14. Subdivision curves

Reading

Recommended:

- Stollnitz, DeRose, and Salesin. *Wavelets for Computer Graphics: Theory and Applications,* 1996, section 6.1-6.3, A.5.

Note: there is an error in Stollnitz, et al., section A.5. Equation A.3 should read:

$$\mathbf{MV} = \mathbf{V}\Lambda$$

Subdivision curves

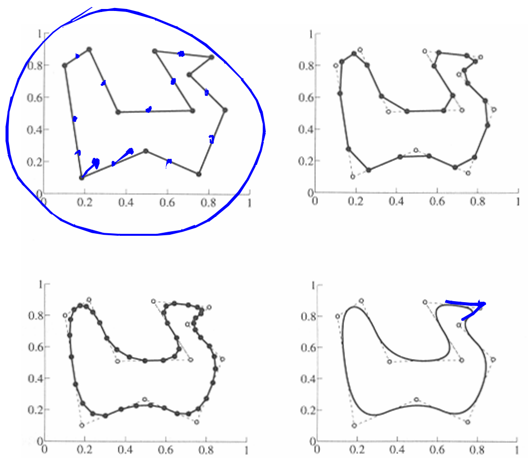
Idea:

- repeatedly refine the control polygon

$$P^1 \to P^2 \to P^3 \to \cdots$$

- curve is the limit of an infinite process

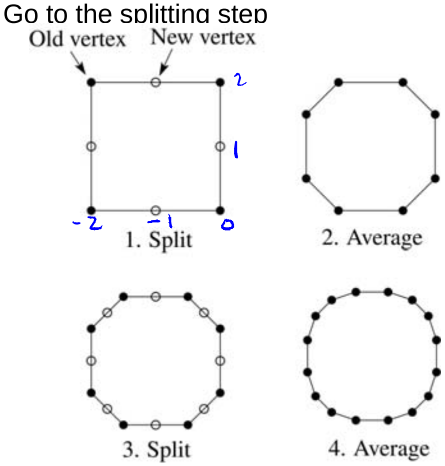
$$Q = \lim_{j \to \infty} P^j$$



Chaikin's algorithm

Chakin introduced the following "corner-cutting" scheme in 1974:

- Start with a piecewise linear curve
- Insert new vertices at the midpoints (the **splitting step**)
- Average each vertex with the "next" (clockwise) neighbor (the **averaging step**)
- Go to the splitting step



Averaging masks

The limit curve is a quadratic B-spline!

Instead of averaging with the nearest neighbor, we can generalize by applying an **averaging mask** during the averaging step:

$$r = (\ldots, r_{-1}, r_0, r_1, \ldots)$$

In the case of Chaikin's algorithm:

$r =$ 0.5 0.5

$\frac{1}{2}(1, 1)$

5

Can we generate other B-splines?

Answer: Yes

Lane-Riesenfeld algorithm (1980)

Use averaging masks from Pascal's triangle:

$$r = \frac{1}{2^n}\left(\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}\right)$$

$\frac{1}{2}(1 \quad 1)$

Gives B-splines of degree n+1.

n=1 quadratic B spline

n=0: **1**

n=1: 1
1 1

n=2: 1
1 1
1 2 1

cubic B spline

$r = \frac{1}{4}(1 \quad 2 \quad 1)$

6

Subdivide ad nauseum?

After each split-average step, we are closer to the **limit curve**.

How many steps until we reach the final (limit) position?

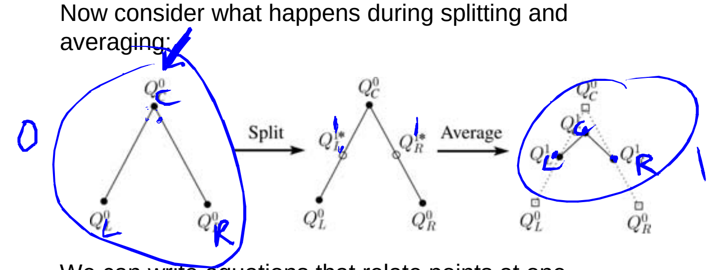
Can we push a vertex to its limit position without infinite subdivision? Yes!

Local subdivision matrix

Consider the cubic B-spline subdivision mask:

$$\frac{1}{4}(1 \quad 2 \quad 1)$$

Now consider what happens during splitting and averaging:



We can write equations that relate points at one subdivision level to points at the previous:

$$Q_L^{1*} = \frac{1}{2}\left(Q_L^0 + Q_C^0\right)$$

$$Q_R^{1*} = \frac{1}{2}\left(Q_C^0 + Q_R^0\right)$$

$$Q_L^1 = \frac{1}{4}\left(Q_L^0 + 2Q_L^{1*} + Q_C^{0*}\right) = \frac{1}{4}\left(2Q_L^0 + 2Q_C^0\right) = \frac{1}{8}\left(4Q_L^0 + 4Q_C^0\right)$$

$$Q_C^1 = \frac{1}{4}\left(Q_L^{1*} + 2Q_C^0 + Q_R^{1*}\right) = \frac{1}{8}\left(Q_L^0 + 6Q_C^0 + Q_R^0\right)$$

$$Q_R^1 = \frac{1}{4}\left(Q_C^0 + 2Q_R^{1*} + Q_R^0\right) = \frac{1}{4}\left(2Q_C^0 + 2Q_R^0\right) = \frac{1}{8}\left(4Q_C^0 + 4Q_R^0\right)$$

Local subdivision matrix

We can write this as a recurrence relation in matrix form:

$$\begin{pmatrix} Q_L^j \\ Q_C^j \\ Q_R^j \end{pmatrix} = \frac{1}{8}\begin{pmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{pmatrix}\begin{pmatrix} Q_L^{j-1} \\ Q_C^{j-1} \\ Q_R^{j-1} \end{pmatrix}$$

$$Q^j = SQ^{j-1}$$

Where the Q's are (for convenience) *row* vectors and S is the **local subdivision matrix**.

We can think about the behavior of each coordinate independently. For example, the x-coordinate:

$$\begin{pmatrix} x_L^j \\ x_C^j \\ x_R^j \end{pmatrix} = \frac{1}{8}\begin{pmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{pmatrix}\begin{pmatrix} x_L^{j-1} \\ x_C^{j-1} \\ x_R^{j-1} \end{pmatrix}$$

$$X^j = SX^{j-1}$$

Local subdivision matrix, cont'd

Tracking just the x components through subdivision:

$$X^j = SX^{j-1} = S \cdot SX^{j-2} = S \cdot S \cdot SX^{j-3} = \cdots = S^j X^0$$

The limit position of the x's is then:

$$X^\infty = \lim_{j\to\infty} S^j X^0$$

OK, so how do we apply a matrix an infinite number of times??

Eigenvectors and eigenvalues

To solve this problem, we need to look at the eigenvectors and eigenvalues of *S*. First, a review…

Let *v* be a vector such that:

1 [eigenvector, v
eigenvalue, λ

$$Sv = \lambda v$$

2 [eigenvector 2
eigenvalue 2

We say that *v* is an eigenvector with eigenvalue λ.

An *n*x*n* matrix can have *n* eigenvalues and eigenvectors:

$$Sv_1 = \lambda_1 v_1$$
$$\vdots$$
$$Sv_n = \lambda_n v_n$$

If the eigenvectors are linearly independent (which means that *S* is *non-defective*), then they form a basis, and we can re-write *X* in terms of the eigenvectors:

$$X = \sum_i^n a_i v_i$$

To infinity, but not beyond…

Now let's apply the matrix to the vector X:

$$X^1 = SX^0 = S\sum_i^n a_i v_i = \sum_i^n a_i S v_i = \sum_i^n a_i \lambda_i v_i$$

Applying it *j* times:

$$X^j = S^j X = S^j \sum_i^n a_i v_i = \sum_i^n a_i S^j v_i = \sum_i^n a_i \lambda_i^j v_i$$

Let's assume the eigenvalues are non-negative and sorted so that:

$$\lambda_1 > \lambda_2 > \lambda_3 \geq \cdots \geq \lambda_n \geq 0$$

Now let *j* go to infinity:

$$X^\infty = \lim_{j\to\infty} S^j X^0 = \lim_{j\to\infty} \sum_i^n a_i \lambda_i^j v_i$$

If $\lambda_1 > 1$, then:

If $\lambda_1 < 1$, then:

If $\lambda_1 = 1$, then:

Evaluation masks

What are the eigenvalues and eigenvectors of our cubic B-spline subdivision matrix?

limit position — limit derivative

$$\lambda_1 = 1 \qquad \lambda_2 = \frac{1}{2} \qquad \lambda_3 = \frac{1}{4}$$

$$v_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad v_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \qquad v_3 = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}$$

We're OK!

But where did the x-coordinates end up?

What about the y-coordinates?

Evaluation masks, cont'd

To finish up, we need to compute a_1. First, we can reorganize the expansion of X into the eigenbasis:

$$X^0 = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n = \begin{bmatrix} \vdots & \vdots & & \vdots \\ v_1 & v_2 & \cdots & v_n \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \mathbf{V}A$$

We can then solve for the coefficients in this new basis:

$$A = \mathbf{V}^{-1} X^0$$

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \cdots & u_1^T & \cdots \\ \cdots & u_2^T & \cdots \\ & \vdots & \\ \cdots & u_n^T & \cdots \end{bmatrix} X^0$$

Now we can compute the limit position of the x-coordinate:

$$x_C^\infty = a_1 = u_1^T X^0$$

We call u_1 the **evaluation mask**.

Evaluation masks, cont'd

Note that we need not start with the 0th level control points and push them to the limit.

If we subdivide and average the control polygon *j* times, we can push the vertices of the refined polygon to the limit as well:

$$x^{\infty} = S^{\infty} X^{j} = u_1^T X^{j}$$

The same result obtains for the y-coordinate:

$$y^{\infty} = S^{\infty} Y^{j} = u_1^T Y^{j}$$

Left eigenvectors

What are these *u*-vectors? Consider the eigenvector relation:

$$Sv_i = \lambda_i v_i$$

We can re-write this as a matrix:

$$S\mathbf{V} = \mathbf{V}\Lambda$$

where Λ is a diagonal matrix filled with the eigenvalues of S.

Now lets multiply both sides by $\mathbf{V}^{-1}$ from the left and right and then simplify:

$$\mathbf{V}^{-1}(S\mathbf{V})\mathbf{V}^{-1} = \mathbf{V}^{-1}(\mathbf{V}\Lambda)\mathbf{V}^{-1}$$
$$\mathbf{V}^{-1}S = \Lambda\mathbf{V}^{-1}$$
$$\mathbf{U}S = \Lambda\mathbf{U}$$

Thus, we find that the *u*-vectors obey the relation:

$$u_i^T S = \lambda_i u_i^T$$

These are the "**left eigenvectors**" of S. (Alternatively, they are the eigenvectors of S^T.)

Recipe for subdivision curves

The evaluation mask for the cubic B-spline is:

$$\frac{1}{6}\begin{pmatrix}1 & 4 & 1\end{pmatrix}$$

Now we can cook up a simple procedure for creating subdivision curves:

- Subdivide (split+average) the control polygon a few times. Use the averaging mask.
- Push the resulting points to the limit positions. Use the evaluation mask.

Tangent analysis

What is the tangent to the cubic B-spline curve?

First, let's consider how we represent the x and y coordinate neighborhoods:

$$X^0 = a_1 v_1 + a_2 v_2 + a_3 v_3$$
$$Y^0 = b_1 v_1 + b_2 v_2 + b_3 v_3$$

We can view the point neighborhoods then as:

$$Q^0 = \begin{bmatrix} X^0 & Y^0 \end{bmatrix} = v_1\begin{bmatrix} a_1 & b_1 \end{bmatrix} + v_2\begin{bmatrix} a_2 & b_2 \end{bmatrix} + v_3\begin{bmatrix} a_3 & b_3 \end{bmatrix}$$

After *j* subdivisions, we would get:

$$\begin{aligned} Q^j &= S^j\left\{v_1\begin{bmatrix} a_1 & b_1 \end{bmatrix} + v_2\begin{bmatrix} a_2 & b_2 \end{bmatrix} + v_3\begin{bmatrix} a_3 & b_3 \end{bmatrix}\right\} \\ &= \lambda_1^j v_1\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \lambda_2^j v_2\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j v_3\begin{bmatrix} a_3 & b_3 \end{bmatrix} \end{aligned}$$

We can write this more explicitly as:

$$\begin{bmatrix} Q_L^j \\ Q_C^j \\ Q_R^j \end{bmatrix} = \lambda_1^j \begin{bmatrix} v_{1,L} \\ v_{1,C} \\ v_{1,R} \end{bmatrix}\begin{bmatrix} a_1 & b_1 \end{bmatrix} + \lambda_2^j \begin{bmatrix} v_{2,L} \\ v_{2,C} \\ v_{2,R} \end{bmatrix}\begin{bmatrix} a_2 & b_2 \end{bmatrix} + \lambda_3^j \begin{bmatrix} v_{3,L} \\ v_{3,C} \\ v_{3,R} \end{bmatrix}\begin{bmatrix} a_3 & b_3 \end{bmatrix}$$

Tangent analysis (cont'd)

The tangent to the curve is along the direction:

$$\mathbf{t} = \lim_{j\to\infty}\left(Q_R^j - Q_C^j\right)$$

What's wrong with this definition?

Instead, we'll find the *normalized* tangent direction :

$$\mathbf{t} = \lim_{j\to\infty}\frac{Q_R^j - Q_C^j}{\left\|Q_R^j - Q_C^j\right\|}$$

Now, let's look at the "right" and "center" points in isolation:

$$\begin{aligned}Q_R^j &= \lambda_1^j v_{1,R}\begin{bmatrix}a_1 & b_1\end{bmatrix} + \lambda_2^j v_{2,R}\begin{bmatrix}a_2 & b_2\end{bmatrix} + \lambda_3^j v_{3,R}\begin{bmatrix}a_3 & b_3\end{bmatrix}\\ Q_C^j &= \lambda_1^j v_{1,C}\begin{bmatrix}a_1 & b_1\end{bmatrix} + \lambda_2^j v_{2,C}\begin{bmatrix}a_2 & b_2\end{bmatrix} + \lambda_3^j v_{3,C}\begin{bmatrix}a_3 & b_3\end{bmatrix}\end{aligned}$$

The difference between these is:

$$\begin{aligned}Q_R^j - Q_C^j = {} & \lambda_1^j\left(v_{1,R} - v_{1,C}\right)\begin{bmatrix}a_1 & b_1\end{bmatrix} + \\ & \lambda_2^j\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix} + \lambda_3^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix}a_3 & b_3\end{bmatrix}\\ = {} & \lambda_2^j\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix} + \lambda_3^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix}a_3 & b_3\end{bmatrix}\end{aligned}$$

The tangent mask

And now computing the tangent:

$$\begin{aligned}\lim_{j\to\infty}\frac{Q_R^j - Q_C^j}{\left\|Q_R^j - Q_C^j\right\|} &= \lim_{j\to\infty}\frac{\lambda_2^j\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix} + \lambda_3^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix}a_3 & b_3\end{bmatrix}}{\left\|\lambda_2^j\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix} + \lambda_3^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix}a_3 & b_3\end{bmatrix}\right\|}\\ &= \lim_{j\to\infty}\frac{\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix} + \left(\frac{\lambda_3}{\lambda_2}\right)^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix}a_3 & b_3\end{bmatrix}}{\left\|\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix} + \left(\frac{\lambda_3}{\lambda_2}\right)^j\left(v_{3,R} - v_{3,C}\right)\begin{bmatrix}a_3 & b_3\end{bmatrix}\right\|}\\ &= \frac{\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix}}{\left\|\left(v_{2,R} - v_{2,C}\right)\begin{bmatrix}a_2 & b_2\end{bmatrix}\right\|}\\ &= \frac{\begin{bmatrix}a_2 & b_2\end{bmatrix}}{\left\|\begin{bmatrix}a_2 & b_2\end{bmatrix}\right\|}\\ &= \frac{\begin{bmatrix}u_2^T X^0 & u_2^T Y^0\end{bmatrix}}{\left\|\begin{bmatrix}u_2^T X^0 & u_2^T Y^0\end{bmatrix}\right\|}\\ &= \frac{u_2^T Q^0}{\left\|u_2^T Q^0\right\|}\end{aligned}$$

Thus, we can compute the tangent using the *second* left eigenvector! This analysis holds for general subdivision curves and gives us the **tangent mask**.

Approximation vs. Interpolation of Control Points

Previous subdivision scheme *approximated* control points. Can we *interpolate* them?

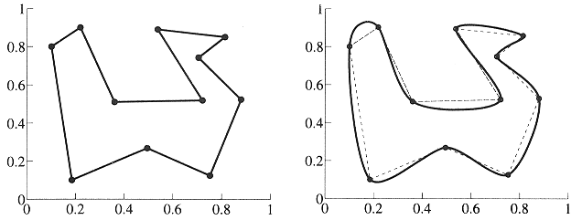
Yes: **DLG interpolating scheme (1987)**

Slight modification to subdivision algorithm:

- splitting step introduces midpoints
- averaging step *only changes midpoints*

For DLG (Dyn-Levin-Gregory), use:

$$r = \frac{1}{16}(-2, 5, 10, 5, -2)$$



Since we are only changing the midpoints, the points after the averaging step do not move.