3.1 Parametric Curves

- The graph of a function \( y = 2x - x^2 \) is a set of points \([y] = [2x - x^2]\).
- A parametric curve is of the form \([y] = [f(t)]\).
- A parametric curve for a straight line:
  \[ [x, y] = [(1-t)ax + t bx, (1-t)ay + t by] \]

**Ex 3.1**
- The curve defined by \([y] = [2t - t^2]\) is identical to the curve given as the graph of \(y = 2x - x^2\).

**Ex 3.2**
- By rotating the above curve by 90°, we get a curve \([x, y] = \left[ t - 2t^2 + t^3 \right] \)

An important difference between parametric curves and the graphs of functions:

- The concept of “zero slope” or “horizontal tangents” is important. It characterizes extreme points.
- But, for parametric curves, zero slopes do not signify geometric properties. A simple rotation changes horizontal tangents. The geometry of a curve does not change under rotations or other affine maps.

- A parametric curve in 3D is \([x, y, z] = [f(t), f(t), f(t)]\)

**Ex:** The helix is given by \([x, y, z] = [\cos(t), \sin(t), t]\)
3.2 Cubic Bézier Curves

**Example 3.3**

\[
X(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 2t & -3(1-t)+t^3 \\ 3(1-t)^2+3(1-t)t^2 & 0 \end{bmatrix}
\]

\[
= (1-t)^3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + 3(1-t)^2t \begin{bmatrix} 0 \\ 1 \end{bmatrix} + x(t)t^2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} + t^3 \begin{bmatrix} 1 \\ 0 \end{bmatrix}.
\]

The polynomial curve is expressed in terms of a combination of points. We may compute \(X(0.5) = \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix}\).

A cubic Bézier curve is defined by

\[
X(t) = (1-t)^3 \mathbf{b}_0 + 3(1-t)^2t \mathbf{b}_1 + 3(1-t)t^2 \mathbf{b}_2 + t^3 \mathbf{b}_3,
\]

where \(\mathbf{b}_i\), the Bézier control points, form the Bézier polygon of the curve.

\[
X(t) = B^3(t) \cdot \mathbf{b}_0 + B^3_1(t) \cdot \mathbf{b}_1 + B^3_2(t) \cdot \mathbf{b}_2 + B^3_3(t) \cdot \mathbf{b}_3.
\]

\(\rightarrow\) the cubic Bernstein polynomials.

**Properties of Cubic Bézier Curves**

1. **Endpoint interpolation**: \(X(0) = \mathbf{b}_0\), \(X(1) = \mathbf{b}_3\).

2. **Symmetry**: Two polygons \(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\) and \(\mathbf{b}_3, \mathbf{b}_2, \mathbf{b}_1, \mathbf{b}_0\) describe the same curve; but the directions are different.

3. **Invariance under rotations**:

4. **Invariance under affine maps**: If an affine map is applied to the control polygon, the curve is mapped by the same map.

5. **Convex hull property**: For \(t \in [0, 1]\), the point \(X(t)\) is on the convex hull of the control polygon.

6. **Linear precision**: If \(\mathbf{b}_1 = \frac{1}{3} \mathbf{b}_0 + \frac{2}{3} \mathbf{b}_3\) and \(\mathbf{b}_2 = \frac{1}{3} \mathbf{b}_0 + \frac{2}{3} \mathbf{b}_1\), then the curve \(X(t) = (1-t)\mathbf{b}_0 + t \mathbf{b}_3\); linear interpolation.

7. For \(t \in [0, 1]\), \(X(t)\) may not stay within the convex hull of the control polygon.
3.3 Derivatives

\[ x(t) = -3(1-t)^2l_0 + \left[ 3(1-t)^2 - 6(1-t)t \right]l_1 + \left[ 6(1-t)t - 3t^2 \right]l_2 + 3t^2 l_3 \]

\[ = (1-t)^2 \cdot 3[1b_1 - l_0] + 2(1-t)t \cdot 3[1b_2 - l_1] + t^2 \cdot 3[1b_3 - l_2] \]

\[ = (1-t)^2 \cdot 3\Delta l_0 + 2(1-t)t \cdot 3\Delta l_1 + t^2 \cdot 3\Delta l_2 \]

\[
\Rightarrow \text{the forward difference} \quad ^{2}
\]

\[ = 3 \left( B_0^2(t) \cdot \Delta l_0 + B_1^2(t) \cdot \Delta l_1 + B_2^2(t) \cdot \Delta l_2 \right) \]

\[
\Rightarrow \text{the quadratic Bernstein basis functions}
\]

- For parametric curves, "derivative curves" produce vectors rather than points.
- The coefficients are the difference vectors of the polygon, scaled by 3, the degree of the curve.
- \( x'(0) = 3 \Delta l_0, x'(1) = 3 \Delta l_2 \)

3.4 The de Casteljau Algorithm

\[
\begin{cases}
  l_0'(t) = (1-t)l_0 + tl_1 \\
  l_1'(t) = (1-t)l_1 + tl_2 \\
  l_2'(t) = (1-t)l_2 + tl_3
\end{cases}
\]

\[
\Rightarrow \begin{cases}
  l_0^2(t) = (1-t)l_0'(t) + tl_1'(t) \\
  l_1^2(t) = (1-t)l_1'(t) + tl_2'(t)
\end{cases}
\]

\[ l_0^3(t) = (1-t)l_0^2(t) + tl_1^2(t) \]

\[ \Rightarrow l_0^3(t) \text{ is the control point} \]

A convenient schematic tool for the algorithm

- In the implementation, \( l_0^1 \) is calculated and loaded into \( l_0^0 \) since \( l_0^0 \) is never needed.
- \( l_0^0 \) array of control points is sufficient!

- \( x'(t) = 3 [l_0^2(t) - l_0^0(t)] \):

The derivative is essentially a byproduct of point evaluation!
3.5 Subdivision

o. Two polygons \( l^0, l^1_0, l^2_0, l^3_0 \) and \( l^0, l^1_1, l^2_1, l^3_1 \)
define two segments corresponding to \([0, \frac{1}{2}]\) and \([\frac{1}{2}, 1]\),

o. Subdivision at \( t=0.5 \) splits the curve \( \gamma(t) \) respectively.

at the parameter midpoint, but the two arcs are not

o. Subdivision may be repeated: \( \gamma(t) \) of equal length.

Each of the two new control polygons may be subdivided,

The resulting sequence of control polygons converges to the

o. Another application is the intersection of a curve with a line

1. Find the AABB (axis aligned bounding box) of the polygon.
2. If no intersection between the AABB and the line, EXIT.
3. Else if the AABB is smaller than \( \varepsilon \),
    report the center of the AABB.
4. Else, subdivide the curve at \( t=0.5 \) into two
    and repeat the same procedure to each segment.

The curve-curve intersection can be done in a similar
way. In this case, the curve with a bigger AABB
is subdivided.
Chap 4. Bézier Curves

4.1 Bézier Curves

A Bézier curve of degree $n$ is defined by

$$x(t) = l_0 B_0^n(t) + l_1 B_1^n(t) + \ldots + l_n B_n^n(t),$$

where $B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i$ is Bernstein polynomial.

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

if $0 \leq i \leq n$

otherwise.

4.2 Derivatives Revisited

$$x'(t) = \frac{d}{dt} \left[ l_0 B_0^n(t) + \ldots + l_{n-1} B_{n-1}^n(t) \right],$$

where $\Delta l_{i-1} = l_i - l_{i-1}$.

$$\triangledown^k x(t) = \frac{d^k}{dt^k} \left[ l_0 B_0^n(t) + \ldots + l_{n-k} B_{n-k}^n(t) \right],$$

where $\Delta^k l_{i-1} = \Delta^{k-1} l_i - \Delta^{k-1} l_{i-1}$ and $\Delta^0 l_{i-1} = l_i$.

Ex.

$$\Delta^2 l_{i-1} = \Delta^2 l_i - \Delta l_{i-1} = l_{i+2} - 2l_{i+1} + l_i$$

$$\Delta^3 l_{i-1} = \Delta^3 l_i - \Delta^2 l_{i-1} = l_{i+3} - 3l_{i+2} + 3l_{i+1} - l_i$$

$$\Delta^4 l_{i-1} = \Delta^4 l_i - \Delta^3 l_{i-1} = l_{i+4} - 4l_{i+3} + 6l_{i+2} - 4l_{i+1} + l_i$$

$$\begin{bmatrix}
  x^{(k)}(0) = \frac{n!}{(n-k)!} \Delta^k l_0 \\
  x^{(k)}(1) = \frac{n!}{(n-k)!} \Delta^k l_{n-k}
\end{bmatrix}$$

Ex.

For a cubic Bézier curve $x(t)$,

$$x''(0) = 3 \Delta^2 l_0 = 3 (l_2 - 2l_1 + l_0).$$
4.3 The de Casteljau Algorithm Revisited

\[
\text{for } i = 1, \ldots, n
\]
\[
\text{for } r = 0, \ldots, n - r
\]
\[
b_i^r(t) = (1-t)b_{i-1}^r(t) + t b_{i+1}^r(t)
\]

\( X(t) = b_n^0(t) \) is the point on the curve.

The de Casteljau algorithm subdivides the curve into a left and a right segment. Their control polygons are given by

\[ b_0, b_1, \ldots, b_n \text{ and } b_0, b_1, \ldots, b_n. \]

The de Casteljau algorithm provides a way for computing the first derivative and the second derivative

\[
X'(t) = n \left[ b_i^{n-1}(t) - b_0^{n-1}(t) \right]
\]
\[
X''(t) = n(n-1) \left[ b_i^{n-2}(t) - 2b_i^{n-2}(t) + b_0^{n-2}(t) \right]
\]

4.4 The Matrix Form and Monomials Revisited

\( X(t) = N^T B, \) where

\[ N = \begin{bmatrix} B_{21}(t) \\ B_{22}(t) \end{bmatrix} \]

and

\[ B = \begin{bmatrix} b_0 \\ b_n \end{bmatrix}. \]

\( X(t) = a_0 + a_1 t + \cdots + a_n t^n, \)

where

\[ a_0 = b_0 \text{ and } a_i = \frac{a_i}{(n-i)!} \Delta^i b_0, \]

for \( i = 1, \ldots, n. \)
4.5 Degree Elevation

A Bézier curve of degree $n$ may be represented as a Bézier curve of degree $(n+1)$.

**Ex**

$$x(t) = (1-t)^3 b_0 + 2(1-t)t b_1 + t^2 b_2 : \text{a quadratic curve.}$$

Multiplying by $1 = [(1-t) + t]$, we get

$$x(t) = [(1-t)^3 + (1-t)^2 t] b_0 + 2[(1-t)^2 t + (1-t)t^2] b_1 + [(1-t)t^2 + t^3] b_2$$

$$= (1-t)^3 b_0 + 3(1-t)^2 t \left[ \frac{1}{3} b_0 + \frac{2}{3} b_1 \right] + 3(1-t)t^2 \left[ \frac{2}{3} b_1 + \frac{1}{3} b_2 \right] + t^3 b_2.$$

$$\therefore x(t) = B_0^3(t) b_0 + B_1^3(t) \left[ \frac{1}{3} b_0 + \frac{2}{3} b_1 \right] + B_2^3(t) \left[ \frac{2}{3} b_1 + \frac{1}{3} b_2 \right] + B_3^3(t) b_2.$$

**Ex 4.2**

$$x(t) = (1-t)^3 b_0 + 2(1-t)t b_1 + t^2 b_2,$$

where $b_0 = \left[ \begin{array}{c} 0 \\ 0 \end{array} \right]$, $b_1 = \left[ \begin{array}{c} 3 \\ 2 \end{array} \right]$, $b_2 = \left[ \begin{array}{c} 6 \\ 0 \end{array} \right]$.

Degree elevation will produce

$$x(t) = (1-t)^3 c_0 + 3(1-t)^2 t c_1 + 3(1-t)t^2 c_2 + t^3 c_3,$$

where $c_0 = b_0 = \left[ \begin{array}{c} 0 \\ 0 \end{array} \right]$, $c_3 = b_2 = \left[ \begin{array}{c} 6 \\ 0 \end{array} \right]$

$$c_1 = \frac{1}{3} b_0 + \frac{2}{3} b_1 = \left[ \begin{array}{c} 2 \\ 2 \end{array} \right], \quad c_2 = \frac{2}{3} b_1 + \frac{1}{3} b_2 = \left[ \begin{array}{c} 4 \\ 2 \end{array} \right].$$

$$\rightarrow 2\text{길\ 토끼\ 동\ 얇}}$$
Degree elevation of a Bézier curve of degree \( n \) to a Bézier curve of degree \( (n+1) \):

\[
\begin{align*}
C_0 &= l b_0 \\
C_1 &= \frac{c}{n+1} l b_{c-1} + (1 - \frac{c}{n+1}) l b_c \\
& \vdots \\
C_{n+1} &= l b_n
\end{align*}
\]

\[
\begin{bmatrix}
1 & * & * \\
* & * & * \\
* & * & 1
\end{bmatrix}
\begin{bmatrix}
l b_0 \\
l b_1 \\
l b_n
\end{bmatrix}
=
\begin{bmatrix}
C_0 \\
C_1 \\
C_{n+1}
\end{bmatrix}
\]

\[
\Rightarrow \text{ \((n+2) \times (n+1)\) matrix} \Rightarrow DB = C
\]

The process of degree elevation may be repeated. The resulting sequence of control polygons converges to the curve. However, convergence is too slow.

**Ex 4.3**

For \( n = 2 \),

\[
\begin{bmatrix}
1 & 0 & 0 \\
\frac{1}{3} & \frac{2}{3} & 0 \\
0 & \frac{2}{3} & \frac{1}{3} \\
0 & \frac{1}{3} & \frac{1}{3} \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
l b_0 \\
l b_1 \\
l b_2 \\
l b_3
\end{bmatrix}
=
\begin{bmatrix}
C_0 \\
C_1 \\
C_2 \\
C_3
\end{bmatrix}
\]
4.6 Degree Reduction

0. The inverse process of degree reduction is more important. Some CAD systems allow degrees up to 40, others use cubic curves. Reducing a curve of degree 40 to cubic is non-trivial. In practice, several cubic segments will be needed, involving an interplay between subdivision and degree reduction.

0. Degree reduction approximates a curve of degree \( (n+1) \) by a curve of degree \( n \). To approximate the solution of \( DB = C \), we solve \( DTDDB = DTC \).

0. The matrix \( DTD \) is independent of the given data, but dependent only on \( n \). To solve many degree reduction problems, we store the LU factorization of \( DTD \).

**Ex 4.4**

For \( n+1 = 3 \), \( DTD = \frac{1}{9} \begin{bmatrix} 10 & 2 & 0 \\ 2 & 8 & 2 \\ 0 & 2 & 10 \end{bmatrix} \)

For the degree-elevated cubic curve \( x(t) \) with \( c_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \), \( c_1 = \begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \\ \frac{1}{2} \end{bmatrix} \), \( c_2 = \begin{bmatrix} \frac{4}{3} \\ \frac{2}{3} \\ \frac{2}{3} \end{bmatrix} \), \( c_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \), we get

\( DTC = \frac{1}{3} \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \) corresponds to the \( y \)-components

\( \) corresponds to the \( x \)-components

\( \Rightarrow B = (DTD)^{-1} (DTC) = \begin{bmatrix} \frac{3}{3} \\ 0 \end{bmatrix} \)

\( \therefore b_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \), \( b_1 = \begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \end{bmatrix} \), \( b_2 = \begin{bmatrix} \frac{2}{3} \\ \frac{1}{3} \end{bmatrix} \).

0. In general, the degree-reduced curve may not pass through the original curve endpoint \( c_0 \) and \( c_{n+1} \).
4.8 Functional Bézier Curves

The graph of a functional curve can be thought of as a parametric curve of the form

\[
\begin{bmatrix}
y(t)
\end{bmatrix} = \begin{bmatrix}
x(t)
\end{bmatrix} = \begin{bmatrix}
g(t)
\end{bmatrix}.
\]

(A different name is a nonparametric curve)

1. One dimension is restricted to be a linear polynomial.

2. How to write a (polynomial) functional curve in Bézier form:

\[
g(t) = b_0 B_0^n(t) + \cdots + b_n B_n^n(t),
\]

\[
\Rightarrow \quad \begin{bmatrix}
1
\end{bmatrix} b(t) = \begin{bmatrix}
b_0 & \ldots & b_n
\end{bmatrix} B_0^n(t) + \cdots + \begin{bmatrix}
b_0 & \ldots & b_n
\end{bmatrix} B_n^n(t).
\]

4.9 More on Bernstein Polynomials

**Properties**

1. Partition of unity:

\[
B_0^n(t) + \cdots + B_n^n(t) = 1
\]

2. \(0 \leq B_i^n(t) \leq 1\).

3. Convex hull property:

\[x(t) \in CH(Ib_0, \ldots, Ib_n)\]

4. Symmetry of the Bernstein polynomials:

\[
B_i^n(t) = B_{n-i}^n(1-t)
\]

\[\Rightarrow\] This is reflected in Bézier curves by the symmetry property.