting functions of this type is to identify the xy-plane with the complex numbers, and then to investigate functions of a complex variable.

Example 3.4: *Complex Exponential Functions*

Let $F(z) = e^z$, where z is a complex number. Then $F^{-1}(z) = log(z)$. Again let S be a subdivision algorithm consisting of d rounds $S_1, \ldots, S_d$ of repeated midpoint averaging. If $\widehat{S} = F \circ S \circ F^{-1}$, then just as in Example 3.1 at each round $\widehat{S}_j$ of the subdivision algorithm $\widehat{S}$.

$$\widehat{S}_j(\widehat{p}) = F \circ S \circ F^{-1}(\widehat{p}) = F \circ S_j(log(\widehat{p})) = e^{\frac{log(\widehat{p}_k)+log(\widehat{p}_{k+1})}{2}} = \sqrt{\widehat{p}_k}\sqrt{\widehat{p}_{k+1}},$$

but here the entries of $\widehat{p}$ are complex numbers, so the square root here is the complex square root. If we adopt polar coordinates and set $\widehat{p}_k = \widehat{r}_k e^{i\widehat{\theta}_k}$, $-\pi < \theta < \pi$, then

$$\widehat{S}_j(\widehat{p}) = \sqrt{\widehat{p}_k}\sqrt{\widehat{p}_{k+1}} = \sqrt{\widehat{r}_k \widehat{r}_{k+1}} e^{i(\widehat{\theta}_k+\widehat{\theta}_{k+1})/2}.$$

Thus the averaging rule here takes the geometric mean of the radial coordinates and the arithmetic mean of the angular coordinates. Since both log and complex square root have a branch cut discontinuity along the negative x-axis, to get a single-valued function we must restrict our attention to control points $\widehat{p}$ lying in a convex subset of C that does not contain the negative real numbers. What kinds of functions can be generated by such subdivision algorithms? If the initial control points $\widehat{p}$ lie along a circle of radius r centered at the origin, then the points at each round $\widehat{S}^k(\widehat{p})$ will necessarily lie along the same circle; moreover, in this case the point

$$\sqrt{\widehat{p}_k}\sqrt{\widehat{p}_{k+1}} = \widehat{r} e^{i(\widehat{\theta}_k+\widehat{\theta}_{k+1})/2}$$

lies midway along the arc between $\widehat{p}_k$ and $\widehat{p}_{k+1}$. (Here the arc that does not contain the negative x-axis is chosen to avoid the branch cut discontinuity in the square root. This arc is not necessarily the shorter arc.) Thus if we replace midpoint averaging by the product of

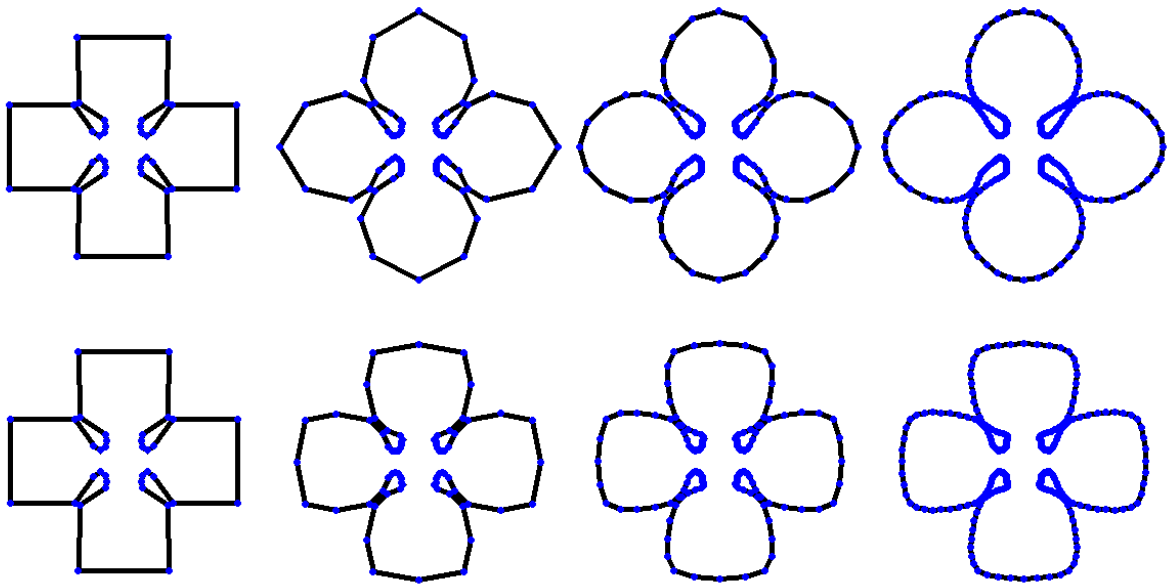

Figure 7. A clover-shaped curve generated by the four-point subdivision algorithm conjugated with a Mobius transformation (top). Contrast with the same control points subdivided with the ordinary four-point scheme (bottom).

the complex square roots, then both the de Casteljau subdivision algorithm and the Lane-Riesenfeld algorithm will reproduce arcs of circles centered at the origin. Moreover, if the initial control points are uniformly spaced along the circle, then the new control points after each round of subdivision will also be uniformly spaced along the circle. Nevertheless, these subdivision algorithms cannot reproduce the entire circle because of the branch cut discontinuity along the negative x-axis (see Figure 6). In general, by Theorem 3.1 and Proposition 3.4, the curves generated by the de Casteljau subdivision algorithm and the Lane-Riesenfeld algorithm when we replace midpoint averaging by the product of the complex square roots are the images under the map $F(z) = e^z$ of the curves – polynomials and splines – generated by the standard versions of the de Casteljau subdivision algorithm and the Lane-Riesenfeld algorithm using midpoint averaging. In particular, if p are the Bezier control points for the planar polynomial curve $g(t) = (g_1(t), g_2(t))$, then the curve generated by the de Casteljau subdivision algorithm using the product of the complex square roots and the control points $\widehat{p} = e^p$ is

$$e^{g(t)} = e^{g_1(t)+ig_2(t)} = (e^{g_1(t)}\cos(g_2(t)), e^{g_1(t)}\sin(g_2(t))).$$

Similar results hold for the Lane-Riesenfeld algorithm, where polynomials are replaced by splines.

Example 3.5: *Mobius Transformations*

Consider the Mobius transformation

$$F(z) = \frac{az+b}{cz+d},$$

where a, b, c, d, z are complex numbers and $ad - bc \neq 0$. Then

$$F^{-1}(z) = \frac{dz - b}{-cz + a}.$$

Once again let S be a subdivision algorithm consisting of d rounds $S_1, \ldots, S_d$ of repeated midpoint averaging. If $\widehat{S} = F \circ S \circ F^{-1}$, then at each round $\widehat{S}_j$ of the subdivision algorithm $\widehat{S}$

$$\begin{aligned} \widehat{S}_j(\widehat{p}) &= F \circ S \circ F^{-1}(\widehat{p}) = F\left(\frac{1}{2}\left(\frac{d\widehat{p}_k - b}{-c\widehat{p}_k + a} + \frac{d\widehat{p}_{k+1} - b}{-c\widehat{p}_{k+1} + a}\right)\right) \\ &= \frac{a(\widehat{p}_k + \widehat{p}_{k+1}) - 2c(\widehat{p}_k\widehat{p}_{k+1})}{2a - c(\widehat{p}_k + \widehat{p}_{k+1})}. \end{aligned}$$

Mobius transformations are closed under composition and therefore form a group. This group is isomorphic to the group of projective 2×2 nonsingular complex-valued matrices. Thus every Mobius transformation is the composition of transformations corresponding to unitary and diagonal projective matrices – that is, to translations, dilations, and inversions [1]. Each of these elementary transformations maps lines and circles to lines and circles, so all Mobius transformations map lines and circles to lines and circles. For example, under the Mobius transformation $F(z) = \frac{z+i}{-z}$, the image of the horizontal line $l(t) = (t + i)$ is $F(l) = \frac{-t^2-2}{t^2+1} - \frac{it}{t^2+1}$, which is a circle with center at $-\frac{3}{2}$ and radius $\frac{1}{2}$. Figure 7 (top) illustrates subdivision using the four-point algorithm conjugated with a Mobius transformation.

3.5. The Geometric Approach: Changing the Geodesics

Standard subdivision algorithms for approximating schemes such as the de Casteljau subdivision algorithm for Bezier curves and the Lane-Riesenfeld subdivision algorithm for uniform B-splines generate curves that lie in the convex hull of their control points and satisfy the variation diminishing property. These two properties are often critical for good approximation schemes. Moreover, the convex hull property is crucial for intersection algorithms based on recursive subdivision (see Section 3.5.2). But if we replace the arithmetic mean in these subdivision algorithms by nonlinear averaging rules, then these properties no longer seem to hold (see Figures 5 and 6 from the previous section.) It turns out that these properties break down because in this context straight lines are no longer the appropriate notions for geodesics in R^n.

On curved manifolds if there is a unique geodesic joining two points, then midpoint averaging can be replaced by the midpoint of the geodesic joining the two points. With this approach, the de Casteljau subdivision algorithm and the Lane-Riesenfeld algorithm can be extended to curved manifolds [21], [22]. We can adopt this same geodesic approach to subdivision in Euclidean space: here we are going to change the geodesics, but not the underlying geometry of Euclidean space. While this geodesic interpretation may seem elementary, we shall see shortly that it is necessary to adopt this point of view to construct intersection algorithms for several common transcendental functions using nonlinear subdivision.

Suppose then that $F : U \to \widehat{U}$ is a continuous 1-1 function on an open set $U \subseteq R^n$. We can think of F as warping the set U into the set $\widehat{U}$. Under this warp, lines L are mapped into curves $F(L)$. If we treat the curves $F(L)$ as the geodesics in $\widehat{U}$, then F is

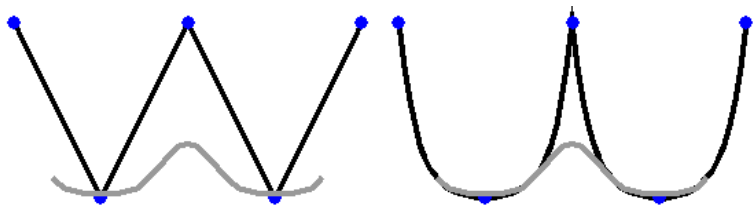

Figure 8. A Gaussian generated by nonlinear subdivision using the control points in blue. On the left, the control polygon is drawn in black with straight lines. The Gaussian (gray) does not satisfy the convex hull or variation-diminishing properties with respect to this control polygon. On the right, the control polygon is drawn using the geodesics generated from the map $F(x, y) = (x, e^y)$. Notice that the Gaussian does indeed satisfy the convex hull and variation-diminishing properties with respect to this geodesic control polygon.

an isometry from U to $\widehat{U}$. Since F is 1-1, if $\widehat{P}$, $\widehat{Q}$ are two points in $\widehat{U}$, then we can define a metric D on $\widehat{U}$ by setting

$$D(\widehat{P}, \widehat{Q}) = Dist_{Euclidean}(F^{-1}(\widehat{P}), F^{-1}(\widehat{Q})).$$

Since the lines L represent shortest distances in U, the curves $F(L)$ represent geodesics in $\widehat{U}$.

3.5.1. The Convex Hull and Variation Diminishing Properties

To prepare the way for our theorem on the convex hull and variation diminishing properties for curves generated by nonlinear subdivision algorithms, we need a few simple definitions relating these properties to geodesics.

A set C is *convex with respect to the geodesics* G if the geodesic connecting any two points in the set C lies completely within the set C. The *convex hull of the points* p *with respect to the geodesics* G is the smallest convex set with respect to the geodesics G containing the points p.

Similarly, a curve C with control points p satisfies *the variation diminishing property with respect to the geodesics* G if for every geodesic g, the number of intersections of g with C is always less than or equal to the number of intersections of g with the geodesic control polygon of p.

Theorem 3.5: *Suppose that* $F : U \to \widehat{U}$ *is a continuous 1-1 function on an open set* $U \subseteq R^n$. *Let* S *be a subdivision algorithm that maps points in* U *to points in* U *and let* $\widehat{S} = F \circ S \circ F^{-1}$.

a. If the subdivision scheme S *satisfies the convex hull property with respect to straight lines* L, *then the subdivision scheme* $\widehat{S}$ *satisfies the convex hull property with respect to the geodesics* $F(L)$.

b. If the subdivision scheme S *satisfies the variation diminishing property with respect to straight lines* L, *then the subdivision scheme* $\widehat{S}$ *satisfies the variation diminishing property with respect to the geodesics* $F(L)$.

Proof: Part *a* follows easily from the fact that F maps convex sets in U (with respect to straight lines L) to convex sets in $\widehat{U}$ (with respect to the geodesics $F(L)$), and part *b*

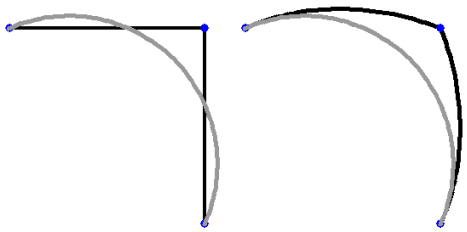

Figure 9. The analogue of Figure 8 with the geodesics on the right generated by the complex exponential function: $F(z) = e^z \iff F(x, y) = (e^x \cos(y), e^x \sin(y))$.

follows directly from the fact that F maps straight lines in U to geodesics in $\widehat{U}$.

Suppose that $F : U \to \widehat{U}$ is a continuous 1-1 function on an open set $U \subseteq R$. To apply the geometric approach to subdivision and to take full advantage of Theorem 3.5, it is best to extend F to a bivariate function so that we can consider the graph of F in the xy-plane. To extend F to a bivariate function, we introduce the map $H(x, y) = (x, F(y))$. For example, if $F(t) = e^t$, then we consider the bivariate function $H(x, y) = (x, e^y)$. In this case, in order to reproduce x, we typically perform the standard linear subdivision algorithm on the first component and we perform nonlinear subdivision only on the second component. Figures 8-10 illustrate the convex hull property and the variation diminishing property for several different nonlinear subdivision schemes in the Euclidean plane.

3.5.2. Intersection Algorithms

For two curves generated by recursive subdivision that lie in the convex hulls of their control points, we have the following standard intersection algorithm:
Intersection Algorithm

- If the convex hulls of the control points of the two curves fail to intersect, then the curves themselves do not intersect.
- Otherwise if each curve can be approximated by a straight line segment, then intersect these line segments.
- Otherwise *subdivide* the two curves and intersect the curve segments recursively.

Now suppose that $F : U \to \widehat{U}$ is a continuous 1-1 function on an open set $U \subseteq R^n$. By Theorem 3.5 if S is a subdivision scheme that maps points in U to points in U and satisfies the convex hull property with respect to straight lines L, then the subdivision scheme $\widehat{S} = F \circ S \circ F^{-1}$ satisfies the convex hull property with respect to the geodesics $F(L)$. Therefore the intersection algorithm presented above is still valid for curves generated by the nonlinear subdivision scheme $\widehat{S}$ provided that we replace straight lines L bounding the convex hull by geodesics $F(L)$. We illustrate this intersection algorithm in Figure 11, where we apply recursive subdivision to intersect a Gaussian with a straight line. The figure illustrates the recursive procedure calls that find the two intersection points. Notice

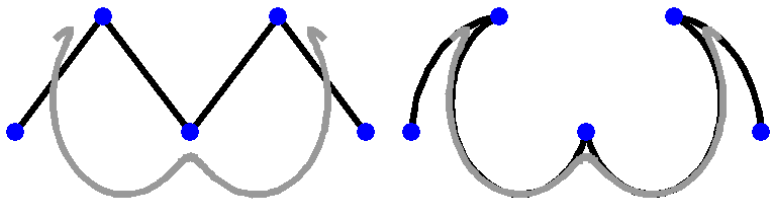

Figure 10. The analogue of Figure 8 with the geodesics on the right generated by the Mobius transformation $F(z) = \frac{1}{z}$, which turns the plane inside-out.

that this algorithm is complete; unlike Newton iteration, recursive subdivision finds all the intersection points.

In general, the geodesics $F(L)$ are not straight lines but rather are smooth curves (for instance $y = e^x$). Computing the convex hull of a set bounded by smooth functions may be difficult and can complicate the intersection algorithm. For certain functions, however, (such as e^x), we can bound each geodesic $F(L)$ of the control polygon with a triangle. In particular, if the derivative of $F(L)$ is monotonic, we can construct a triangle that bounds $F(L)$ by intersecting the tangent lines at the end-points of the geodesic. Computing these triangles for all the geodesics yields a set of points whose convex hull contains the convex hull of the geodesics. Because this convex hull is bounded by straight line segments, the intersection algorithm simplifies. Notice that this algorithm does not require that the function produced by subdivision or its derivative is monotonic, which is almost certainly not the case. Instead the only requirement is on the derivative of the nonlinear function F.

Figure 12 illustrates the same intersection calculation as in Figure 11 except that the convex hull of the geodesics is replaced by a piecewise linear hull bounding the geodesics. Figure 13 depicts a similar intersection calculation between a Gaussian and a cubic curve. We subdivide the cubic curve with a linear subdivision scheme and the Gaussian with a nonlinear subdivision scheme. Since both curves satisfy the convex hull property, we can still efficiently intersect these two curves.

4. Conclusion

We have presented a systematic approach to generating a new class of nonlinear subdivision schemes by replacing linear averages with nonlinear averages in linear subdivision algorithms. We showed that replacing linear averages with nonlinear averages is equivalent to composing a linear subdivision algorithm with a locally invertible nonlinear map. The smoothness of our nonlinear subdivision schemes is inherited from the smoothness of the original linear subdivision schemes and the differentiability of the locally invertible maps. Using our method, we are able to build simple nonlinear stationary subdivision schemes for many functions, including trigonometric and exponential functions, for which no known linear stationary subdivision schemes exist.

This paper is based on four key insights: three algebraic and one geometric.

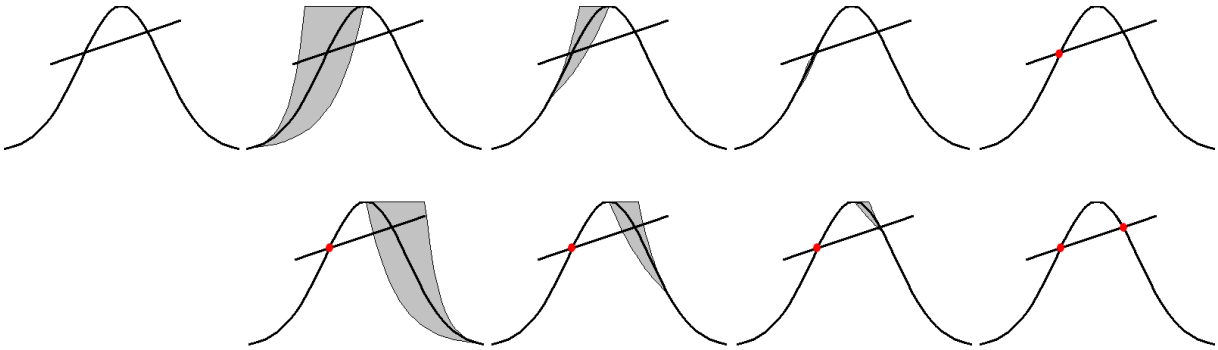

Figure 11. Using recursive subdivision to intersect a Gaussian with a straight line. The convex hulls of the exponential control polygons are shaded.

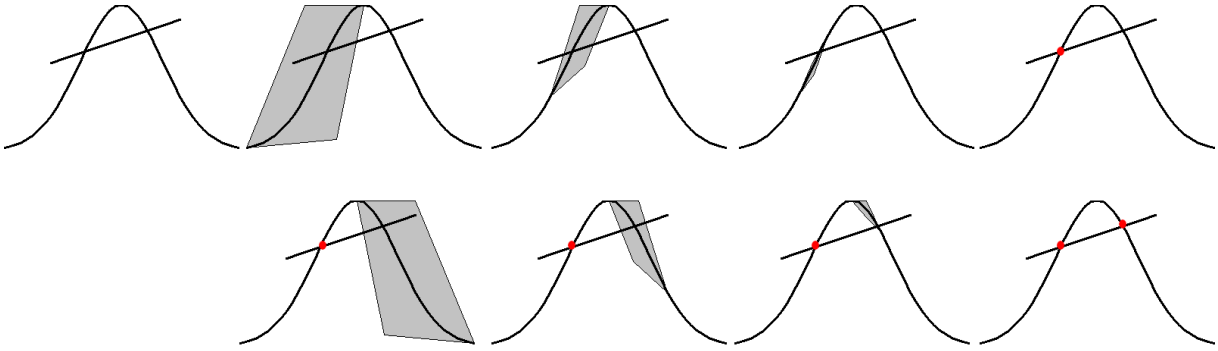

Figure 12. Using recursive subdivision to intersect a Gaussian with a straight line. The piecewise linear approximations to the convex hulls of the exponential control polygons are shaded. Compare with Figure 11.

- new averaging rules – replace midpoint averaging in linear subdivision algorithms
- functional equations – generate new averaging rules with especially nice properties
- conjugation – provides insight into why new averaging rules lead to smooth functions
- geodesics – present a geometric context for understanding nonlinear subdivision and for building intersection algorithms based on recursive subdivision

Algebra guided our intuitions and led to rigorous proofs; geometry linked our new subdivision schemes for functions in Euclidean space to standard subdivision techniques on curved manifolds and allowed us to recover the convex hull and variation diminishing properties. The convex hull property relative to geodesics permitted us to extend standard intersection algorithms for polynomial and piecewise polynomial functions based on recursive subdivision to intersection algorithms for transcendental functions generated from nonlinear subdivision.

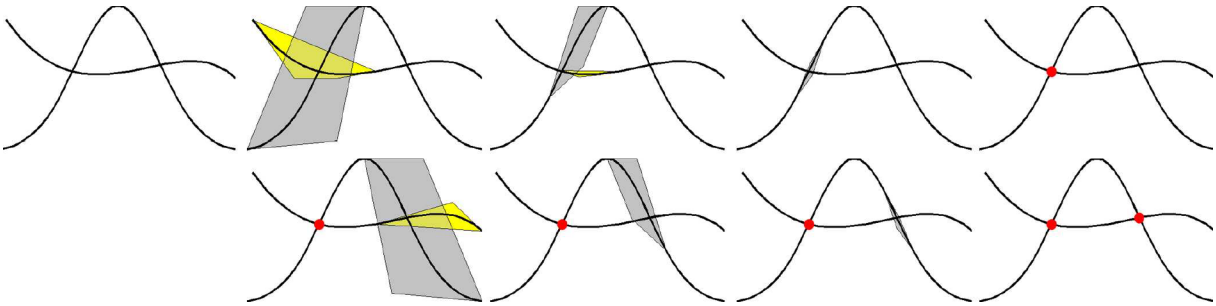

Figure 13. Using recursive subdivision to intersect a Gaussian with a cubic curve. The piecewise linear approximations to the convex hulls of the exponential control polygons are shaded.

The interpretation of nonlinear subdivision in terms of geodesics also allows us to recover other geometric properties. For example, functions generated from linear subdivision algorithms are invariant under translation along straight lines. Similarly, functions generated from nonlinear subdivision based on nonlinear averages are invariant under translation along the corresponding geodesics.

Because replacing linear averages with nonlinear averages is equivalent to conjugating a linear subdivision algorithm with a locally invertible nonlinear map, many other properties of functions generated from linear subdivision algorithms extend to functions generated by these nonlinear subdivision algorithms. For example, subdivision schemes generate self-similar functions. Therefore functions generated by linear subdivision schemes are attractors [19]. Similarly, the functions generated from these nonlinear subdivision schemes are also attractors. The iterated function systems for functions generated from these nonlinear subdivision schemes are built by conjugating the iterated function systems for the corresponding linear subdivision scheme with the locally invertible map that represents the nonlinear average.

Several important polynomial and piecewise polynomial functions that can be built by linear subdivision procedures can also be generated from recursive evaluation algorithms based on repeated linear interpolation: For Lagrange interpolation there is Nevilles algorithm; for Bezier approximation, the de Casteljau algorithm; and for B-splines, the de Boor algorithm. If we replace linear interpolation in these evaluation algorithms by a nonlinear rule derived from replacing weighted linear averages with the corresponding weighted nonlinear averages, then the result is a recursive evaluation algorithm for the functions generated from the corresponding nonlinear subdivision scheme. The proofs that this works are much the same as the proofs of convergence for these nonlinear subdivision schemes. In fact, Theorem 3.1, Corollaries 3.2, 3.3, and Proposition 3.4 for recursive subdivision algorithms have straightforward analogues for recursive evaluation procedures.

In the future, we would like to find additional interesting maps to create new and different nonlinear subdivision schemes for univariate functions. In particular, we would like to explore maps that are not analytic. For instance, Theorem 3.1 bears a striking

similarity to the technique used by Schaefer *et al* [20] to modify the Catmull-Clark scheme to interpolate networks of curves. We would also like to pay more attention to bivariate functions and perhaps study nonlinear subdivision algorithms for surfaces with extraordinary points. Finally we would like to use our approach to develop nonlinear subdivision techniques for new applications such as Fourier and wavelet transforms.

Acknowledgements
We would like to thank Nira Dyn for her help and advice with this paper as well as for suggesting [11] as a reference.

REFERENCES

1. M. Artin. *Algebra*. Prentice-Hall, 1991.
2. Nicolas Aspert, Touradj Ebrahimi, and Pierre Vandergheynst. Non-linear subdivision using local spherical coordinates. *Computer Aided Geometric Design*, 20(3):165–187, 2003.
3. E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10:350–355, 1978.
4. A. Cohen, N. Dyn, and B. Matei. Quasilinear subdivision schemes with applications to ENO interpolation. *ACHA*, 15:89–116, 2003.
5. P. de Castlejau. *Formes a Poles*. Hermes, 1985.
6. G. Deslauriers and S. Dubuc. Symmetric iterative interpolation processes. *Constructive Approximation*, 5:49–68, 1989.
7. N. Dyn, J. Gregory, and D. Levin. A four point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4:257–268, 1987.
8. Nira Dyn. Three families of nonlinear subdivision schemes. In Jetter, Buhmann, Haussman, Schaback, and Stoeckler, editors, *Topics in Multivariate Approximation And Interpolation*, pages 23–38. Elsevier, 2005.
9. Michael Floater and Charles Micchelli. Nonlinear stationary subdivision. *Approximation Theory*, 212:209–224, 1998.
10. R. Goldman, E. Vouga, and S. Schaefer. Smoothness of nonlinear subdivision schemes based on nonlinear averaging. In preparation, 2007.
11. G.H. Hardy, J.E. Littlewood, and G. Polya. *Inequalities*. Cambridge University Press, 1952.
12. J. Lane and R. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:35–46, 1980.
13. Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.
14. M. Marinov, N. Dyn, and D. Levin. Geometrically controlled 4-point interpolatory schemes. In N.A. Dodgson, M.S. Floater, and M.A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*. Springer-Verlag, 2005.
15. Charles A. Micchelli. Interpolatory subdivision schemes and wavelets. *J. Approx. Theory*, 86(1):41–71, 1996.
16. L. Noaks. Nonlinear corner cutting. *Adv. Comp. Math.*, 8:165–177, 1998.
17. Ulrich Reif. A unified approach to subdivision algorithms near extraordinary vertices.

Computer Aided Geometric Design, 12(2):153–174, 1995.
18. M. Sabin and N. Dogson. A circle-preserving variant of the four-point subdivision scheme. In *Mathematical Methods for Curves and Surfaces: Tromso 2004*, pages 275–286, 2005.
19. S. Schaefer, D. Levin, and R. Goldman. Subdivision schemes and attractors. In *Proceedings of the Eurographics Symposium on Geometry Processing 2005*, pages 171–180. Eurographics, 2005.
20. Scott Schaefer, Joe Warren, and Denis Zorin. Lofting curve networks using subdivision surfaces. In *Proceedings of the Eurographics Symposium on Geometry Processing 2004*, pages 105–116. Eurographics, 2004.
21. J. Wallner and N. Dyn. Convergence and C^1 analysis of subdivision schemes on manifolds by proximity. *Computer Aided Geometric Design*, 22(7):593–622, 2005.
22. G. Xie and T. Yu. Smoothness equivalence properties of manifold-valued data subdivision schemes based on the projection approach. *In preparation*, 2006.