Freeform Curves on Spheres of Arbitrary Dimension

Scott Schaefer and Ron Goldman

Rice University
6100 Main St.
Houston, TX 77005
sschaefe@rice.edu and rng@rice.edu

Abstract
Recursive evaluation procedures based on spherical linear interpolation and stationary subdivision algorithms based on geodesic midpoint averaging are used to construct the analogues on spheres of arbitrary dimension of Lagrange and Hermite interpolation, Bezier and B-spline approximation, Catmull-Rom splines and four point interpolatory subdivision schemes. Applications include constructing surface patches on spheres and building splines in quaternion space for key frame animation in 3-dimensions.

1. Introduction

Spheres are the simplest curved surfaces. Naturally, spheres in 3-dimensions are useful for Geometric Modeling, but spheres in 4-dimensions are also important in Computer Graphics, since the unit sphere in 4-dimensions is the space of unit quaternions, which represents rotations in 3-space. Curves on the sphere in 3-dimensions can be used to bound surface patches on the sphere. Curves on the unit sphere in 4-dimensions represent families of rotations in 3-space, and smooth families of rotations are useful in Computer Graphics for key frame animation. Thus it is natural to investigate various kinds of curves on spheres of different dimensions.

Many methods are known for generating curves in n-dimensional Euclidean spaces. The goal of this paper is to extend some of these approaches to generate curves on spheres of arbitrary dimension. We shall focus our attention on two intrinsic techniques: recursive evaluation procedures and stationary subdivision algorithms.

Recursive evaluation procedures are a convenient device for generating many common curve schemes [4, 8]. For Lagrange interpolation, there is Neville's algorithm, for Bezier approximation de Casteljau's algorithm, for B-splines de Boor's algorithm, and for Catmull-Rom splines a combination of Neville's algorithm and de Boor's algorithm [1]. In Euclidean space, all of these algorithms are based on repeated linear interpolation. In Section 2, we shall show how to use spherical linear interpolation to construct the analogous algorithms on spheres of arbitrary dimension. Using this approach, we shall generate the analogues of Lagrange and Hermite interpolation, Bezier and B-spline approximation, and Catmull-Rom splines on spheres of arbitrary dimension.

In a landmark paper, Shoemake introduced spherical linear interpolation to construct the analogues of Bezier curves in the space of unit quaternions [7]. Spherical linear interpolation extends readily to spheres of arbitrary dimension. Our contribution here is to show that spherical linear interpolation is compatible not only with approximating methods such as Bezier curves and B-splines, but also with interpolatory techniques such as Lagrange interpolation and Catmull-Rom splines. We also extend recursive evaluation algorithms to include Hermite interpolation on spheres of arbitrary dimension where, along with spherical linear interpolation, spherical translation is required. Finally we show how to construct the analogues of rational curves – curves where the control points have weights – on spheres of arbitrary dimension.

Stationary subdivision algorithms are procedures for building smooth curves from discrete data by recursively inserting values between the data. To construct smooth curves in Euclidean space, we commonly insert new data by averaging the given data. Thus, to extend subdivision algorithms to generate smooth curves on spheres, we need to interpret what averaging means on spheres of arbitrary dimension. We shall do so in Section 3. We will then show how to apply subdivision to construct the analogues of Bezier, B-spline, and four point interpolating curves on spheres of arbitrary dimension. Our averaging procedures are based on calculating the midpoints of the arcs of great circles on the sphere. Other researchers have applied geodesics to extend the theory of subdivision from Euclidean spaces to arbitrary manifolds [2, 6]; our contribution here is to provide simple explicit formulas for subdivision algorithms on spheres of arbitrary dimension.

2. Recursive Evaluation Procedures on Spheres

Many well known schemes for generating freeform curves in Euclidean space can be evaluated by recursive procedures based on repeated linear interpolation [4, 8]: Lagrange interpolation invokes Neville's algorithm, Bezier approximation de Casteljau's algorithm, B-splines de Boor's algorithm, and Catmull-Rom splines have a recursive evaluation algorithm based on a combination of Neville's algorithm for Lagrange interpolation and de Boor's algorithm for B-spline approximation. We shall see that each of these schemes can be extended to spheres by replacing linear interpolation in Euclidean spaces by spherical linear interpolation on spheres of arbitrary dimension.

Linear interpolation ($lerp$) in Euclidean space takes in two points P, Q and a scaling parameter α and creates a point that splits the line from P to Q into 2 segments whose ratio is $\frac{1-\alpha}{\alpha}$.

$$lerp(P,Q,\alpha) = (1-\alpha)P + \alpha Q$$

Likewise, spherical linear interpolation ($slerp$) operates on two points on a unit sphere represented by unit vectors u, v and generates a point along the geodesic connecting u and v that splits the arc into two arcs whose ratio is $\frac{1-\alpha}{\alpha}$.

$$slerp(u,v,\alpha) = \frac{\sin(\phi(1-\alpha))u + \sin(\phi\alpha)v}{\sin(\phi)}$$

where $\phi = \cos^{-1}(u \cdot v)$ [7].

Recursive evaluation procedures based on repeated linear interpolation typically have the following structure: Fix two collections of scalar parameters $\{s_j^k\}, \{t_j^k\}$, $j = 0,\ldots,n-k$, $1 \le k \le n$. Let $P_0,\ldots,P_n$ be a set of control points, and define

$$\begin{aligned} P_j^0(t) &= P_j \quad j = 0,\ldots,n \\ P_j^k(t) &= lerp(P_j^{k-1}(t), P_{j+1}^{k-1}(t), \tfrac{t-s_j^k}{t_j^k - s_j^k}) \quad j = 0,\ldots,n-k. \end{aligned} \tag{1}$$

The function

$$P(t) = P_0^n(t)$$

is the curve generated by this recursive evaluation procedure. The scalars $\{s_j^k\}, \{t_j^k\}$ specify the type of curve generated by the algorithm – different choices of scalars lead to different types of curves schemes, e.g., Lagrange, Bezier, B-spline, Catmull-Rom splines; the control points $P_0,\ldots,P_n$ govern the shape of the curve.

To define the corresponding recursive evaluation procedures on spheres, we replace linear interpolation in the recurrence – Equation 1 – by spherical linear interpolation. Thus if $u_0,\ldots,u_n$ are a collection of unit vectors representing points on a sphere, we set

$$\begin{aligned} su_j^0(t) &= u_j \quad j = 0,\ldots,n \\ su_j^k(t) &= slerp(su_j^{k-1}(t), su_{j+1}^{k-1}(t), \tfrac{t-s_j^k}{t_j^k - s_j^k}) \end{aligned}$$

for $j = 0,\ldots,n-k$. The function

$$S(t) = su_0^n(t)$$

is the curve on the sphere generated by this recursive evaluation procedure.

Generally, the curves $S(t)$ on spheres inherit properties analogous to the properties of the corresponding curves $P(t)$ in Euclidean spaces. For example, in Euclidean 3-space curves generated by recursive evaluation procedures are rotation invariant; rotating all the control points around a common axis rotates all the points on the curve $P(t)$ around the same axis. Rotation invariance holds in Euclidean space because rotation commutes with linear interpolation. Similarly, rotation invariance holds for the curves generated by recursive evaluation procedures on the sphere because spherical rotation commutes with spherical linear interpolation since spherical rotation maps geodesics to geodesics.

2.1. Lagrange Interpolation and Neville's Algorithm

Consider the Lagrange interpolating polynomial $L(t)$ of degree n with nodes $t_0,\ldots,t_n$ and interpolation points $P_0,\ldots,P_n$ – that is, the unique degree n polynomial curve $L(t)$ such that $L(t_i) = P_i, i = 0,\ldots,n$. The recursive step for Neville's algorithm is

$$P_j^k(t) = lerp(P_j^{k-1}(t), P_{j+1}^{k-1}(t), \frac{t-t_j}{t_{j+k}-t_j}) \quad j = 0,\ldots,n-k.$$

The function

$$L(t) = P_0^n(t)$$

interpolates the points $P_0,\ldots,P_n$ in Euclidean space at the parameter values $t_0,\ldots,t_n$. Notice here $s_j^k = t_j$ and $t_j^k = t_{j+k}$.

To generate interpolating curves on spheres, simply replace linear interpolation by spherical linear interpolation in the recursive step of the evaluation algorithm. Thus if $u_0,\ldots,u_n$ are a collection of unit vectors representing points on a sphere and $t_0,\ldots,t_n$ are a set of parameter values, the recursive step for Lagrange interpolation on the sphere is

$$su_j^k(t) = slerp(su_j^{k-1}(t), su_{j+1}^{k-1}(t), \frac{t-t_j}{t_{j+k}-t_j}) \quad j = 0,\ldots,n-k.$$

The function

$$SL(t) = su_0^n(t)$$

is the analogue of a Lagrange interpolating curve on the sphere – that is, $SL(t_i) = u_i, i = 0,\ldots,n$ (see Figure 1).

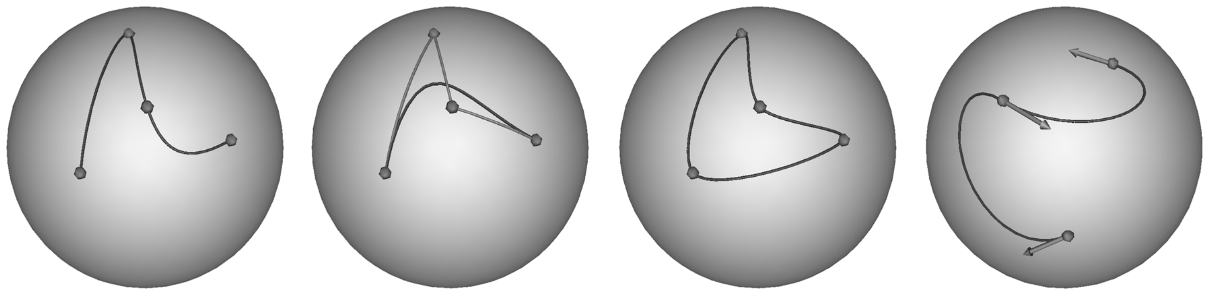


Figure 1: *Recursive evaluation procedures generate different free-form curves on a sphere. From left to right: Lagrange interpolant, Bezier curve (with control polygon shown), a C^1 Catmull-Rom spline and a C^1 Hermite curve with two segments.*

The proof that this spherical version of Neville's algorithm generates an interpolant on the sphere is essentially the same as the inductive proof that the standard version of Neville's algorithm generates an interpolant in Euclidean space [4]. To prove interpolation on the sphere, we need only show that the functions $su_j^k(t),\ j=0,\dots,n-k,\ 0\le k\le n$, interpolate the vectors $u_j,\dots,u_{j+k}$ at the nodes $t_j,\dots,t_{j+k}$. Clearly this result is true for $k=0$. Suppose this result is true for $k-1$. Then

$$su_j^k(t_{j+i}) = slerp(su_j^{k-1}(t_{j+i}), su_{j+1}^{k-1}(t_{j+i}), \tau_j^k(t_{j+i}))$$

where $\tau_j^k(t_{j+i}) = \frac{t_{j+i}-t_j}{t_{j+k}-t_j}$. Hence, by the inductive hypothesis

$$\tau_j^k(t_j) = 0 \Rightarrow su_j^k(t_j) = su_j^{k-1}(t_j) = u_j$$
$$\tau_j^k(t_{j+k}) = 1 \Rightarrow su_j^k(t_{j+k}) = su_{j+1}^{k-1}(t_{j+k}) = u_{j+k}.$$

Moreover, if $1\le i\le k-1$, then by the inductive hypothesis, $su_j^{k-1}(t_{j+i}) = u_{j+i} = su_{j+1}^{k-1}(t_{j+i})$, so it follows that $\phi(t)\to 0$ as $t\to t_{j+i}$ where $\phi(t)=\cos^{-1}(su_j^{k-1}(t)\cdot su_{j+1}^{k-1}(t)$. Therefore for $1\le i\le k-1$

$$Lim_{t\to t_{j+i}} su_j^k(t) = u_{j+i}.$$

Notice that Neville's algorithm for interpolation is valid on any manifold, simply replace $slerp$ by the geodesic connecting two points. The proof is essentially the same.

2.2. Bezier Curves and de Casteljau's Algorithm

Consider the Bezier curve of degree n over the interval $[a,b]$ with control points $P_0,\dots,P_n$. The recursive step for de Casteljau's algorithm is

$$P_j^k(t) = lerp(P_j^{k-1}(t), P_{j+1}^{k-1}(t), \frac{t-a}{b-a}) \quad j=0,\dots,n-k.$$

The function

$$B(t) = P_0^n(t)$$

is the Bezier curve with control points $P_0,\dots,P_n$. Notice that for Bezier curves $s_j^k=a$ and $t_j^k=b$.

To generate Bezier curves on spheres, we again replace linear interpolation by spherical linear interpolation in the recursive step of the evaluation algorithm. Thus if $u_0,\dots,u_n$ are a collection of unit vectors representing points on a sphere, the recursive step is

$$su_j^k(t) = slerp(su_j^{k-1}(t), su_{j+1}^{k-1}(t), \frac{t-a}{b-a}) \quad j=0,\dots,n-k.$$

The function

$$SB(t) = su_0^n(t)$$

is the analogue of a Bezier curve on the sphere. The points $u_0,\dots,u_n$ joined by geodesic arcs form the control polygon for the Bezier curve (see Figure 1).

If we replace straight lines in Euclidean spaces with great circles on spheres, then many of the standard properties of Bezier curves in Euclidean spaces carry over to Bezier curves on spheres. For example, the convex hull and variation diminishing properties of Bezier curves relative to straight lines in Euclidean space become the convex hull and variation diminishing properties relative to great circles on the sphere. The proofs of the more difficult properties such as the variation diminishing property are based on subdivision algorithms for Bezier curves, which we shall investigate in Section 3.

2.3. B-splines and de Boor's Algorithm

Consider a B-spline curve of degree n over the interval $[t_0,t_r]$ with knots $\{t_j\}$ and control points $\{P_k\}$. The recursive step for de Boor's algorithm is

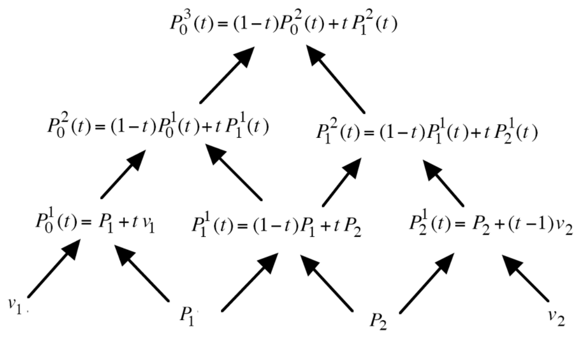
$$P_j^k(t) = lerp(P_j^{k-1}(t), P_{j+1}^{k-1}(t), \frac{t-t_{j+k}}{t_{j+n+1}-t_{j+k}})$$

for $j=\alpha-n,\dots,\alpha-k$, when $t\in[t_\alpha,t_{\alpha+1}]$. The function

$$BS(t) = P_{\alpha-n}^n(t)$$

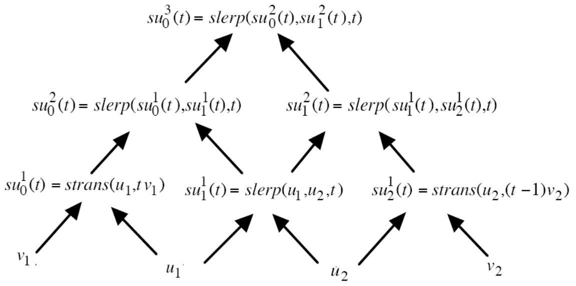
is the segment of the B-spline curve with control points $\{P_{\alpha-n},\dots,P_\alpha\}$ over the interval $[t_\alpha,t_{\alpha+1}]$. Notice that for B-spline curves $s_j^k=t_{j+k}$ and $t_j^k=t_{j+n+1}$.

To generate B-spline curves on spheres, we again replace linear interpolation by spherical linear interpolation in the recursive step of the evaluation algorithm. Thus if $u_0,\dots,u_n$



$P_0^3(t) = (1-t)P_0^2(t) + tP_1^2(t)$
$P_0^2(t) = (1-t)P_0^1(t) + tP_1^1(t)$
$P_1^2(t) = (1-t)P_1^1(t) + tP_2^1(t)$
$P_0^1(t) = P_1 + tv_1$
$P_1^1(t) = (1-t)P_1 + tP_2$
$P_2^1(t) = P_2 + (t-1)v_2$
v_1
P_1
P_2
v_2

Figure 2: *Neville's algorithm for cubic Hermite interpolation in Euclidean space.*



$su_0^3(t) = slerp(su_0^2(t), su_1^2(t), t)$
$su_0^2(t) = slerp(su_0^1(t), su_1^1(t), t)$
$su_1^2(t) = slerp(su_1^1(t), su_2^1(t), t)$
$su_0^1(t) = strans(u_1, tv_1)$
$su_1^1(t) = slerp(u_1, u_2, t)$
$su_2^1(t) = strans(u_2, (t-1)v_2)$
v_1
u_1
u_2
v_2

Figure 3: *The analogue of Neville's algorithm for cubic Hermite interpolation on the sphere, replacing linear interpolation by spherical linear interpolation ($slerp$) and replacing translation by spherical translation ($strans$). Compare to Figure 2.*

are a collection of unit vectors representing points on a sphere, the recursive step is

$$su_j^k(t) = slerp(su_j^{k-1}(t), su_{j+1}^{k-1}(t), \frac{t - t_{j+k}}{t_{j+n+1} - t_{j+k}})$$

for $j = 0, \ldots, n-k$. The function

$$SBS(t) = su_{\alpha-n}^n(t)$$

is the analogue of a segment of a B-spline curve on the sphere.

As with Bezier curves, standard properties of B-spline curves in Euclidean spaces carry over with only minor modifications to B-spline curves on spheres. The proofs of the more difficult properties such as the variation diminishing property are based on knot insertion or subdivision algorithms for B-splines, which we shall investigate in Section 3.

2.4. Catmull-Rom Splines

A Catmull-Rom spline of degree $2n-1$ is a piecewise polynomial $CR(t)$ that interpolates the control points $\{P_j\}$ at the knots $\{t_j\}$ – that is, $CR(t_j) = P_j$ – and has continuity of order $n-1$ at the knots. Catmull-Rom splines can be generated by running Neville's algorithm for n levels followed by de Boor's algorithm for $n-1$ levels; equivalently one can run Neville's algorithm for $n-1$ levels followed by de Boor's algorithm for n levels [1]. Thus the recursive steps of the evaluation algorithm for Catmull-Rom splines are

$$P_j^k(t) = lerp(P_j^{k-1}(t), P_{j+1}^{k-1}(t), \frac{t - t_j}{t_{j+k} - t_j})$$

for $j = \alpha, \ldots, \alpha + n - k$, $1 \le k \le n$ and

$$P_j^k(t) = lerp(P_j^{k-1}(t), P_{j+1}^{k-1}(t), \frac{t - t_{j+k+1-n}}{t_{j+n+1} - t_{j+k+1-n}})$$

for $j = \alpha - n, \ldots, \alpha + n - 1 - k$, $n + 1 \le k \le 2n - 1$ when $t \in [t_\alpha, t_{\alpha+1}]$. The function

$$CR(t) = P_{\alpha-n}^{2n-1}(t)$$

is the segment of the Catmull-Rom spline with control points $\{P_\alpha, \ldots, P_{\alpha+2n-1}\}$ over the interval $[t_\alpha, t_{\alpha+1}]$.

To generate the analogue of Catmull-Rom splines on spheres, we again replace linear interpolation by spherical linear interpolation in the recursive steps of the evaluation algorithm. Thus if $\{u_j\}$ are a collection of unit vectors representing points on a sphere, the recursive steps are

$$su_j^k(t) = slerp(su_j^{k-1}(t), su_{j+1}^{k-1}(t), \frac{t - t_j}{t_{j+k} - t_j})$$

for $j = \alpha, \ldots, \alpha + n - k$, $1 \le k \le n$, and

$$su_j^k(t) = slerp(su_j^{k-1}(t), su_{j+1}^{k-1}(t), \frac{t - t_{j+k+1-n}}{t_{j+n+1} - t_{j+k+1-n}})$$

for $j = \alpha - n, \ldots, \alpha + n - 1 - k$, $1 \le k \le 2n - 1$. The function

$$SCR(t) = su_{\alpha-n}^{2n-1}(t)$$

is the analogue of a segment of a spherical Catmull-Rom spline with control points $\{u_\alpha, \ldots, u_{\alpha+2n-1}\}$ over the interval $[t_\alpha, t_{\alpha+1}]$.

Since the smoothness and interpolatory properties of Neville's algorithm and de Boor's algorithm extend from Euclidean spaces to spheres of arbitrary dimension, spherical Catmull-Rom splines inherit many of the standard properties of Catmull-Rom splines in Euclidean space. In particular, spherical Catmull-Rom splines with $2n-1$ recursive steps have continuity of order $n-1$ at the knots and interpolate their control points at the knots (see Figure 1). Thus we can use Catmull-Rom splines to generate smooth interpolating splines in the space of unit quaternions. This technique might be useful for performing key frame animation in 3-dimensions, but there is one drawback to this approach: even though Catmull-Rom splines are smooth we cannot specify their derivatives at the knots. To specify derivatives, we need to employ Hermite interpolation.

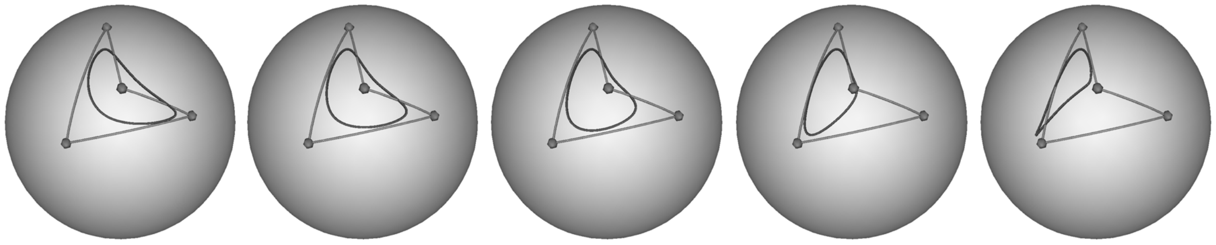


Figure 4: *Rational B-spline on the sphere with unit weights everywhere except at the lower right vertex, which has weights* $2, 1, \frac{1}{2}, \frac{1}{10}, \frac{-1}{10}$ *(from left to right).*

2.5. Hermite Interpolation and Neville's Algorithm

Hermite polynomials interpolate derivatives as well as positions at the nodes [4]. The simplest and probably the most important version of Hermite interpolation is cubic Hermite interpolation, where a single cubic polynomial $H(t)$ interpolates two points P_1, P_2 as well as the first derivative vectors v_1, v_2 at each of the points – that is,

$$\begin{array}{ll} H(0) = P_1 & H(1) = P_2 \\ H'(0) = v_1 & H'(1) = v_2 \end{array}$$

Given a sequence (P_j, v_j) of points and derivative vectors, cubic Hermite interpolation can be applied to generate a C^1 cubic spline that interpolates all the data.

Neville's algorithm for Lagrange interpolation can be generalized to handle Hermite interpolation [4]. Figure 2 illustrates Neville's algorithm for cubic Hermite interpolation in Euclidean space. Notice that except for the first and last steps on the first level of this algorithm, each step is once again just linear interpolation. To generalize cubic Hermite interpolation to spheres, we shall, as usual, simply replace these linear interpolations by spherical linear interpolations. The expressions $P_0^1(t) = P_1 + tv_1$ and $P_2^1(t) = P_2 + tv_2$ that appear on the first level of Neville's algorithm for cubic Hermite interpolation represent translations in Euclidean space. Therefore to extend cubic Hermite interpolation to spheres, we shall need to replace these Euclidean translations by translations on spheres.

To perform translation on a sphere, we define an operator *strans* that takes as input a point u on the sphere and a tangent vector v at u, and translate u along the geodesic in the direction of v by the distance $|v|$. Equivalently, *strans* rotates u in the plane determined by u, v by the angle $|v|$. Thus, since v is perpendicular to u,

$$strans(u,v) = \cos(|v|)u + \sin(|v|)v\frac{|u|}{|v|}.$$

We illustrate this algorithm for Hermite interpolation on the sphere in Figure 3. In Figure 1, we show an example of a C^1 spline on the sphere generated by Hermite interpolation.

The proof that this spherical version of Neville's algorithm generates a Hermite interpolant on the sphere is essentially the same as the proof that the standard version of Neville's algorithm generates a Hermite interpolant in Euclidean space [4]. It is easy to verify directly from the definitions of *slerp* and *strans* that:

$$\begin{array}{lll} s_0^1(0) = u_0 & s_1^1(0) = P_0 & s_2^1(1) = u_1 \\ s'^1_0(0) = v_0 & s_1^1(1) = P_1 & s'^1_2(1) = v_1. \end{array}$$

Using these properties and the definition of *slerp*, a more lengthy computation reveals that:

$$\begin{array}{lll} s_0^2(0) = u_0 & s_0^2(1) = u_1 & s'^2_0(0) = v_0 \\ s_1^2(0) = u_0 & s_1^2(1) = u_1 & s'^2_1(1) = v_1. \end{array}$$

Finally using these results and the definition of *slerp*, we can show, as required, that:

$$\begin{array}{ll} s_0^3(0) = u_0 & s_0^3(1) = u_1 \\ s'^3_0(0) = v_0 & s'^3_0(1) = v_1. \end{array}$$

Since Hermite interpolation works on spheres of arbitrary dimension, we can use Hermite interpolation to generate splines in the space of unit quaternions. This technique may be useful for performing key frame animation in 3-dimensions, since we can specify both a value and a derivative vector for each quaternion.

2.6. Rational Curves

By replacing affine control points with mass-points in recursive evaluation procedures based on repeated linear interpolation, we can generate rational curves in Euclidean spaces [4]. The only additional step we need is to divide by the mass after we complete the recursive evaluation procedure. Rational curves have two advantages: there exist rational curves, such as circles, that cannot be represented as polynomials; in addition, the weights of a rational curve can serve as shape parameters. Similarly, by replacing unit vectors with vectors of arbitrary length, we can introduce mass into the spherical control vectors in recursive evaluation procedures based on repeated spherical linear interpolation. This approach works because *slerp* is defined even for vectors of arbitrary length. The only additional step we need is to normalize the vectors to unit length – that is, to divide by the total mass – after we complete the recursive evaluation procedure. Thus we can generate the analogues of rational curves on spheres of arbitrary dimension.

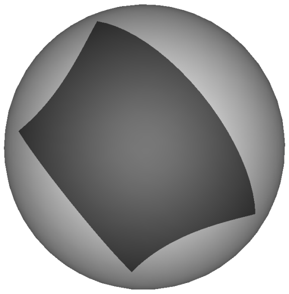


Figure 5: *Analogue of a tensor product biquadratic Bezier surface on the sphere.*

Figure 4 shows a rational B-spline with several different weights for a single vertex. These rational curves on the sphere behave similar to rational B-splines in Euclidean space. As the weight becomes large, the curve is pulled towards the vertex; as the weight becomes small or even negative, the curve is pushed away from the vertex.

2.7. Tensor Product Patches on Spheres

Tensor product surfaces of bidegree (m,n) in Euclidean space can be generated from recursive evaluation procedures for curves by using the output of the procedure for $n+1$ curves of degree m in s as the input for a single curve of degree n in t. Similarly, tensor product surfaces on spheres can be generated from spherical recursive evaluation procedures for curves on spheres by using the output of the procedure for $n+1$ curves in s as the input for a single curve in t. This approach permits us to generate surface patches on spheres of arbitrary dimensions, whose boundary curves are curves of a fixed type. We illustrate this procedure for generating a Bezier patch on the sphere in 3-dimensions in Figure 5.

3. Stationary Subdivision Algorithms: Midpoint Averaging on Spheres

Subdivision is a technique for generating smooth curves from discrete data by recursively inserting values between the data. Although we have seen that many freeform curves, such as Bezier curves and B-splines, can be generated from recursive evaluation procedures based on linear interpolation, in practice these curves are almost always built using recursive subdivision. Recursive subdivision has many advantages: Subdivision generates piecewise linear approximations that converges rapidly to the desired curve, and subdivision algorithms often require only simple repeated averaging, Moreover, there are curve schemes that can be generated from recursive subdivision, such as the four-point subdivision scheme which we will discuss below, for which there is no known recursive evaluation procedure. Here we shall show how to extend subdivision algorithms from Euclidean spaces to spheres. As an added advantage, we will see that subdivision on spheres can often avoid $slerp$ altogether.

$$\begin{tikzcd}
 & & & Q_3 = \frac{P_0+3P_1+3P_2+P_3}{8} = R_0 & & & \\
 & & Q_2 = \frac{P_0+2P_1+P_2}{4} \arrow[ur] & & R_1 = \frac{P_1+2P_2+P_3}{4} \arrow[ul] & & \\
 & Q_1 = \frac{P_0+P_1}{2} \arrow[ur] & & \frac{P_1+P_2}{2} \arrow[ul] \arrow[ur] & & R_2 = \frac{P_2+P_3}{2} \arrow[ul] & \\
Q_0 = P_0 \arrow[ur] & & P_1 \arrow[ul] \arrow[ur] & & P_2 \arrow[ul] \arrow[ur] & & R_3 = P_3 \arrow[ul]
\end{tikzcd}$$

Figure 6: *The de Casteljau subdivision algorithm for cubic Bezier curves in n-dimensional Euclidean space. The Bezier control points are placed at the base of the triangle. At each node above the base, the midpoint is computed of the two values on the nodes from which the arrows entering the node emerge. The points denoted by $\{Q_j\}$ and $\{R_k\}$ subdivide the original Bezier curve at the parameter $\frac{1}{2}$. Iterating this procedure on the points $\{Q_j\}$ and $\{R_k\}$ generates a sequence of piecewise linear functions that converge to the Bezier curve determined by the original control points $\{P_i\}$. Here we illustrate the cubic case, but the algorithm extends in an obvious manner to Bezier curves of arbitrary degree.*

The two most famous subdivision algorithms are the de Casteljau subdivision algorithm for Bezier curves and the Lane-Riesenfeld subdivision algorithm for uniform B-splines [5]. We illustrate these algorithms for cubic curves in Figures 6, 8. The main step in each of these subdivision procedures is midpoint averaging. Therefore to extend these two algorithms to spheres, we need to extend the notion of midpoint averaging to spheres of arbitrary dimension.

The midpoint between two points on a sphere is the midpoint of the shortest arc of the great circle joining these two points. (To make the midpoint unique, we shall exclude pairs of antipodal points.) Therefore, we need a simple procedure for finding this midpoint. We could invoke $slerp$ and evaluate at $s=\frac{1}{2}$, but there is an easier way.

A point on the sphere can be represented by a unit vector. Two points on the sphere are represented by two unit vectors, and their midpoint is simply the vector from the origin to the arc through the tips of the two vectors pointing midway between the two vectors. Thus the midpoint between two vectors on the sphere can be represented by the vector that bisects the angle between the two vectors normalized to unit length. Therefore, if v_1, v_2 are two unit vectors representing points on a unit sphere, then their midpoint on the sphere is represented by the unit vector

$$v_m = \frac{(v_1+v_2)/2}{|(v_1+v_2)/2|} = \frac{v_1+v_2}{\sqrt{2(1+v_1\cdot v_2)}}. \qquad (2)$$

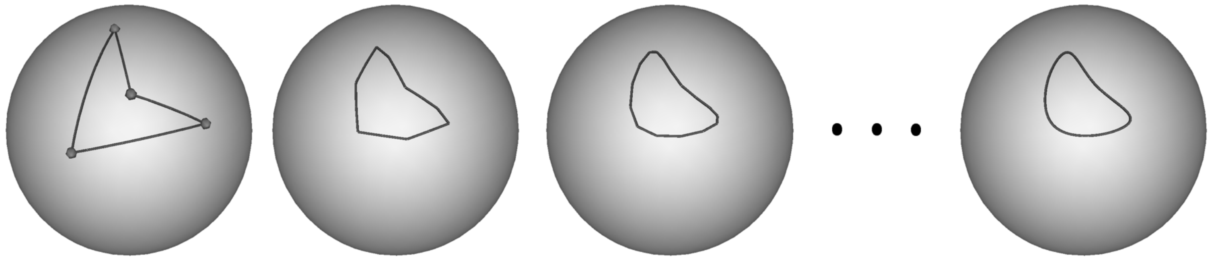


Figure 7: *Subdivision for a uniform cubic B-spline curve on a sphere.*

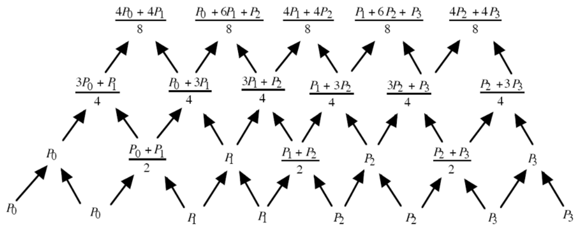


Figure 8: *The Lane-Riesenfeld subdivision algorithm for uniform B-spline curves (knots at the integers) in n-dimensional Euclidean space. The B-spline control points $\{P_i\}$ are repeated and placed at the base of the diagram. At each node above the base, the midpoint is computed of the two values on the nodes from which the arrows entering the node emerge. The points on the n^{th} level of the diagram represent the control points for the same B-spline curve with knots at the half integers. Iterating this procedure generates a sequence of piecewise linear functions that converge to the B-spline curve for the original control points $\{P_i\}$. Here we illustrate the case of a cubic B-spline curve with one segment, but the algorithm generalizes in the obvious fashion to B-spline curves of arbitrary degree with arbitrarily many polynomial segments.*

It is straightforward to verify that the right hand side of Equation 2 is identical to the value of $slerp(v_1, v_2, s)$ at $s = \frac{1}{2}$.

Now all we need to generate splines on spheres is to reinterpret the midpoint averaging rule in the de Casteljau and Lane-Riesenfeld algorithms – that is, we replace the Euclidean midpoint averaging rule

$$P_m = \frac{P_0 + P_1}{2}$$

by the spherical midpoint averaging rule

$$v_m = \frac{(v_1+v_2)/2}{|(v_1+v_2)/2|} = \frac{v_1+v_2}{\sqrt{2(1+v_1\cdot v_2)}}.$$

Notice that there is no need for $slerp$ here. We illustrate this subdivision procedure for generating the analogues of B-spline curves on spheres in Figure 7.

In Euclidean space if the control points are evenly spaced along a straight line, then both the de Casteljau algorithm and the Lane-Riesenfeld algorithm generate a straight line with a linear parameterization. Similarly, on the sphere, if the control points are evenly spaced along a great circle, then both the de Casteljau algorithm and the Lane-Riesenfeld algorithm generate a great circle with a uniform parameterization. In particular, since a circle is a sphere in 2-dimensions, if we start with evenly spaced points on a circle in the plane, then the curve generated by each of these spherical subdivision algorithms is a circle with a uniform parameterization.

Using midpoint averaging, we can also develop subdivision algorithms for the four point subdivision scheme [3] on spheres of arbitrary dimensions. In the plane, the subdivision rules for the four-point scheme are

$$\begin{aligned} p^k_{2i} &= p^{k-1}_i \\ p^k_{2i+1} &= \tfrac{-1}{16}p^{k-1}_{i-1} + \tfrac{9}{16}p^{k-1}_i + \tfrac{9}{16}p^{k-1}_{i+1} + \tfrac{-1}{16}p^{k-1}_{i+2} \end{aligned}$$

where p^k_i is the i^{th} control point at the k^{th} level of subdivision.

Arbitrary affine combinations are difficult to extend to the sphere. However, there is an equivalent geometric construction for these rules. If we find the midpoint of each line segment:

$$m^k_{i,i+1} = \frac{p^k_i + p^k_{i+1}}{2},$$

then we can rewrite the vertex insertion rule using only affine combinations of two vertices at a time.

$$p^k_{2i+1} = \frac{-1}{4}\left(\frac{1}{2}m^{k-1}_{i-1,i} + \frac{1}{2}m^{k-1}_{i+1,i+2}\right) + \frac{5}{4}m^{k-1}_{i,i+1}$$

Notice that this rule only uses midpoints with the exception of one extrapolation (weights of $\frac{5}{4}$ and $\frac{-1}{4}$).

We have seen that

$$v_m = \frac{v_1+v_2}{\sqrt{2(1+v_1\cdot v_2)}}$$

corresponds to interpolation at the midpoint of v_1, v_2. Similarly,

$$v_d = 2(v_1\cdot v_2)v_2 - v_1 \tag{3}$$

corresponds to extrapolation of v_1 past v_2 so that v_2 is the midpoint of v_1 and v_d . Equation 3 can be derived by

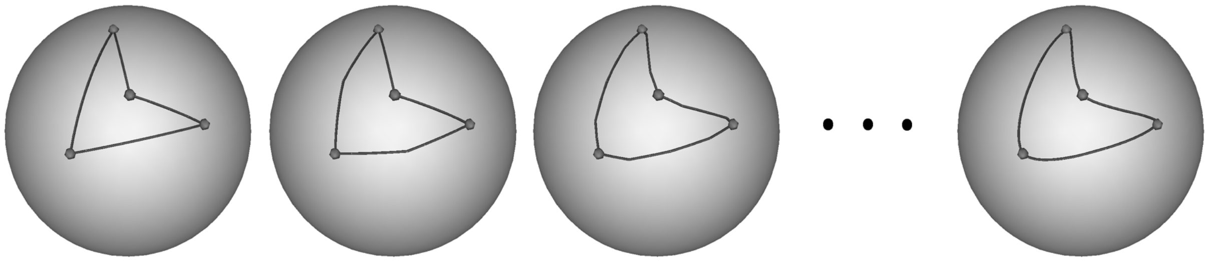


Figure 9: *Subdivision curve on a sphere generated using the four-point rule.*

evaluating $slerp(v_1, v_2, s)$ at $s = 2$. Moreover, Equation 3 is easy to verify simply by observing that $v_d \cdot v_d = 1$ and $v_d \cdot v_2 = v_1 \cdot v_2$. We shall use Equation 3 for the four point subdivision procedure on the sphere.

All of the steps except one in the geometric version of the four point scheme use only midpoint averaging (v_m on the sphere). We can avoid using slerp to perform the extrapolation by first computing v_d followed by two v_m's to generate the desired vertex position on the sphere.

Since midpoint averaging and extrapolation work for spheres of arbitrary dimension, we can use subdivision to generate splines in the space of unit quaternions. Once again, this technique may be useful for performing key frame animation in 3-dimensions.

4. Conclusions and Open Questions

We have investigated two methods for generating curves on spheres of arbitrary dimension: recursive evaluation procedures and stationary subdivision algorithms. Recursive evaluation procedures can be applied to generate the analogues on spheres of arbitrary dimension of Lagrange and Hermite interpolation, Bezier and B-spline approximation, and Catmull-Rom splines. Stationary subdivision algorithms are convenient for constructing Bezier curves, uniform B-splines, and curves generated by four point subdivision on spheres. Subdivision algorithms based on midpoint averaging have two advantages over recursive evaluation procedures: subdivision algorithms on spheres can be implemented without recourse to slerp, and subdivision algorithms can be used to generate curves such as those created by the four point subdivision scheme for which there is no known recursive evaluation procedure.

Recursive evaluation procedures and stationary subdivision algorithms depend only on intrinsic properties of the sphere (geodesics) and are independent of any parameterization of the sphere. Therefore these methods can be extended to arbitrary manifolds. To extend recursive evaluation algorithms to arbitrary manifolds, all we need is a method like $slerp$ for computing points at an arbitrary distance along an arbitrary geodesic joining two points. To generalize stationary subdivision algorithms to arbitrary manifolds, all we require is a method for calculating the midpoint between two points on an arbitrary geodesic. Many manifolds permit such calculations; spheres are particularly nice because the calculations are particularly simple.

Several problems remain open. We constructed the analogues of cubic Hermite interpolants and tensor product patches on spheres of arbitrary dimension. We would like to extend curve techniques to arbitrary Hermite interpolants and surface methods to arbitrary multisided patches. Both of these topics will require further research.

References

[1] P. Barry and R. Goldman. A recursive evaluation algorithm for a class of catmull-rom splines. In *Proceedings of SIGGRAPH '88*, pages 199–204, 1988.

[2] N. Dyn. Three families of nonlinear subdivision schemes. *Private communication*, 2005.

[3] N. Dyn, J. Gregory, and D. Levin. A four point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4:257–268, 1987.

[4] R. Goldman. *Pyramid Algorithms: A Dynamic PRogramming Approach to Curves and Surfaces for Geometry Modeling.* Morgan Kaufmann, 2002.

[5] 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.

[6] L. Noaks. Nonlinear corner cutting. *Advances in Computational Mathematics*, 8:165–177, 1998.

[7] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254. ACM Press, 1985.

[8] K. Shoemake. *Linear form curves in Graphics Gems V.* AP Professional, 1995.