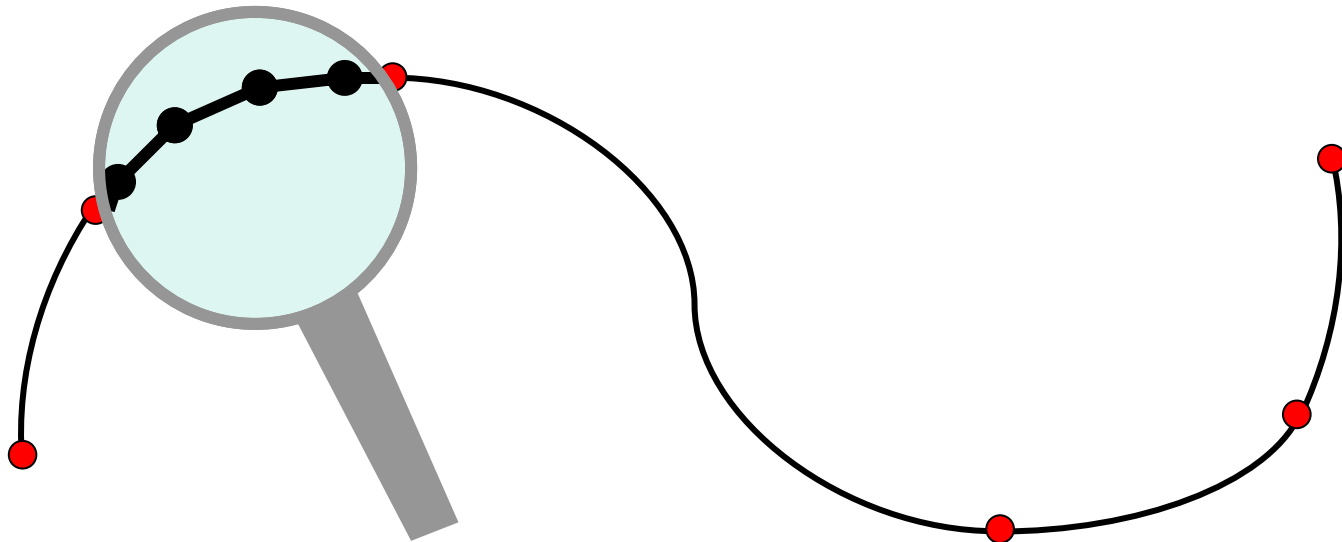


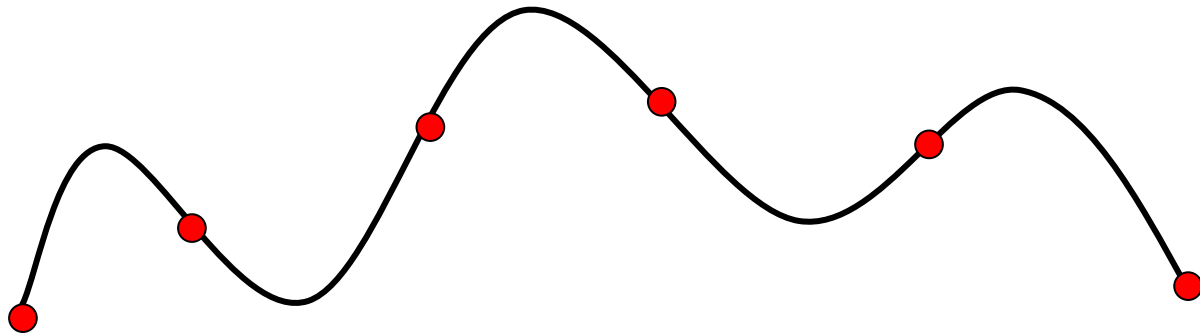
Kapitel 7: 2D-Kurven

Spezifikation einer Kurve



Stützpunkte P_0, P_1, \dots, P_n

Algebraischer Ansatz



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

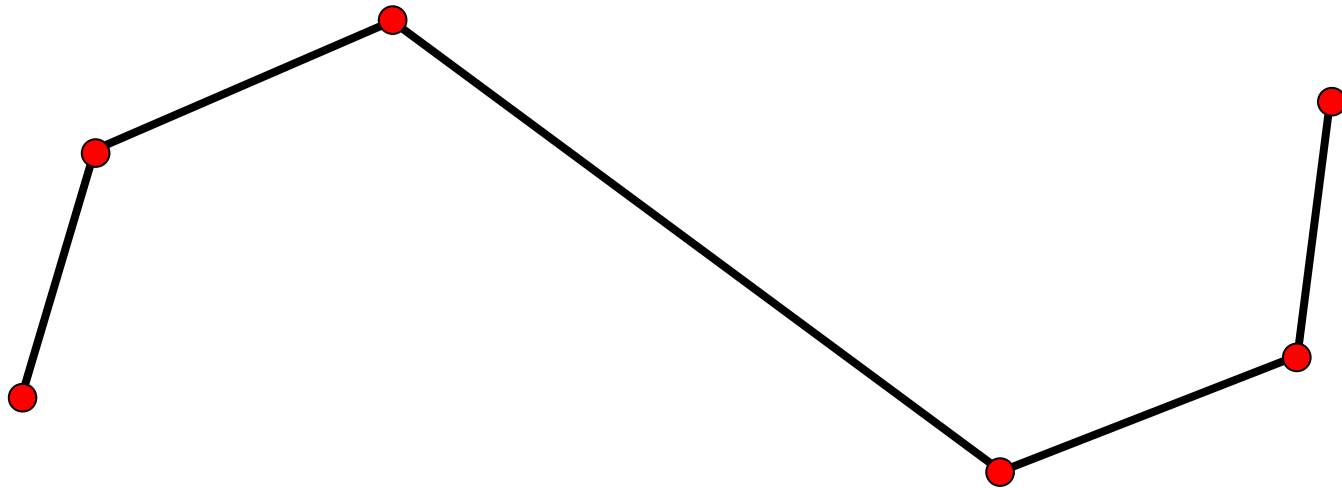
$$y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \dots + a_1 \cdot x + a_0$$

Oszillation!

Rechenaufwand!

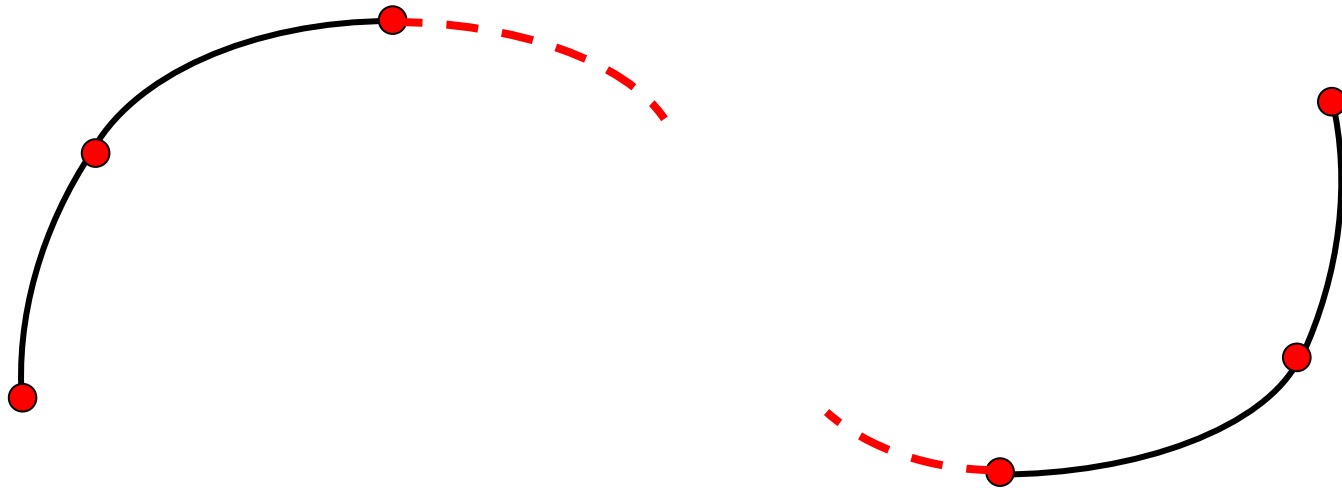
Rundungsfehler !

lineare Splines



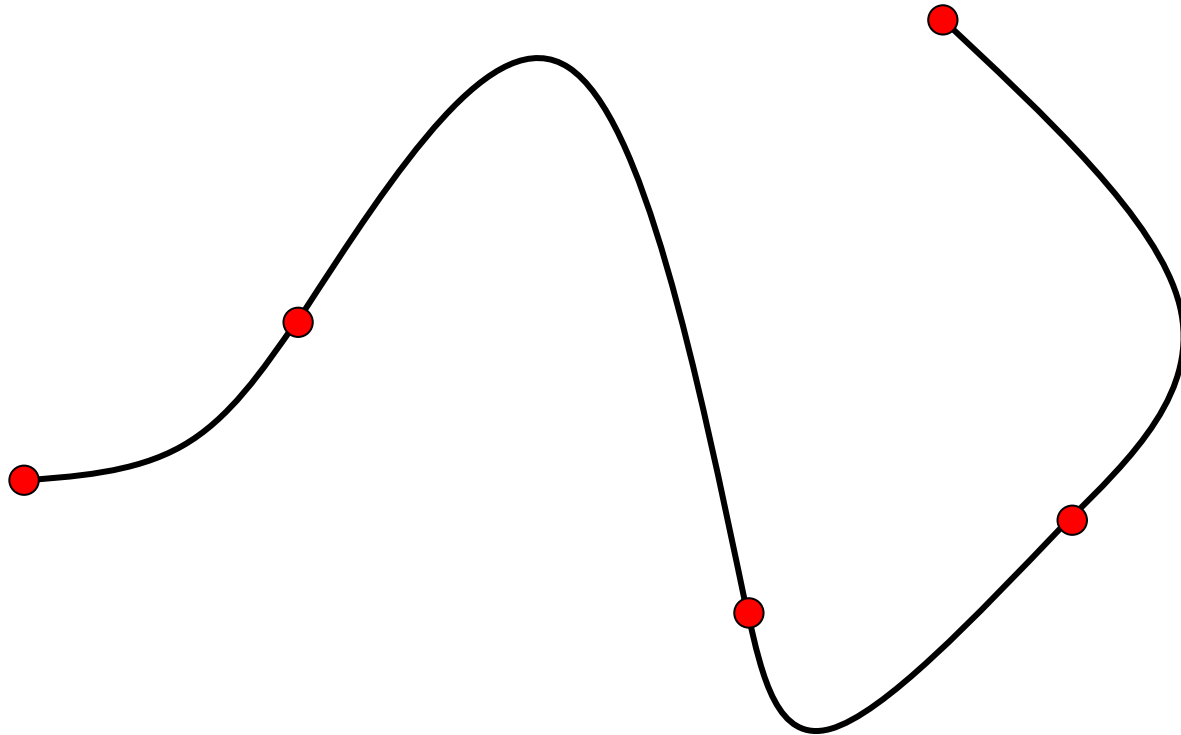
verbinde zwei aufeinanderfolgende Punkte
durch eine Gerade

quadratische Splines

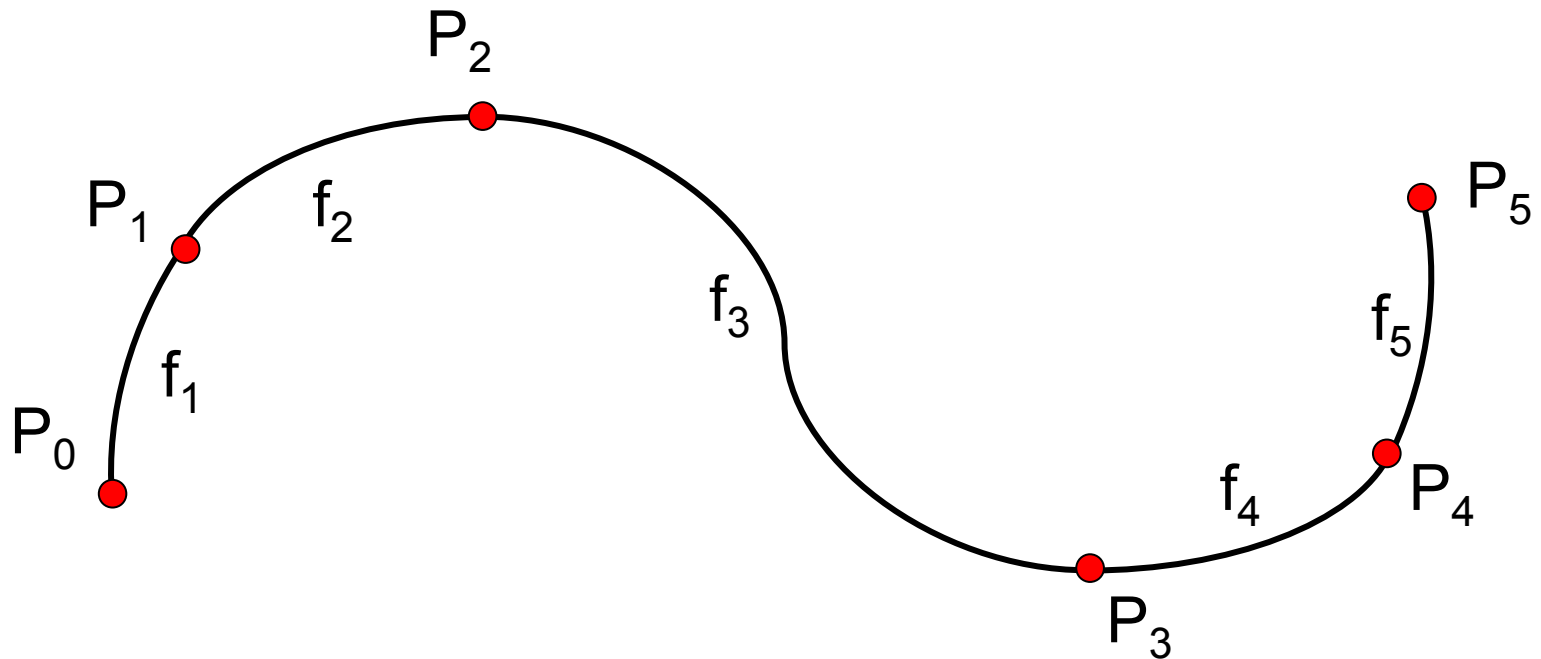


verbinde zwei aufeinanderfolgende Punkte
durch eine Kurve 2. Grades

quadratische Splines

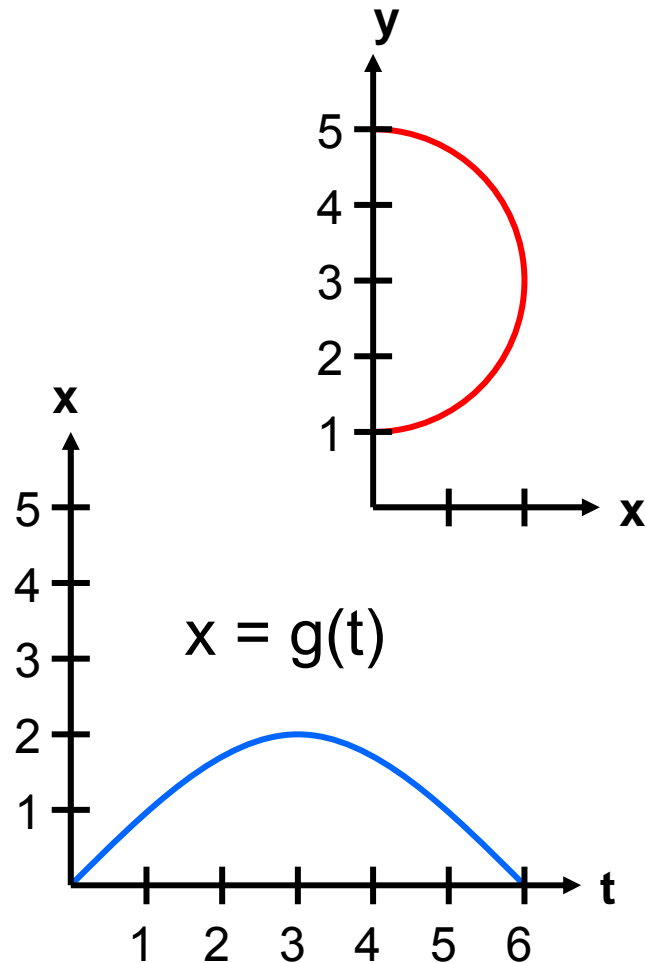


kubische Splines

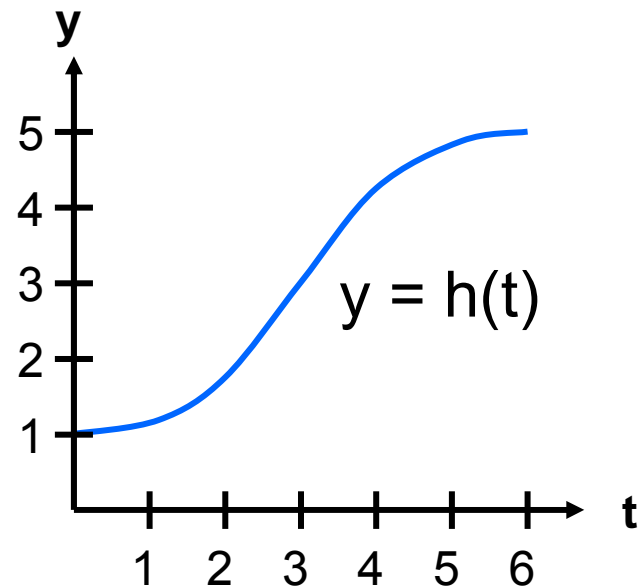


Verbinde zwei aufeinanderfolgende Punkte
durch eine Kurve 3. Grades

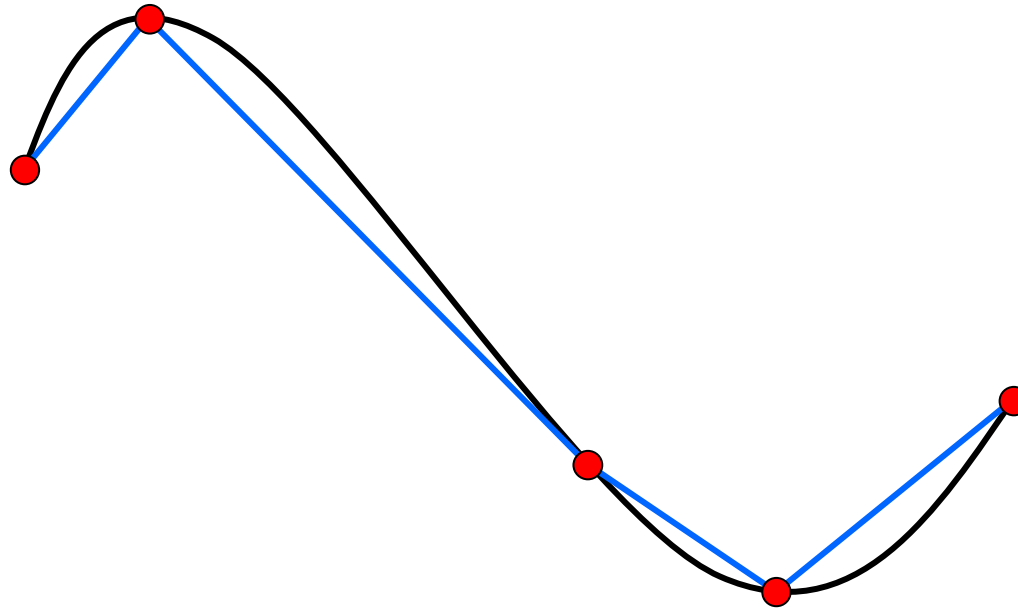
Parametrisierte Kurvengleichung



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



Intervallgrenzen



$$\Delta_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$$



Kurvenabschnitte

Gesucht ist pro Intervall i , $i=1, \dots, n$

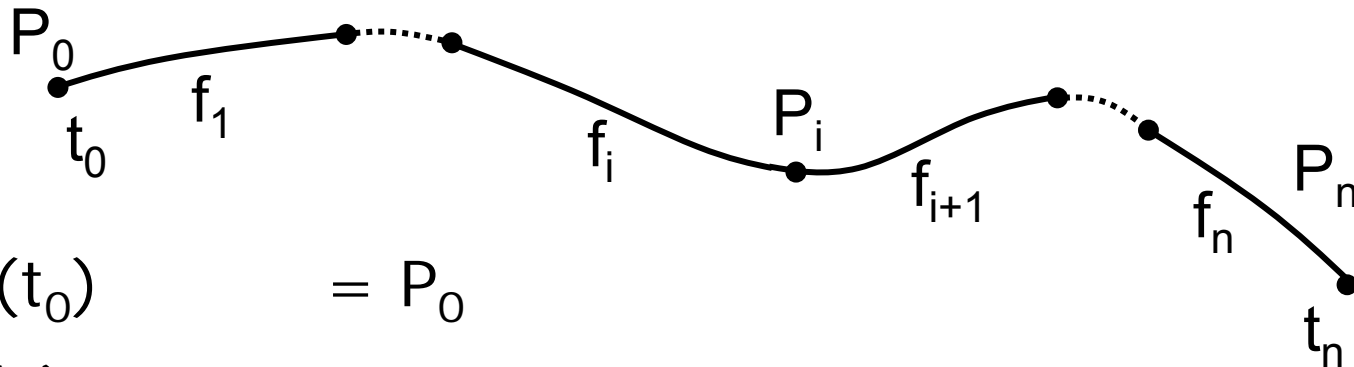
$$f_i(t) = a_i + b_i \cdot t + c_i \cdot t^2 + d_i \cdot t^3$$

genauer:

$$g_i(t) = a^{g_i} + b^{g_i} \cdot t + c^{g_i} \cdot t^2 + d^{g_i} \cdot t^3$$

$$h_i(t) = a^{h_i} + b^{h_i} \cdot t + c^{h_i} \cdot t^2 + d^{h_i} \cdot t^3$$

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

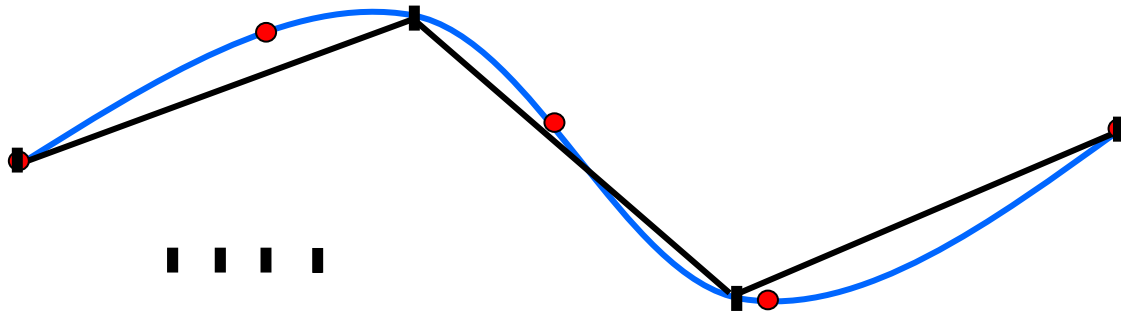
Gegeben n Stützpunkte P_0, P_1, \dots, P_n

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

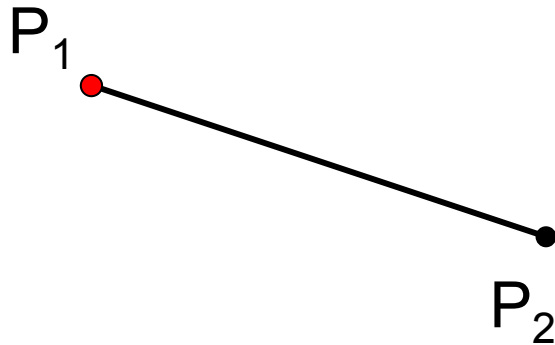
[~cg/2006/skript/Applets/Splines/App.html](#)

Bewertung von Splines

bei vielen Stützpunkten:

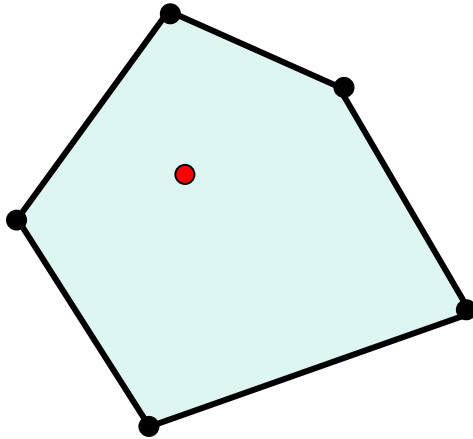
- hoher Rechenaufwand zur Lösung des Gleichungssystems
- Kein lokaler Einfluss möglich

Kontrollpunkte



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

1.0000

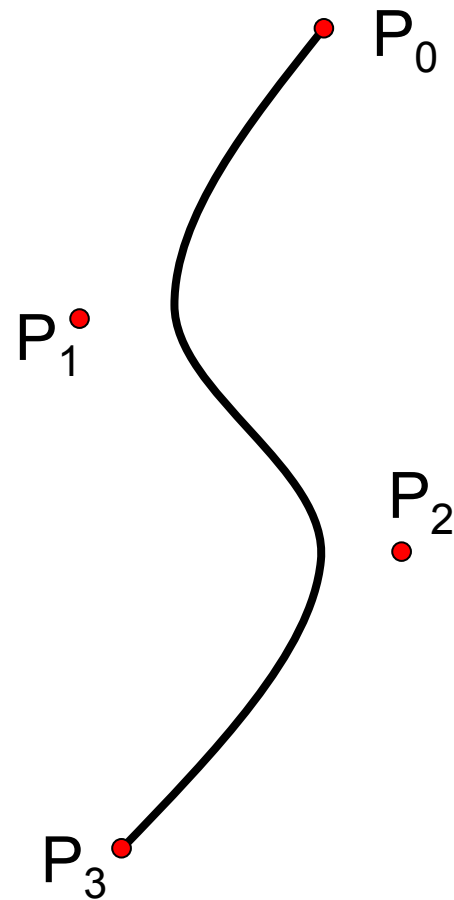


$$P = \sum_{i=0}^n m_i \cdot P_i$$

Bézier-Kurven

Pierre Bézier
1911-1999
Ingenieur bei Renault

Paul deCasteljau
1910 –1999
Ingenieur bei Citroen



Bernstein-Polynome

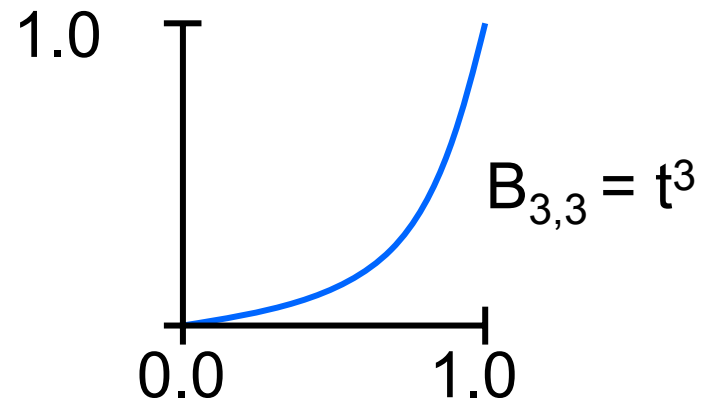
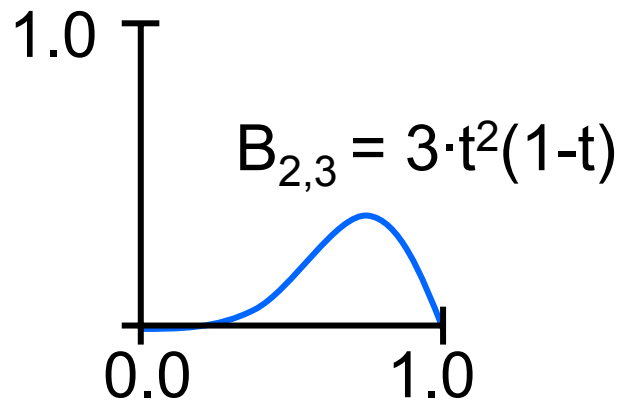
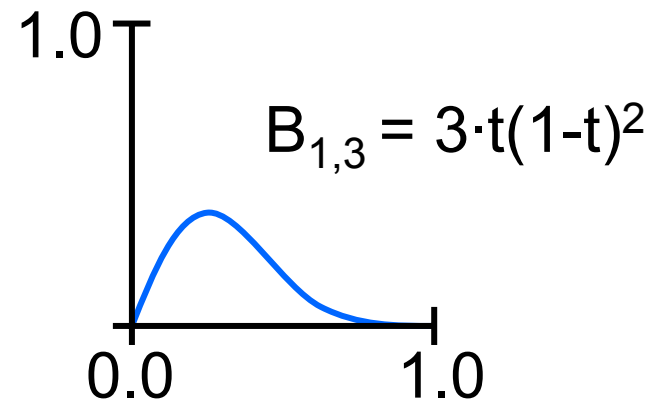
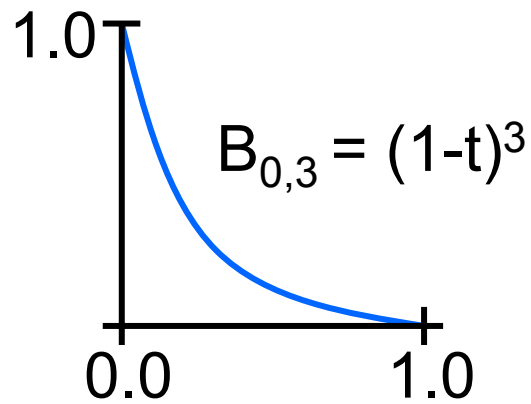
Jeder Kontrollpunkt P_i
wird gewichtet mit Bernstein-Polynom B_i

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

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

$$\binom{n}{i}$$

Bernstein-Polynome für $n=3$



Eigenschaften der Bernstein-Polynome

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

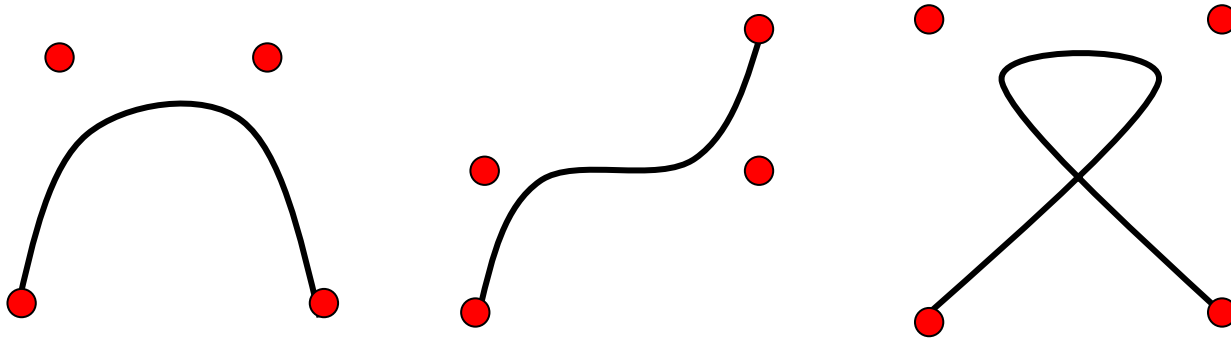
für jedes feste t : $\sum_{i=0}^n B_{i,n}(t) = 1$

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

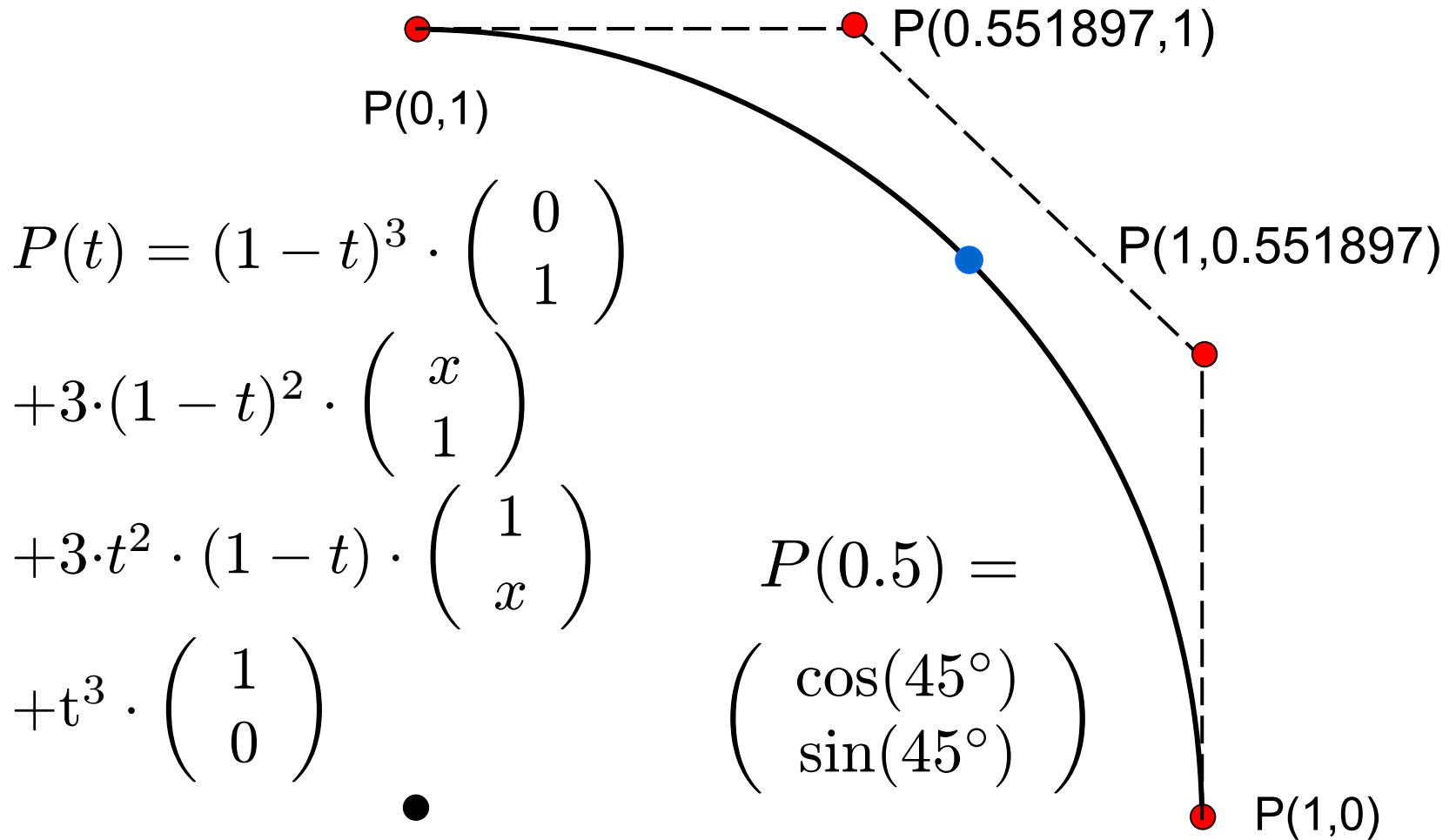
Maxima von $B_{i,n}(t)$ bei $t = \frac{i}{n}, i = 0, \dots, n$

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

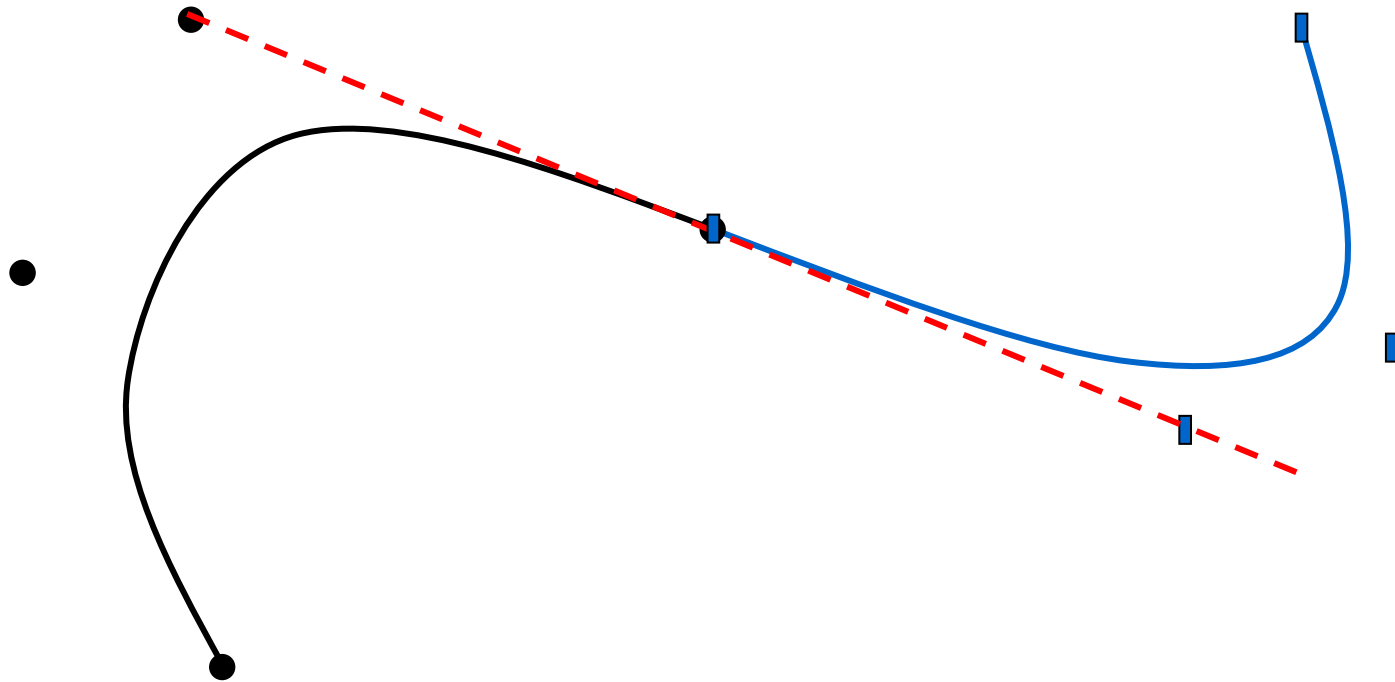
Beispiele für Bézierkurven



Approximation von Viertelkreis



Bézier-Kurven aneinandersetzen



Bézier-Kurve nach de Casteljau

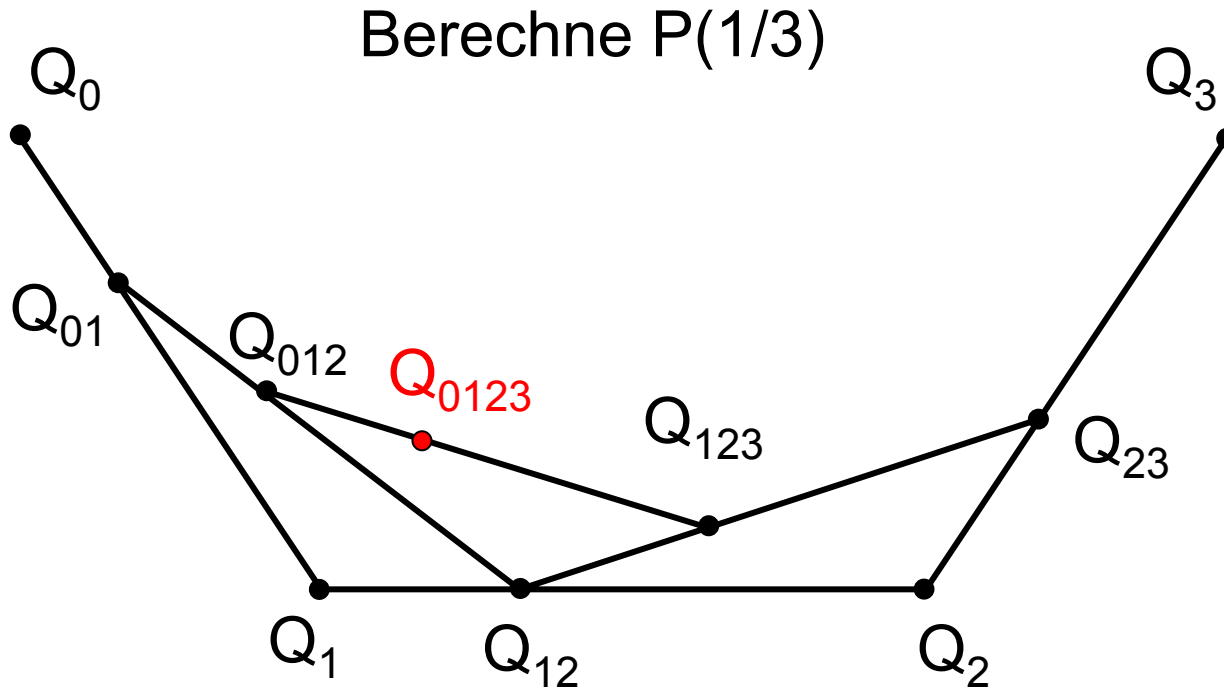
$$P_{0,n}(t) = (1 - t) \cdot P_{0,n-1}(t) + t \cdot P_{1,n}(t)$$

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

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

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

Berechnung nach de Casteljau

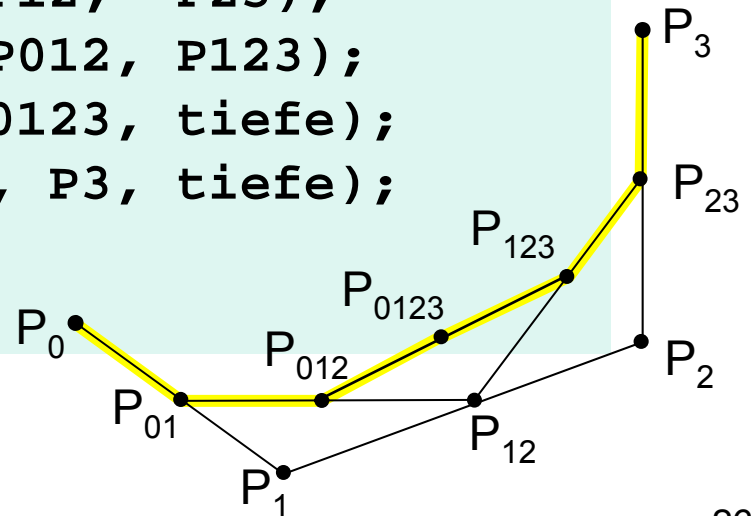


halbiere

```
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/2006/skript/Applets/Splines/App.html](#)

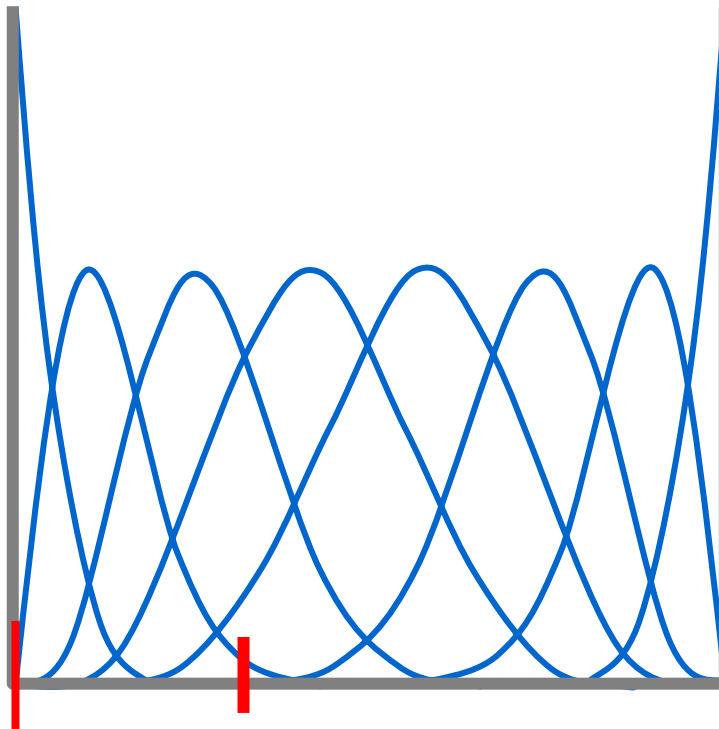
Bewertung von Bézier-Kurven

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

Wunsch:

- fester Polynomgrad
- lokaler Einfluss

Wunsch: Lokaler Einfluss



Einfluss von maximal vier Kontrollpunkten

B-Splines

- $n+1$ Kontrollpunkte P_0, P_1, \dots, P_n
- wähle k
- konstruiere $n+1$ Gewichtsfunktionen
- Gewichtsfunktion wirkt nur auf max. k Abschnitte
- Gewichtsfunktion ist Polynom vom Grad $k-1$ (abschnittsweise)

Konstruktion von B-Splines

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

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

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

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

Beispiel für Knotenvektor

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

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

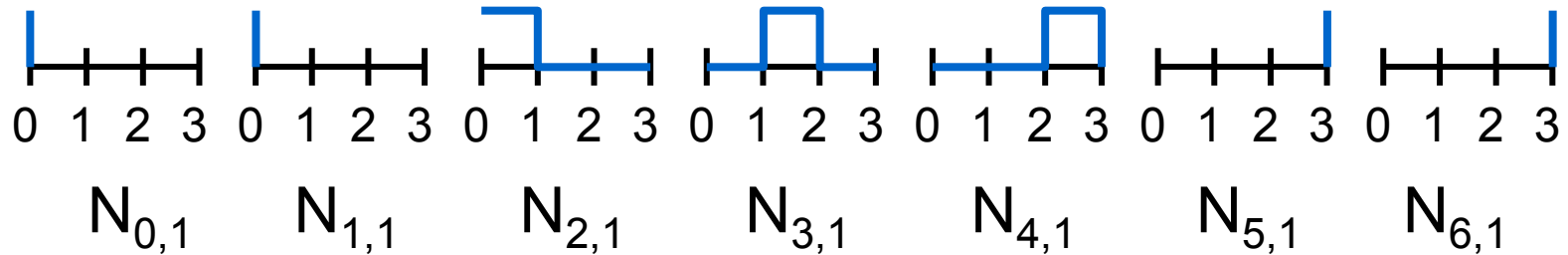
Beispiel: $n = 4, k = 3$

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

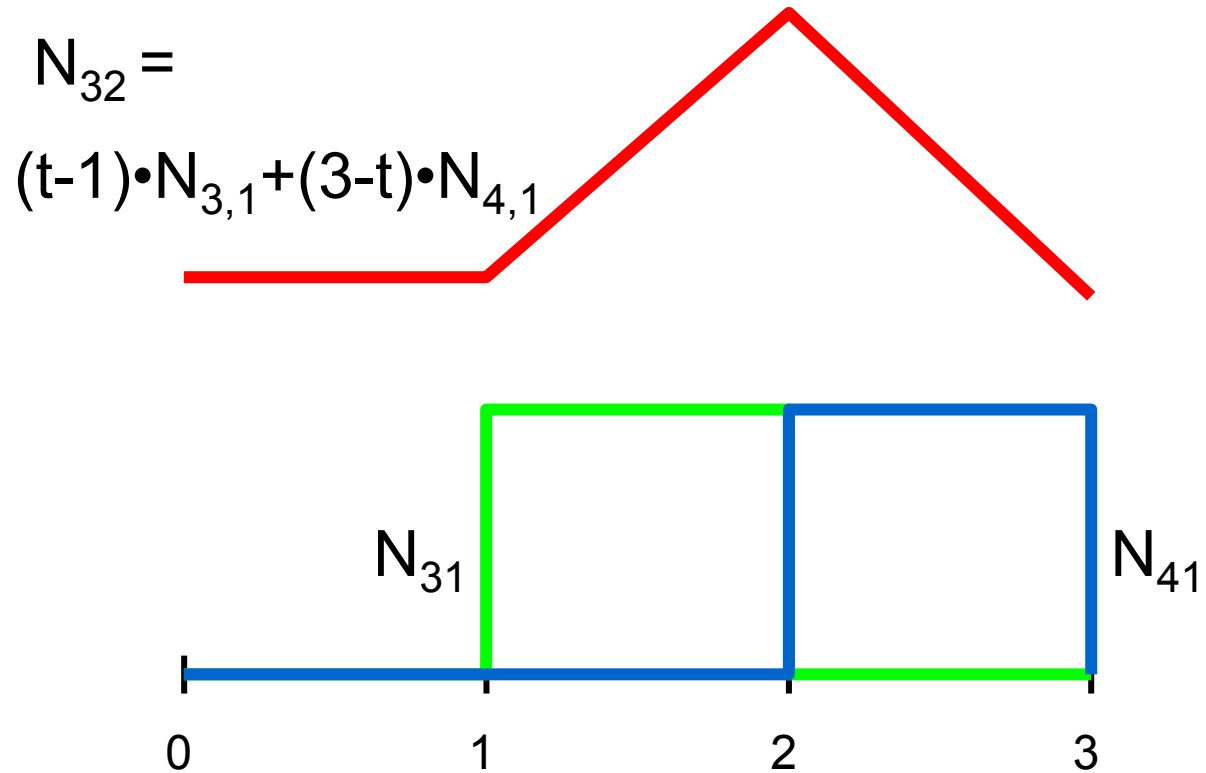
Beispiel: $n = 8, k = 4$

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

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

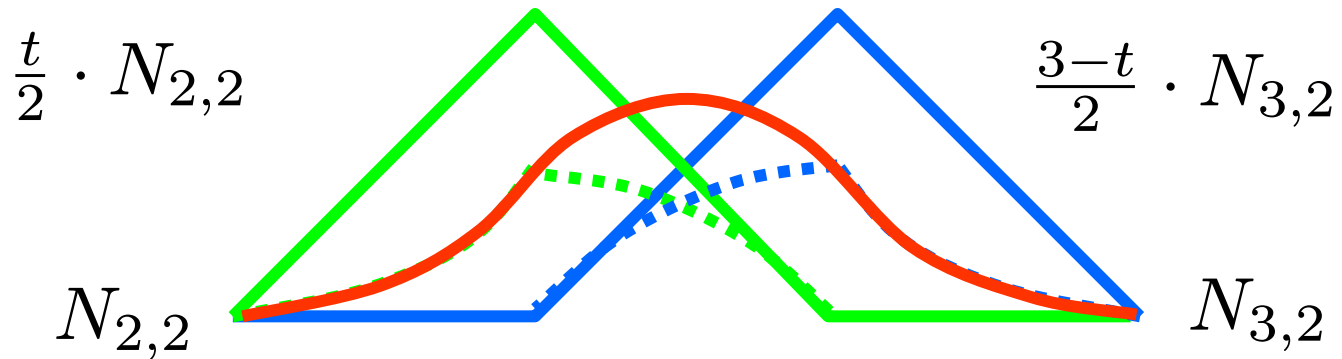


Konstruktion von $N_{3,2}$

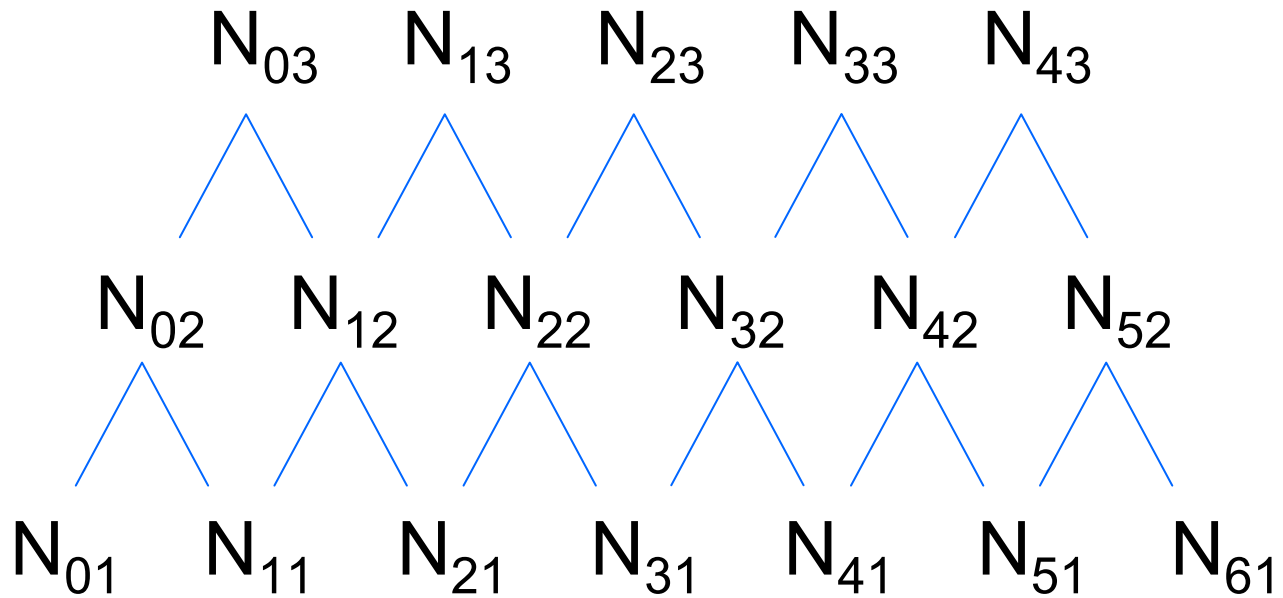


Konstruktion von $N_{2,3}$

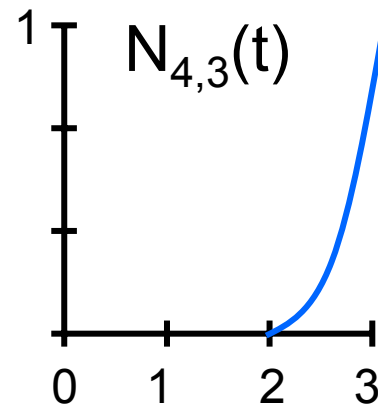
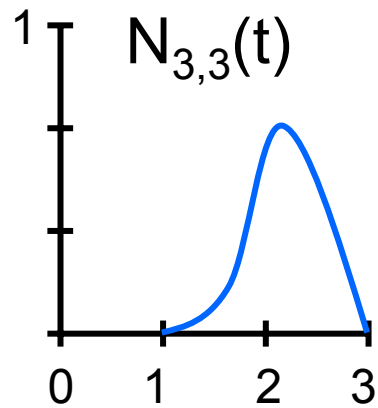
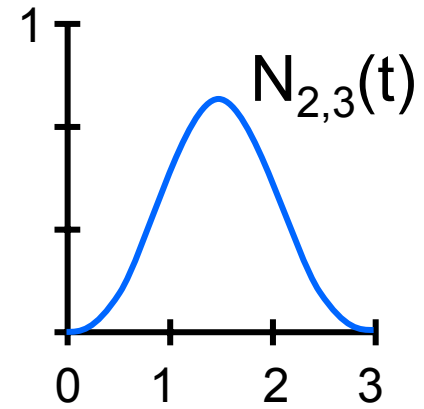
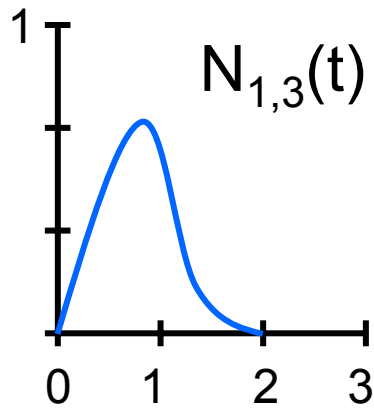
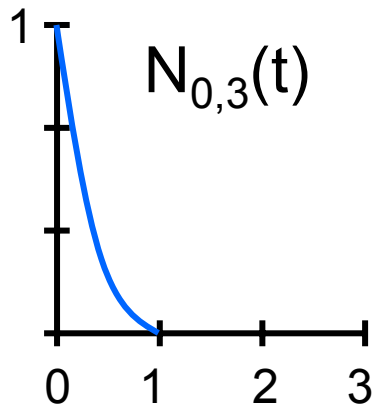
$$N_{2,3} = \frac{t}{2} \cdot N_{2,2} + \frac{3-t}{2} \cdot N_{3,2}$$



Konstruktion von $N_{i,k}$



Gewichtspolynome für $n=4, k=3$



Sonderfall

$$k = n + 1$$

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

$$N_{i,k}(t) = \frac{(k-1)!}{i! \cdot (k-1-i)!} \cdot t^i \cdot (1-t)^{k-1-i}$$
$$i = 0, \dots, k-1$$

⇒ Bernstein-Polynome sind Spezialfall
von B-Splines

Java-Applet von B-Splines

[~cg/2006/skript/Applets/Splines/App.html](#)

Bewertung von B-Splines

- tolle Sache !

aber:

- kein Kreis
- nur invariant bzgl. affiner Abbildungen

Affine Abbildung

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

Erhalten bleiben:

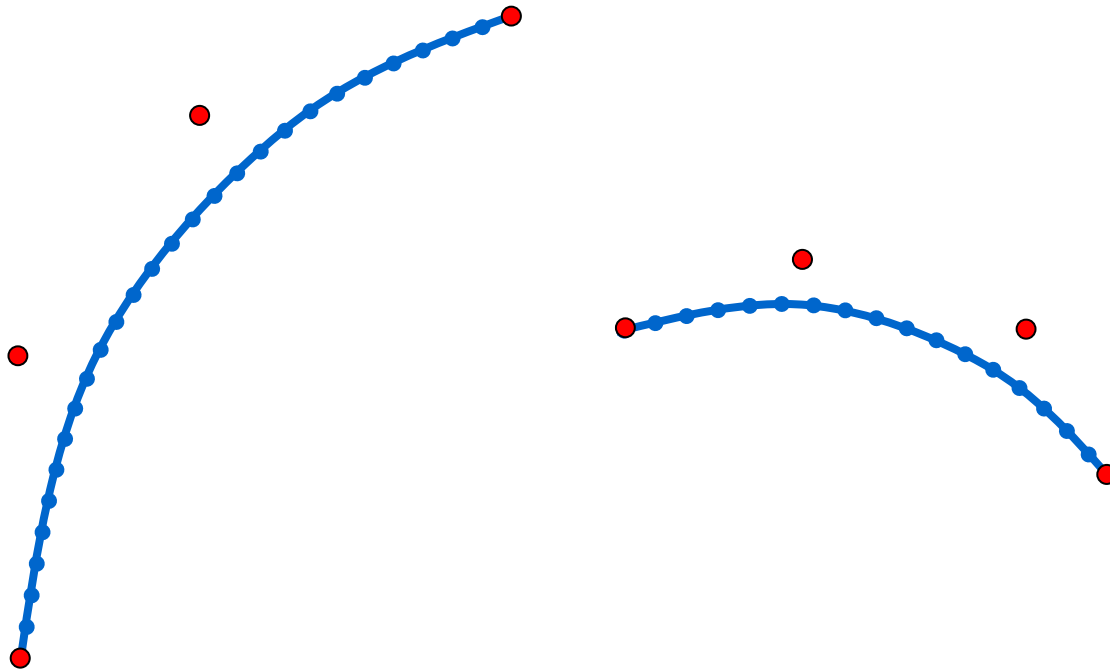
- Gradlinigkeit
- Parallelität von Geraden

nicht erhalten bleiben:

- Winkel
- Flächeninhalte

- Translation
 - Skalierung
 - Rotation
- sind affine Abbildungen

Invarianz bzgl Abbildung



B-Splines sind nicht invariant unter Projektion !

NURBS

Non Uniform

t_j nicht äquidistant

Rational

Gewichtsquotient

B-Splines

Gewichtsfunktion

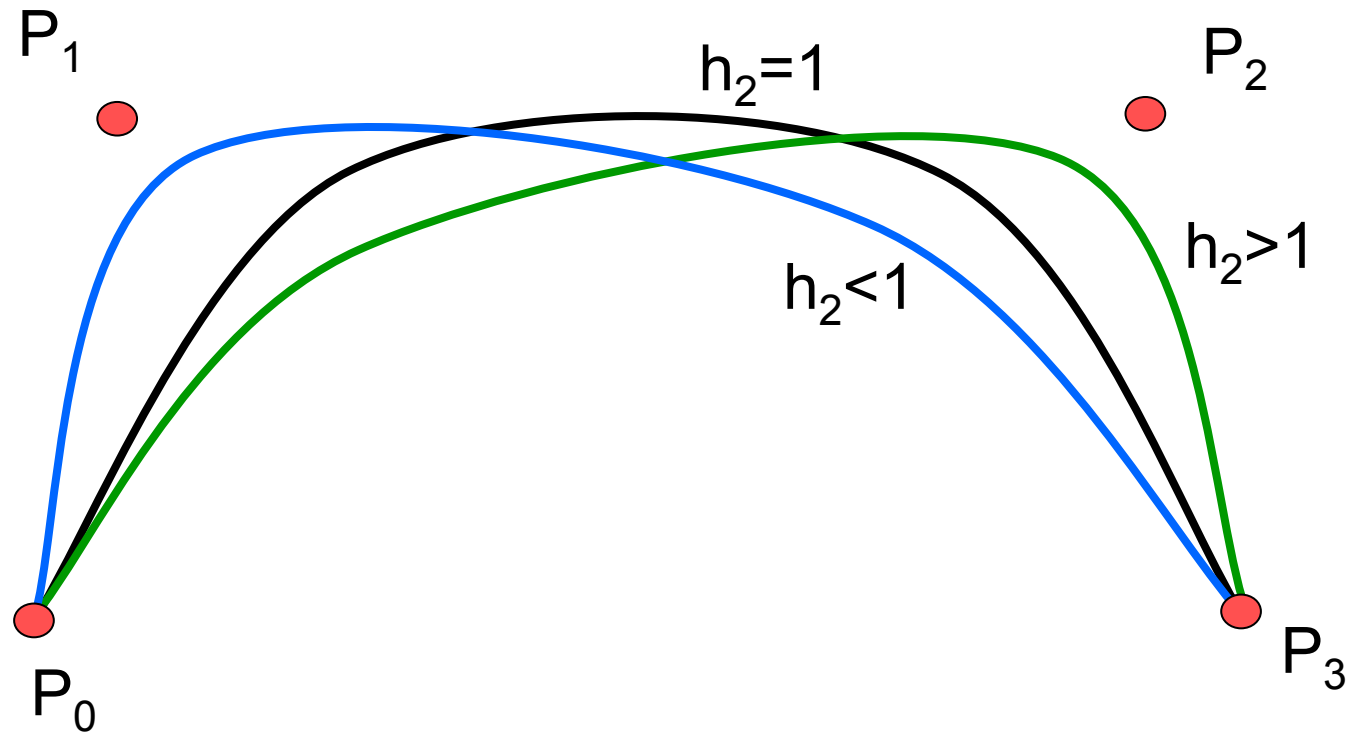
NURBS

Punkte P_i , Gewichte h_i , Knoten t_j

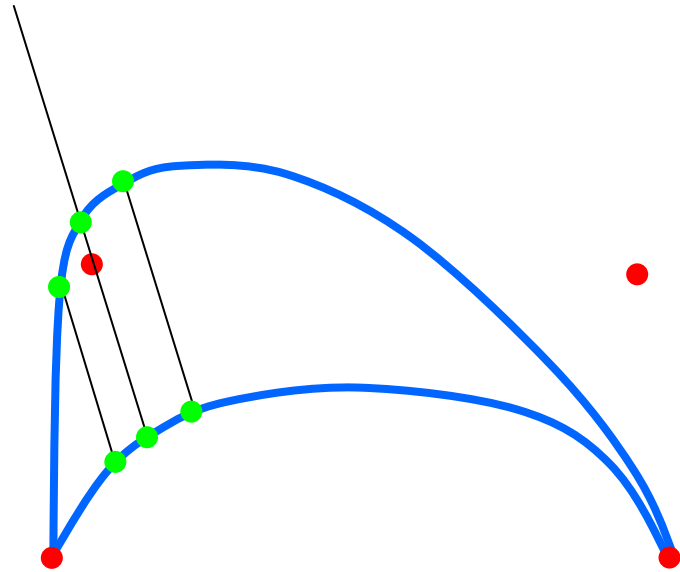
$$P(t) = \sum_{i=0}^n R_{i,k}(t) \cdot P_i$$

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

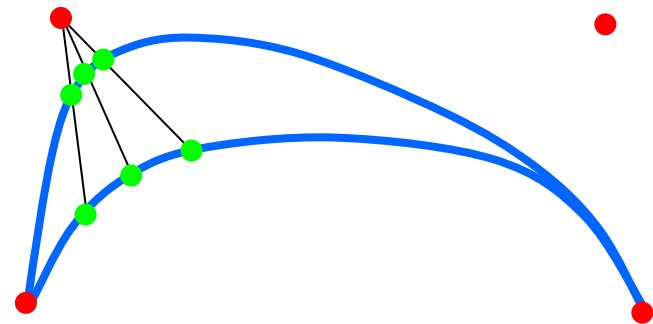
Auswirkung der NURBS-Gewichte



Möglichkeiten der Einflussnahme

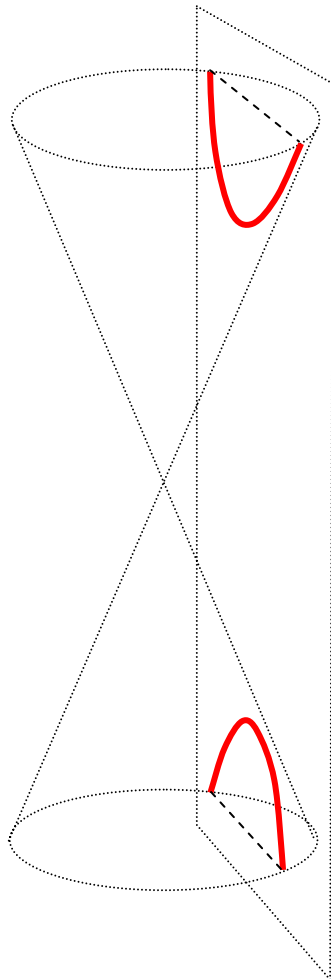


Kontrollpunkt verschieben:
Punkte wandern parallel



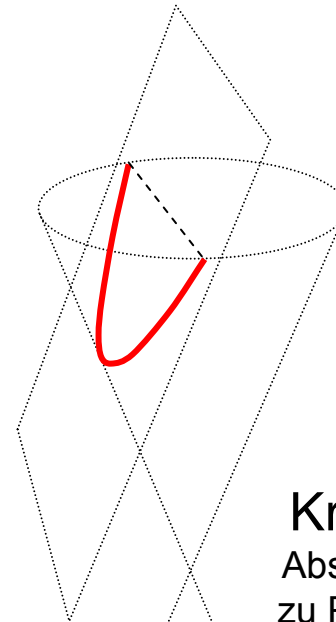
Gewicht erhöhen:
Punkte wandern auf
Kontrollpunkt zu

Kegelschnitte



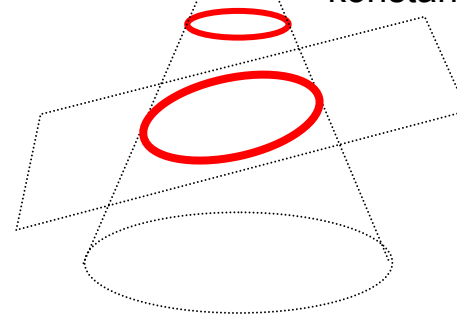
Hyperbel
Differenz der
Abstände
konstant

Parabel
Abstand zu
Punkt und
Gerade
gleich

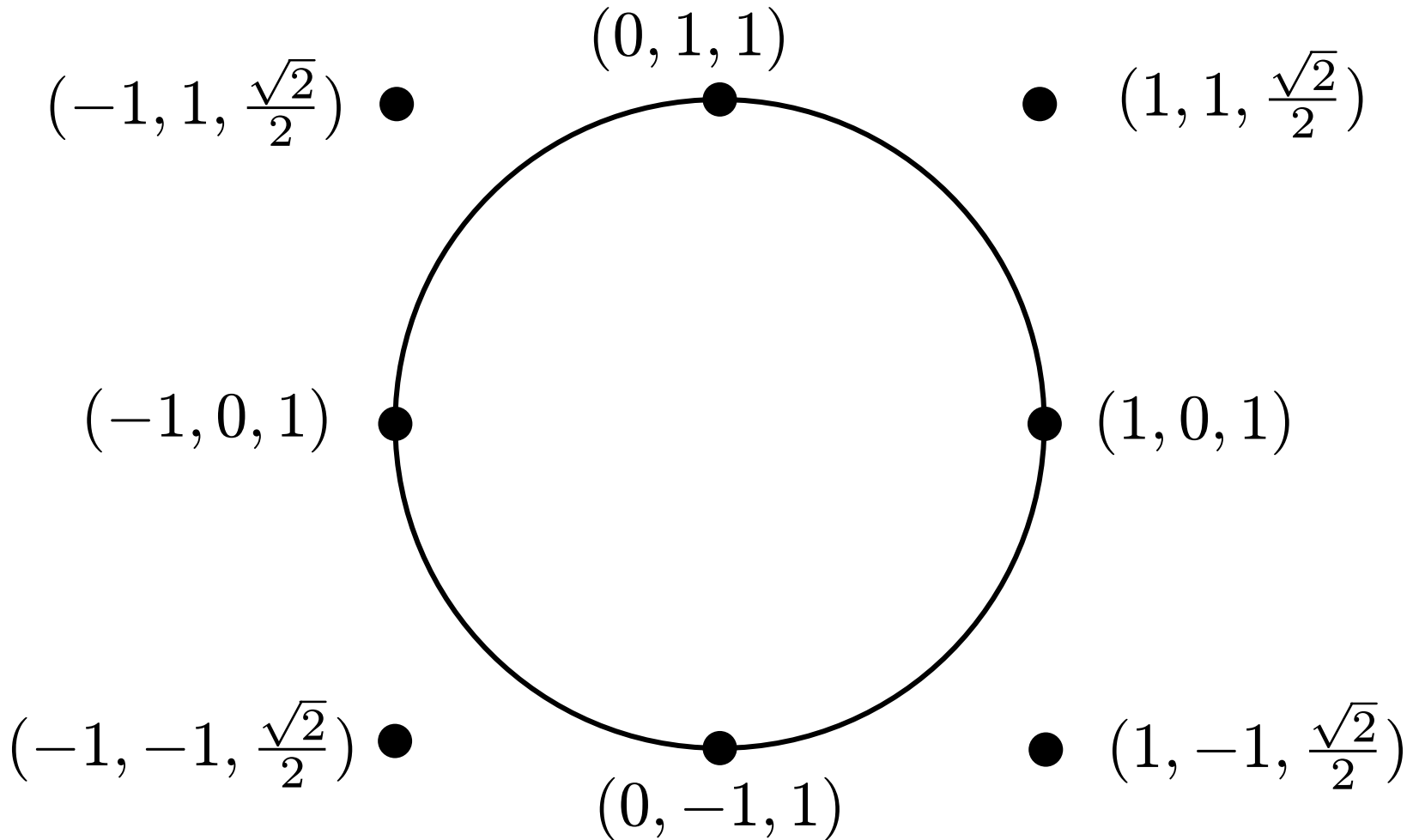


Kreis
Abstand
zu Punkt
konstant

Ellipse
Summe der
Abstände
konstant



NURBS-Kreis



Knotenvektor: $0, 0, 1, 1, 2, 2, 3, 3, 4, 4$, erster + letzter Kontrollpunkt verdoppelt

NURBS + homogene Koordinaten

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

Punktkoordinaten \rightarrow Homogene Koordinaten

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} \rightarrow \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} h_i \cdot x_i \\ h_i \cdot y_i \\ h_i \end{pmatrix}$$
$$P_i \longrightarrow P'_i$$

NURBS + homogene Koordinaten

verwende B-Splines: $P'(t) = \sum_{i=0}^n N_{i,k}(t) \cdot P'_i$

1.+2. Komponente: $\sum_{i=0}^n h_i \cdot N_{i,k}(t) \cdot P_i$

3. Komponente: $\sum_{j=0}^n h_j \cdot N_{j,k}(t)$

Punktkoordinaten \leftarrow Homogene Koordinaten

$$\begin{pmatrix} x/z \\ y/z \end{pmatrix} \leftarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{h_i \cdot N_{i,k}(t) \cdot P_i}{\sum_{j=0}^n h_j \cdot N_{j,k}(t)} = P(t)$$



Wer kurvt am besten ?

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