

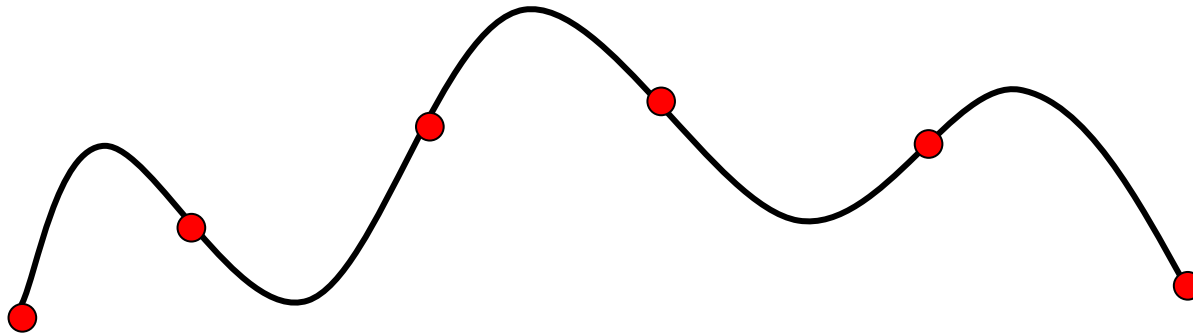
Computergrafik SS 2016

Oliver Vornberger

Vorlesung vom 19.04.2016

noch Kapitel 7:
2D-Kurven

Algebraischer Ansatz



Bestimme $n+1$ Koeffizienten für Polynom n -ten Grades

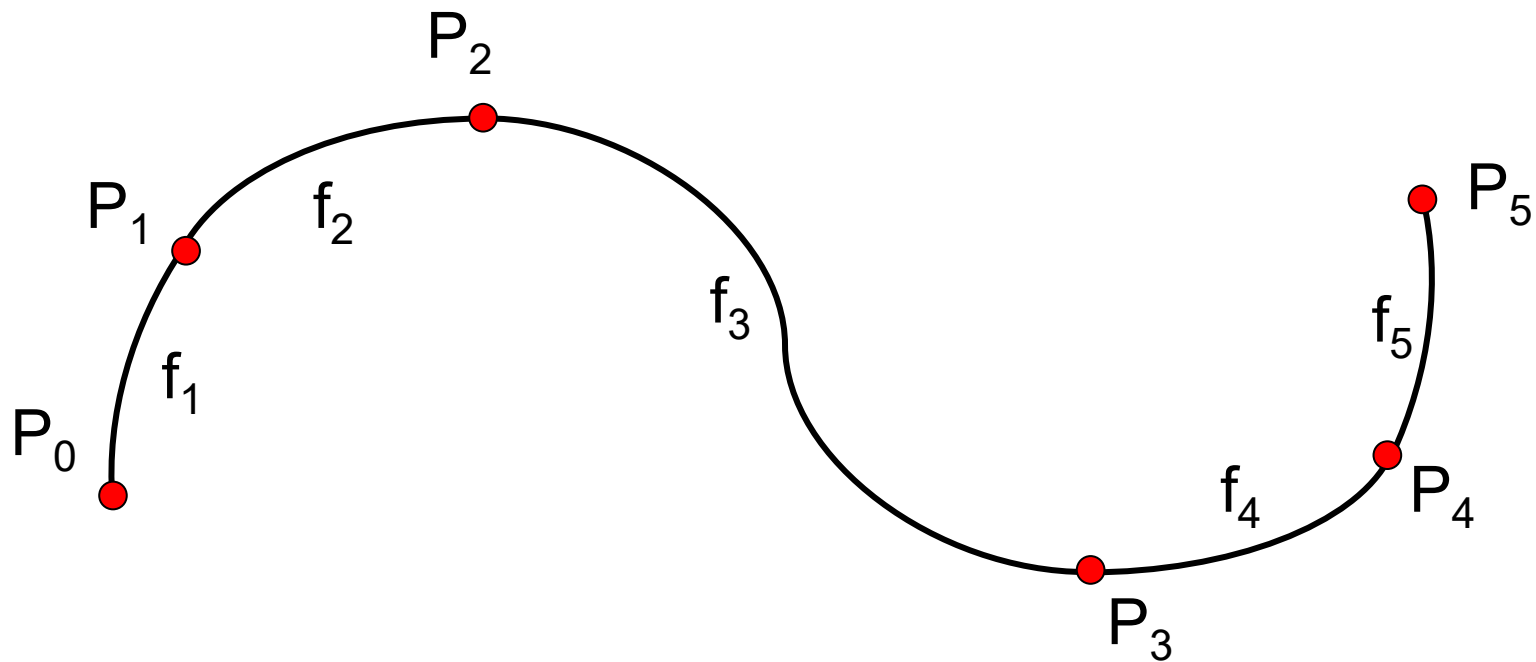
$$y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \dots + a_1 \cdot x + a_0$$

Oszillation!

Rechenaufwand!

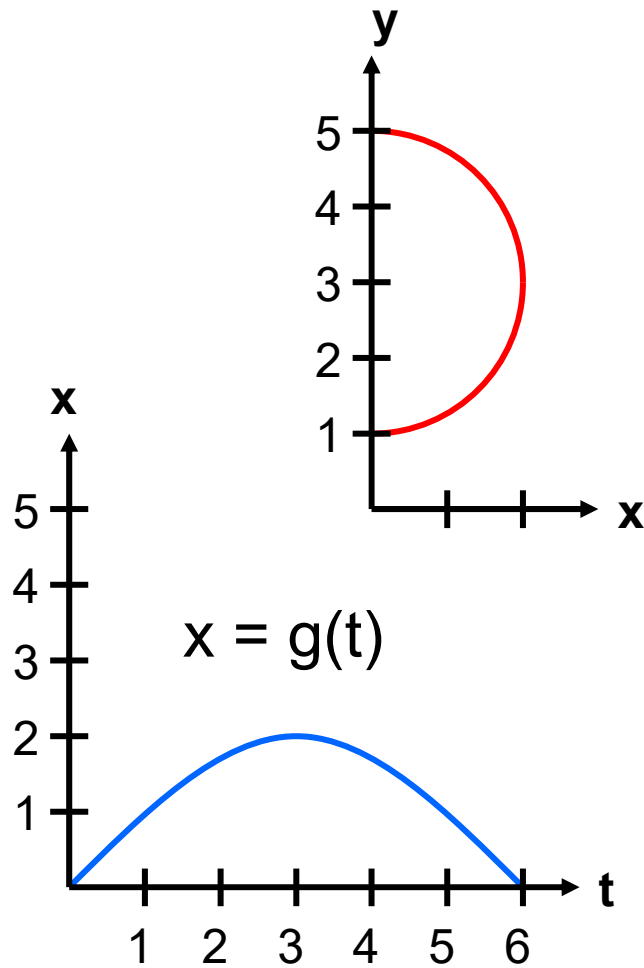
Rundungsfehler !

kubische Splines

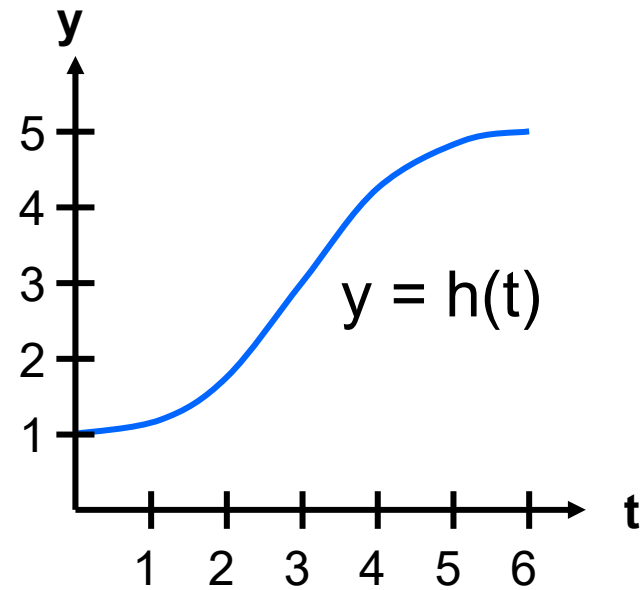


Verbinde zwei aufeinanderfolgende Punkte
durch eine Kurve 3. Grades

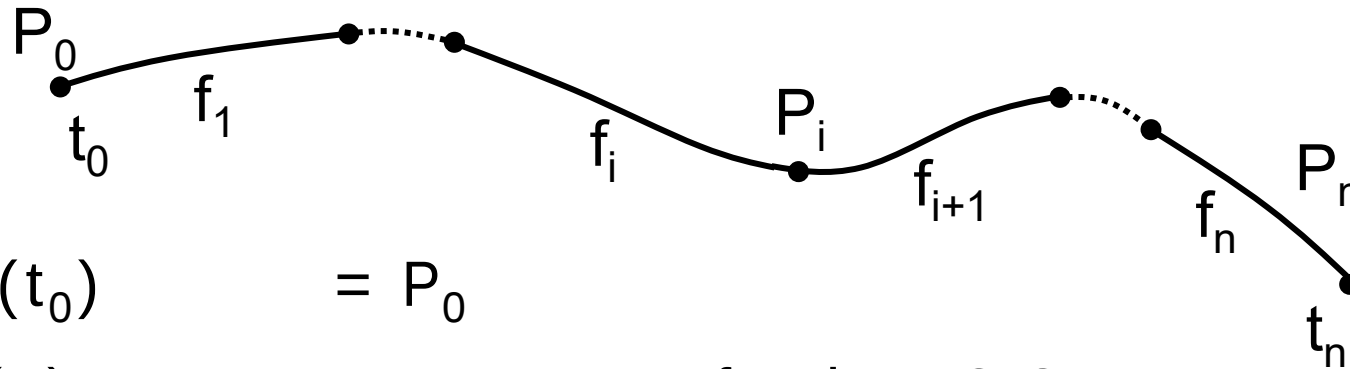
Parametrisierte Kurvengleichung



$$f(t) = [g(t), h(t)]$$



Gleichungssystem



$$f_1(t_0) = P_0$$

$$f_i(t_i) = P_i \quad \text{für } i=1,2,3,\dots,n$$

$$f_i(t_i) = f_{i+1}(t_i) \quad \text{für } i=1,2,3,\dots,n-1$$

$$f'_i(t_i) = f'_{i+1}(t_i) \quad \text{für } i=1,2,3,\dots,n-1$$

$$f''_i(t_i) = f''_{i+1}(t_i) \quad \text{für } i=1,2,3,\dots,n-1$$

$$f_1''(t_0) = 0 \quad 2 \cdot 4n \text{ Gleichungen mit}$$

$$f_n''(t_n) = 0 \quad 2 \cdot 4n \text{ Unbekannten}$$

Approximation

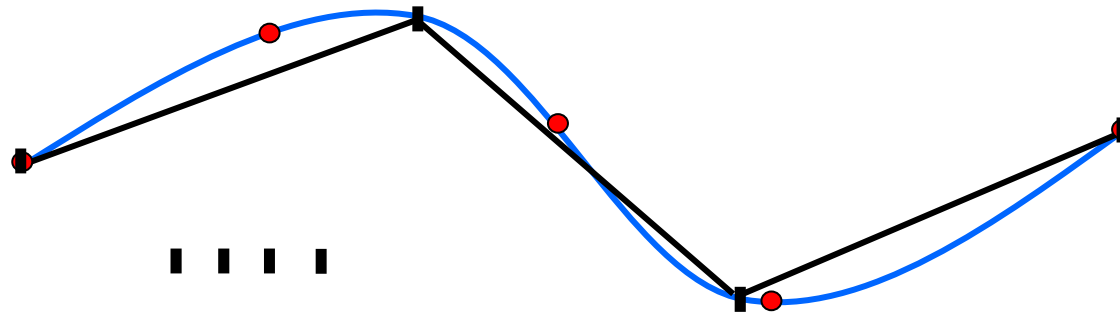
Gegeben $n+1$ Stützpunkte $P_0, P_1, \dots, P_n$

Berechne Kurvenabschnitte $f_1, f_2, \dots, f_n$

Bestimme Zahl der Interpolationspunkte k

Verteile längs der Kurvenabschnitte

Zeichne $k-1$ Geradenabschnitte



Java-Applet zu kubischen Splines

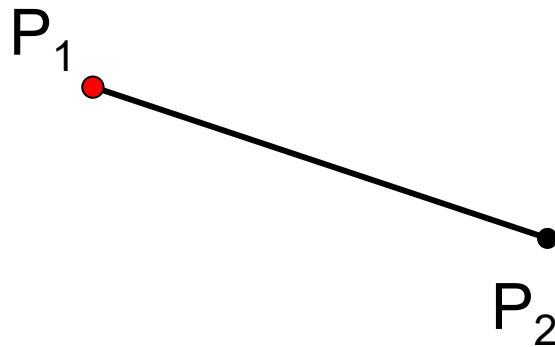
[~cg/2016/skript/Applets/Splines/App.html](#)

Bewertung von Splines

bei vielen Stützpunkten:

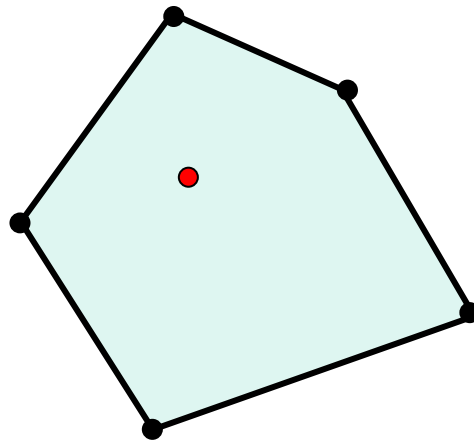
- hoher Rechenaufwand zur Lösung des Gleichungssystems
- Kein lokaler Einfluss möglich

Kontrollpunkte



$$P = (1 - t) \cdot P_1 + t \cdot P_2$$

1.0000



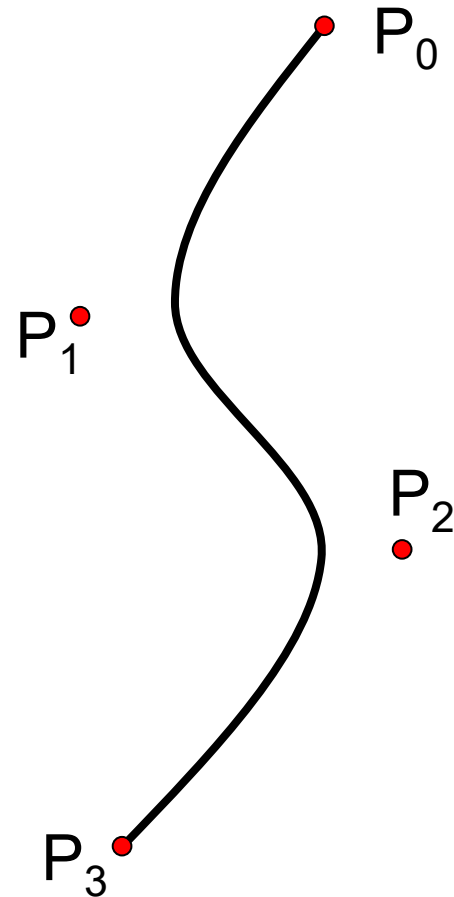
$$M = \sum_{i=0}^n m_i$$

$$P = \frac{1}{M} \sum_{i=0}^n m_i \cdot P_i$$

Bézier-Kurven

Pierre Bézier
1911-1999
Ingenieur bei Renault

Paul deCasteljau
1930 -
Ingenieur bei Citroen



Bernstein-Polynome

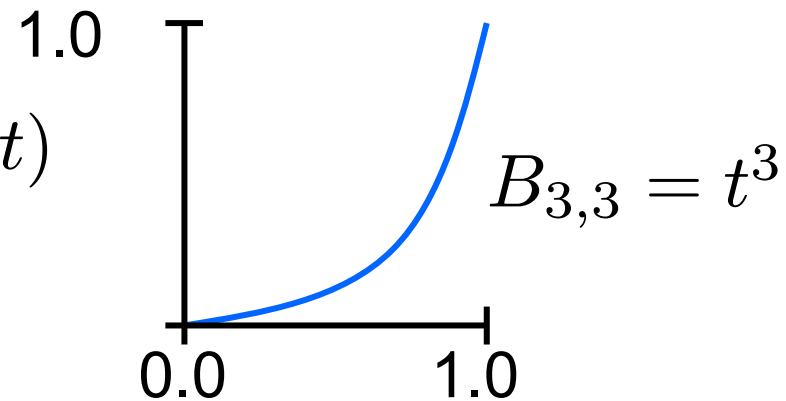
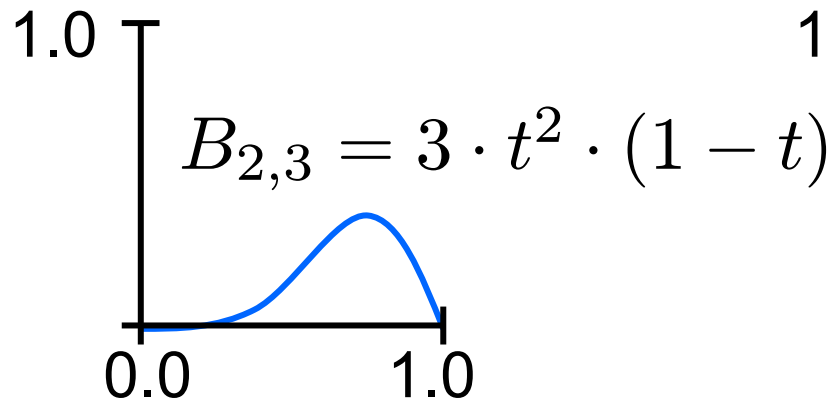
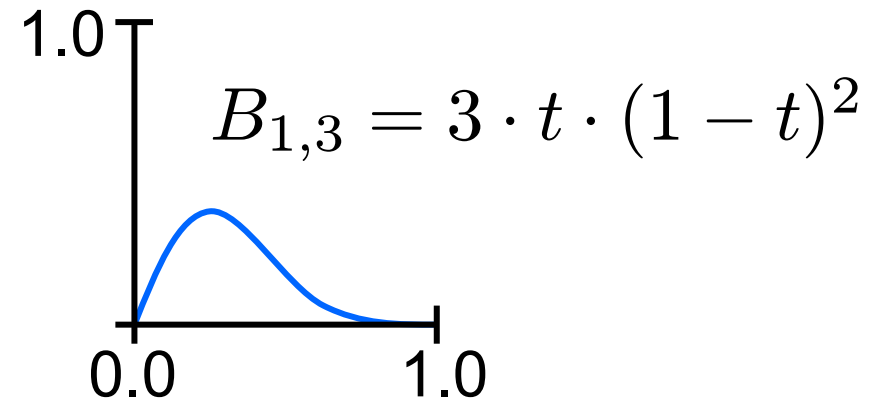
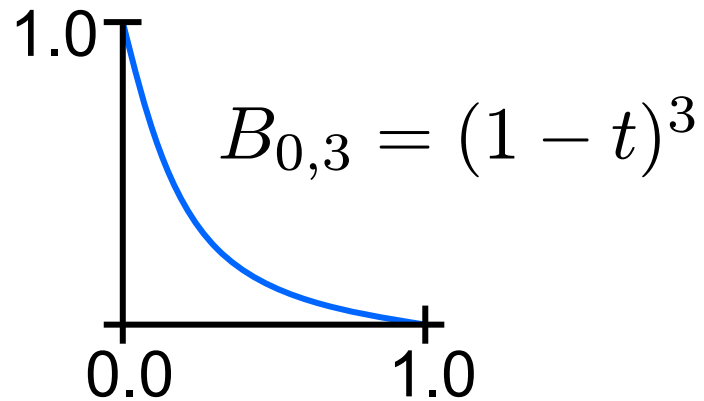
$n+1$ Kontrollpunkte

Jeder Kontrollpunkt P_i
wird gewichtet mit Bernstein-Polynom B_i

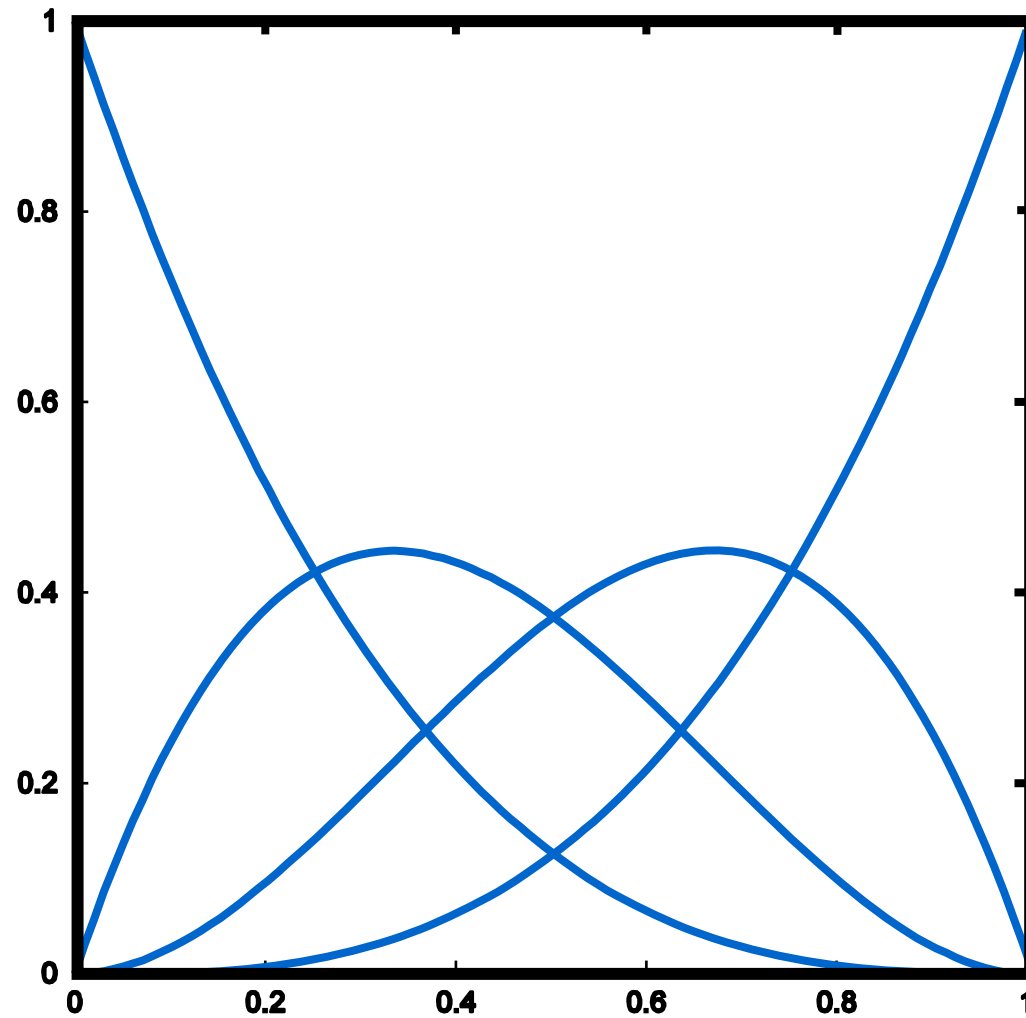
$$P(t) = \sum_{i=0}^n B_{i,n}(t) \cdot P_i, \quad 0 \leq t \leq 1$$

$$\begin{aligned} B_{i,n}(t) &= \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i}, \quad i = 0, \dots, n \\ &= \frac{n!}{i! \cdot (n-i)!} \cdot t^i \cdot (1-t)^{n-i} \end{aligned}$$

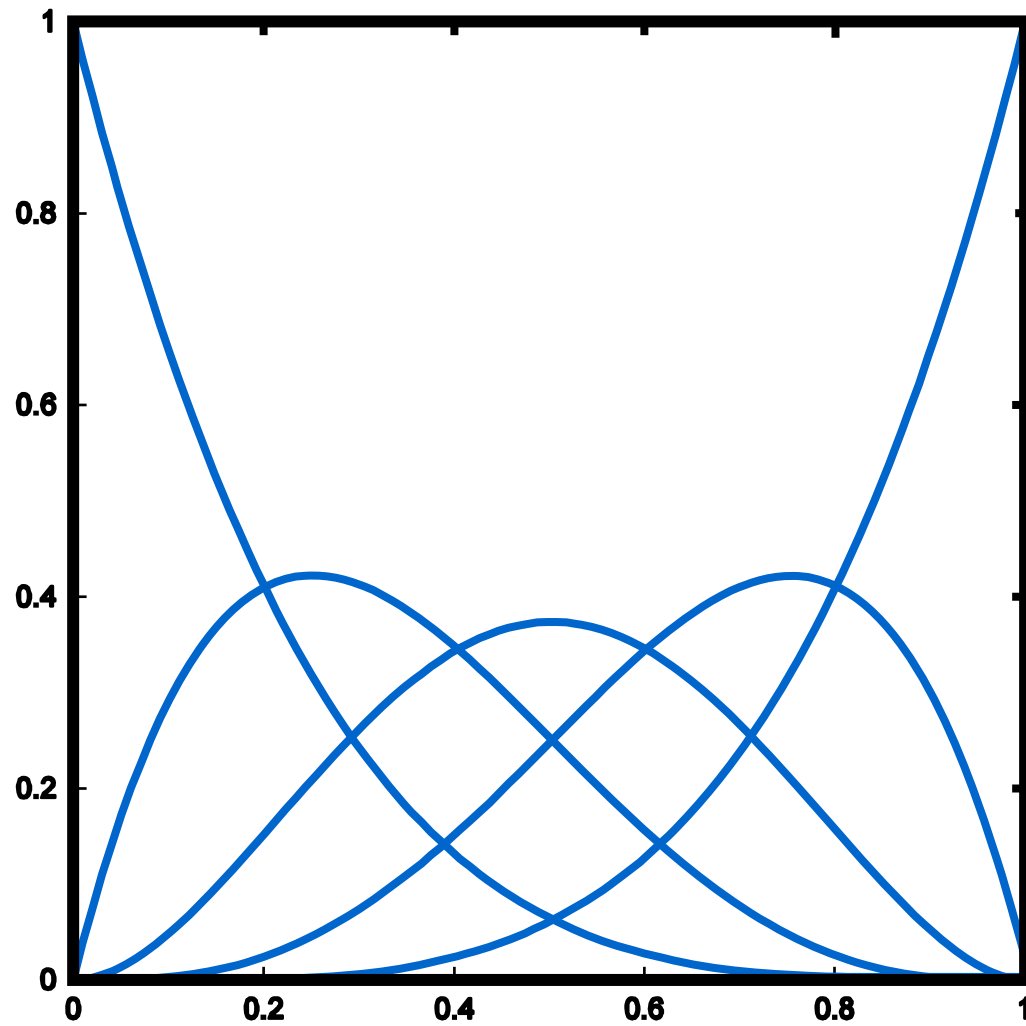
Bernstein-Polynome 3. Grades



Bernsteinpolynome 3. Grades



Bernsteinpolynome 4. Grades



Eigenschaften der Bernstein-Polynome

Alle Bernsteinpolynome sind positiv auf $[0,1]$

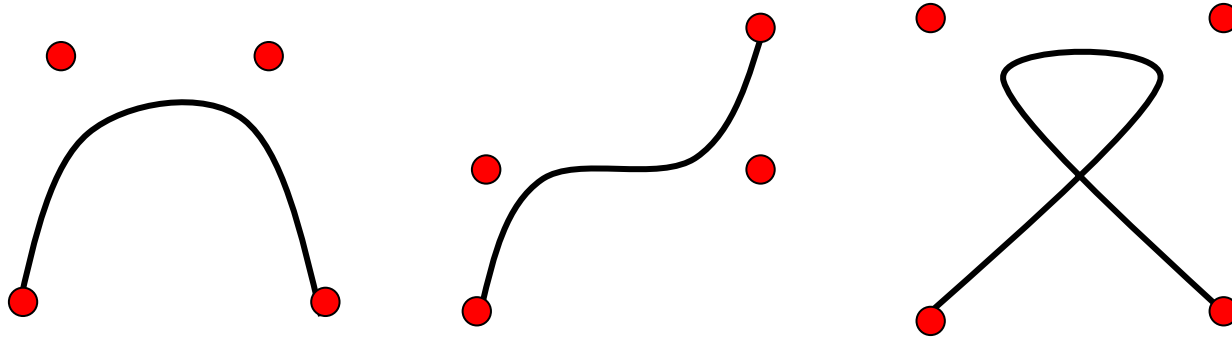
$$\sum_{i=0}^n B_{i,n}(t) = 1 \quad \text{für jedes feste } t$$

$$B_{i,n}(t) = B_{n-i,n}(1-t)$$

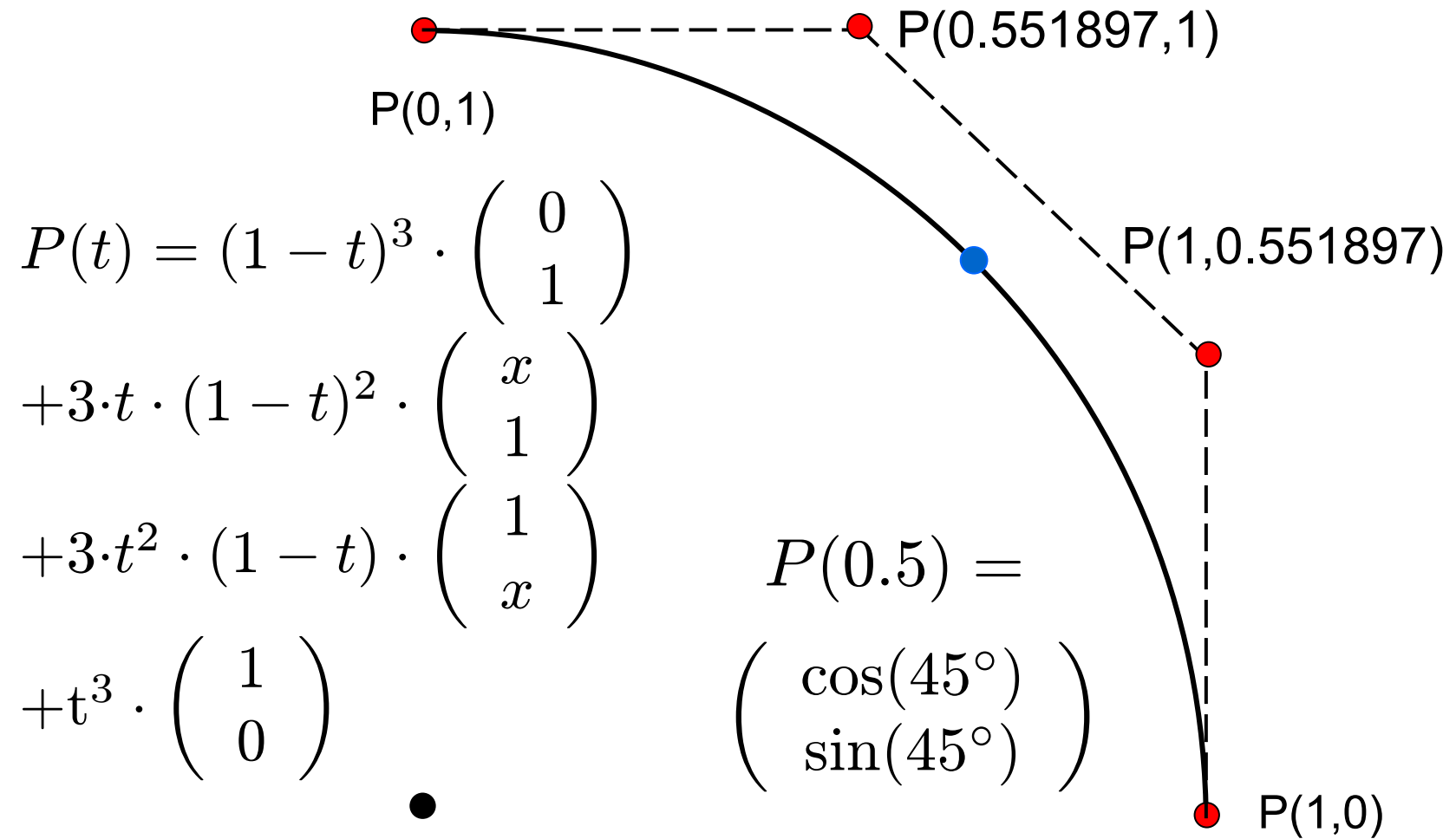
Maxima von $B_{i,n}(t)$ bei $t = \frac{i}{n}, i = 0, \dots, n$

$$\frac{dB_{i,n}(t)}{dt} = n \cdot (B_{i-1,n-1}(t) - B_{i,n-1}(t)), i > 0$$

Beispiele für Bézierkurven



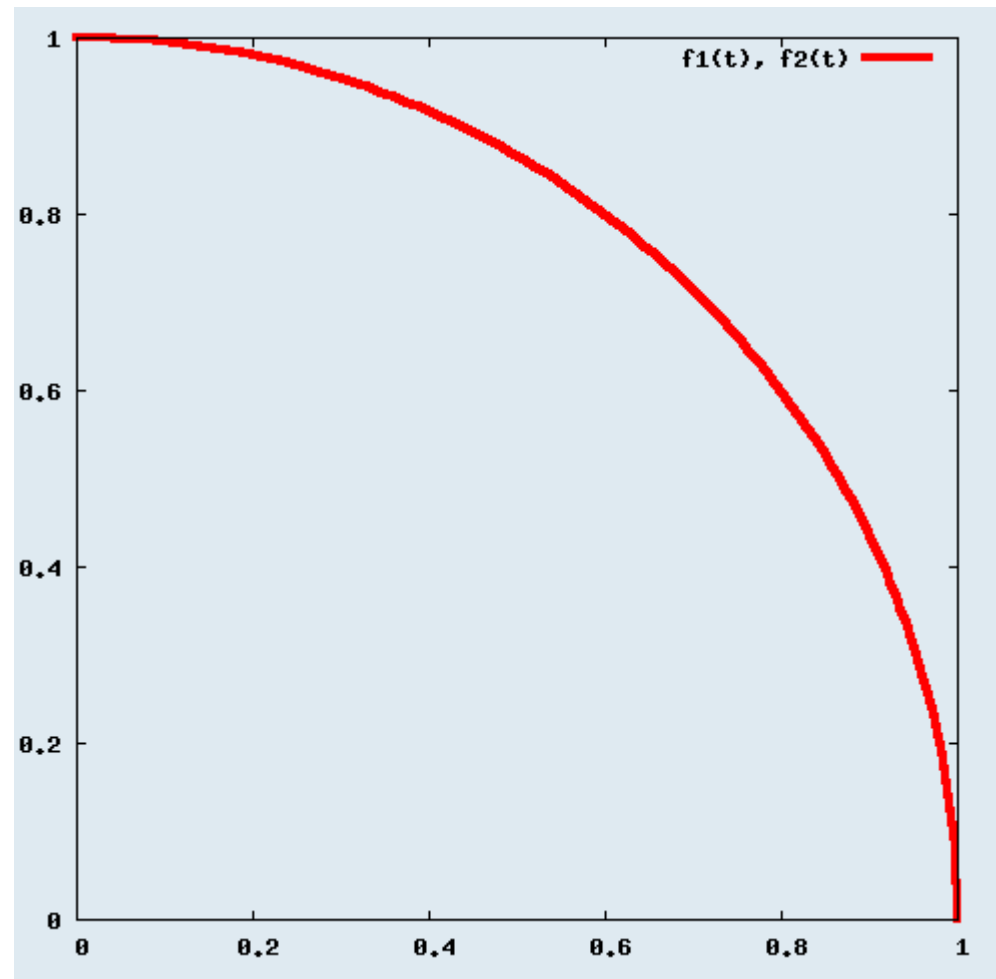
Approximation von Viertelkreis



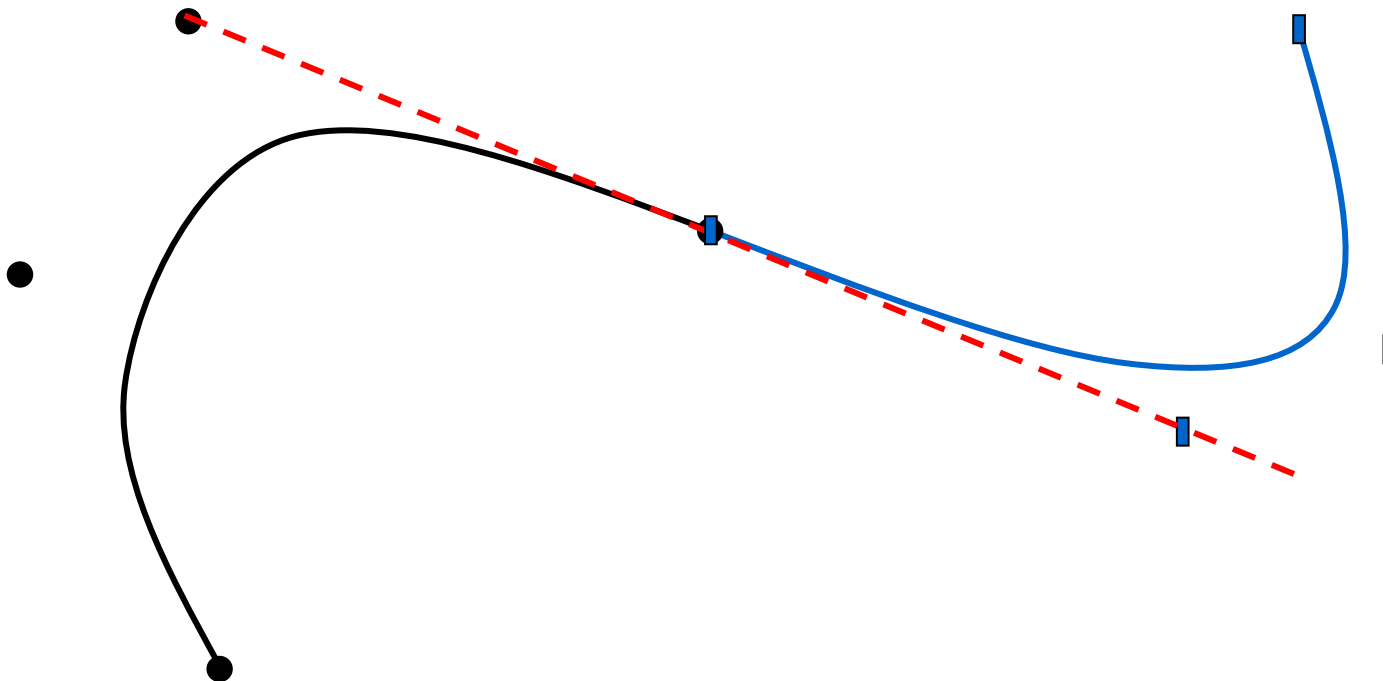
Bezier-Viertelkreis

$$f_1(t) = 3 \cdot t \cdot (1 - t)^2 \cdot 0.551897 + 3 \cdot t^2 \cdot (1 - t) + t^3$$

$$f_2(t) = (1 - t)^3 + 3 \cdot t \cdot (1 - t)^2 + 3 \cdot t^2 \cdot (1 - t) \cdot 0.551897$$



Bézier-Kurven aneinandersetzen



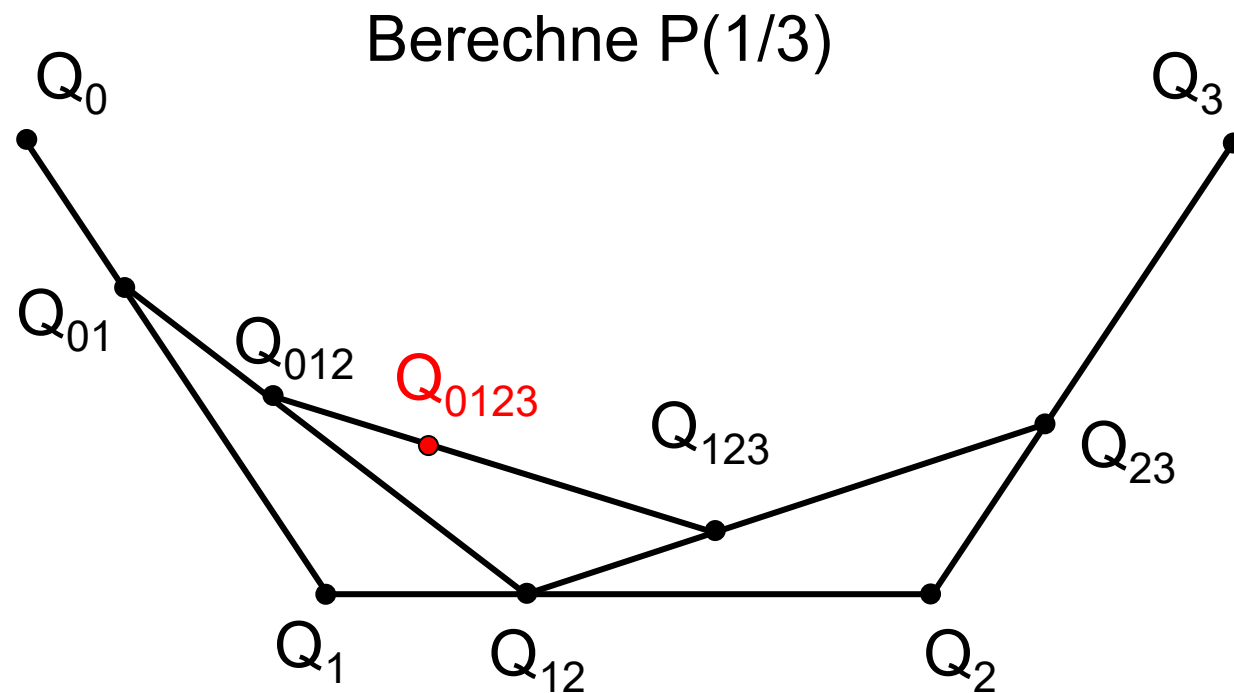
Bézier-Kurve nach de Casteljau

$$P_{0,n}(t) = (1 - t) \cdot P_{0,n-1}(t) + t \cdot P_{1,n}(t)$$

$$P_{0,3}\left(\frac{1}{3}\right)$$

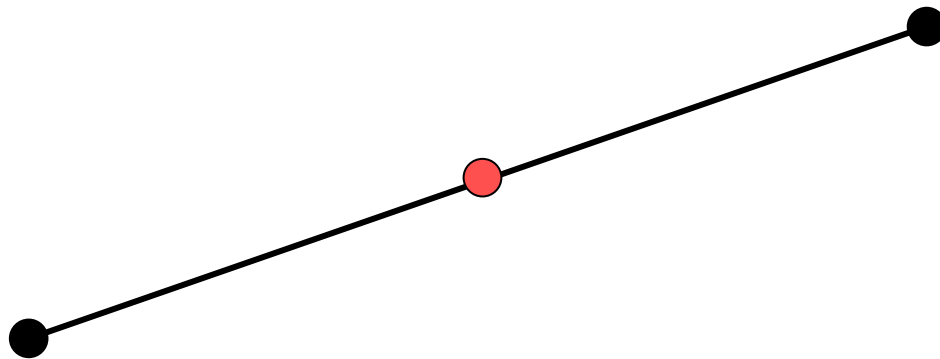
$$\underbrace{\frac{2}{3} \cdot P_{0,2}\left(\frac{1}{3}\right) + \frac{1}{3} \cdot P_{1,3}\left(\frac{1}{3}\right)}_{\underbrace{\frac{2}{3} \cdot P_{0,1}\left(\frac{1}{3}\right) + \frac{1}{3} \cdot P_{1,2}\left(\frac{1}{3}\right)} \quad \underbrace{\frac{2}{3} \cdot P_{1,2}\left(\frac{1}{3}\right) + \frac{1}{3} \cdot P_{2,3}\left(\frac{1}{3}\right)}}$$

Berechnung nach de Casteljau



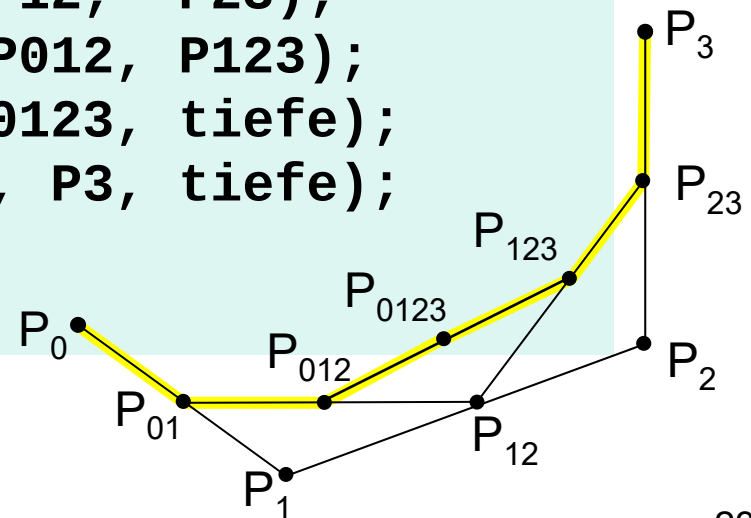
halbiere

```
Point halbiere(Point P1, Point P2){  
    Point P = new Point();  
    P.x = (P1.x + P2.x)/2;  
    P.y = (P1.y + P2.y)/2;  
    return P;  
}
```



Rekursion nach de Casteljau

```
void bezier(Point P0, Point P1,  
            Point P2, Point P3, int tiefe) {  
    if (tiefe == 0) drawLine(P0, P3); else {  
        tiefe--;  
        Point P01 = halbiere(P0, P1);  
        Point P12 = halbiere(P1, P2);  
        Point P23 = halbiere(P2, P3);  
        Point P012 = halbiere(P01, P12);  
        Point P123 = halbiere(P12, P23);  
        Point P0123 = halbiere(P012, P123);  
        bezier(P0, P01, P012, P0123, tiefe);  
        bezier(P0123, P123, P23, P3, tiefe);  
    }  
}
```



Java-Applet zu Bézier-Kurven

[~cg/2016/skript/Applets/Splines/App.html](http://cg/2016/skript/Applets/Splines/App.html)

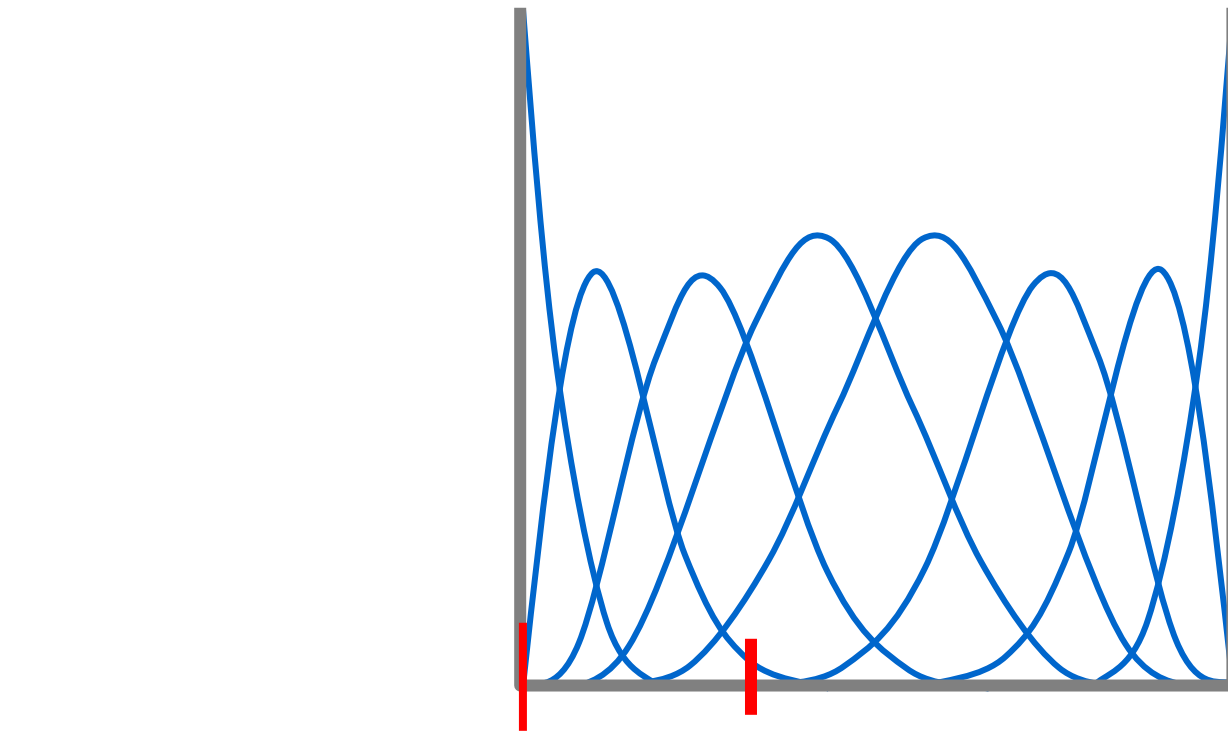
Bewertung von Bézier-Kurven

- Grad der Polynome
ist abhängig von Zahl der Kontrollpunkte
- Alle Kontrollpunkte
wirken auf die gesamte Kurve

Wunsch:

- fester Polynomgrad
- lokaler Einfluss

Wunsch: Lokaler Einfluss



z.B. Einfluss von maximal vier Kontrollpunkten

B-Splines

- wähle $n+1$ Kontrollpunkte $P_0, P_1, \dots, P_n$
- wähle k
- wähle Knotenvektor $t_0, t_1, t_2, \dots, t_{n+k}$
- konstruiere $n+1$ Gewichtsfunktionen
- Gewichtsfunktion wirkt auf max. k Abschnitte
- Gewichtsfunktion ist Polynom vom Grad $k-1$ (abschnittsweise)

Konstruktion von B-Splines

$$T = (t_0, t_1, \dots, t_{n+k}), t_j \leq t_{j+1}$$

$$N_{j,1}(t) = \begin{cases} 1 & \text{falls } t_j \leq t < t_{j+1} \\ 0 & \text{sonst} \end{cases} \quad j = 0, \dots, n+k-1$$

$$N_{j,i}(t) = \frac{t-t_j}{t_{j+i-1}-t_j} \cdot N_{j,i-1}(t) + \frac{t_{j+i}-t}{t_{j+i}-t_{j+1}} \cdot N_{j+1,i-1}(t)$$

$$P(t) = \sum_{i=0}^n N_{i,k}(t) \cdot P_i$$

Beispiel für Knotenvektor

$$t_j = \begin{cases} 0 & \text{falls } j < k \\ j - k + 1 & \text{falls } k \leq j \leq n \\ n - k + 2 & \text{falls } j > n \end{cases}$$

$$j \in \{0, \dots, n + k\} \quad t \in [0, n - k + 2]$$

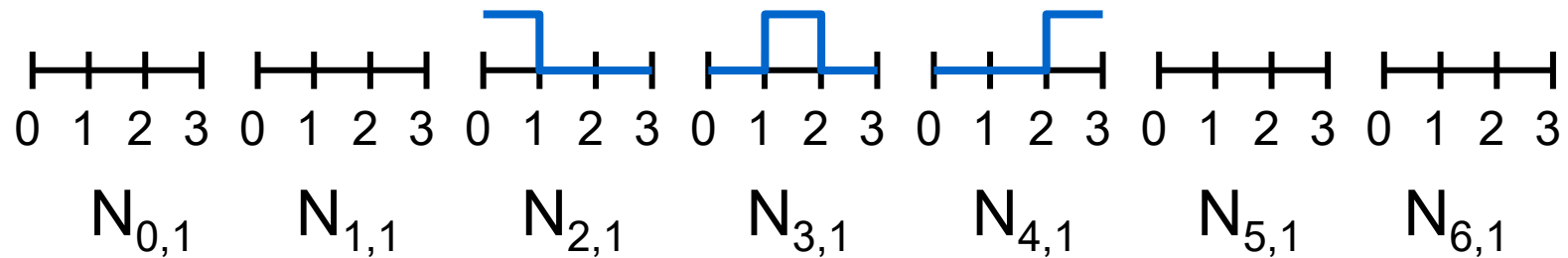
Beispiel: $n = 4, k = 3$

Knotenvektor: $T = (0, 0, 0, 1, 2, 3, 3, 3)$

Beispiel: $n = 8, k = 4$

Knotenvektor: $T = (0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6)$

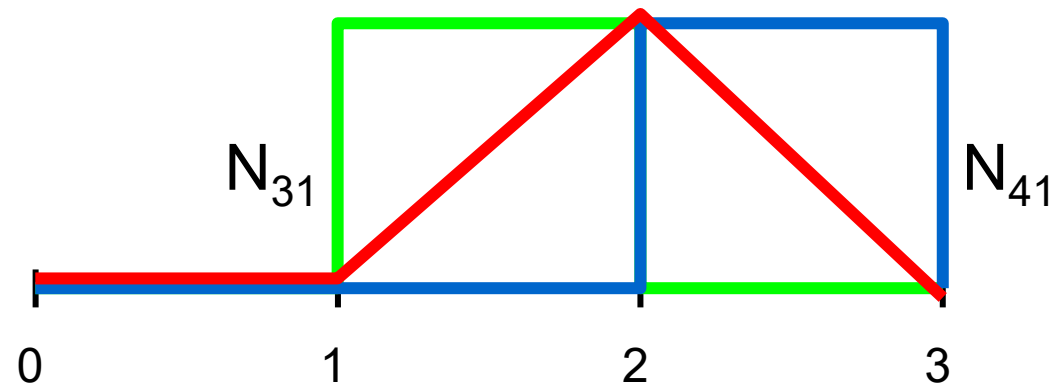
Konstruktion von $N_{i,1}$ für $n=4$, $k=3$



$n+k$ Ausgangsfunktionen

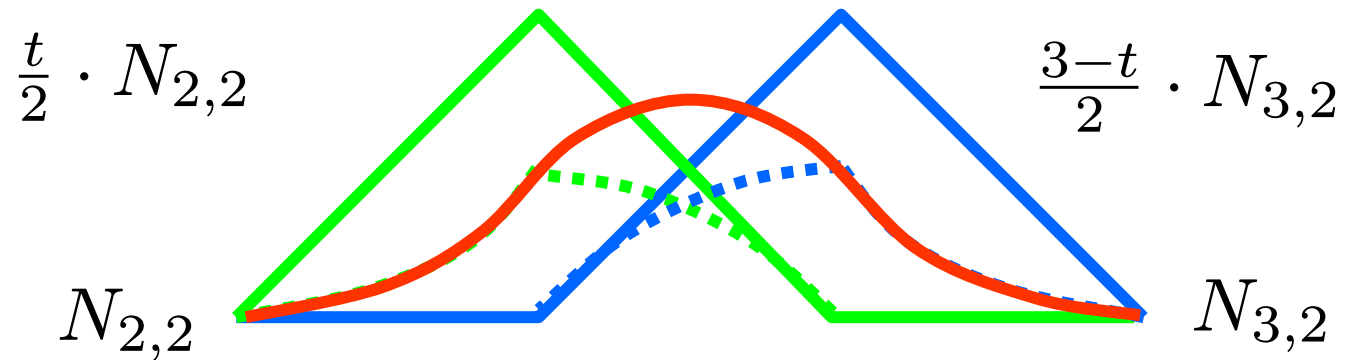
Konstruktion von $N_{3,2}$

$$N_{3,2} := (t - 1) \cdot N_{3,1} + (3 - t) \cdot N_{4,1}$$



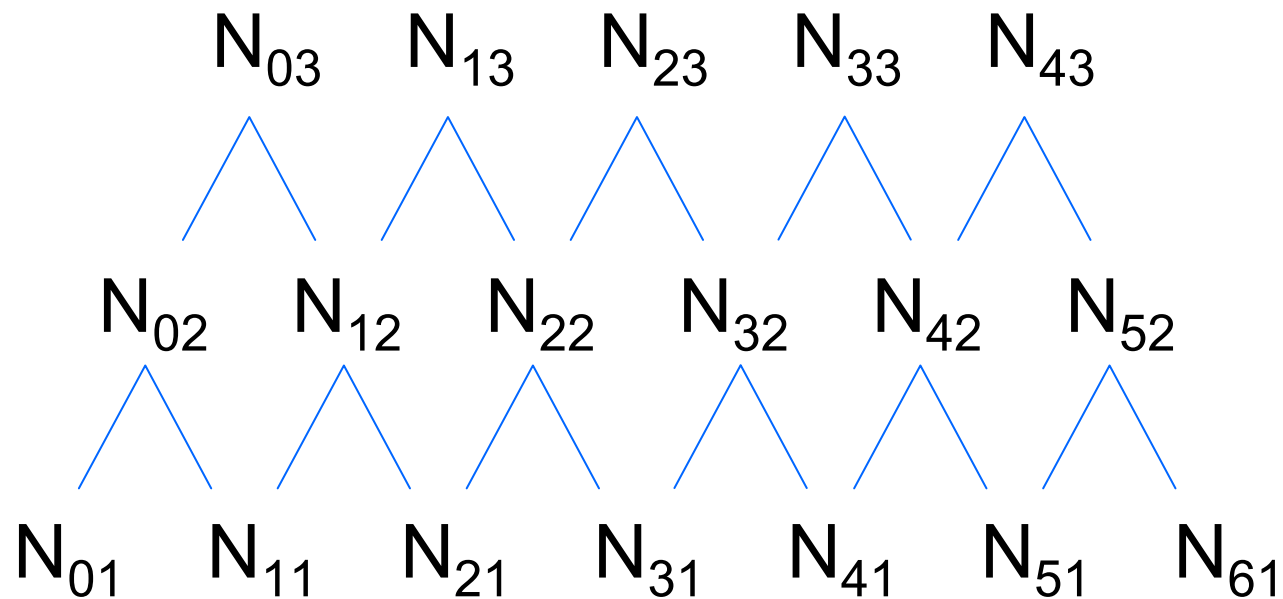
Konstruktion von $N_{2,3}$

$$N_{2,3} = \frac{t}{2} \cdot N_{2,2} + \frac{3-t}{2} \cdot N_{3,2}$$



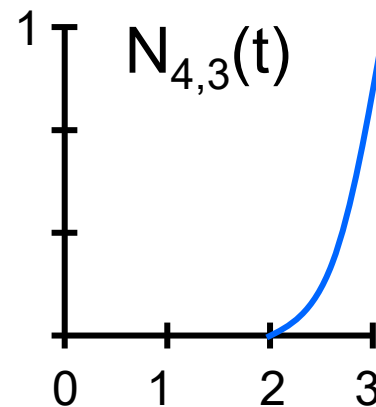
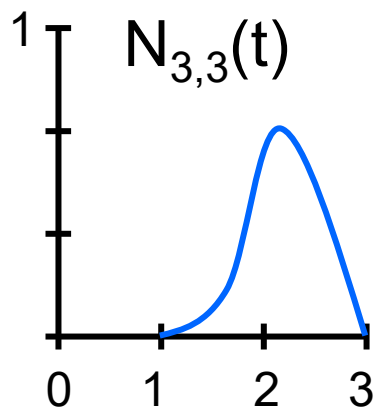
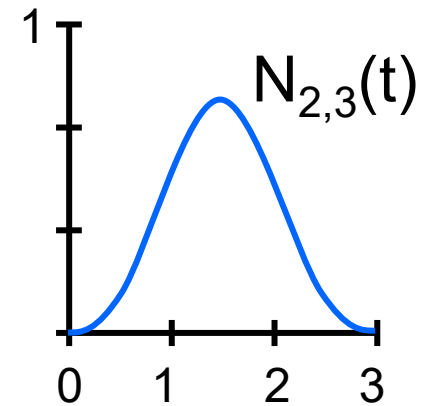
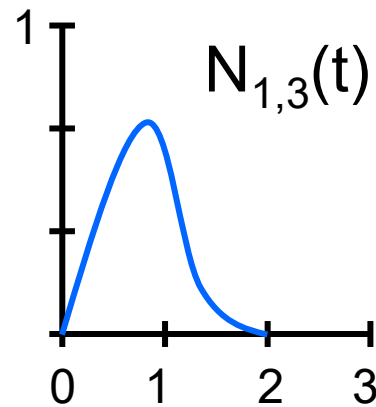
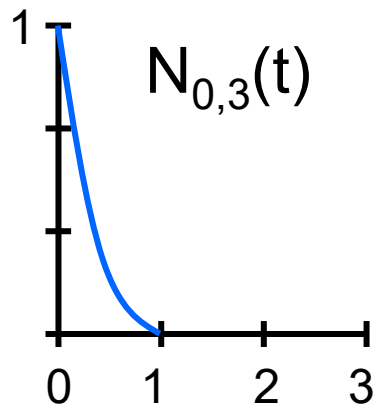
Konstruktion von $N_{i,k}$

$n+1$ Gewichtsfunktionen



$n+k$ Ausgangsfunktionen

Gewichtspolynome für $n=4$, $k=3$



$$P(t) = \sum_{i=0}^4 N_{i,3}(t) \cdot P_i$$

Sonderfall

$$k = n + 1$$

$$T = (\underbrace{0, \dots, 0}_{k \text{ mal}}, \underbrace{1, \dots, 1}_{k \text{ mal}}).$$

$$\begin{aligned} N_{i,k}(t) &= \frac{(k-1)!}{i! \cdot (k-1-i)!} \cdot t^i \cdot (1-t)^{k-1-i} \\ &= \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i} = B_{i,n}(t) \end{aligned}$$

⇒ Bernstein-Polynome sind Spezialfall von B-Splines

Java-Applet von B-Splines

[~cg/2016/skript/Applets/Splines/App.html](#)

Bewertung von B-Splines

- tolle Sache !

aber:

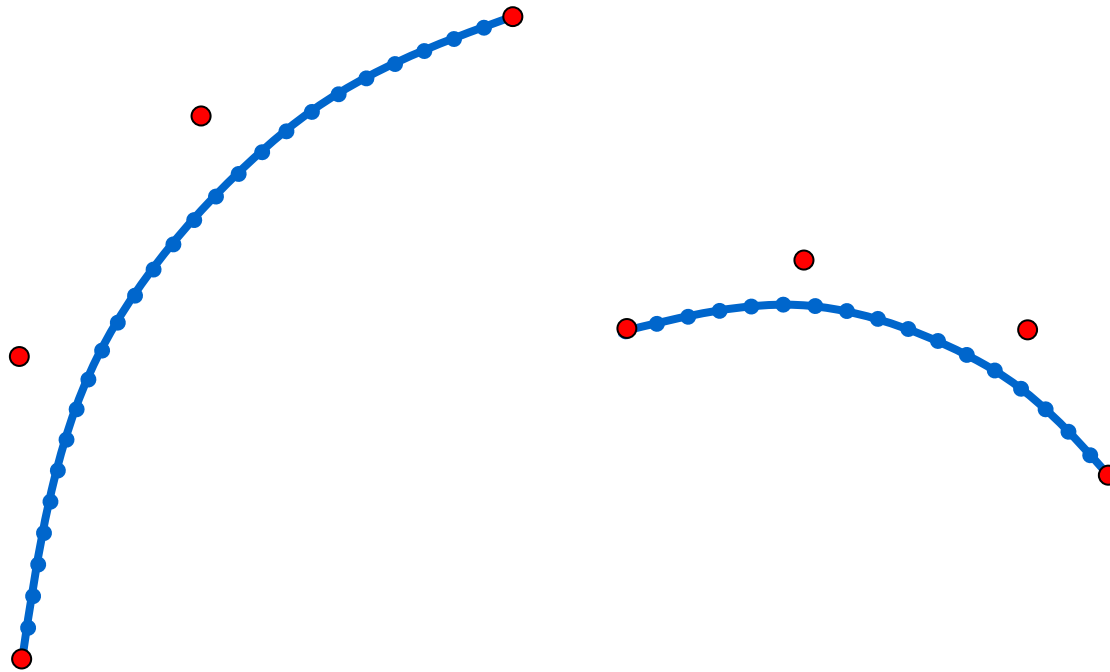
- kein Kreis
- nur invariant bzgl. affiner Abbildungen

Affine Abbildung

$$\vec{y} = A \cdot \vec{x} + \vec{b}$$

- Translation
 - Skalierung
 - Rotation
 - Scherung
 - Spiegelung
- sind affine Abbildungen

Invarianz bzgl Abbildung



B-Splines sind nicht invariant unter Projektion !

NURBS

Non Uniform

Rational

B-Splines

t_j nicht äquidistant

Gewichtsquotient

Gewichtsfunktion

NURBS

Punkte P_i Knoten t_j Gewichte h_i

$$\sum_{i=0}^n N_{i,k}(t) = 1$$

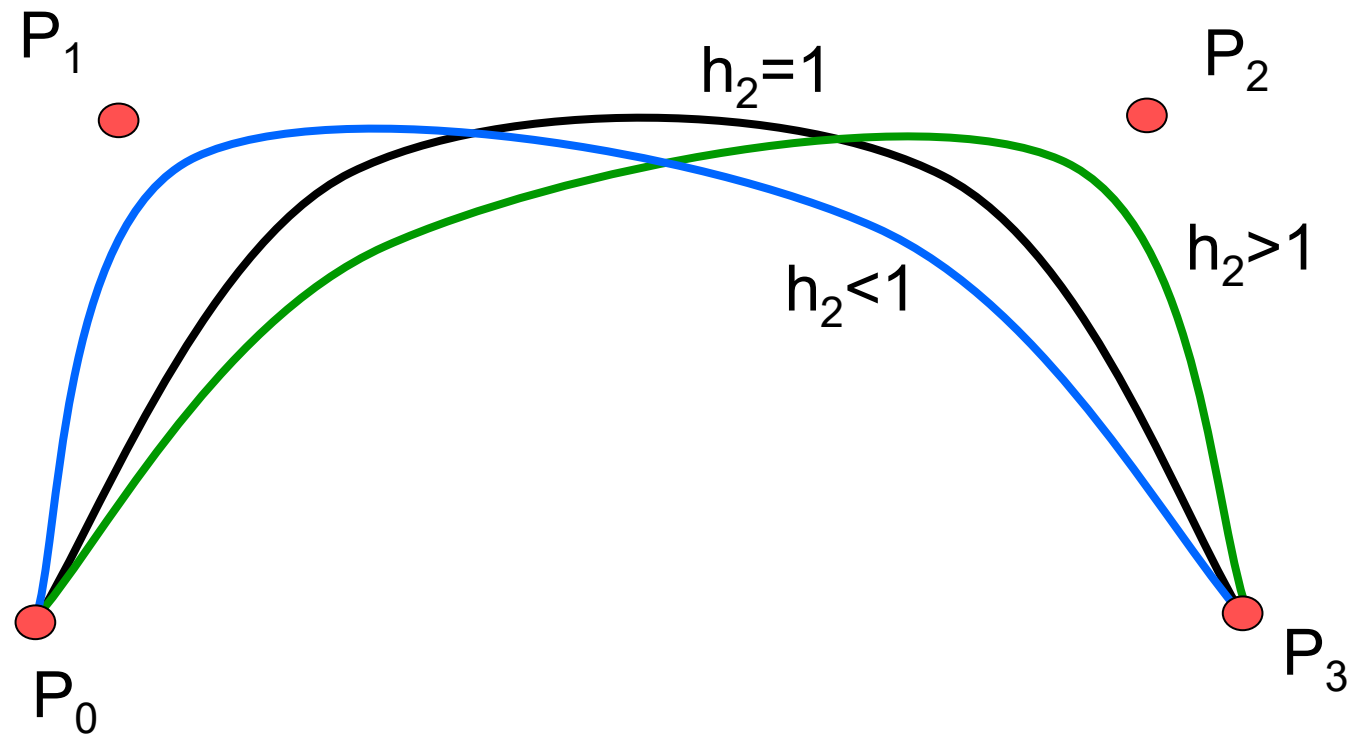
$$\sum_{i=0}^n h_i \cdot N_{i,k}(t) = z$$

normiere auf 1

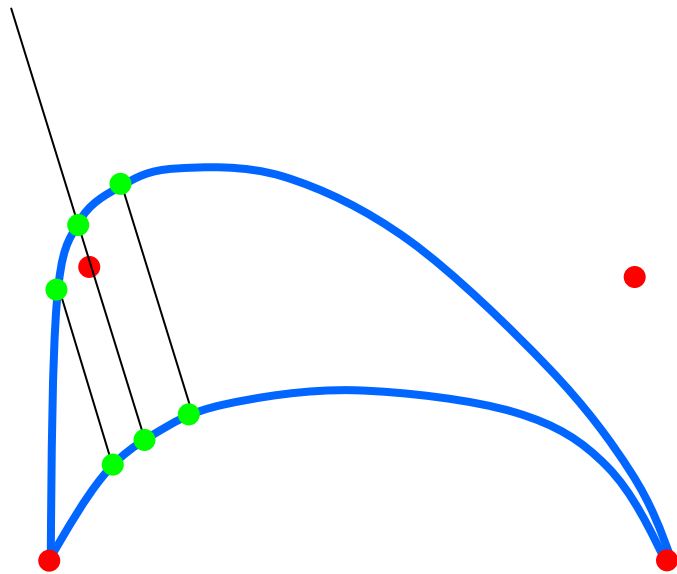
$$R_{i,k}(t) = \frac{h_i \cdot N_{i,k}(t)}{\sum_{j=0}^n h_j \cdot N_{j,k}(t)}$$

$$P(t) = \sum_{i=0}^n R_{i,k}(t) \cdot P_i$$

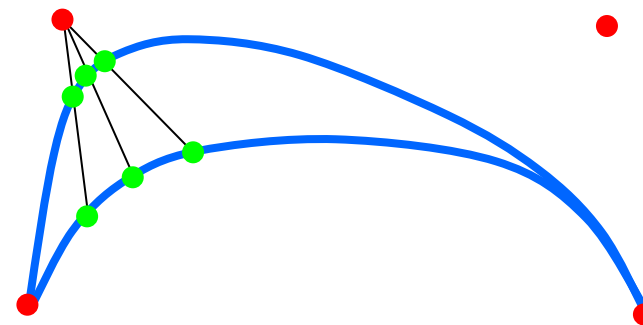
Auswirkung der NURBS-Gewichte



Möglichkeiten der Einflussnahme

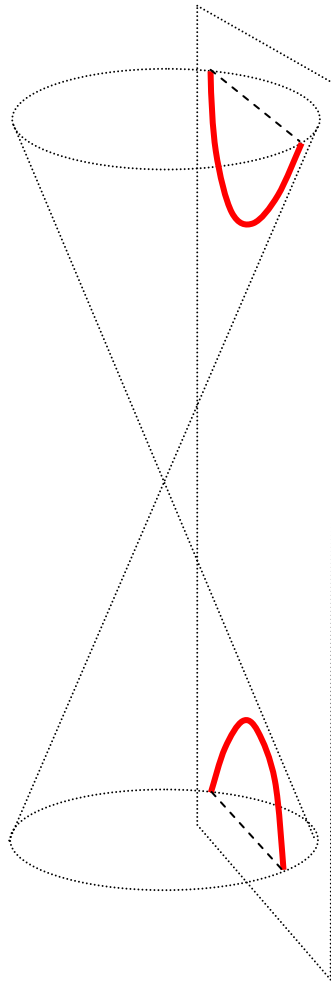


Kontrollpunkt verschieben:
Punkte wandern parallel



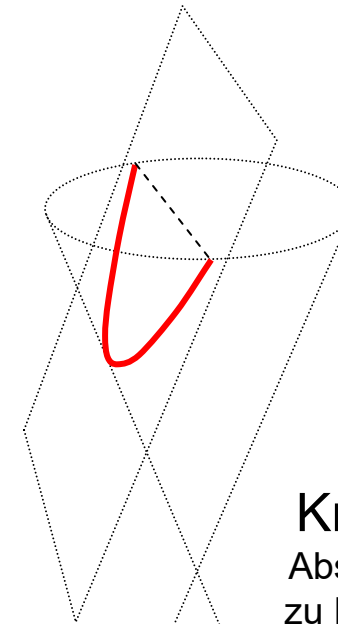
Gewicht erhöhen:
Punkte wandern auf
Kontrollpunkt zu

Kegelschnitte



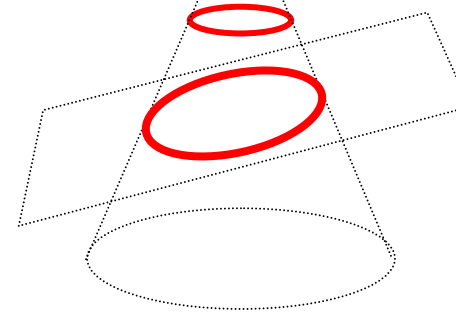
Hyperbel
Differenz der
Abstände
konstant

Parabel
Abstand zu
Punkt und
Gerade
gleich

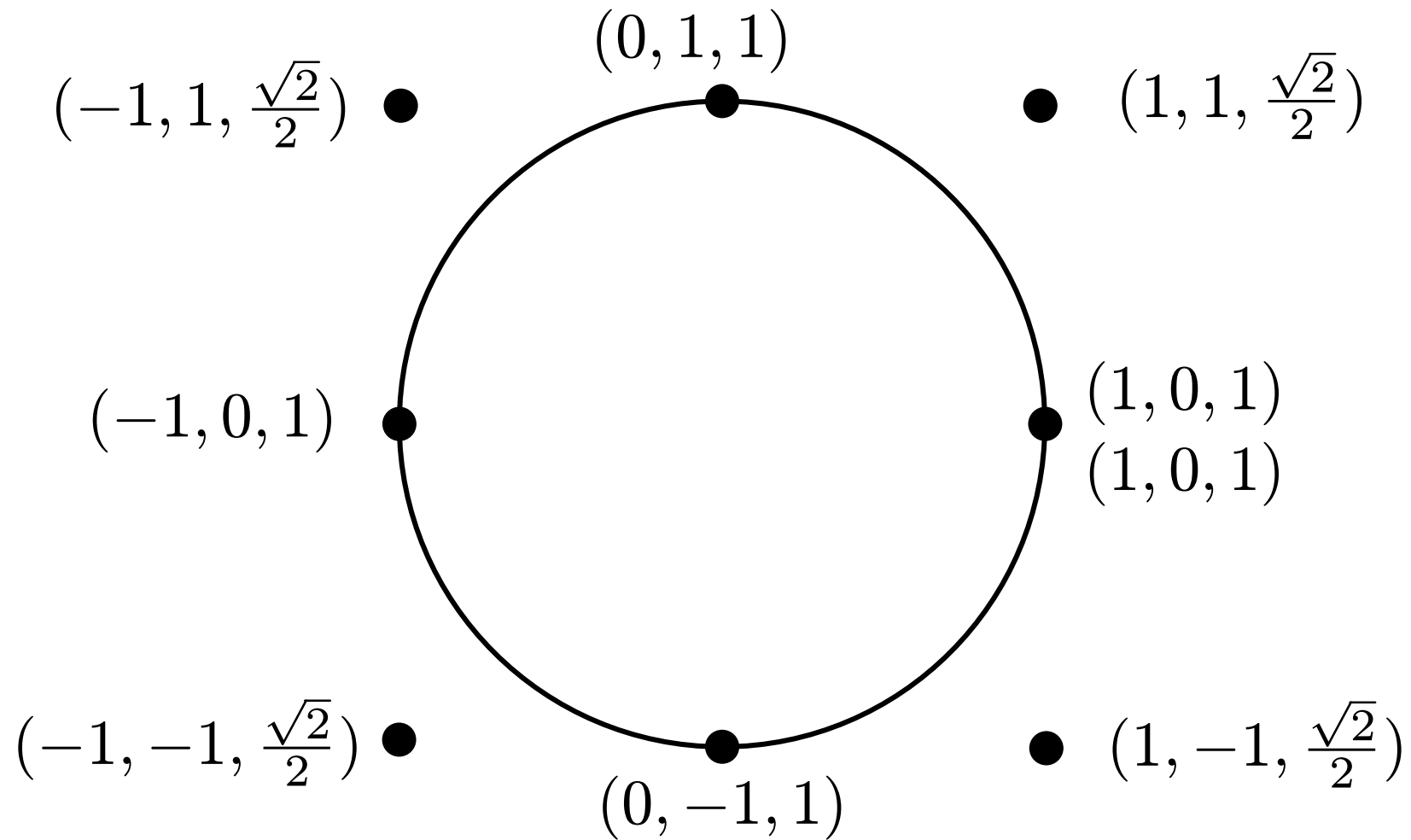


Kreis
Abstand
zu Punkt
konstant

Ellipse
Summe der
Abstände
konstant



NURBS-Kreis



Viertelkreis durch NURBS

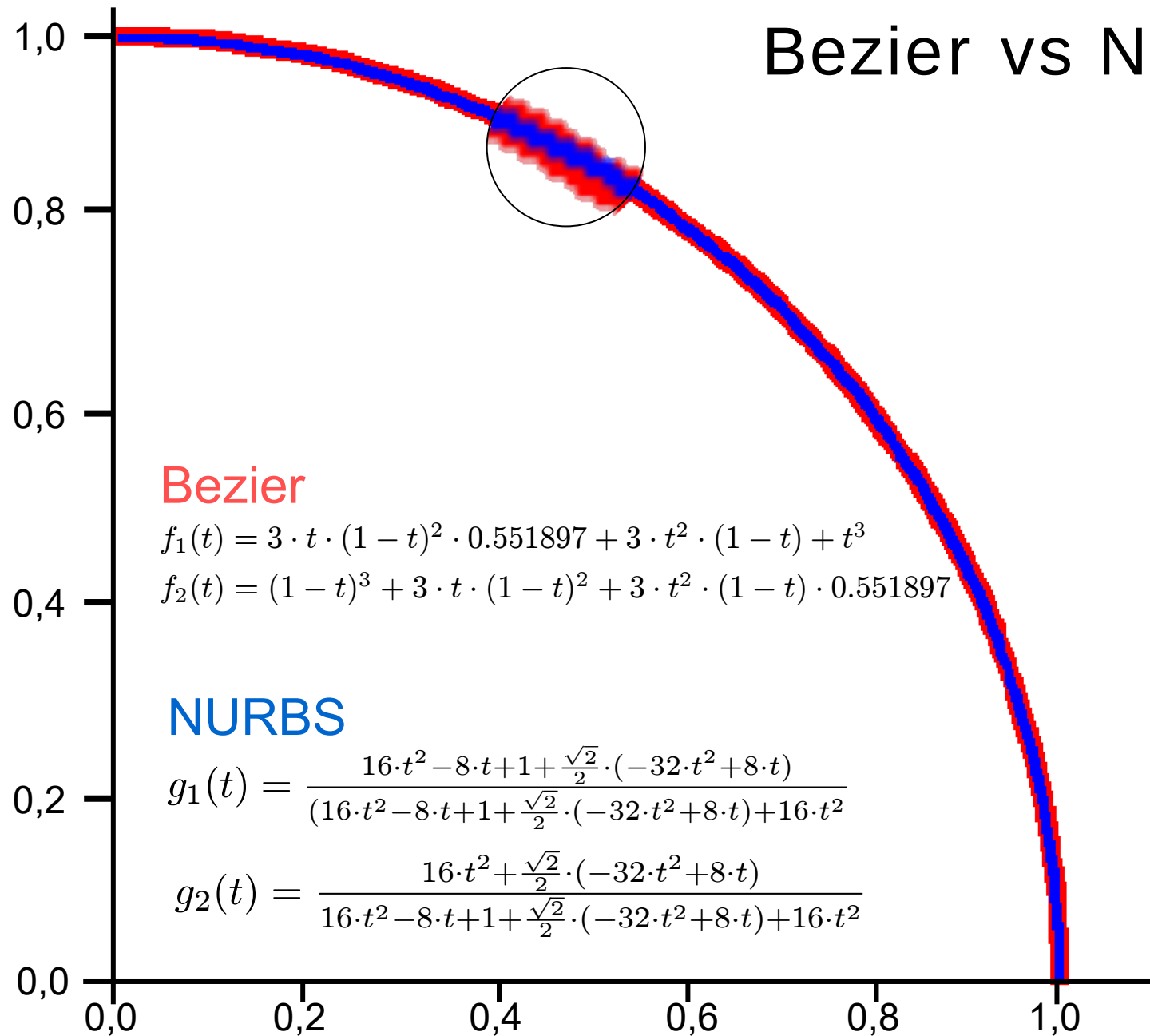
$$P_x(t) = \left(\frac{16t^2 - 8t + 1 + \frac{\sqrt{2}}{2}(-32t^2 + 8t)}{16t^2 - 8t + 1 + \frac{\sqrt{2}}{2}(-32t^2 + 8t) + 16t^2} \right)$$

$$P_y(t) = \left(\frac{\frac{\sqrt{2}}{2}(-32t^2 + 8t) + 16t^2}{16t^2 - 8t + 1 + \frac{\sqrt{2}}{2}(-32t^2 + 8t) + 16t^2} \right)$$

korrekter Kreis, denn $P_x^2(t) + P_y^2(t) =$

$$\left(\frac{16t^2 - 8t + 1 + \frac{\sqrt{2}}{2}(-32t^2 + 8t)}{16t^2 - 8t + 1 + \frac{\sqrt{2}}{2}(-32t^2 + 8t) + 16t^2} \right)^2 + \left(\frac{\frac{\sqrt{2}}{2}(-32t^2 + 8t) + 16t^2}{16t^2 - 8t + 1 + \frac{\sqrt{2}}{2}(-32t^2 + 8t) + 16t^2} \right)^2 = 1$$

Bezier vs NURBS





Wer kurvt am besten ?

Splines Bézier B-Splines NURBS

konstanter Polynomgrad	+	-	+	+
lokaler Einfluss	-	-	+	+
effiziente Speicherung	+	+	+	+
Kreis möglich	-	-	-	+
invariant bzgl. Affine Abb.	+	+	+	+
invariant bzgl. Projektion	-	-	-	+