**Subdivision curves**

Idea:
- repeatedly refine the control polygon
  \[ p^1 \rightarrow p^2 \rightarrow p^3 \rightarrow \ldots \]
- 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

**Reading**

Recommended:

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

\[ MV = V \Lambda \]
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:

\[ r = \]

Lane-Riesenfeld algorithm (1980)

Use averaging masks from Pascal’s triangle:

\[ r = \frac{1}{2^n} \binom{n}{0} \binom{n}{1} \cdots \binom{n}{n} \]

Gives B-splines of degree \( n+1 \).

\[ n=0: \]

\[ n=1: \]

\[ n=2: \]

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:

\[ \frac{1}{4} (1 \ 2 \ 1) \]

Now consider what happens during splitting and averaging in a small neighborhood:

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

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

\[
\begin{pmatrix}
L_j \\
C_j \\
R_j
\end{pmatrix} = \frac{1}{8} \begin{pmatrix}
4 & 4 & 0 \\
1 & 6 & 1 \\
0 & 4 & 4
\end{pmatrix} \begin{pmatrix}
L_{j-1} \\
C_{j-1} \\
R_{j-1}
\end{pmatrix}
\]

\[Q_j = SQ_{j-1}\]

Where the \( L, R, C \)’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:

\[
\begin{pmatrix}
x_j^L \\
x_j^C \\
x_j^R
\end{pmatrix} = \frac{1}{8} \begin{pmatrix}
4 & 4 & 0 \\
1 & 6 & 1 \\
0 & 4 & 4
\end{pmatrix} \begin{pmatrix}
x_{j-1}^L \\
x_{j-1}^C \\
x_{j-1}^R
\end{pmatrix}
\]

\[x_j^L = SX_{j-1}^L\]

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:

\[ Sv = \lambda v \]

We say that \( v \) is an eigenvector with eigenvalue \( \lambda \).

An \( nxn \) 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:

\[ X = \sum_{i=1}^{n} a_i v_i \]

To infinity, but not beyond...

Now let’s apply the matrix to the vector \( X \):

\[ X' = SX' = S \sum_{i=1}^{n} a_i v_i = \sum_{i=1}^{n} a_i S v_i = \sum_{i=1}^{n} a_i \lambda_i v_i \]

Applying it \( j \) times:

\[ X' = S^jX' = S^j \sum_{i=1}^{n} a_i v_i = \sum_{i=1}^{n} a_i S^j v_i = \sum_{i=1}^{n} a_i \lambda_i^j v_i \]

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:

\[ X' = \lim_{j \to \infty} S^jX' = \lim_{j \to \infty} \sum_{i=1}^{n} a_i \lambda_i^j v_i \]

If \( \lambda_i > 1 \), then:

If \( \lambda_i < 1 \), then:

If \( \lambda_i = 1 \), then:
Evaluation masks

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

\[
\begin{align*}
\lambda_1 &= 1 \\
\lambda_2 &= \frac{1}{2} \\
\lambda_3 &= \frac{1}{4}
\end{align*}
\]

\[
\begin{align*}
v_1 &= \begin{pmatrix} 1 \end{pmatrix} \\
v_2 &= \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \\
v_3 &= \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}
\end{align*}
\]

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:

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

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

\[
\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \mathbf{V}^{-1} X^0
\]

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

\[
x_{x}^\infty = a_1 = u_1^T X^0
\]

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:

\[
S v_j = \lambda_j v_j
\]

We can re-write this as a matrix:

\[
S \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \lambda_1 v_1 \\ \lambda_2 v_2 \\ \lambda_3 v_3 \end{bmatrix}
\]

\[
S \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}
\]

\[
SV = \mathbf{V} \Lambda
\]

where \( \mathbf{V} \) is the concatenation of the eigenvectors into a matrix and \( \Lambda \) is a diagonal matrix filled with the eigenvalues of \( S \).
**Left eigenvectors (cont’d)**

Now let’s multiply both sides by $V^{-1}$ from the left and right and then simplify:

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

$$U^T S = \Lambda U$$

If we “de-construct” this relation, we get:

$$U^T S = \Lambda U$$

$$\begin{bmatrix}
 u_1^T \\
 u_2^T \\
 u_3^T 
\end{bmatrix} S =
\begin{bmatrix}
 \lambda_1 & 0 & 0 \\
 0 & \lambda_2 & 0 \\
 0 & 0 & \lambda_3 
\end{bmatrix}
\begin{bmatrix}
 u_1^T \\
 u_2^T \\
 u_3^T 
\end{bmatrix}$$

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 $S^T$.)

**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:

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

After $j$ subdivisions, we would get:

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

We can write this more explicitly as:

$$L = \begin{bmatrix} \lambda_1 & 0 & 0 \\
 \lambda_2 & 0 & 0 \\
 \lambda_3 & 0 & 0 
\end{bmatrix}
\begin{bmatrix} v_1 & a_1 & b_1 \\
 v_2 & a_2 & b_2 \\
 v_3 & a_3 & b_3 
\end{bmatrix}$$

**Recipe for subdivision curves**

The evaluation mask for the cubic B-spline is:

$$\frac{1}{6}(1 \ 4 \ 1)$$

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 (cont’d)**

The tangent to the curve is along the direction:

$$t = \lim_{j \to \infty} (R^j - C^j)$$

What’s wrong with this definition?

Instead, we’ll find the normalized tangent direction:

$$t = \lim_{j \to \infty} \frac{R^j - C^j}{\|R^j - C^j\|}$$

Now, let’s look at the “right” and “center” points in isolation:

$$R^j = \lambda_1 v_{1R} + \lambda_2 v_{2R} + \lambda_3 v_{3R}$$

$$C^j = \lambda_1 v_{1C} + \lambda_2 v_{2C} + \lambda_3 v_{3C}$$

The difference between these is:

$$R^j - C^j = \lambda_1 (v_{1R} - v_{1C}) + \lambda_2 (v_{2R} - v_{2C}) + \lambda_3 (v_{3R} - v_{3C})$$

$$= \lambda_1 (v_{2R} - v_{2C}) + \lambda_2 (v_{3R} - v_{3C})$$
The tangent mask

And now computing the tangent:

\[
\lim_{j \to \infty} \frac{R' - C'}{R' - C} = \lim_{j \to \infty} \frac{\lambda_1 (v_{s,b} - v_{s,c}) [a_j b_j] + \lambda_2 (v_{s,b} - v_{s,c}) [a_j b_j] + \lambda_3 (v_{s,b} - v_{s,c}) [a_j b_j]}{\|v_{s,b} - v_{s,c}\| [a_j b_j] + \frac{\lambda_1}{\lambda_2} [v_{s,b} - v_{s,c}] [a_j b_j] + \frac{\lambda_2}{\lambda_3} [v_{s,b} - v_{s,c}] [a_j b_j]}
\]

\[
= \frac{\|v_{s,b} - v_{s,c}\| [a_j b_j]}{\|v_{s,b} - v_{s,c}\| [a_j b_j]}
\]

\[
= \frac{[a_j b_j]}{[a_j b_j]}
\]

\[
= \frac{[u_i^x X^0 \ y^0]}{[u_i^x X^0 \ y^0]}
\]

\[
= \frac{u_i^x Q_i^x}{[u_i^x Q_i^x]}
\]

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

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.