Subdivision curves

Idea:
- repeatedly refine the control polygon
- 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:

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

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:

\[
\begin{pmatrix}
1 & 1 & 1 \\
0 & 2 & 0
\end{pmatrix}
\]

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:

\[
\begin{pmatrix}
L^i & C^i & R^i
\end{pmatrix} = \begin{pmatrix}
1/2 & 1/2 & 0 \\
1/8 & 3/4 & 1/8 \\
0 & 1/2 & 1/2
\end{pmatrix} \begin{pmatrix}
L^{i-1} & C^{i-1} & R^{i-1}
\end{pmatrix}
\]

\[
Q^i = SQ^{i-1}
\]

Where the \(L, R, C\)'s are (for convenience) row vectors and \(S\) is the local subdivision matrix.

We can expand these row vectors:

\[
\begin{pmatrix}
L_x^j & L_y^j & C_x^j & C_y^j & R_x^j & R_y^j
\end{pmatrix} = \begin{pmatrix}
1/2 & 1/2 & 0 \\
1/8 & 3/4 & 1/8 \\
0 & 1/2 & 1/2
\end{pmatrix} \begin{pmatrix}
L_{x}^{j-1} & L_{y}^{j-1} & C_{x}^{j-1} & C_{y}^{j-1} & R_{x}^{j-1} & R_{y}^{j-1}
\end{pmatrix}
\]

and now think in terms of the behaviors of the \(x\) and \(y\) components, treating them as column vectors:

\[
\begin{pmatrix}
x^i \\
y^i
\end{pmatrix} = S \begin{pmatrix}
x^{i-1} \\
y^{i-1}
\end{pmatrix}
\]

Local subdivision matrix, cont’d

Let’s focus on just the behavior of the \(x\) components:

\[
\begin{pmatrix}
L_x^i & L_y^i & C_x^i & C_y^i & R_x^i & R_y^i
\end{pmatrix} = \begin{pmatrix}
1/2 & 1/2 & 0 \\
1/8 & 3/4 & 1/8 \\
0 & 1/2 & 1/2
\end{pmatrix} \begin{pmatrix}
L_{x}^{i-1} & L_{y}^{i-1} & C_{x}^{i-1} & C_{y}^{i-1} & R_{x}^{i-1} & R_{y}^{i-1}
\end{pmatrix}
\]

\[
\chi^j = S \chi^{j-1}
\]

(The analysis of the \(y\) component will be the same.)

Tracking the \(x\) components through subdivision:

\[
\chi^j = S \chi^{j-1} = S \cdot S \chi^{j-2} = S \cdot S \cdot S \chi^{j-3} = \ldots = S^j \chi^0
\]

The limit position of the \(x\)'s is then:

\[
\chi^\infty = \lim_{j \to \infty} S^j \chi^0
\]

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:

\[
Sv = \lambda v
\]

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:

\[
X = \sum_{i} a_i v_i
\]

To infinity, but not beyond…

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

\[
X^j = S^j \chi^0 = S^j \sum_{i} a_i v_i = \sum_{i} a_i S^j v_i = \sum_{i} a_i \lambda_i^j v_i
\]

Applying it \(j\) times:

\[
X^j = S^j X = S^j \sum_{i} a_i v_i = \sum_{i} a_i S^j v_i = \sum_{i} a_i \lambda_i^j v_i
\]

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

\[
\lambda_1 > \lambda_2 > \lambda_3 \geq \cdots \geq \lambda_n \geq 0
\]

(The form of the inequalities is important.) Now let \(j\) go to infinity:

\[
\chi^\infty = \lim_{j \to \infty} S^j \chi^0 = \lim_{j \to \infty} \sum_{i} a_i \lambda_i^j v_i
\]

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?

\[ \lambda_1 = 1 \quad \lambda_2 = \frac{1}{2} \quad \lambda_3 = \frac{1}{4} \]

\[
\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{v}_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{v}_3 = \begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix}
\]

We're OK! (In fact, for proper subdivision matrices, the first eigenvector will always be \([1\ldots1]^T\). Why?)

So, the first thing we do is expand the \(x\) and \(y\) components of a vertex in terms of the eigenbasis:

\[ X = \sum_{i} a_i \mathbf{v}_i \quad Y = \sum_{i} b_i \mathbf{v}_i \]

Then, after infinite subdivision, the \(x\) components will end up at...?

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:

\[ X^0 = a_1 \mathbf{v}_1 + a_2 \mathbf{v}_2 + \cdots + a_n \mathbf{v}_n \]

\[
\begin{bmatrix}
\vdots & \vdots & \vdots \\
\mathbf{v}_1 & \mathbf{v}_2 & \cdots \\
\vdots & \vdots & \vdots 
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_n 
\end{bmatrix} = \mathbf{v} A
\]

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

\[
A = \mathbf{v}^{-1} X^0
\]

\[
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_n 
\end{bmatrix} = \begin{bmatrix}
\cdots & \mathbf{u}_1^T & \cdots \\
\vdots & \vdots & \vdots \\
\cdots & \mathbf{u}_n^T & \cdots 
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_n 
\end{bmatrix} = \mathbf{v} A
\]

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

\[ X^\infty = a_1 = \mathbf{u}_1^T X^0 \]

We call \(\mathbf{u}_1\) the evaluation mask.

Left eigenvectors

What are these \(u\)-vectors? Consider the eigenvector relation:

\[ \mathbf{S} \mathbf{v}_i = \lambda_i \mathbf{v}_i \]

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

\[ X^\infty = \mathbf{S}^\infty X^j = \mathbf{u}_1^T X^j \]

The same result obtains for the \(y\)-coordinate:

\[ Y^\infty = \mathbf{S}^\infty Y^j = \mathbf{u}_1^T Y^j \]

We can re-write this as a matrix (we’ll use the 3x3 case for illustration here):

\[
\mathbf{S} = \begin{bmatrix}
\lambda_1 & \lambda_2 & \lambda_3 \\
\lambda_2 & \lambda_1 & \lambda_3 \\
\lambda_3 & \lambda_2 & \lambda_1 
\end{bmatrix}
\]

\[
\lambda_1 \begin{bmatrix}
0 & 0 \\
0 & \lambda_2 \\
0 & 0 
\end{bmatrix}
\]

\[
\mathbf{S} \mathbf{v} = \mathbf{V} \Lambda
\]

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

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

$$V^{-1}(SV)V^{-1} = V^{-1}(AV)V^{-1}$$

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

$$US = \Lambda U$$

where $U \equiv V^{-1}$. If we “de-construct” this relation, we get:

$$US = \Lambda U$$

$$\begin{bmatrix}
  u'_1 \\
  u'_2 \\
  u'_3 \\
  u'_4
\end{bmatrix}
= \begin{bmatrix}
  \lambda_1 & 0 & 0 & u'_1 \\
  0 & \lambda_2 & 0 & u'_2 \\
  0 & 0 & \lambda_3 & u'_3 \\
  0 & 0 & 0 & u'_4
\end{bmatrix}$$

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

$$u'_i S = \lambda_i u'_i$$

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:

$$\begin{pmatrix}
  1 & 2 & 1 \\
  6 & 3 & 6
\end{pmatrix}$$

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:

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

After $j$ subdivisions, we would get:

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

We can write this more explicitly as:

$$\begin{bmatrix}
  L^j \\
  C^j \\
  R^j
\end{bmatrix} = \begin{bmatrix}
  v_{1,1}^j & \lambda_1 j v_{1,c} \\
  v_{2,1} & \lambda_2 j v_{2,c} \\
  v_{3,1} & \lambda_3 j v_{3,c}
\end{bmatrix}$$

$$\begin{bmatrix}
  v_{1,1}^j & \lambda_1 j v_{1,r} \\
  v_{2,1} & \lambda_2 j v_{2,r} \\
  v_{3,1} & \lambda_3 j v_{3,r}
\end{bmatrix}$$

$$\begin{bmatrix}
  a_1 v_{1,1} & b_1 v_{1,1} \\
  a_2 v_{2,1} & b_2 v_{2,1} \\
  a_3 v_{3,1} & b_3 v_{3,1}
\end{bmatrix} + \begin{bmatrix}
  a_1 v_{1,c} & b_1 v_{1,c} \\
  a_2 v_{2,c} & b_2 v_{2,c} \\
  a_3 v_{3,c} & b_3 v_{3,c}
\end{bmatrix}$$

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:

$$R^j = \lambda_{1,j} ^j \left[ v_{1,1} \left[ a_1, b_1 \right] + \lambda_{2,j} ^j \left[ v_{2,1} \left[ a_2, b_2 \right] + \lambda_{3,j} \left[ v_{3,1} \left[ a_3, b_3 \right] \right] \right] \right]$$

$$C^j = \lambda_{1,j} \left[ v_{1,c} \left[ a_1, b_1 \right] + \lambda_{2,j} \left[ v_{2,c} \left[ a_2, b_2 \right] + \lambda_{3,j} \left[ v_{3,c} \left[ a_3, b_3 \right] \right] \right] \right]$$

The difference between these is:

$$R^j - C^j = \lambda_{1,j} \left[ v_{1,1} \left[ a_1, b_1 \right] + \lambda_{2,j} \left[ v_{2,1} \left[ a_2, b_2 \right] + \lambda_{3,j} \left[ v_{3,1} \left[ a_3, b_3 \right] \right] \right] \right]$$
The tangent mask

And now computing the tangent:

\[
\lim_{j \to \infty} \frac{R' - C'}{R' - C} = \lim_{j \to \infty} \frac{\lambda_j \left( (v_{3,0} - v_{2,1}) \begin{bmatrix} a_3 & b_3 \\ a_2 & b_2 \end{bmatrix} + \lambda_j \left( (v_{1,0} - v_{3,1}) \begin{bmatrix} a_1 & b_1 \\ a_3 & b_3 \end{bmatrix} \right) \right)}{\lambda_j \left( (v_{3,0} - v_{2,1}) \begin{bmatrix} a_3 & b_3 \\ a_2 & b_2 \end{bmatrix} + \lambda_j \left( (v_{1,0} - v_{3,1}) \begin{bmatrix} a_1 & b_1 \\ a_3 & b_3 \end{bmatrix} \right) \right)}
\]

\[
\lambda_j \left( (v_{3,0} - v_{2,1}) \begin{bmatrix} a_3 & b_3 \\ a_2 & b_2 \end{bmatrix} + \lambda_j \left( (v_{1,0} - v_{3,1}) \begin{bmatrix} a_1 & b_1 \\ a_3 & b_3 \end{bmatrix} \right) \right) = \lambda_j \left( (v_{3,0} - v_{2,1}) \begin{bmatrix} a_3 & b_3 \\ a_2 & b_2 \end{bmatrix} + \lambda_j \left( (v_{1,0} - v_{3,1}) \begin{bmatrix} a_1 & b_1 \\ a_3 & b_3 \end{bmatrix} \right) \right)
\]

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.