Algorithmic Geometry WS 2017/2018

Prof. Dr. Hans Hagen
Benjamin Karer M.Sc.

http://gfx.uni-kl.de/~alggeom

Bézier Curves

Contents

- Affine geometry
- Bernstein Polynomial
- Bézier Segments
- de Casteljau-algorithm
- Implementation and integration
- Interpolation
- End conditions for joining Bézier segments
- Rational Bézier curves

Affine Geometry

- An *Affine map* Φ can be described as a combination of a linear map A and a translation T:

$$\Phi(x) = Ax + T.$$

- An *Affine combination* is a linear combination

$$x = \sum_{i=0}^{n} \alpha_i x_i$$

such that the weights of all points sum up to 1: $\sum \alpha_i = 1$.

Properties of affine maps

- An affine map maps a line segment to a line segment.
- Linear interpolation is *affinly invariant*:

$$\Phi((1-\lambda)P + \lambda Q) = ((1-\lambda)\Phi(P) + \lambda\Phi(Q).$$

- Affine maps are ratio preserving.
- Parallel lines stay parallel under affine mapping.

Some examples of affine maps

- The identity. Set $A = I$ the identity matrix, and $T = 0$.
- Translation. It is given by $A = I$
- A scaling. It is given by $T = 0$, and A diagonal matrix.
- A rotation around z axis. $T = 0$, $A = \begin{pmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- A shear. $T = 0$. $A = \begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$

Bernstein Polynomials

We will express Bézier Curves in terms of Bernstein Polynomials. First, let's define Bernstein Polynomials.

Definition

Bernstein Polynomials, are defined explicitly by

$$B_i^m(t) := \frac{1}{(b-a)^m} \binom{m}{i} (t-a)^i (b-t)^{m-i}; 0 \leq i \leq m; t \in [a,b],$$

where binomial coefficients are given by

$$\binom{m}{i} = \begin{cases} \frac{m!}{i!(m-i)!} & \text{if } 0 \leq i \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

Bernstein Polynomials - Properties

1. The Bernstein polynomials of degree m are linearly independent and span the linear space of degree m. They form a basis of this space.
2. $\sum_{i=0}^{m} B_i^m(t) = 1$.
3. $B_i^m(t) \geq 0; t \in [a, b]; 0 \leq i \leq m$.
4. $\max B_i^m(t) = \frac{(b-a)\cdot i}{m} + a; t \in [a, b]; 0 < i < m$.
5. $B_i^m(t + a) = B_{m-i}^m(b - t)$.
6. $\frac{\mathrm{d}^p}{\mathrm{d}t^p} B_i^m(t) = \frac{1}{(b-a)^p} \cdot \frac{m!}{(m-p)!} \sum_{j=0}^{p} (-1)^{p-j} \cdot \binom{p}{j} \cdot B_{i-j}^{m-j}(t)$

Basic Idea

- approximate a polygon using Bernstein-polynomials $\rightarrow$ curve
- consider the polygon as a "'control-structure"' for generating curves

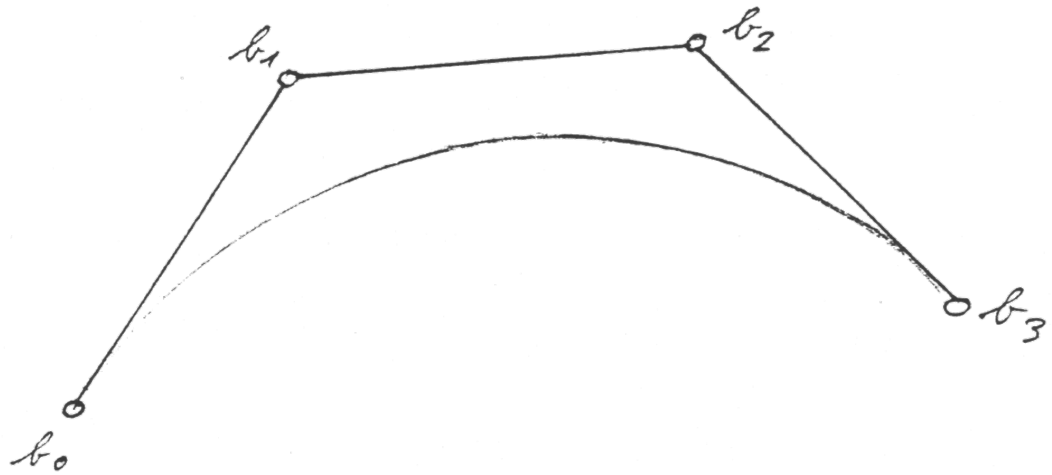

Figure: Bézier Curve

Bézier Curves

We consider a continuous mapping $F : [0,1] \to \mathbb{R}$

$$B_m(F;t) := \sum_{i=0}^{m} F\left(\frac{i}{m}\right) \cdot B_i^m(t) \quad 0 \le t \le 1$$

is the so-called Bernstein approximation of degree m of F.
In the special case of a polygon F_m with vertices $b_0 \dots b_n$,

$$F_m(t) = (m \cdot t - i)b_{i+1} + (i + 1 - m \cdot t)b_i$$

$$\text{with} \quad 0 \le i \le m-1 \text{ and } t \in \left[\frac{i}{m}, \frac{i+1}{m}\right],$$

we get:

$$B_m(F_m;t) = \sum_{i=0}^{m} b_i \cdot B_i^m(t) \quad t \in [0,1]$$

Bézier Curves

This motivates the following

Definition

$$X : [a, b] \to \mathbb{E}^3$$

$$X(t) := \sum_{i=0}^{m} b_i \cdot B_i^m(t) = \frac{1}{(b-a)^m} \sum_{i=0}^{m} b_i \binom{m}{i} (t-a)^i (b-a)^{m-i}$$

is called a Bézier-Segment of degree m,
with $b_i \in \mathbb{E}^3$, $i = 0, \ldots, m$ its Bézier-points.
The polygon is called *Bézier polygon*.

Convex Hull and Variation Diminishing Property

- The Bézier-segment is completely inside the convex hull of the control-polygon.
- Each point $X(t_0)$ is the point of gravity of the Bézier-points b_i with the weights $B_i^m(t_0)$.
- Control polygon and Bézier segment have start point and end point in common.
- Each straight line does not intersect the Bézier segment more often than it intersects the control polygon

Bézier Curves

The "Convex Hull Property" offers a good bounding-box strategy.
The "Variation Diminishing Property" "guarantees" a smooth curve.
An example of Bézier segment in matrix formulation:
$m = 3, [a, b] = [0, 1]$

$$
\begin{aligned}
X(t) &= b_0 \cdot (-t^3 + 3t^2 - 3t + 1) \\
&\quad + b_1 \cdot (3t^3 - 6t^2 + 3t) \\
&\quad + b_2 \cdot (-3t^3 + 3t^2) + b_3(t^3) \\
&= (t^3 t^2 t^1 t^0) \cdot \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}
\end{aligned}
$$

Bézier Curves

The Bernstein polynomials have another interesting property, which we can use for subdivision.

$$
\begin{aligned}
X(t_0) &= b_0(1-t_0)^3 + 3b_1(1-t_0)^2 \cdot t_0 + 3b_2(1-t_0)^2 + b_3 t_0^3 \\
&= b_0(1-t_0)^2(1-t_0) + b_1(1-t_0)^2 t_0 + 2b_1(1-t_0)^2 t_0 \\
&\quad + 2b_2(1-t_0)t_0^2 + b_2(1-t_0)t_0^2 + b_3 t_0^3 \\
&= \underbrace{\big(b_0(1-t_0) + b_1 \cdot t_0\big)}_{=:b_1^1} \cdot \underbrace{(1-t_0)^2}_{B_0^2} \\
&\quad + \underbrace{\big(b_1 \cdot (1-t_0) + b_2 \cdot t_0\big)}_{=:b_2^1} \cdot \underbrace{\big(2(1-t_0) \cdot t_0\big)}_{B_1^2} \\
&\quad + \underbrace{b_2 \cdot (1-t_0) + b_3 \cdot t_0\big)}_{=:b_3^1} \cdot \underbrace{t_0^2}_{B_2^2}
\end{aligned}
$$

Bézier Curves

$$
\begin{aligned}
= \quad & \underbrace{\left(b_1^1 \cdot (1-t_0) + b_2^1 \cdot t_0\right)}_{=:b_2^2} \cdot \underbrace{(1-t_0)}_{B_0^1} \\
& + \underbrace{\left(b_2^1 \cdot (1-t_0) + b_3^1 \cdot t_0\right)}_{=:b_3^2} \cdot \underbrace{t_0}_{B_1^1} \\
= \quad & \left(b_2^2(1-t_0) + b_3^2 t_0\right) = b_3^3 B_0^0 = b_3^3
\end{aligned}
$$

General principles

$$
\begin{aligned}
X(\widetilde{t_0}) &= b_0 B_0^m(\widetilde{t_0}) + \cdots + b_m B_m^m(\widetilde{t_0}) \quad \widetilde{t_0} := \frac{t_0 - a}{b - a} \\
&= b_0 B_0^{m-1}(\widetilde{t_0})(1 - \widetilde{t_0}) + b_1 B_0^{m-1}(\widetilde{t_0}) \cdot \widetilde{t_0} + \cdots \\
&\quad + b_{m-1} B_{m-1}^{m-1}(\widetilde{t_0})(1 - \widetilde{t_0}) + b_{m-1} B_{m-1}^{m-1}(\widetilde{t_0}) \cdot \widetilde{t_0} \\
&= \underbrace{(b_0(1 - \widetilde{t_0}) + b_1 \widetilde{t_0})}_{=: b_1^1} B_0^{m-1}(\widetilde{t_0}) \\
&\quad + \cdots + \underbrace{(b_{m-1}(1 - \widetilde{t_0}) + b_m \widetilde{t_0})}_{=: b_m^1} \cdot B_{m-1}^{m-1}
\end{aligned}
$$

linear interpolation $b_i^1 = b_{i-1}(1 - \widetilde{t_0}) + b_i \widetilde{t_0}$

Bézier Curves

Iterate:

$$
\begin{aligned}
X(\widetilde{t_0}) &= b_r^r \cdot B_0^{m-r}(\widetilde{t_0}) + \cdots + b_m^r \cdot B_{m-r}^{m-r}(\widetilde{t_0}) \\
&\quad \text{with } b_i^r = b_{i-1}^{r-1}(1-\widetilde{t_0}) + b_i^{r-1}\widetilde{t_0}, \\
&\quad \text{and } i = r, \ldots, m \\
&\quad \text{until we reach } r = m, \\
X(\widetilde{t_0}) &= b_m^m B_0^0(\widetilde{t_0}) = b_m^m
\end{aligned}
$$

De Casteljau Algorithm

The De Casteljau algorithm is used to obtain a point on the curve at parameter value $\widetilde{t_0}$ from the control polygon consisting of the points b_i.

Input: control polygon $P_m = b_0, \ldots, b_m$.

Steps:

1 $b_i^1 := (1 - \widetilde{t_0}) \cdot b_{i-1} + \widetilde{t_0} \cdot b_i \quad i = 1, \ldots, m$

2 $b_i^2 := (1 - \widetilde{t_0}) \cdot b_{i-1}^1 + \widetilde{t_0} \cdot b_i^1 \quad i = 2, \ldots, m$

m-1 $b_i^{m-1} := (1 - \widetilde{t_0}) \cdot b_{i-1}^{m-2} + \widetilde{t_0} \cdot b_i^{m-2} \quad i = m-1, m$

m $b_i^m := (1 - \widetilde{t_0}) \cdot b_{i-1}^{m-1} + \widetilde{t_0} b_i^{m-1} \quad i = m$

m+1 $X(\widetilde{t_0}) := b_m^m \qquad \widetilde{t_0} := \frac{t_0 - a}{b - a}$

Visual Scheme for the Algorithm

With the control points int the leftmost column, each pair of consecutive points is used to compute a point of a new control polygon with one fewer point, when arranged in a triangular shape, the target point $X(t_0)$ is then found at the lower right corner.

$$\left.\begin{matrix}(1-t_0)\cdot b_i^m \\ +t_0\cdot b_{i+1}^m\end{matrix}\right\rangle \; b_{i+1}^{m+1}$$

$$\begin{matrix} b_0^0 & & & \\ b_1^0 & b_1^1 & & \\ \vdots & \vdots & \ddots & \\ b_m^0 & b_m^1 & \cdots & b_m^m \end{matrix}$$

Fact

The De Casteljau Algorithm is affine invariant.

Example

$m = 3$, 4, points on the control polygon are given as

$$P_3 = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 4.5 \end{pmatrix}, \begin{pmatrix} 8.5 \\ 6.5 \end{pmatrix}, \begin{pmatrix} 11 \\ 2 \end{pmatrix} \right),$$

compute $X(0.6)$.

$$\begin{array}{cccc} \begin{pmatrix} 0 \\ 0 \end{pmatrix} & & & \\ \begin{pmatrix} 2 \\ 4.5 \end{pmatrix} & \begin{pmatrix} 1.2 \\ 2.7 \end{pmatrix} & & \\ \begin{pmatrix} 8.5 \\ 6.5 \end{pmatrix} & \begin{pmatrix} 5.9 \\ 5.7 \end{pmatrix} & \begin{pmatrix} 4.02 \\ 4.50 \end{pmatrix} & \\ \begin{pmatrix} 11 \\ 2 \end{pmatrix} & \begin{pmatrix} 10 \\ 3.8 \end{pmatrix} & \begin{pmatrix} 8.36 \\ 4.56 \end{pmatrix} & \begin{pmatrix} 6.63 \\ 4.54 \end{pmatrix} \end{array}$$

Graphical example

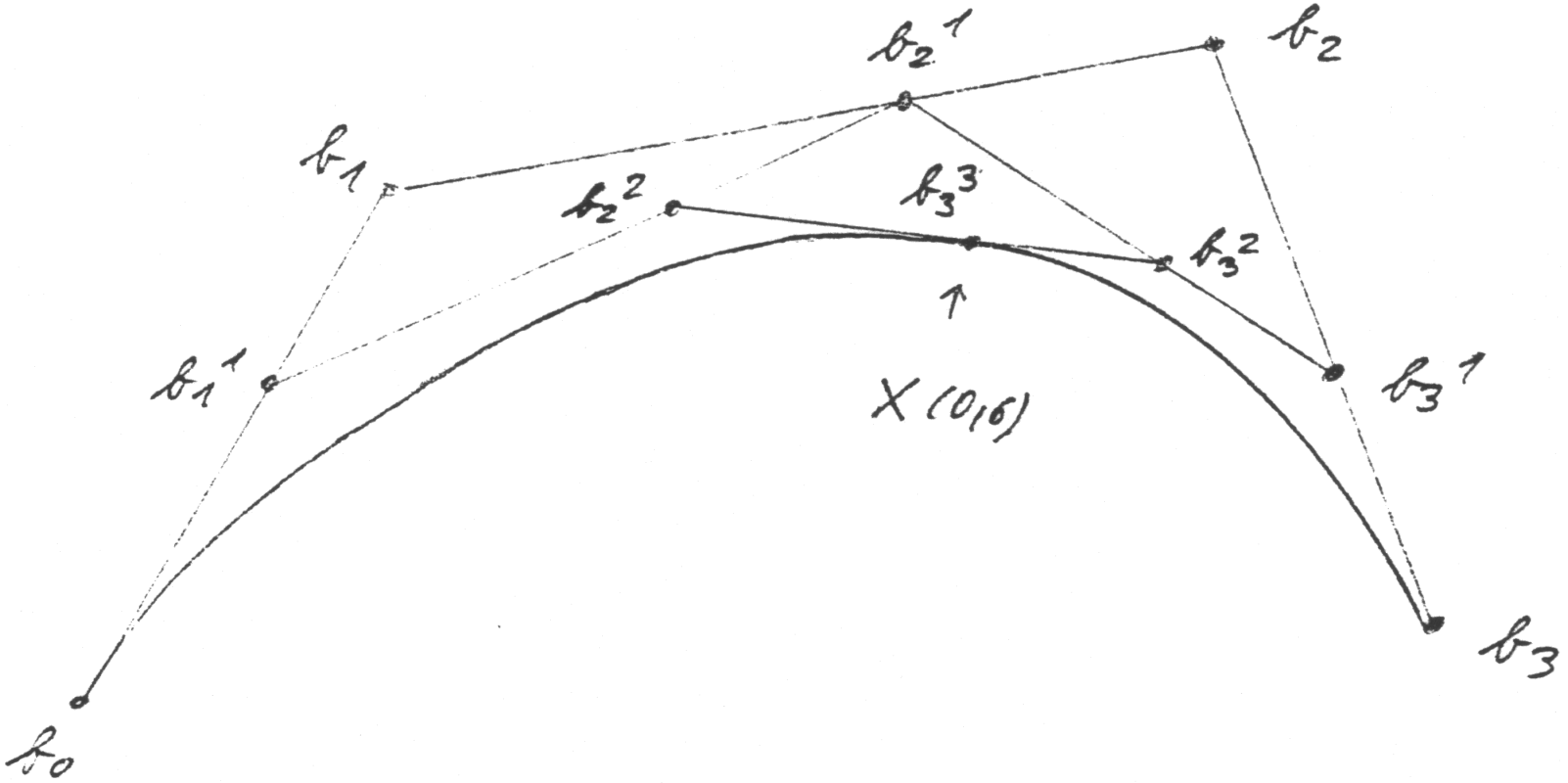

Figure: Visualization of the De Casteljau algorithm

Degree elevation

The Bézier curve generated by the control polygon $P_m = b_0, \ldots, b_m$, can also generated using one more control point by the polygon $P_{m+1} = \widetilde{b_0}, \ldots, \widetilde{b_{m+1}}$, with the new control points given by:

$$\widetilde{b_i} = \frac{i}{m+1} \cdot b_{i-1} + \left(1 - \frac{i}{m+1}\right) \cdot b_i \qquad i = 1 \ldots m$$

Because Bézier curves interpolate the end points, $\widetilde{b_0} = b_0$ and $\widetilde{b_{m+1}} = b_m$.

Bézier Curves

For the pth derivative of a Bézier segment $X(t) := \sum_{i=0}^{m} b_I B_i^m(t), \quad t \in [a, b]$, one obtains:

$$\frac{\mathrm{d}^p}{\mathrm{d}t^p} X(t_0) = \frac{m!}{(m-p)!} \cdot \frac{1}{(b-a)^p} \cdot \sum_{i=0}^{m-p} \Delta^p b_i \cdot B_i^{m-p}(t_0)$$

$$\text{with} \quad \Delta^p b_i := \Delta^{p-1} b_{i+1} - \Delta^{p-1} b_i,$$

$$\text{and} \quad \Delta^0 b_i = b_i$$

Fact

The first and last sides of the control polygon are tangential to the Bézier curve.

Some Special Cases

$$X'(t_0) = \frac{m}{b-a}\left(b_m^{m-1} - b_{m-1}^{m-1}\right)$$
$$X''(t_0) = \frac{m(m-1)}{(b-a)^2}\left(b_m^{m-2} - 2b_{m-1}^{m-2} + b_{m-2}^{m-2}\right)$$

In our example:

$$X'(0.6) = 3\left(b_3^2 - b_2^2\right) = 3\left(\begin{pmatrix}8.36\\4.56\end{pmatrix} - \begin{pmatrix}4.02\\4.50\end{pmatrix}\right) = \begin{pmatrix}13.02\\0.18\end{pmatrix}$$
$$X''(0.6) = 6\left(b_3^1 - 2b_2^1 + b_1^1\right)$$
$$= 6\left(\begin{pmatrix}10.00\\3.80\end{pmatrix} - 2\begin{pmatrix}5.9\\5.7\end{pmatrix} + \begin{pmatrix}1.2\\2.7\end{pmatrix}\right) = \begin{pmatrix}-3.6\\-29.4\end{pmatrix}$$

Integrating a Bézier segment

With $X(t)$ a Bézier segment on the interval $t \in [a, b]$,

$$\int_a^b X(t)\mathrm{d}t = \frac{b-a}{m+1}(b_0 + b_1 + \cdots + b_m)$$

Interpolation with Bézier curves

To fit a Bézier curve through a given set of points $p_o, \ldots, p_r$, one needs to solve the linear system of equations:

$$X(t_j) := \sum_{i=0}^{r} b_i B_i^r(t_j)$$

Once the t_j are chosen, this system has a unique solution b_i, as the basis functions B_i^r are linearly independent.

Example Interpolation

Fitting a Bézier curve through the points

$$p_0 = \begin{pmatrix}1\\2\end{pmatrix}, p_1 = \begin{pmatrix}2\\1\end{pmatrix}, p_2 = \begin{pmatrix}4\\2\end{pmatrix} p_3 = \begin{pmatrix}6\\6\end{pmatrix}.$$

With *chordal parametrization*, the t_i are chosen according to the Euclidean distances, i.e. $t_i - t_{i-1} = \|p_i - p_{i-1}\|$ for $i = 1, \ldots, m$. For the points in this example, $t_0 = 0$, $t_1 = \sqrt{2}$, $t_2 = t_1 + \sqrt{5}$, $t_3 = t_2 + 2\sqrt{5}$, so $[a, b] = [0, 8.12]$.

$$X(t_0) = X(0) = \begin{pmatrix}1\\2\end{pmatrix} \qquad X(t_1) = X(1.41) = \begin{pmatrix}2\\1\end{pmatrix}$$
$$X(t_2) = X(3.65) = \begin{pmatrix}4\\2\end{pmatrix} \quad X(t_3) = X(8.12) = \begin{pmatrix}6\\6\end{pmatrix}$$

Bézier Curves

The equations then read:

$$\begin{pmatrix}1\\2\end{pmatrix} = b_0 \quad \text{(Because of end-point interpolation)}$$

$$\begin{pmatrix}2\\1\end{pmatrix} = \sum_{i=0}^{3} b_i B_i^3(1.41)$$

$$\begin{pmatrix}4\\2\end{pmatrix} = \sum_{i=0}^{3} b_i B_i^3(3.65)$$

$$\begin{pmatrix}6\\6\end{pmatrix} = b_3$$

Bézier Curves

Only b_1 and b_2 need to be solved for:

$$\begin{pmatrix} B_1^3(t_1) & B_2^3(t_1) \\ B_1^3(t_2) & B_2^3(t_2) \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} p_1 - B_0^3(t_1)p_0 - B_3^3(t_1)p_3 \\ p_2 - B_0^3(t_2)p_0 - B_3^3(t_2)p_3 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 0.36 & 0.08 \\ 0.41 & 0.33 \end{pmatrix}^{(-1)} \begin{pmatrix} p_1 - B_0^3(t_1)p_0 - B_3^3(t_1)p_3 \\ p_2 - B_0^3(t_2)p_0 - B_3^3(t_2)p_3 \end{pmatrix}$$

$$\Rightarrow \quad b_1 = \begin{pmatrix} 2.52 \\ -1.55 \end{pmatrix},$$

$$b_2 = \begin{pmatrix} 6.78 \\ 5.27 \end{pmatrix}$$

Bézier Curves

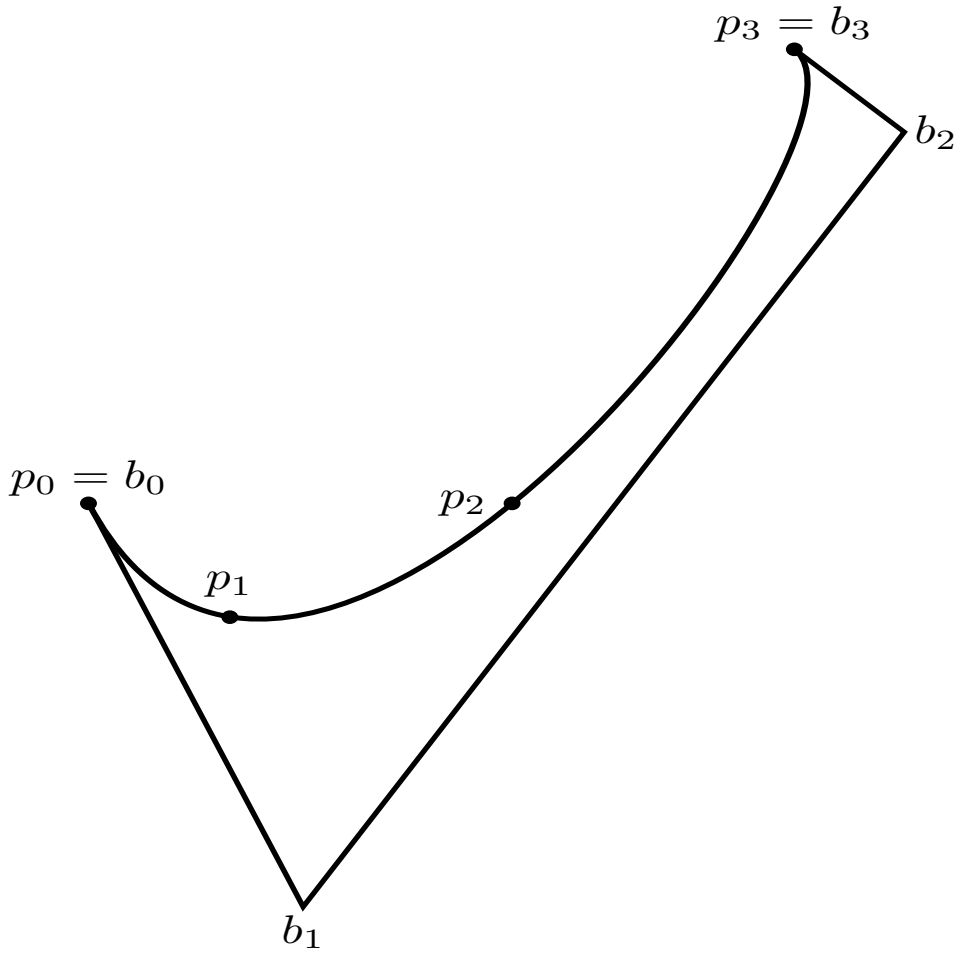

Figure: Interpolating Bézier curve

Further remarks on Interpolation

- previous method offers no additional interaction opportunities like tangent or curvature adjustments.
- another possibility: use the points $p_0, \ldots, p_m$ as corner points for combined Bézier segments

Definition

A *Bézier curve* is a spline consisting of Bézier segments X_l, $l = 0, \ldots, k$, each of degree m, over the parameter interval $u_l \leq u \leq u_{l+1}$.

$$X_l(u) := \sum_{i=0}^{m} b_{m_{l+i}} \cdot B_i^m \left(\frac{u - u_l}{u_{l+1} - u_l} \right)$$

The Bernstein polynomials are used as "blending functions".

Joining the Bézier segments

To obtain a C^n-continuous curve, the function values, as well as derivatives up to the nth, need to be matched.
Function values:

$$X_l(u) := \sum_{i=0}^{m} b_{m(l+i)} \cdot B_i^m \left(\frac{u - u_l}{u_{l+1} - u_l} \right); \quad u_l \leq u \leq u_{l+1}$$

$$X_{l+1}(u) ;= \sum_{j=0}^{m} b_{m(l+i)+j} \cdot B_j^m \left(\frac{u - u_{l+1}}{u_{l+2} - u_{l+1}} \right); \quad u_{l+1} \leq u \leq u_{l+2}$$

Joining the Bézier segments (cont.)

Derivatives in the corner points, with $\lambda_l := u_{l+1} - u_l$:

$$
\begin{aligned}
X_l^{(1)}(u_{l+1}) &:= \frac{m}{\lambda_l}\left(b_{m(l+1)} - b_{m(l+1)-1}\right) \\
X_l^{(2)}(u_{l+1}) &:= \frac{m(m-1)}{\lambda_l^2}\left(b_{m(l+1)} - 2b_{m(l+1)-1} + b_{m(l+1)-2}\right) \\
X_l^{(3)}(u_{l+1}) &:= \frac{m(m-1)(m-2)}{\lambda_l^3} \\
&\qquad \cdot \left(b_{m(l+1)} - 3b_{m(l+1)-1} + 3b_{m(l+1)-2} - b_{m(l+1)-3}\right)
\end{aligned}
$$

Joining the Bézier segments (cont.)

$$
\begin{aligned}
X_{l+1}^{(1)}(u_{l+1}) &:= \frac{m}{\lambda_{l+1}} \left(b_{m(l+1)+1} - b_{m(l+1)} \right) \\
X_{l+1}^{(2)}(u_{l+1}) &:= \frac{m(m-1)}{\lambda_{l+1}^2} \left(b_{m(l+1)+2} - 2b_{m(l+1)+1} + b_{m(l+1)} \right) \\
X_{l+1}^{(3)}(u_{l+1}) &:= \frac{m(m-1)(m-2)}{\lambda_{l+1}^3} \\
&\quad \cdot \left(b_{m(l+1)+3} - 3b_{m(l+1)+2} + 3b_{m(l+1)+1} - b_{m(l+1)} \right)
\end{aligned}
$$

Conditions for Continuity

- C^1-continuity:
  $b_{m(l+1)} - b_{m(l+1)-1} = \frac{\lambda_l}{\lambda_{l+1}} \cdot \left(b_{m(l+1)+1} - b_{m(l+1)}\right)$
- C^2-continuity: additionally

$$b_{m(l+1)} - 2b_{m(l+1)1} + b_{m(l+1)-2} = \frac{\lambda_l^2}{\lambda_{l+1}^2} \cdot \left(b_{m(l+1)+2} - 2b_{m(l+1)+1} + b_{m(l+1)}\right) \quad (1)$$

Polygon construction

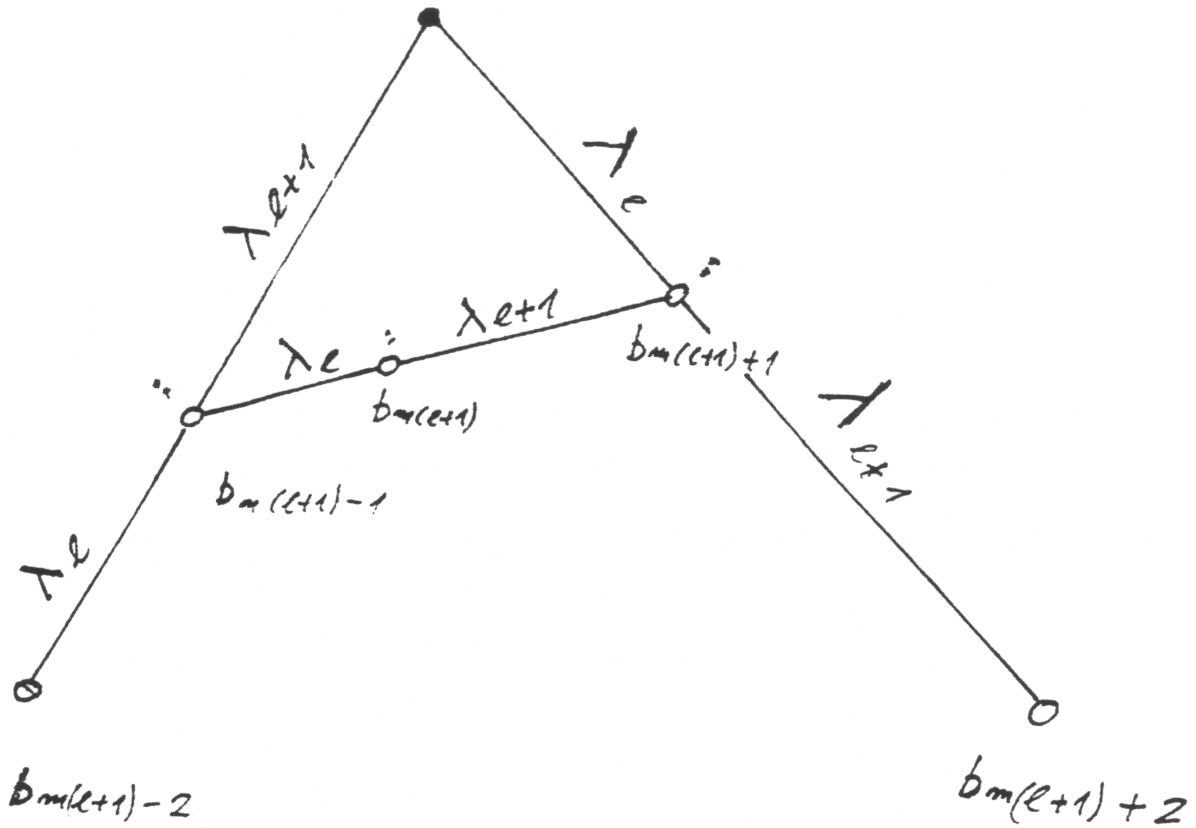

Figure: Two Bézier segments joined C^2-continuously at $b_{m(l+1)}$

C^3-continuity

$$
\begin{aligned}
b_{m(l+1)} - b_{m(l+1)-1} &= \frac{\lambda_l}{\lambda_{l+1}} \cdot \left(b_{m(l+1)+1} - b_{m(l+1)}\right) \\
b_{m(l+1)} - 2b_{m(l+1)-1} & \\
+b_{m(l+1)-2} &= \frac{\lambda_l^2}{\lambda_{(l+1)}^2} \\
& \quad \cdot \left(b_{m(l+1)+2} - 2b_{m(l+1)+1} + b_{m(l+1)}\right) \\
b_{m(l+1)} - 3b_{m(l+1)-1} & \\
+3b_{m(l+1)-2} - b_{m(l+1)-3} &= \frac{\lambda_l^3}{\lambda_{(l+1)}^3} \\
& \quad \cdot \left(b_{m(l+1)+3} - 3b_{m(l+1)+2}\right. \\
& \qquad \left. +3b_{m(l+1)+1} - b_{m(l+1)}\right)
\end{aligned}
$$

Polygon construction

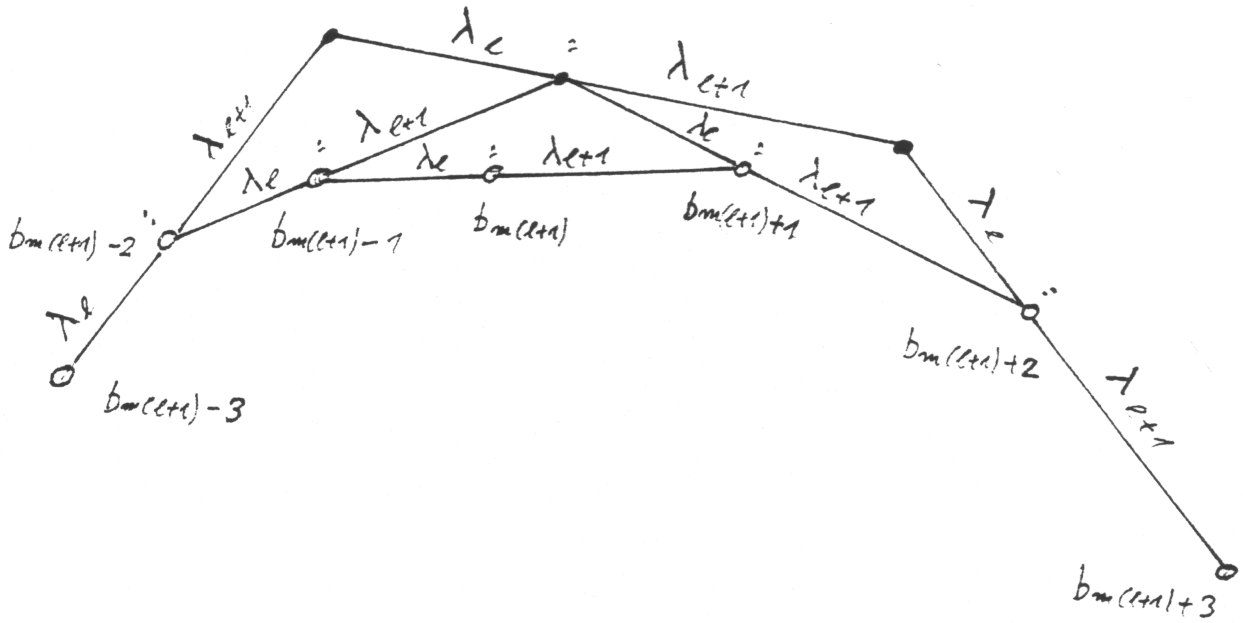

Figure: Two Bézier segments joined with C^3-continuity

Remark

This is “equivalent” to the De Casteljau recursion

Cubic C^2-continuous Bézier splines

Fitting the condition for C^1-continuity,

$$\lambda_{(I+1)} \left(b_{3(I+1)} - b_{3(I+1)-1} \right) = \lambda_I \left(b_{3(I+1)+1} - b_{3(I+1)} \right),$$

into the C^2-condition:

$$\begin{aligned} &\lambda_{(I+1)}^2 \left(b_{3(I+1)} - 2b_{3(I+1)-1} + b_{3(I+1)-2} \right) \\ &\quad = \lambda_I^2 \left(b_{3(I+1)} - 2b_{3(I+1)+1} + b_{3(I+1)} \right), \end{aligned}$$

we get:

$$\frac{\lambda_{I+1}}{\lambda_I} \left(b_{3(I+1)} - \frac{\lambda_I + \lambda_{(I+1)}}{\lambda_{(I+1)}} b_{3(I+2)} \right) = \frac{\lambda_I}{\lambda_{(I+1)}} \left(b_{3(I+5)} - \frac{\lambda_I + \lambda_{(I+1)}}{\lambda_I} b_{3(I+4)} \right)$$

Bézier Curves

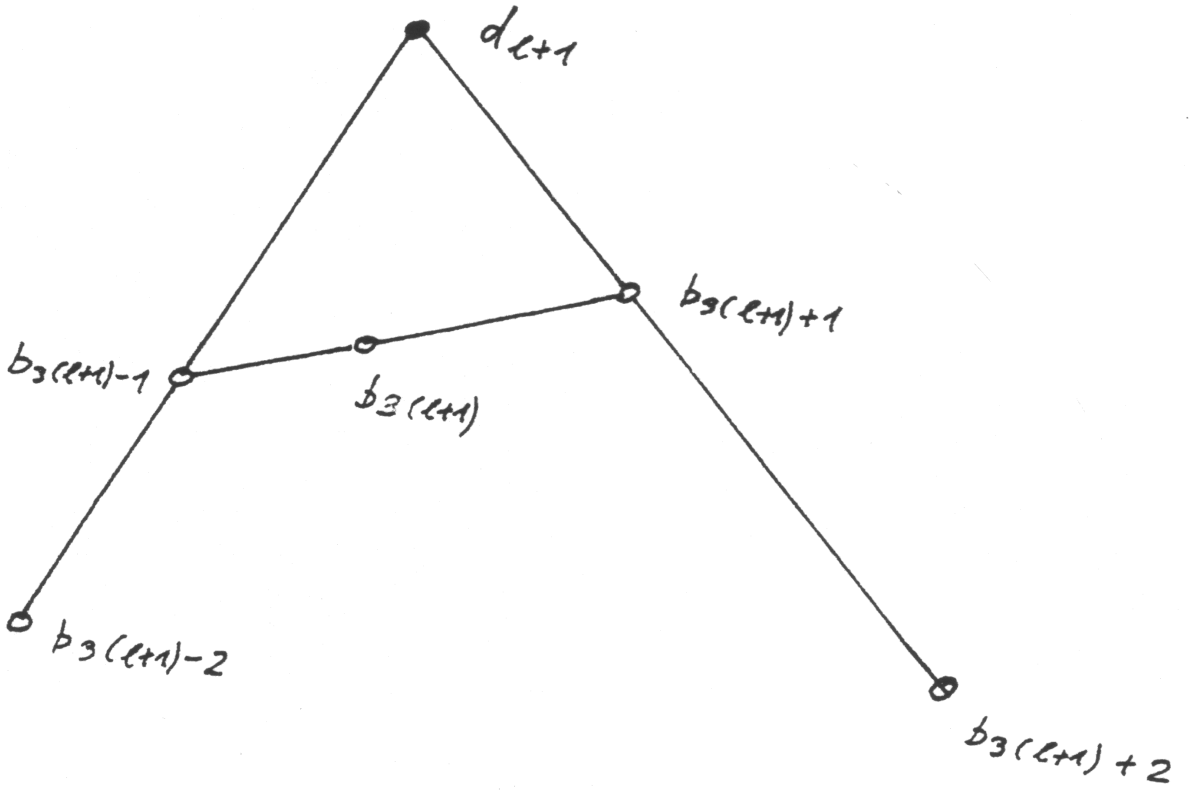

Figure: geometric background for the cubic case. See next slide for the definition of d

Definition

The *weight point* d_{I+1} completes a geometric construction that allows for C^2-continuous connection of Bézier segments (see previous figure).

$$\begin{aligned} d_{I+1} &:= \frac{\lambda_{I+1}}{\lambda_I}\left(-b_{3(I+1)-2} + \frac{\lambda_I + \lambda_{I+1}}{\lambda_{(I+1)}} b_{3(I+1)-1}\right) \\ &= \frac{\lambda_I}{\lambda_{(I+1)}}\left(-b_{3(I+1)+2} + \frac{\lambda_I + \lambda_{I+1}}{\lambda_{(I+1)}} b_{3(I+1)+1}\right) \end{aligned}$$

Continuity in three steps

To connect a new Bézier segment with C^2 continuity, one can follow three steps:

connect the tangents

$$b_{3(l+1)+1} = b_{3(l+1)} + \frac{\lambda_{(l+1)}}{\lambda_l}\left(b_{3(l+1)} - b_{3(l+1)-1}\right)$$

find the weight point

$$d_{l+1} = b_{3(l+1)-1} + \frac{\lambda_{(l+1)}}{\lambda_l}\left(b_{3(l+1)-1} - b_{3(l+1)-2}\right)$$

adjust for C^2-continuity

$$b_{3(l+1)+2} = b_{3(l+1)+1} + \frac{\lambda_{(l+1)}}{\lambda_l}\left(-d_{l+1} + b_{3(l+1)+1}\right)$$

Algorithm for interpolating with Bézier segments

$$
\begin{aligned}
(\lambda_{l-1} + \lambda_l) \cdot d_{l-1} + \lambda_{l-2} d_l &= b_{3l-2} \cdot C_{l-1} && l = 1 \dots k \\
A_l \cdot d_{l-1} + D_l \cdot d_l + A_{l-1} \cdot d_{l+1} &= b_{3l} \cdot B_l && l = 1 \dots k-1 \\
\lambda_{l+1} \cdot d_l + (\lambda_{l-1} + \lambda_l) \cdot d_{l+1} &= b_{3l+2} \cdot C_l && l = 0 \dots k-1
\end{aligned}
$$

with $\lambda_{-1} = \lambda_0$ and $\lambda_k = \lambda_{k-1}$

$$
\begin{aligned}
C_l &= (\lambda_{l-1} + \lambda_l + \lambda_{l+1}) \\
A_l &= \lambda_l^2 (\lambda_{l-1} + \lambda_l + \lambda_{l+1}) \\
D_l &= \lambda_l (\lambda_{l-2} + \lambda_{l-1})(\lambda_{l-1} + \lambda_l + \lambda_{l+1}) \\
&\quad + \lambda_{l-1} (\lambda_l + \lambda_{l+1})(\lambda_{l-2} + \lambda_{l-1} + \lambda_l) \\
B_l &= (\lambda_{l-1} + \lambda_l)(\lambda_{l-1} + \lambda_l + \lambda_{l+1})(\lambda_{l-2} + \lambda_{l-1} + \lambda_l)
\end{aligned}
$$

Bézier Curves

Special case $\lambda_l = 1$ for all $l = 1 \ldots k$.

$$A_l \equiv 3; \quad B_l \equiv 18; \quad C_{l-1} \equiv 3; \quad D_l \equiv 12$$

$$\begin{aligned} 2d_{l-1} + d_l &= 3b_{3l-2} && l = 1, \ldots, k \\ d_{l-1} + 4d_l + d_{l+1} &= 6b_{3l} && l = 1, \ldots, k-1 \\ d_l + 2d_{l+1} &= 3b_{3l+2} && l = 0, \ldots, k-1 \end{aligned}$$

Remarks

- The weight points d_0 and d_k are defined as:

$$\begin{aligned} d_0 &= 2b_1 - b_2 \\ d_k &= -b_{3k-2} + 2b_{3k-1} \end{aligned}$$

- Using the previously pictured algorithm allows for calculation of all Bézier points except for b_0 and b_{3k}.

Example for an interpolating cubic Bézier spline

Input: p_0, p_1, p_2, special parametrization $\lambda_1 = \lambda_2 = 1$.

- set $b_0 = p_0; b_3 = p_1; b_6 = p_2$
- choose the free parameters b_1 and b_5
- solve the linear system:

$$\begin{array}{llll} 2d_0 & +d_1 & & = 3b_1 \\ d_0 & +4d_1 & +d_2 & = 6b_3 \\ & d_1 & +2d_2 & = 3b_5 \end{array}$$

- calculate Bézier points b_2, b_4 using the weight point construction

Bézier Curves

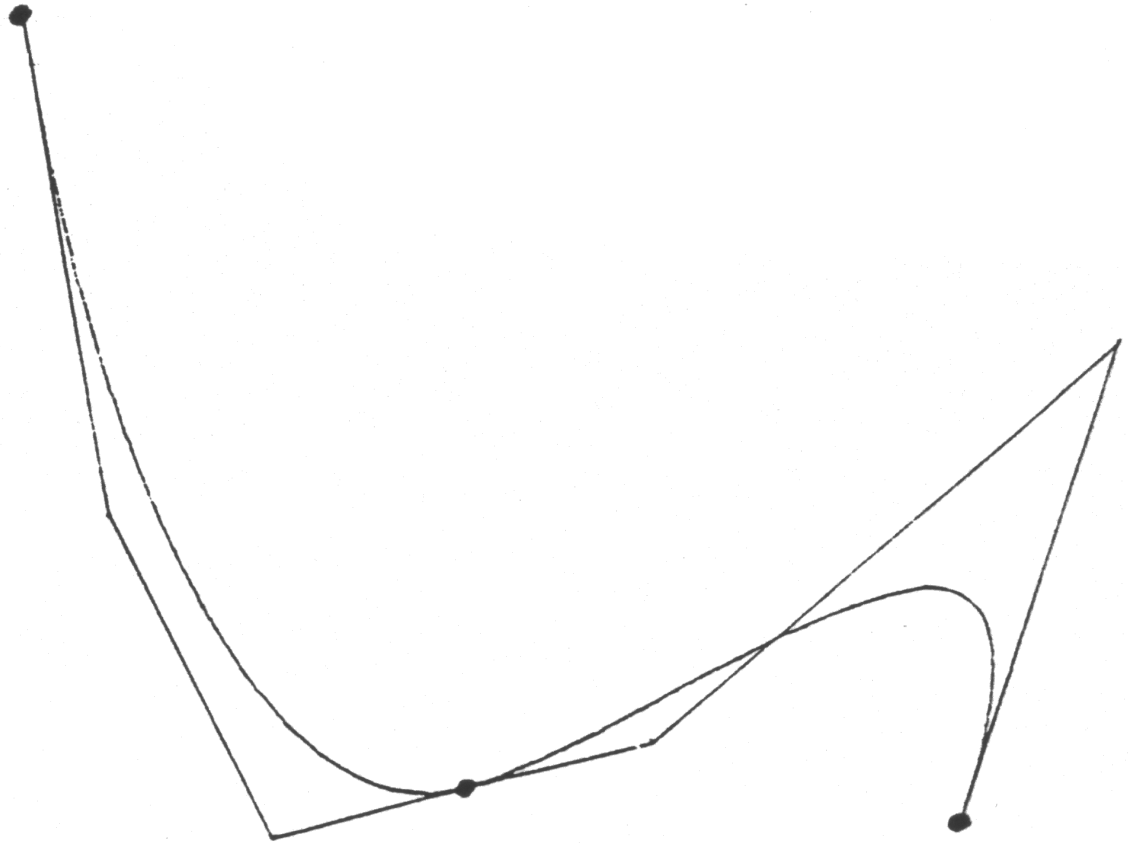

Figure: Interpolating three points with two cubic Bézier curves

Algorithm for Interpolating Bézier Splines

Input: $k+1$ points $p_i \in \mathbb{E}^3, 0 \leq l \leq k$.

1. set $b_0 = p_0, b_3 = p_1, \ldots, b_{3k} - p_k$
2. choose free parameters b_1 and b_{3k-1} (slope at the end points)
3. solve the linear system of equations:

$$\begin{aligned} (\lambda_1 + \lambda_0)d_0 + \lambda_0 \cdot d_1 &= C_0 \cdot b_1 \\ A_l \cdot d_{l-1} + D_l \cdot d_l + A_{l-1} \cdot d_{l+1} &= B_l \cdot b_{3l} \quad l = 1 \ldots k-1 \\ \lambda_{k-1} d_{k-1} + (\lambda_{k-2} + \lambda_{k-1}) \cdot d_k &= C_{k-1} b_{3k-1} \end{aligned}$$

4. calculate the Bézier points $b_{3l-2}, b_{3l-1} (l = 1 \ldots k)$
5. run the De Casteljau algorithm k times:
   $\{b_0, b_1, b_2, b_3\}; \{b_3, b_4, b_5, b_6\};$
   $\ldots; \{b_{3k-3}, b_{3k-2}, b_{3k-2}, b_{3k-1}\}$

Example

Interpolate the points

$$\begin{pmatrix}1\\2\end{pmatrix}\begin{pmatrix}2\\1\end{pmatrix}\begin{pmatrix}4\\4\end{pmatrix}\begin{pmatrix}6\\2\end{pmatrix}$$

by a cubic Bézier Spline (with chordal parametrization),

$$\lambda_0 = \sqrt{2}, \quad \lambda_1 = \sqrt{5}, \quad \lambda_2 = \sqrt{20}$$

$$u_0 = 0, \quad u_1 = u_0 + \lambda_0, \quad u_2 = u_1 + \lambda_1, \quad u_3 = u_2 + \lambda_2$$

Step 1: $b_0 = \begin{pmatrix}1\\2\end{pmatrix}, b_3 = \begin{pmatrix}2\\1\end{pmatrix}, b_6 = \begin{pmatrix}4\\4\end{pmatrix}, b_9 = \begin{pmatrix}6\\2\end{pmatrix}$

Step 2: Choose b_1 and b_8.

Bézier Curves

Step 3:

$$\begin{aligned} d_0 + 0.39 d_1 &= 1.39 b_1 \\ 0.27 d_0 + 0.66 d_1 + 0.068 d_2 &= \begin{pmatrix} 2 \\ 1 \end{pmatrix} \\ 0.39 d_1 + 0.57 d_2 + 0.067 d_3 &= \begin{pmatrix} 4 \\ 2 \end{pmatrix} \\ 0.66 d_2 + d_3 &= 1.67 b_8 \end{aligned}$$

Solution:

$$d_0 = 1.68 b_1 - 0.02 b_8 - \begin{pmatrix} 1.16 \\ 0.58 \end{pmatrix} \qquad d_1 = 0.76 b_1 + 0.03 b_8 + \begin{pmatrix} 3 \\ 1.5 \end{pmatrix}$$

$$d_2 = 0.25 b_8 + \begin{pmatrix} 5.64 \\ 2.82 \end{pmatrix} \qquad d_3 = -0.35 b_1 + 1.84 b_8 - \begin{pmatrix} 3.74 \\ 1.87 \end{pmatrix}$$

Bézier Curves

Step 4: Run the De Casteljau algorithm three times, for each of the segments $b_{0\ldots3}$, $b_{3\ldots6}$, $b_{6\ldots9}$

$$b_2 = 0.18b_1 + 0.004b_8 + \begin{pmatrix}0.72\\0.31\end{pmatrix} \qquad b_4 = -0.47b_1 - 0.02b_8 + \begin{pmatrix}2.97\\1.49\end{pmatrix}$$

$$b_5 = -.0.03b_1 - 0.1b_8 + \begin{pmatrix}3.59\\1.79\end{pmatrix} \qquad b_7 = 0.35b_1 + 0.17b_8 + \begin{pmatrix}3.76\\1.88\end{pmatrix}$$

Bézier Curves

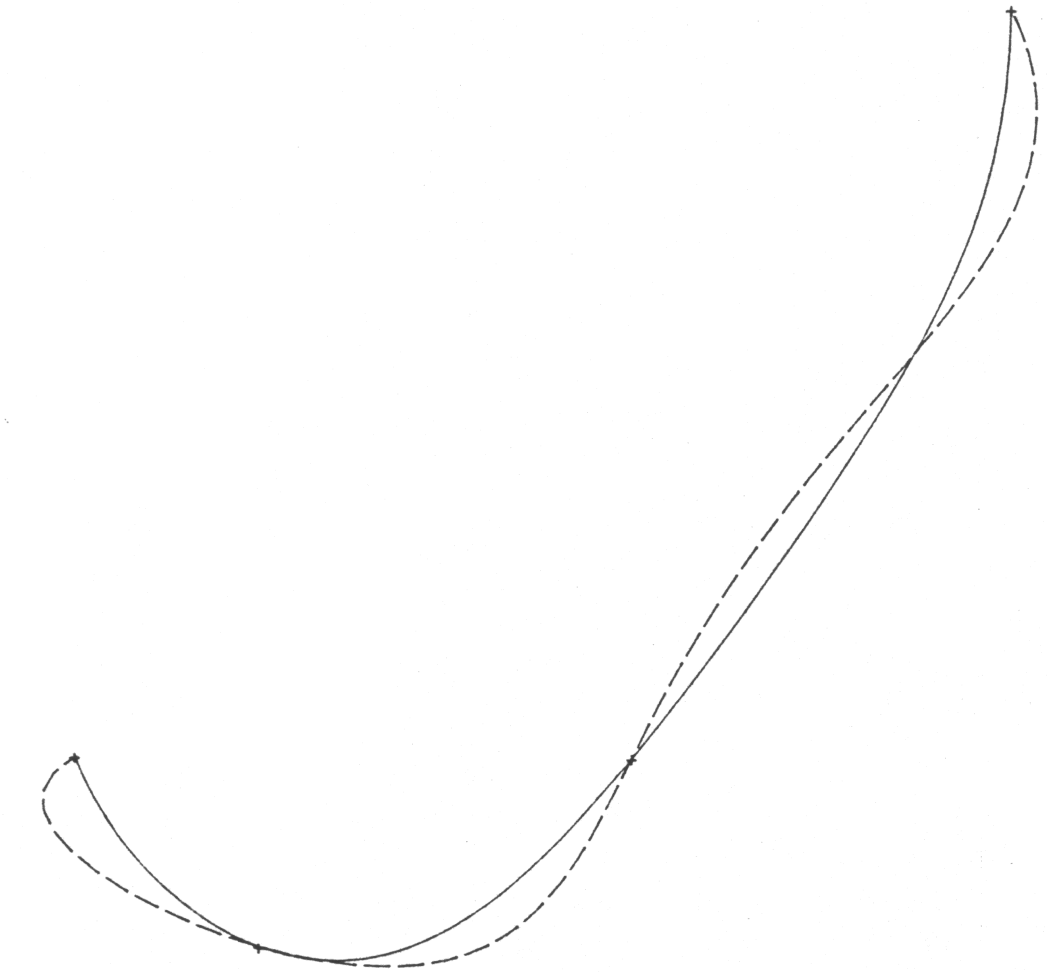

Figure: Different choices for b_1 and b_8

Algorithm for "free modeling" Bézier splines

Input: $k+1$ weight points $d_0, \ldots, d_k$

Step 1 (a) $b_0 = d_0,\ b_{3k} = d_k$
(b) $b_0 = b_{3k} = \frac{1}{2}(b_1 + b_{3k-1}); d_0 = d_k$

Step 2 Calculate all inner Bézier points:

$$\begin{aligned} 2d_{l-1} + d_l &= 3b_{3l-2} && l = 1, \ldots, k \\ d_{l-1} + 4d_l + d_{l+1} &= 6b_{3l} && l = 1, \ldots, k-1 \\ d_l + 2d_{l+1} &= 3b_{3l+2} && l = 0, \ldots, k-1 \end{aligned}$$

Step 3 Run De Casteljau algorithm k times, on $b_{0\ldots3} \ldots b_{3k-3\ldots3k}$

Example

Input: weight points

$$d_0 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}, d_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, d_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, d_3 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}.$$

Step 1: $b_0 = b_9 = \frac{1}{2}(b_1 + b + 8) = \begin{pmatrix} -0.5 \\ 1.5 \end{pmatrix}$

Step 2:

$$b_1 = 1/3(2d_0 + d_1) = \begin{pmatrix} -1/3 \\ 5/3 \end{pmatrix} \quad b_2 = 1/3(d_0 + 2d_1) = \begin{pmatrix} 1/3 \\ 4/3 \end{pmatrix}$$

$$b_3 = 1/6(d_0 + 4d_1 + d_2) = \begin{pmatrix} 1/2 \\ 1 \end{pmatrix} \quad b_4 = 1/3(2d_1 + d_2) = \begin{pmatrix} 2/3 \\ 2/3 \end{pmatrix}$$

$$b_5 = 1/3(d_1 + 2d_2) = \begin{pmatrix} 1/3 \\ 1/3 \end{pmatrix} \quad b_6 = 1/6(d_1 + 4d_2 + d_3) = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix}$$

$$b_7 = 1/3(2d_2 + d_3) = \begin{pmatrix} -1/3 \\ 2/3 \end{pmatrix} \quad b_8 = 1/3(d_2 + 2d + 3) = \begin{pmatrix} -2/3 \\ 4/3 \end{pmatrix}$$

Bézier Curves

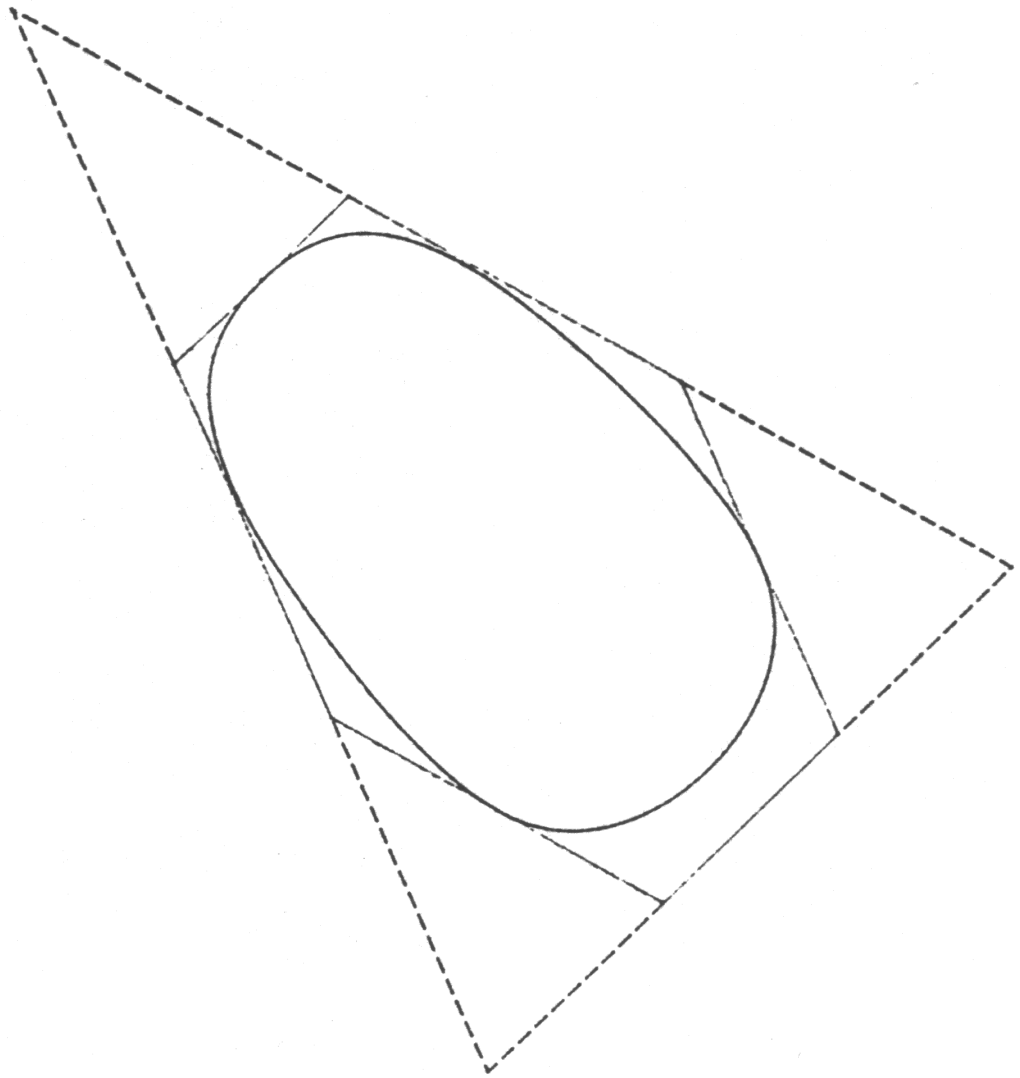

Figure: Result of the previous example calculation

Example

Input: weight points $d_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, d_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, d_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$

Step 1: $b_0 = d_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and $b_6 = d_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$

Step 2:

$$b_1 = 1/3(2d_0 + d_1) = \begin{pmatrix} 1/3 \\ 1/3 \end{pmatrix} \qquad b_2 = 1/3(d_2 + 2d_1) = \begin{pmatrix} 2/3 \\ 2/3 \end{pmatrix}$$

$$b_3 = 1/6(d_0 + 4d_1 + d_2) = \begin{pmatrix} 1 \\ 1/2 \end{pmatrix} \qquad b_4 = 1/3(2d_1 + d_2) = \begin{pmatrix} 4/3 \\ 1/3 \end{pmatrix}$$

$$b_5 = 1/3(d_1 + 2d_2) = \begin{pmatrix} 5/3 \\ -1/3 \end{pmatrix}$$

Bézier Curves

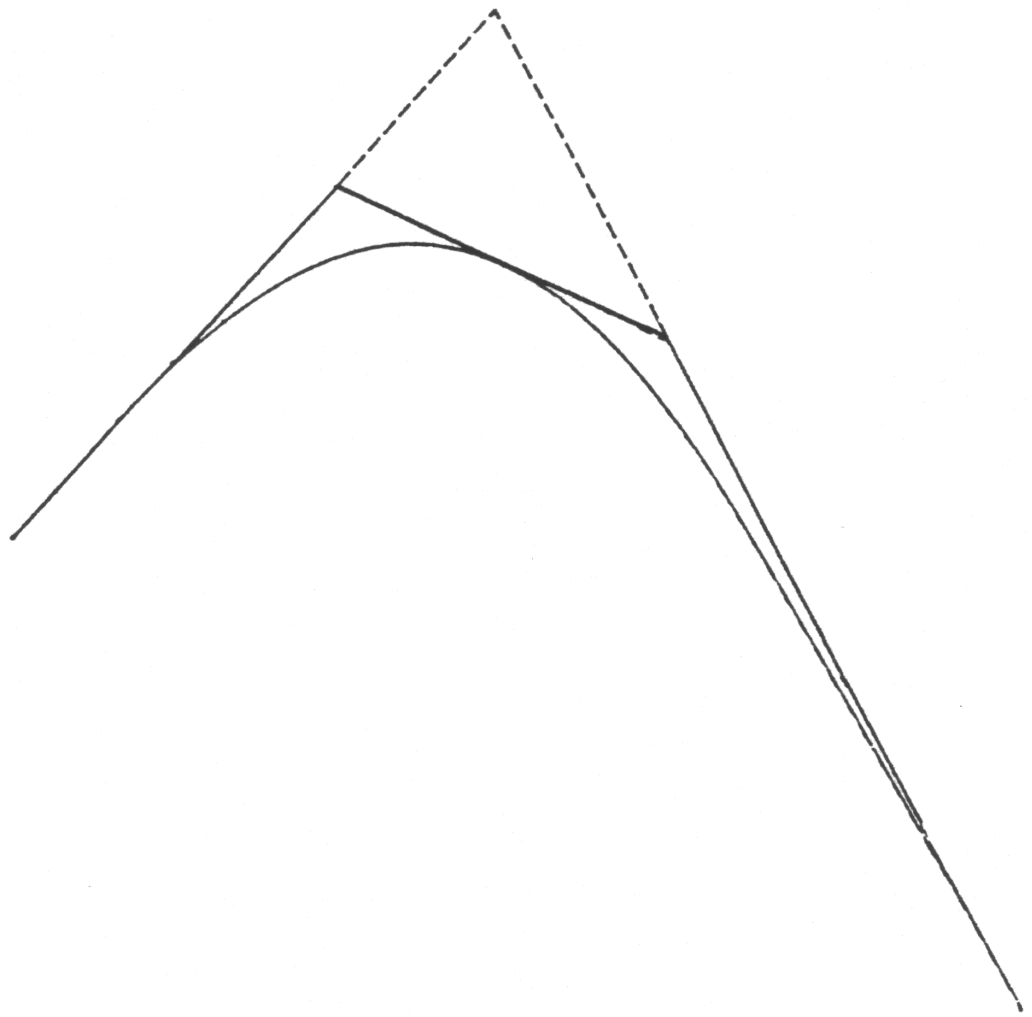

Figure: Result of the previous example

Rational Bézier curves

Blending function methods also work in projective spaces:

$$x(t) = \begin{bmatrix} \sum_{i=0}^{n} \lambda_i b_{ix} \cdot B_i^n(t) \\ \sum_{i=0}^{n} \lambda_i b_{iy} \cdot B_i^n(t) \\ \sum_{i=0}^{n} \lambda_i b_{iz} \cdot B_i^n(t) \\ \sum_{i=0}^{n} \lambda_i \cdot B_i^n(t) \end{bmatrix}$$

Projective Bézier curve with homogeneous coordinates

Bézier Curves

"Going back" to Euclidean space, we gt rational Bézier curves:

$$x(t) = \sum_{i=0}^{n} b_i \cdot \widetilde{B_i^n}(t) = \sum_{i=0}^{n} b_i \cdot \frac{\lambda_i \cdot B_i^n(t)}{\sum_{i=0}^{n} \lambda_i B_i^n(t)}$$

Remarks

1. For $\lambda_i \equiv 1$ for all $i = 0 \ldots n$ we get the "usual" Bézier curves, since $\sum_{i=0}^{n} B_i^n(t) = 1$
2. Under the condition $\lambda_0 > 0$, $\lambda_n > 0$ and $\lambda_i \geq 0$ for $i = 1, \ldots, n-1$, the curve starts in b_0 and ends in b_n.
3. All the other properties like the constriction to the convex hull, variation diminishing etc. are still valid.

Bézier Curves

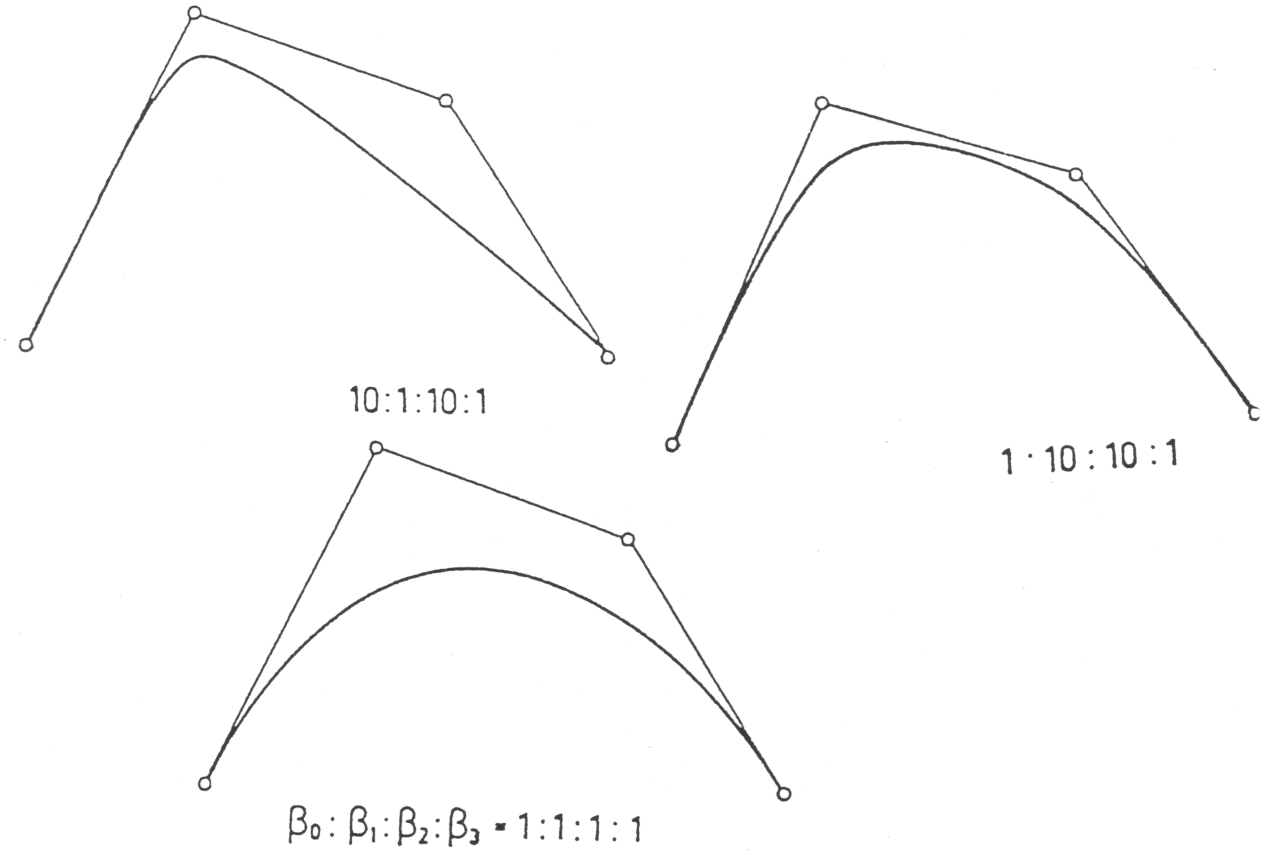

Figure: Influence of weight changes

Main reason for rational Bézier curves

Curves of algebraic order 2 (conics) are special rational Bézier curves of degree 2.
We consider rational, quadratic Bézier curves in this form:

$$x(t) = b_0 \frac{\lambda_0 B_0^2(t)}{\sum_{i=0}^{2} \lambda_i B_i^2(t)} + b_1 \frac{\lambda_1 B_1^2(t)}{\sum_{i=0}^{2} \lambda_i B_i^2(t)} + b_2 \frac{\lambda_2 B_2^2(t)}{\sum_{i=0}^{2} \lambda_i B_i^2(t)},$$

and choose $\lambda_0 = \lambda_2 = 1$.

Bézier Curves

We now define the *center point* $M = \frac{1}{2}(b_0 + b_2)$ and the *shoulder point* $S = M(1-s) + b_1$ with $s := \frac{\lambda_1}{1+\lambda_1}$.

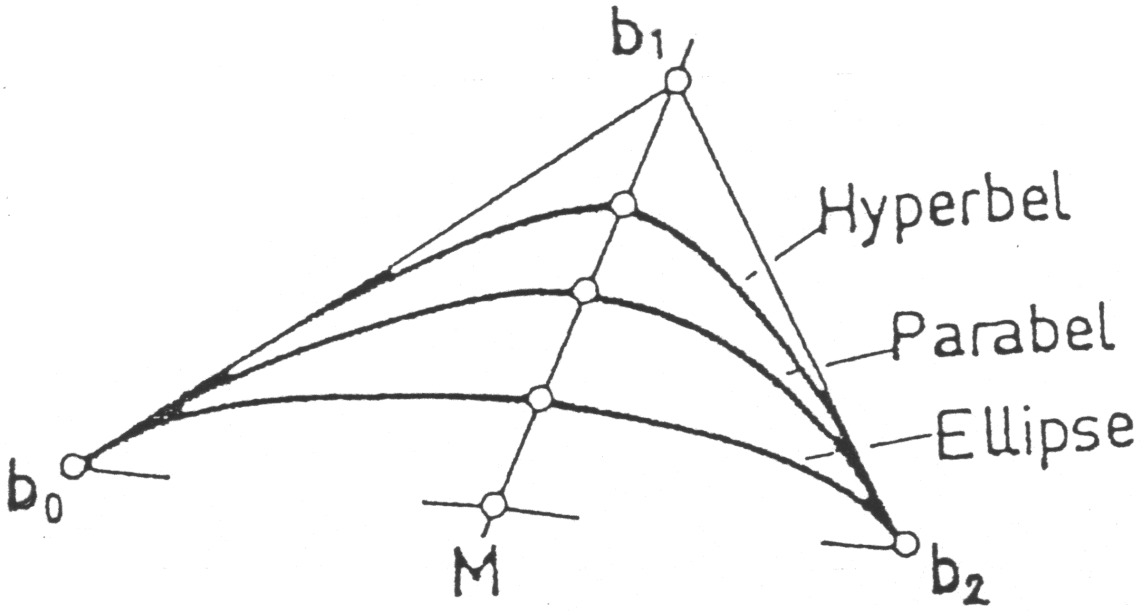

Figure: Shoulder point construction

For $s = 1/2$ we get a parabola, for $s < 1/2$ an ellipse, and for $s > 1/2$ a hyperbola.