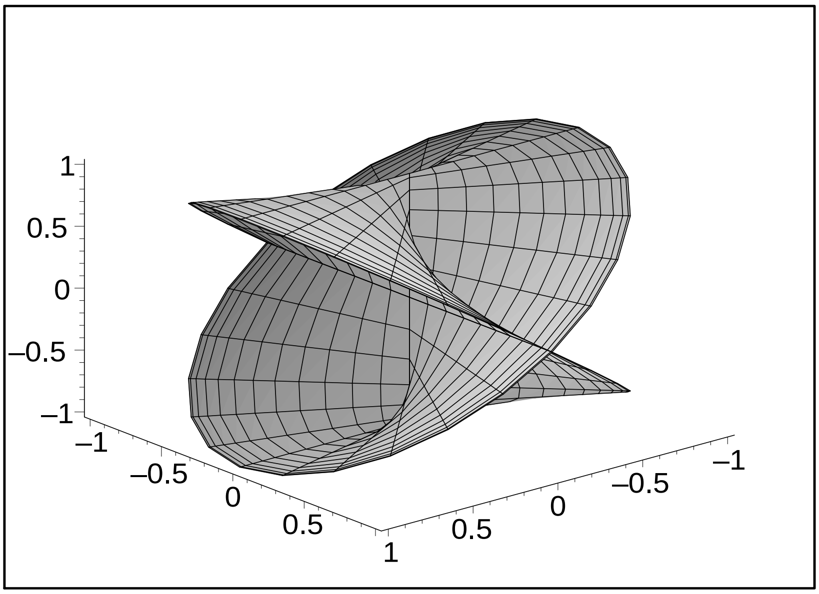
Beispiel 2:

In diesem Beispiel soll gezeigt werden wie sehr sich eine kleine Änderung der Koordinatenfunktionen auf die Form der resultierenden Fläche auswirkt. $x = \cos(u)\sin(v),\ y = \cos(v)\cos(u),\ z = \sin(v)$ beschreibt ein Plückerkonoid, eine Regelfläche weil diese Fläche eine Schar von Geraden trägt.

```
> plot3d([cos(u)*sin(v),cos(v)*cos(u),sin(v)],u=0..Pi,v=0..2*Pi,axes=frame,
> orientation=[50,60]);
```




# 4 Freiformkurven und Freiformflächen

Die Konstruktion und Darstellung von Freiformflächen und Freiformkurven gewinnen im Zeitalter des Computers immer mehr an Bedeutung. Aus dem inzwischen weit entwickelten Gebiet der verschiedenen Kurven- und Flächendarstellungen werden wir exemplarisch die Bezierkurven und Bezierflächen behandeln und kurz auf deren Weiterentwicklungen B-Splines und NURBS eingehen. Grundsätzlich geht es bei der Konstruktion von Freiformflächen oder Freiformkurven immer darum, eine gewisse Anzahl von gegebenen Punkten (Kontrollpolygon bzw. Kontrollpunktenetz) durch eine Kurve bzw. eine Fläche anzunähern *(Approximation)* oder eine Kurve oder Fläche durch eine gewisse Anzahl von gegebenen Punkten (Kurvenpunkte, Flächenpunkte) hindurchzulegen *(Interpolation)*. Für den planenden Architekten ist es nun wichtig eine Kurve oder Fläche interaktiv am Bildschirm verändern zu können. Dies geschieht in beiden Fällen mit Hilfe der Kontrollpunkte, die man sich wie Griffe vorstellen kann an denen der Designer ziehen und schieben kann. Durch Veränderung der Kontrollpunkte wird sich eine Änderung der Kurve ergeben. Ziel dieses Einführungskapitels ist es, die geometrischen und mathematischen Eigenschaften von Freiformobjekten kennenzulerenen, sodass ein Designprozess rational durchgeführt werden kann. Es ist im Rahmen dieser Vorlesung nicht möglich auf Einzelheiten dieses schnell wachsenden Wissens- und Anwendungsgebietes näher einzugehen, aber am Beispiel der relativ einfachen Bezierkurven sollen die geometrischen und mathematischen Algorithmen, die zur Konstruktion von Freiformkurven und Flächen führen dargestellt werden.

## 4.1 Bezierkurven

### 4.1.1 Ein Einführungsbeispiel

Wir geben eine einfache Konstruktion für Parabelpunkte an, deren Verallgemeinerung unmittelbar zu Bezierkurven führt. Gegeben seien 3 Punkte $P_0, P_1, P_2$ im $\mathbb{E}^3$ durch ihre Ortsvektoren $\mathbf{b_0}, \mathbf{b_1}, \mathbf{b_2}$ und ein Parameter $t \in I \subset \mathbb{R}$. Wir konstruieren:

$$\mathbf{b_0^1}(t) = (1-t)\mathbf{b_0} + t\mathbf{b_1} \tag{52}$$

$$\mathbf{b_1^1}(t) = (1-t)\mathbf{b_1} + t\mathbf{b_2} \tag{53}$$

$$\mathbf{b_0^2}(t) = (1-t)\mathbf{b_0^1} + t\mathbf{b_1^1} \tag{54}$$

Setzen wir (52) und (53) in (54) ein, so erhalten wir einen quadratischen Ausdruck in $t$, der eine Parabel bestimmt, wenn $t$ das Intervall $(-\infty, \infty)$ durchläuft.

$$\mathbf{b_0^2}(t) = (1-t)^2\mathbf{b_0} + 2t(1-t)\mathbf{b_1} + t^2\mathbf{b_2}, \tag{55}$$

Eine konstruktive Auswertung von (55) zeigt Abb.14.

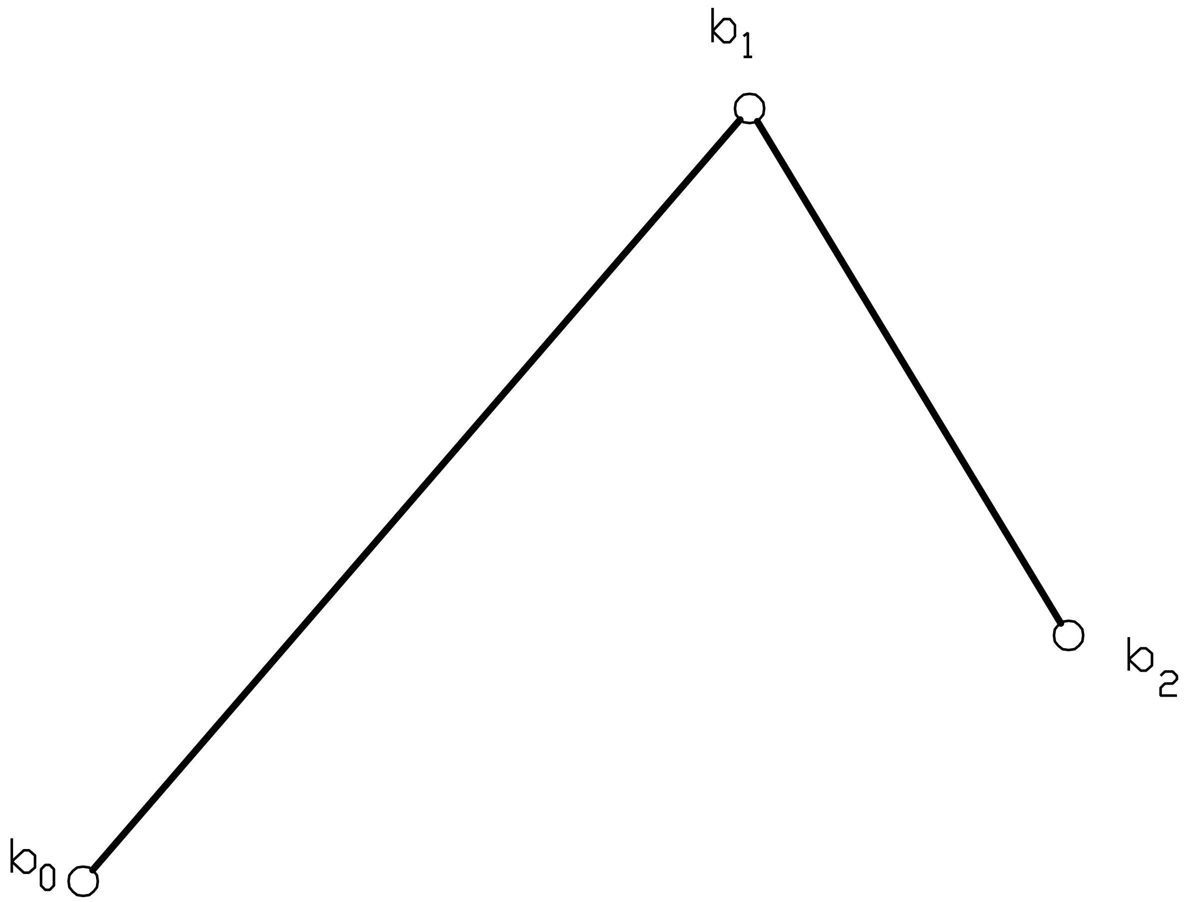



Abbildung 14: Parabelkonstruktion nach (55)

### 4.1.2 Der de Casteljau Algorithmus

Parabeln sind ebene Kurven zweiter Ordnung, da man aber auch Kurven im Raum beliebiger Ordnung $n$ erzeugen möchte, wird man die in (52)-(55) angegebene Konstruktion verallgemeinern: Gegeben seien die Punkte $\mathbf{b_0}, \mathbf{b_1}, \ldots, \mathbf{b_n} \in \mathbb{E}^3$ und ein Parameter $t \in I \subset \mathbb{R}$. Wir setzen:

$$\mathbf{b_i^r}(t) = (1-t)\mathbf{b_i^{r-1}} + t\mathbf{b_{i+1}^{r-1}} \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{cases} \tag{56}$$

und $\mathbf{b_i^0}(t) = \mathbf{b_i}$

Dann beschreibt der Punkt $\mathbf{b_0^n}(t)$ einen Punkt der Bezierkurve zum Parameter $t$ und die Ausgangspunkte $\mathbf{b_i}$ $\quad i = 0, \ldots, n$ heißen ihr *Kontrollpolygon.* Als Anwendung von Formel (56) berechnen wir im Beispiel 4.1 einen Punkt einer Bezierkurve 3. Ordnung ausführlich.

**Beispiel 4.1** ***Bezierkurve 3. Ordnung:*** *Gegeben seien die Punkte* $\mathbf{b_0} = (0,0,0)^t, \mathbf{b_1} = (1,0,0)^t, \mathbf{b_2} = (1,1,0)^t$ *und* $\mathbf{b_3} = (0,1,1)^t$. *Man berechne und zeichne den Bezierkurvenpunkt zum Parameterwert* $t = \frac{1}{2}$.
*Lösung: Aus den gegebenen Punkten des Kontrollpolygons werden schrittweise die Zwischenpunkte ermittelt, aus denen sich dann der Bezierkurvenpunkt* $\mathbf{b_0^3}$ *berechnet:*

*1. Schritt:*

$$\mathbf{b_0^1} = (1-\frac{1}{2})\mathbf{b_0} + \frac{1}{2}\mathbf{b_1} = \frac{1}{2}\begin{bmatrix}0\\0\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\\0\\0\end{bmatrix}$$

$$\mathbf{b_1^1} = (1-\frac{1}{2})\mathbf{b_1} + \frac{1}{2}\mathbf{b_2} = \begin{bmatrix}1\\\frac{1}{2}\\0\end{bmatrix}$$

$$\mathbf{b_2^1} = (1-\frac{1}{2})\mathbf{b_2} + \frac{1}{2}\mathbf{b_3} = \begin{bmatrix}\frac{1}{2}\\1\\\frac{1}{2}\end{bmatrix}$$

*2. Schritt:*

$$\mathbf{b_0^2} = (1-\frac{1}{2})\mathbf{b_0^1} + \frac{1}{2}\mathbf{b_1^1} = \frac{1}{2}\begin{bmatrix}\frac{1}{2}\\0\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1\\\frac{1}{2}\\0\end{bmatrix} = \begin{bmatrix}\frac{3}{4}\\\frac{1}{4}\\0\end{bmatrix}$$

$$\mathbf{b_1^2} = (1-\frac{1}{2})\mathbf{b_1^1} + \frac{1}{2}\mathbf{b_2^1} = \begin{bmatrix}\frac{3}{4}\\\frac{3}{4}\\\frac{1}{4}\end{bmatrix}$$

*3. Schritt:*

$$\mathbf{b_0^3} = (1-\frac{1}{2})\mathbf{b_0^2} + \frac{1}{2}\mathbf{b_1^2} = \frac{1}{2}\begin{bmatrix}\frac{3}{4}\\\frac{1}{4}\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\frac{3}{4}\\\frac{3}{4}\\\frac{1}{4}\end{bmatrix} = \begin{bmatrix}\frac{3}{4}\\\frac{1}{2}\\\frac{1}{8}\end{bmatrix}$$

*Durchläuft der Parameter* $t$ *das Intervall* $[0,1]$, *so beschreibt der Punkt* $\mathbf{b_0^3}$ *die Bezierkurve. Eine konstruktive Auswertung der einzelnen Rechenschritte zeigt Abb.15.*

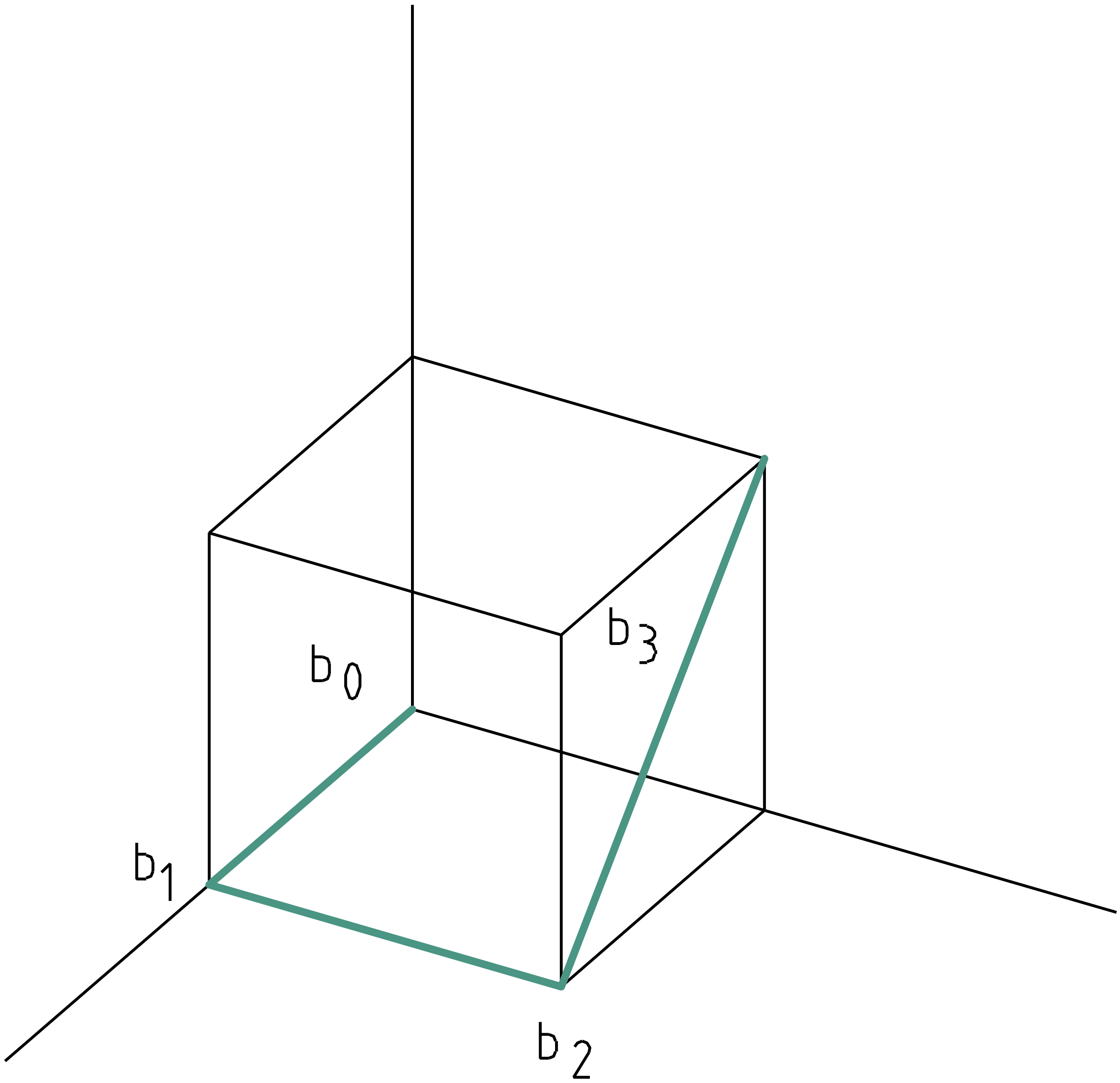



Abbildung 15: Konstruktion einer Bezierkurve

### 4.1.3 Eigenschaften von Bezierkurven:

1. Affine Invarianz: Für die Konstruktion der Bezierpunkte wurden nur Teilverhältnisse verwendet, woraus die affine Invarianz unmittelbar folgt (damit folgt unmittelbar auch die Invarianz bei Parallelprojektionen, denn diese sind ja teilverhältnistreu).

2. Die Bezierkurve liegt in der konvexen Hülle des Kontrollpolygons (wichtige Designeigenschaft!).

3. Anfangspunkt $\mathbf{b_0}$ und Endpunkt $\mathbf{b_n}$ liegen auf der Bezierkurve.

4. Die erste und die letzte Seite des Kontrollpolygons ($\overline{\mathbf{b_0 b_1}}$ und $\overline{\mathbf{b_n b_{n-1}}}$) sind Tangenten im Anfangspunkt und Endpunkt ($b_0$ und $b_n$) der Bezierkurve.

### 4.1.4 Bernstein Darstellung von Bezierkurven

Wir haben bisher einen rekursiven Algorithmus behandelt mit dem einzelne Punkte von Bezierkurven konstruiert werden können. Es ist aber gerade für eine Implementation am Computer notwendig eine explizite Darstellung - wie in Formel (55) des Einführungsbeispieles - zu haben. Dies kann man durch Verallgemeinerung von (55) erreichen. Dazu definieren wir *Bernstein Polynome* durch:

$$B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j} \tag{57}$$

$\binom{n}{j}$ stellt in dieser Formel den Binomialkoeffizenten dar, der sich folgendermaßen berechnet:

$$\binom{n}{j} = \frac{n!}{j!(n-j)!} \qquad \text{und} \qquad n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1, \;\; 0! = 1$$

Bernsteinpolynome haben eine Reihe mathematisch interessanter Eigenschaften (bez. Literatur vgl. z.B. G. FARIN, Curves and Surfaces for CADG, S. 33). In unserem Zusammenhang ist die wichtigste, dass sich Bezierkurven mit Hilfe von Bernsteinpolynomen explizit darstellen lassen:

$$\mathbf{b^n}(t) = \sum_{j=0}^{n} \mathbf{b_j} B_j^n(t) \tag{58}$$

**Darstellung einer Bezierkurve**

```
> with(plots):with(linalg):
Warning, the name changecoords has been redefined
> p:=pointplot3d([[0,0,0],[1,0,0],[1,1,0],[0,1,1]],style=line,colo
  r=red,thickness=3,axes=normal,orientation=[30,60]):
> k:=spacecurve([3*t-3*t^2,3*t^2-2*t^3,t^3],t=0..1,color=black,thi
  ckness=3,axes=normal):
> display(p,k);
```

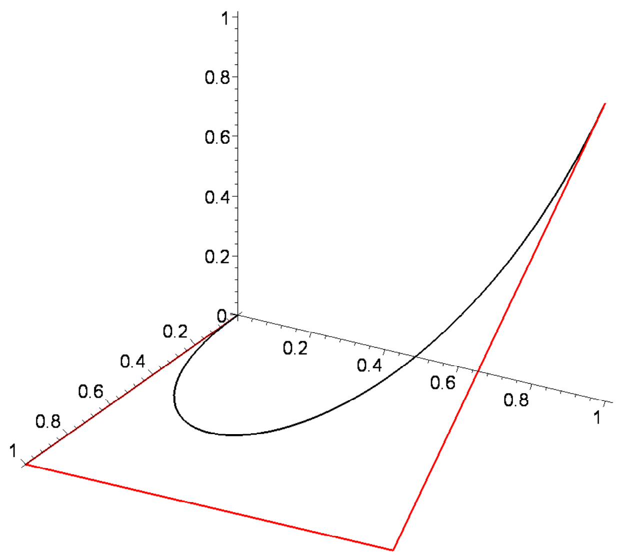



Abbildung 16: Darstellung der Bezierkurve von Beispiel 4.2

**Beispiel 4.2** ***(Bernsteinpolynome 3. Ordnung)*** *Man berechne eine explizite Darstellung der Bezierkurve 3. Ordnung von Beispiel 4.1.*
*Lösung: Wir bestimmen vorerst die Bernsteinpolynome 3. Ordnung $B_j^3(t)$, die wir dann in (58) einsetzen müssen:*

$$\begin{aligned}
B_0^3(t) &= \binom{3}{0} t^0(1-t)^3 = \frac{3!}{0!(3-0)!}(1-t)^3 = (1-t)^3 \\
B_1^3(t) &= \binom{3}{1} t^1(1-t)^2 = 3t(1-t)^2 \\
B_2^3(t) &= \binom{3}{2} t^2(1-t) = 3t^2(1-t) \\
B_3^3(t) &= \binom{3}{3} t^3(1-t)^0 = t^3
\end{aligned}$$

*Einsetzen in (58) liefert die Bezierkurve:*

$$\mathbf{b^3}(t) = \mathbf{b_0} \cdot (1-t)^3 + \mathbf{b_1} \cdot 3t(1-t)^2 + \mathbf{b_2} \cdot 3t^2(1-t) + \mathbf{b_3} \cdot t^3, \tag{59}$$

*bzw. für die konkreten Kontrollpunkte von Beispiel 4.1:*

$$\mathbf{b^3}(t) = \begin{pmatrix}0\\0\\0\end{pmatrix} \cdot (1-t)^3 + \begin{pmatrix}1\\0\\0\end{pmatrix} \cdot 3t(1-t)^2 + \begin{pmatrix}1\\1\\0\end{pmatrix} \cdot 3t^2(1-t) + \begin{pmatrix}0\\1\\1\end{pmatrix} \cdot t^3. \tag{60}$$

*Rechnen wir die einzelnen Koordinatenfunktionen aus, so erhalten wir*

$$\begin{aligned}
x &= 3t(1-t)^2 + 3t^2(1-t) = 3t - 3t^2 \\
y &= 3t^2(1-t) + t^3 = 3t^2 - 2t^3 \\
z &= t^3
\end{aligned} \tag{61}$$

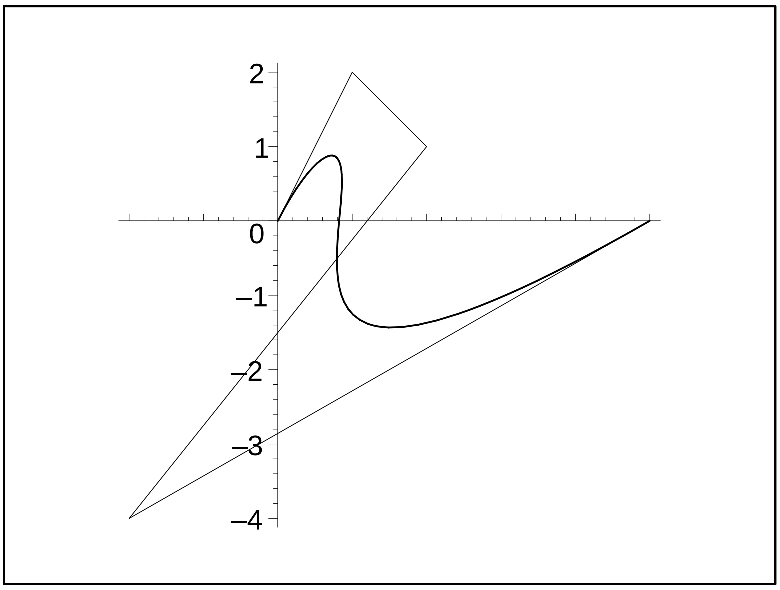



Abbildung 17: Bezierkurve 4.Ordnung

Abb. 17 zeigt eine ebene Bezierkurve zu den Kontrollpunkten $\mathbf{b_0}, \mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}, \mathbf{b_4}$. Die Kurve ist von vierter Ordnung. Hier sieht man auch bereits einen Nachteil von Bezierkurven: mit wachsender Anzahl der Kontrollpunkte steigt die Ordnung der Bezierkurve. Diesen Nachteil kann man wettmachen, indem man Kurven konstruiert, die stückweise aus Bezierkurven bestimmter Ordnung zusammengesetzt sind. Man kommt dadurch zu sogenannten *B-Splines* deren Verallgemeinerung die *NURBS (Non Uniform Rational B-Splines)* sind.

## 4.2 Bezierflächen

### 4.2.1 Bilineare Interpolation

Die lineare Interpolation in 4.1 hat die *einfachste* Kurve zwischen zwei Punkten geliefert. Bilineare Interpolation liefert die *einfachste* Fläche zwischen vier Punkten. Gegeben seien vier verschiedene Punkte $\mathbf{b_{00}}$, $\mathbf{b_{01}}$, $\mathbf{b_{10}}$, $\mathbf{b_{11}}$ in $\mathbb{E}^3$. Die Menge der Punkte, für die gilt:

$$\mathbf{x}(u,v) = \sum_{j=0}^{1} \sum_{i=0}^{1} \mathbf{b_{ij}} B_i^1(u) B_j^1(v) \tag{62}$$

ist ein *hyperbolisches Paraboloid* durch die vier Punkte $b_{i,j}$. Man kann die Gleichung 62 in Matrizenform schreiben

$$\mathbf{x}(u,v) = (1-u, u) \begin{pmatrix} \mathbf{b_{00}} & \mathbf{b_{01}} \\ \mathbf{b_{10}} & \mathbf{b_{11}} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix} \tag{63}$$

Die sich ergebende Fläche (63) wird die bilinear Interpolierende der vier Grundpunkte $\mathbf{b_{i,j}}$ genannt (Abb. 18). Genau wie die Bezierkurven durch Verallgemeinerung der linearen Interpo-

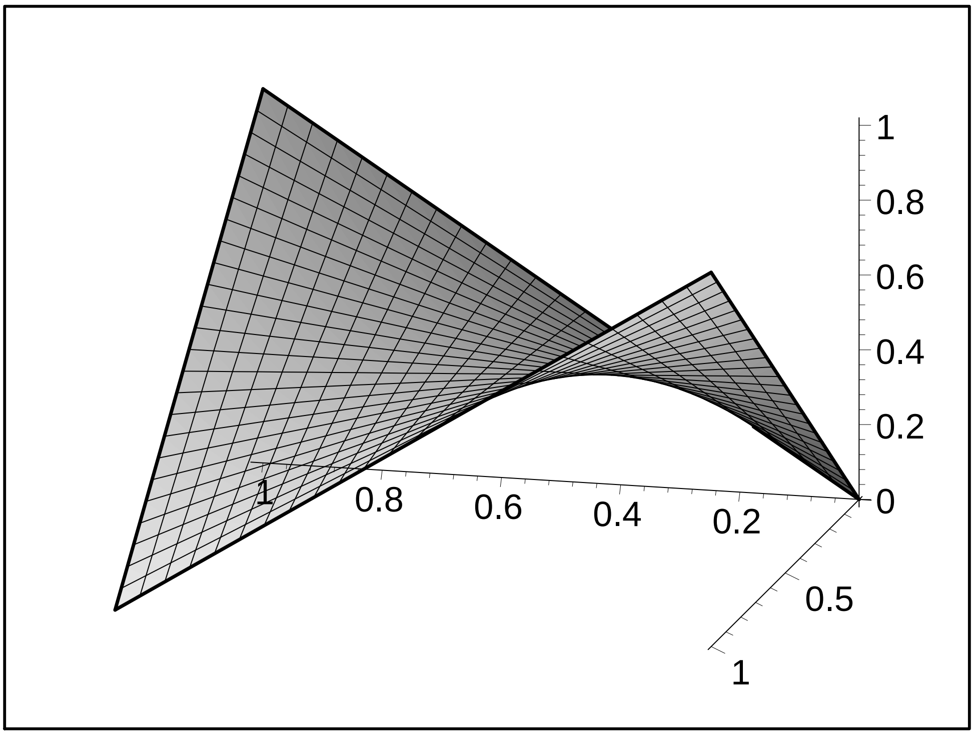



Abbildung 18: Hyperbolisches Paraboloid

lation zwischen zwei Punkten entstehen, so kann man Bezierflächen als die Verallgemeinerung der bilinaren Interpolation auffassen. Gegeben ist nun ein dreidimensionales Kontrollpunktenetz von $n \times m$ Punkten. Durch dieses Kontrollpunktenetz ist die Bezierfläche bestimmt.

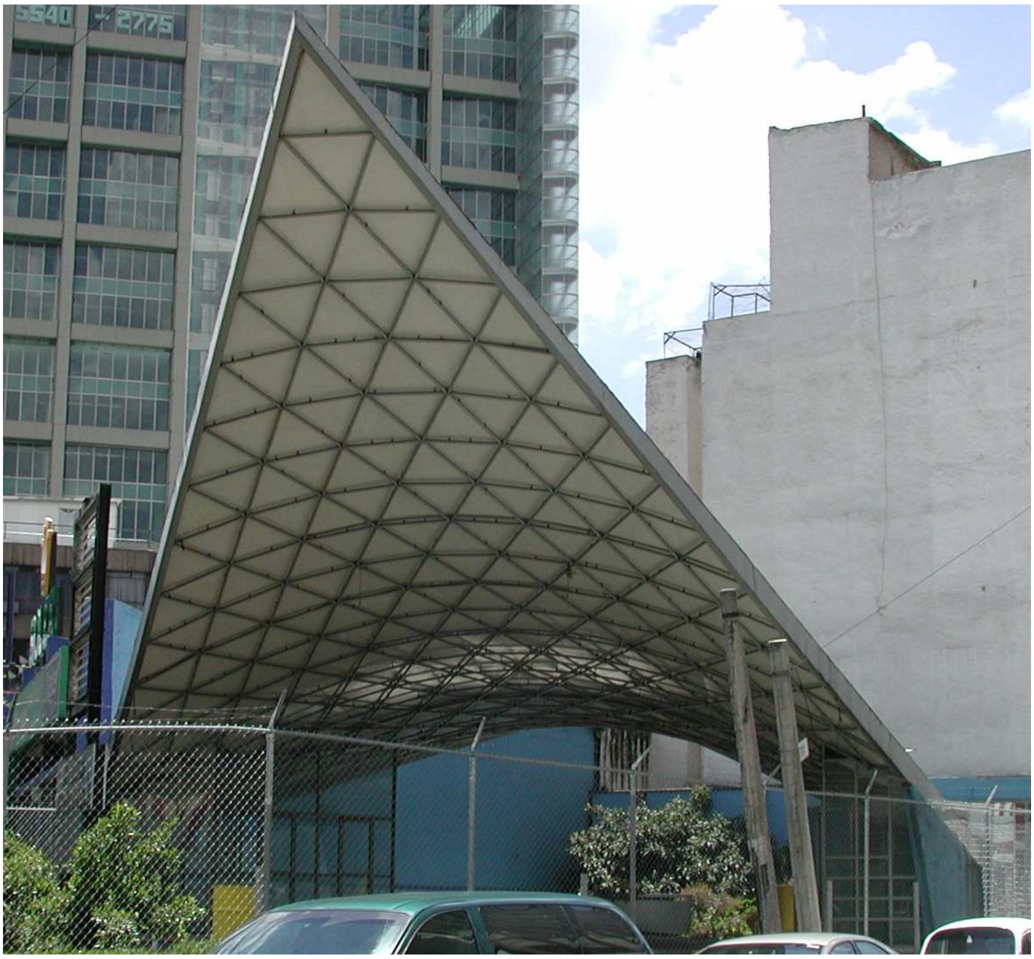


Abbildung 19: Hyperbolisches Paraboloid

### 4.2.2 Darstellung der Bezierflächen als Tensorprodukt

Unter der Vorstellung, dass eine beliebige Fläche $\Phi$ dadurch entsteht, dass eine Ausgangskurve über sie geführt wird und dabei die Gestalt ändert, können Bezierflächen in folgender Weise konstruiert werden: Sei die Ausgangskurve eine Bezierkurve vom Grad $m$

$$\mathbf{b^m}(u) = \sum_{i=0}^{m} \mathbf{b_i} B_i^m(u) \tag{64}$$

und jeder vorkommende Kontrollpunkt $\mathbf{b_i}$ soll wieder eine Bezierkurve bestimmen.

$$\mathbf{b_i} = \mathbf{b_i}(v) = \sum_{j=0}^{n} \mathbf{b_{ij}} B_j^n(v) \tag{65}$$

Setzen wir nun (65) in (64) ein, so ergibt sich die Darstellung der entstehenden Bezierfläche mit

$$\mathbf{b}^{m,n}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{ij} B_j^n(v) B_i^m(u) \tag{66}$$

**Beispiel 4.3** *Es soll die Darstellung (66) einer Bezierfläche mit Parameterkurven 3. Ordnung* *angegeben werden. Gegeben sei dazu das Netz der Kontrollpunkte:*

$$\begin{array}{cccc} \mathbf{b_{00}} & \mathbf{b_{01}} & \mathbf{b_{02}} & \mathbf{b_{03}} \\ \mathbf{b_{10}} & \mathbf{b_{11}} & \mathbf{b_{12}} & \mathbf{b_{13}} \\ \mathbf{b_{20}} & \mathbf{b_{21}} & \mathbf{b_{22}} & \mathbf{b_{23}} \\ \mathbf{b_{30}} & \mathbf{b_{31}} & \mathbf{b_{32}} & \mathbf{b_{33}} \end{array}$$

*Lösung: Wir erhalten durch Einsetzen in (66) und Ausrechnen ausführlich:*

$$\begin{aligned} \mathbf{b^{3,3}}(u,v) = & \mathbf{b_{00}}(1-v)^3(1-u)^3 + \mathbf{b_{01}}3v(1-v)^2(1-u)^3 + \mathbf{b_{02}}3v^2(1-v)(1-u)^3 + \\ & \mathbf{b_{03}}v^3(1-u)^3 + \mathbf{b_{10}}(1-v)^3[3u(1-u)^2] + \mathbf{b_{11}}[3v(1-v)^2][3u(1-u)^2] + \\ & \mathbf{b_{12}}3v^2(1-v)[3u(1-u)^2] + \mathbf{b_{13}}v^2[3u(1-u)^2] + \mathbf{b_{20}}(1-v)^3[3u^2(1-u)] + \\ & \mathbf{b_{21}}[3v(1-v)^2][3u^2(1-u)] + \mathbf{b_{22}}3v^2(1-v)[3u^2(1-u)] + \mathbf{b_{23}}v^3[3u^2(1-u)] + \\ & \mathbf{b_{30}}(1-v)^3u^3 + \mathbf{b_{31}}[3v(1-v)^2]u^3 + \mathbf{b_{32}}3v^2(1-v)u^3 + \mathbf{b_{33}}v^3u^3 \end{aligned}$$

*Wir konstruieren nun eine Bezierfläche mit 16 konkreten Kontrollpunkten:*

$$\begin{array}{cccc} \mathbf{b_{00}} = (0,0,0) & \mathbf{b_{01}} = (1,0,1) & \mathbf{b_{02}} = (2,0,0) & \mathbf{b_{03}} = (3,0,1) \\ \mathbf{b_{10}} = (0,1,1) & \mathbf{b_{11}} = (1,1,0) & \mathbf{b_{12}} = (2,1,-1) & \mathbf{b_{13}} = (3,1,0) \\ \mathbf{b_{20}} = (0,2,1) & \mathbf{b_{21}} = (1,2,0) & \mathbf{b_{22}} = (2,2,0) & \mathbf{b_{23}} = (3,2,1) \\ \mathbf{b_{30}} = (0,3,0) & \mathbf{b_{31}} = (1,3,1) & \mathbf{b_{32}} = (2,3,1) & \mathbf{b_{33}} = (3,3,0) \end{array}$$

*Die explizte Berechnung der Flächengleichung und der dazugehörige Plot wird im folgenden Maple Notebook gezeigt.*

```
> with(plots):
```

Eingabe der Tensorproduktdarstellung der Bezierfläche (Skriptum : Formel (66))

```
> P:=sum(sum(b[i,j]*binomial(m,i)*u^i*(1-u)^(m-i)*binomial(n,j)
> *v^j*(1-v)^(n-j),j=0..n),i=0..m);
```

$$P := \sum_{i=0}^{m} \left( \sum_{j=0}^{n} b_{i,j} \binom{m}{i} u^i (1-u)^{(m-i)} \binom{n}{j} v^j (1-v)^{(n-j)} \right)$$

## Beispiel einer Bezierfläche mit 16 Knotenpunkten:

n=m=3 -> (i=0..3, j=0..3)!

```
> P3:=sum(sum(b[i,j]*binomial(3,i)*u^i*(1-u)^(3-i)*binomial(3,j)*
> v^j*(1-v)^(3-j),i=0..3),j=0..3);
```

$$\begin{aligned}P3 := {} & b_{0,0} (1-u)^3 (1-v)^3 + 3 b_{1,0} u (1-u)^2 (1-v)^3 + 3 b_{2,0} u^2 (1-u) (1-v)^3 \\ & + b_{3,0} u^3 (1-v)^3 + 3 b_{0,1} (1-u)^3 v (1-v)^2 + 9 b_{1,1} u (1-u)^2 v (1-v)^2 \\ & + 9 b_{2,1} u^2 (1-u) v (1-v)^2 + 3 b_{3,1} u^3 v (1-v)^2 + 3 b_{0,2} (1-u)^3 v^2 (1-v) \\ & + 9 b_{1,2} u (1-u)^2 v^2 (1-v) + 9 b_{2,2} u^2 (1-u) v^2 (1-v) + 3 b_{3,2} u^3 v^2 (1-v) \\ & + b_{0,3} (1-u)^3 v^3 + 3 b_{1,3} u (1-u)^2 v^3 + 3 b_{2,3} u^2 (1-u) v^3 + b_{3,3} u^3 v^3\end{aligned}$$

Einsetzen der gegebenen Kontrollpunkte (Skriptum Beispiel 4.3) (in Komponentenform)

```
> P_x:=simplify((subs(b[0,0]=0,b[1,0]=0,b[2,0]=0,b[3,0]=0, b[0,1]=1,
> b[1,1]=1, b[2,1]=1, b[3,1]=1,b[0,2]=2, b[1,2]=2, b[2,2]=2, b[3,2]=2,
> b[0,3]=3, b[1,3]=3, b[2,3]=3, b[3,3]=3, P3)));
```

$$P\_x := 3\,v$$

```
> P_y:=simplify((subs(b[0,0]=0,b[1,0]=1,b[2,0]=2,b[3,0]=3,b[0,1]=0,
> b[1,1]=1,b[2,1]=2,b[3,1]=3,b[0,2]=0,b[1,2]=1,b[2,2]=2,b[3,2]=3,
> b[0,3]=0,b[1,3]=1,b[2,3]=2,b[3,3]=3,P3)));
```

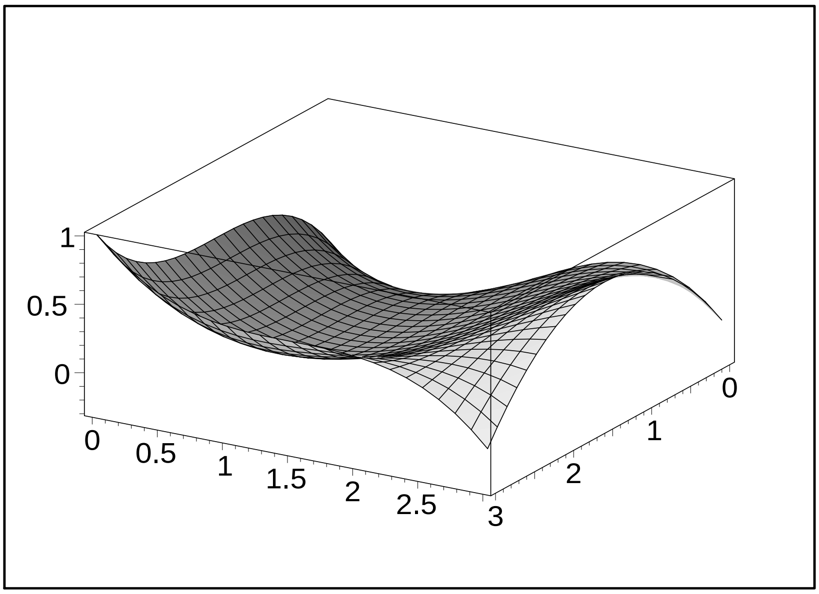
$$P\_y := 3\,u$$

```
> P_z:=simplify((subs(b[0,0]=0,b[1,0]=-1,b[2,0]=1,b[3,0]=0,b[0,1]=1,
> b[1,1]=0,b[2,1]=0,b[3,1]=1,b[0,2]=0,b[1,2]=-1,b[2,2]=0,b[3,2]=1,
> b[0,3]=1,b[1,3]=0,b[2,3]=1,b[3,3]=0,P3)));
```

$$\begin{aligned}P\_z := {} & -6 u^3 + 9 u^2 - 6 v^2 + 4 v^3 + 3 v - 12 u^2 v^3 - 18 u^2 v + 27 u^2 v^2 + 18 u^3 v - 24 u^3 v^2 \\ & + 8 u^3 v^3 - 3 u\end{aligned}$$

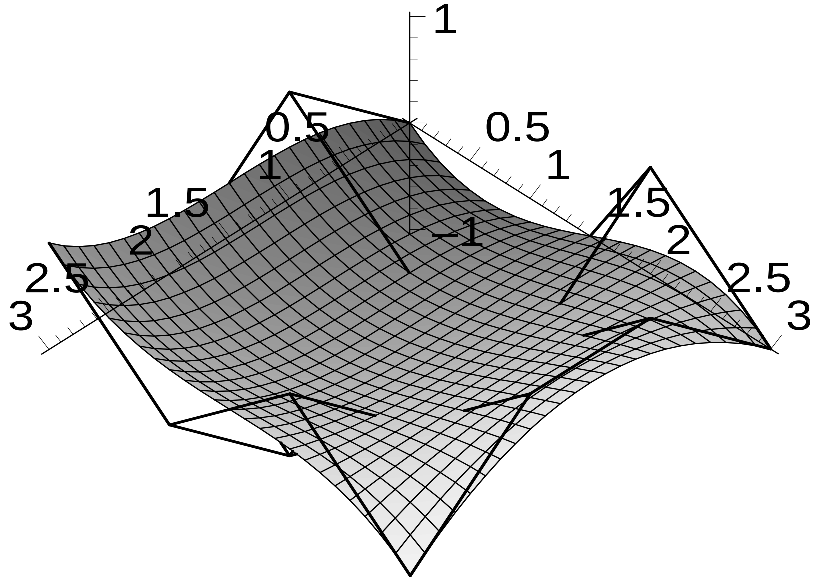
Plot der Bezierfläche:

```
> plot3d([P_x,P_y,P_z],u=0..1,v=0..1,axes=boxed,scaling=constrained,
> orientation=[31,70]);
```




Um die Interaktion mit den Kontrollpunkten zu zeigen plotten wir auch noch das Kontrollpunktenetz:

```
> u0:=pointplot3d([[0,0,0],[0,1,-1],[0,2,1],[0,3,0]],thickness=3,
> style=line,color=black):
> u1:=pointplot3d([[1,0,1],[1,1,0],[1,2,0],[1,3,1]],thickness=3,
> style=line,color=black):
> u2:=pointplot3d([[2,0,0],[2,1,-1],[2,2,0],[2,3,1]],thickness=3,
> style=line,color=black):
> u3:=pointplot3d([[3,0,1],[3,1,0],[3,2,1],[3,3,0]],thickness=3,
> style=line,color=black):
> v0:=pointplot3d([[0,0,0],[1,0,1],[2,0,0],[3,0,1]],thickness=3,
> style=line,color=black):
> v1:=pointplot3d([[0,1,-1],[1,1,0],[2,1,-1],[3,1,0]],thickness=3,
> style=line,color=black):
> v2:=pointplot3d([[0,2,1],[1,2,0],[2,2,0],[3,2,1]],thickness=3,
> style=line,color=black):
> v3:=pointplot3d([[0,3,0],[1,3,1],[2,3,1],[3,3,0]],thickness=3,
> style=line,color=black):
> fl:=plot3d([P_x,P_y,P_z],u=0..1,v=0..1,axes=boxed,
> scaling=constrained):
> display3d({u0,u1,u2,u3,v0,v1,v2,v3,fl},
> scaling=constrained,axes=normal);
```

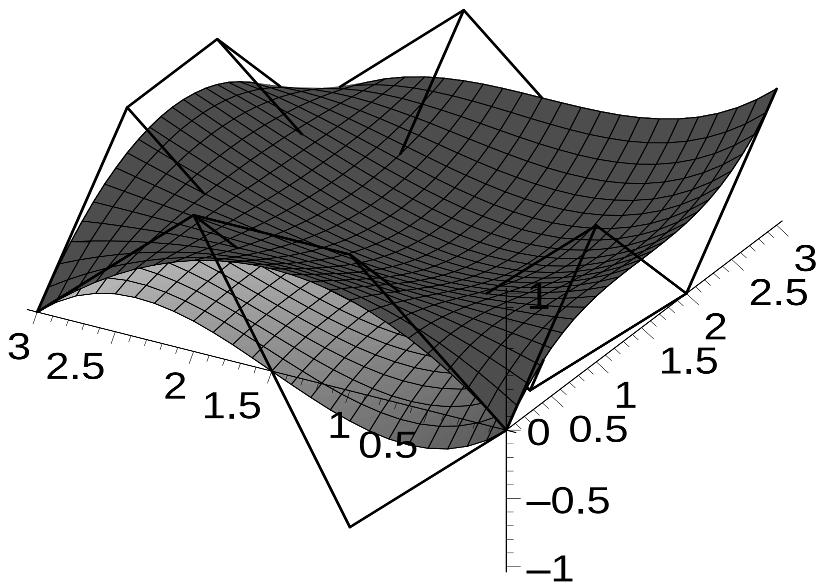



Flächendesign ist nun durch Veränderung der Kontrollstruktur möglich: zum Beispiel der Punkt b_01 wird von (0,1,-1) auf (0,1,1) verändert:

```
> P_z1:=simplify((subs(b[0,0]=0,b[1,0]=1,b[2,0]=1,b[3,0]=0,b[0,1]=1,
> b[1,1]=0,b[2,1]=0,b[3,1]=1,b[0,2]=0,b[1,2]=-1,b[2,2]=0,b[3,2]=1,
> b[0,3]=1,b[1,3]=0,b[2,3]=1,b[3,3]=0,P3)));
```

$$P\_z1 := -3\,u^2 - 6\,v^2 + 4\,v^3 + 3\,v + 18\,u\,v^2 - 6\,u\,v^3 + 18\,u^2\,v - 9\,u^2\,v^2 - 6\,u^3\,v^2 + 2\,u^3\,v^3 + 3\,u - 18\,u\,v$$

```
> u01:=pointplot3d([[0,0,0],[0,1,1],[0,2,1],[0,3,0]],thickness=3,
> style=line,color=black):
> v11:=pointplot3d([[0,1,1],[1,1,0],[2,1,-1],[3,1,0]],thickness=3,
> style=line,color=black):
> fl1:=plot3d([P_x,P_y,P_z1],u=0..1,v=0..1,axes=boxed,
> scaling=constrained,color=red):
> display3d({u0,u01,u1,u2,u3,v0,v11,v2,v3,fl1,fl
> },scaling=constrained,axes=normal,orientation=[-150,60]);
```



## 4.3 NURBS und Modeling mit NURBS

Bezierkurven und Bezierflächen waren historisch die ersten Freiformwerkzeuge im CAD. Sie sind einfach zu behandeln, haben aber einige gravierende Nachteile. Darunter fallen vor allem der hohe Grad der entstehenden Bezierkurve bzw. Bezierfläche, wenn die Anzahl der Kontrollpunkte hoch ist. Man kann diesen Nachteil durch Aneinanderstückeln von Bezierkurven niederer Ordnung aufheben. Beim Zusammenfügen ist aber sehr genau auf den Übergang von einem Kurvenstück zum nächsten zu achten. Als minimale Forderung wird man die Gleichheit der Tangente im Endpunkt der einen und im Anfangspunkt der anderen Kurve beachten. Für praktische Anwendung wird Tangentenübergang aber meist nicht genug sein. Zusammengesetzte Bezierkurven heißen *B-Splines*.

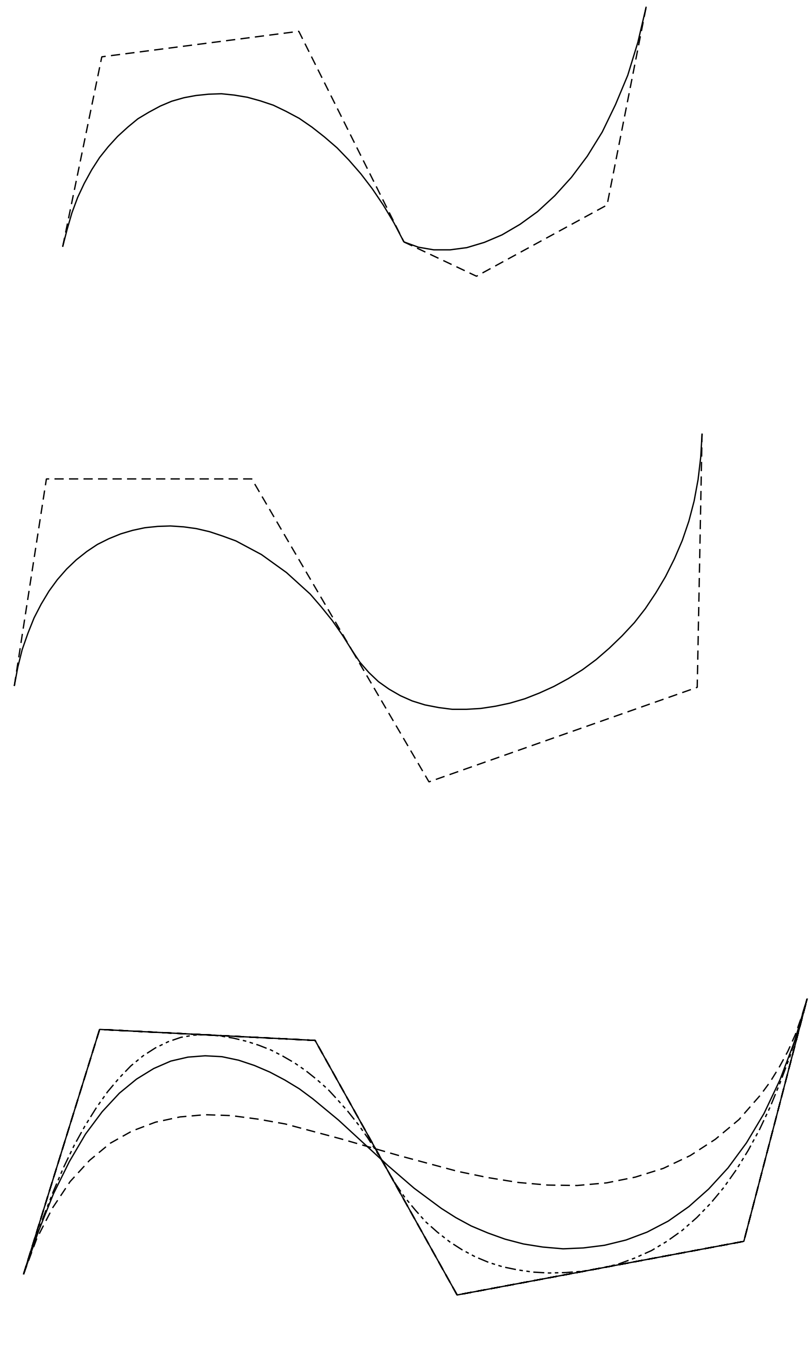


Abbildung 20: B-Spline Kurven

Abbildung 20 zeigt von oben nach unten:

1. Zwei aneinandergestückelte Bezierkurven, die nur Endpunkt und Anfangspunkt gemeinsam haben. Die Kurven sind an der Anschlussstelle zwar stetig (d.h. es gibt kein Loch zwischen den Kurven), aber nicht differenzierbar (die Tangente stimmt nicht überein).

2. Das mittlere Bild zeigt zwei zusammengesetzte Bezierkurven mit Tangentenübergang.

3. Das untere Bild zeigt drei B-Splinekurven, die jeweils aus Bezierkurven unterschiedlicher Ordnungen zusammengesetzt sind. Die strichpunktierte Kurve besteht aus Stücken von Kurven 2. Ordnung (Parabeln). Die durchgezogene Kurve ist ein B-Spline der Ordnung 3 und die strichlierte Kurve ist ein B-Spline der Ordnung 5. Offensichtlich ist die Form der Kurve stark durch die vorgegebene Ordnung des B-Spline abhängig. Grundsätzlich gilt, je niedriger die Ordnung, desto näher ist die Splinekurve zum Kontrollpolygon. Die meisten CAD Systeme bieten die Möglichkeit des Einstellens einer Kurvenordnung an[1]. Wenn man diese Ordnung einstellt, dann sollte man bedenken, dass man dann mindestens einen Kontrollpunkt mehr angeben muss als die Ordnung ist; sonst besteht keine Einschränkung. Warum man einen Kontrollpunkt mehr vorgeben muss als die Ordnung ist unmittelbar aus dem Casteljau Algorithmus klar.

Wir fassen zusammen und listen noch einige weitere wichtige Eigenschaften von B-Splines:

1. B-Splinekurven sind aus Stücken polynomialer Kurven (ihre mathematische Darstellung ist ein Polynom) von einer fester Ordnung $p$ zusammengesetzt.

2. Je niedriger die Ordnung desto näher ist die Kurve dem Kontrollpolygon.

3. Lokale Veränderbarkeit: die Veränderung eines Kontrollpunktes wirkt sich nur auf den unmittelbar in der Nähe gelegenen Teil der Kurve aus (Bei der Änderung eines Kontrollpunktes einer Bezierkurve ändert sich die ganze Kurve!)

4. Affine Invarianz.

5. Bezierkurven sind ein Sonderfall der B-Splines.

Zur Konstruktion von B-Spline Kurven braucht man mehr Information (was übrigens von den meisten CAD-Systemen unterdrückt wird) und es steht eine wesentlich kompliziertere mathematische Theorie dahinter[2]. Die obigen zusätzlichen Eigenschaften zu den Bezierkurveneigenschaften rechtfertigen aber den zusätzlichen rechnerischen Aufwand (den der CAD Benutzer ohnedies nicht spürt).
Die B-Splines haben aber noch einen gravierenden Nachteil: viele bekannte und in CAD Systemen verwendete Kurven, wie z.B. Kreise und Ellipsen können durch sie nicht dargestellt werden. Abbildung 21 zeigt die Annäherung eines Kreises durch B-Splines von der Ordnung 2,3,5,6. Man könnte die Ordnung noch beliebig hinaufschrauben und man würde keinen Kreis bekommen; außerdem ist eine Kurve von der Ordnung 7 mathematisch bereits einigermaßen kompliziert. Um diese Kreise auch als stückweise Splines darzustellen und dadurch als Ausgangskurven für ein Kurvendesign zu bekommen, braucht man noch einen Verallgemeinerungsschritt. Diese Verallgemeinerung sind die NURBS (**N**on-**U**niform **R**ational **B-S**plines). Mit diesen Kurventypen können nun die meisten klassischen Kurven wie Kreise, Ellipsen, Hyperbeln in ein und

[1]Es muss vermerkt werden, dass der Begriff Ordnung einer B-Splinekurve in der Praxis nicht einheitlich gehandhabt wird. Manche CAD-Systeme verwenden den Begriff Ordnung als: mathematische Ordnung+1.

[2]Der/die interesierte StudentIn sei bezüglich der mathematischen Hintergründe auf die Vorlesung "Geometrische Grundlagen des CAD" verwiesen. Dort werden die mathematische Darstellung dieser Kurventypen und auch Beweise für die obigen Eigenschaften gebracht.

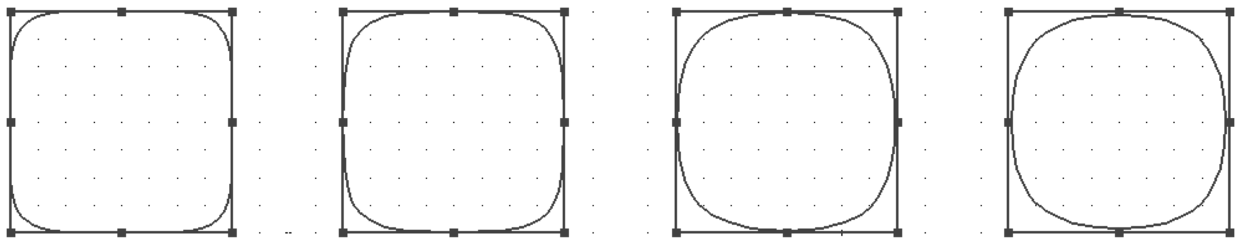


Abbildung 21: B-Spline Kurven zur Kreisannäherung

demselben mathematischen Rahmen dargestellt werden. Es würde den Rahmen dieser Vorlesung sprengen die mathematische Darstellung und alle zusätzlichen Eigenschaften von NURBS zu besprechen. Aber auf zwei in manchen CAD Systemen (z.B. RHINO) bereits implementierte (Design)Eigenschaft sei zum Schluss noch hingewiesen: Bei NURBS ist jeder Punkt des Kontrollpolygons noch zusätzlich mit einem Gewicht versehen. Je höher das Gewicht eines Kontrollpunktes angesetzt wird, desto mehr wird die Kurve von ihm angezogen. Diese Eigenschaft ist in den beiden Kurven in Abbildung 22 gezeigt. Bei der linken Kurve sind alle Kontrollpunkte gleich gewichtet, während auf der rechten Kurve der Kontrollpunkt $K_3$ das 30-fache Gewicht der anderen Punkte hat. Es ist offensichtlich, dass man durch die Gewichte noch eine zusätzliche (lokale) Designmöglichkeit bekommt. Die zweite Eigenschaft ist, zusätzliche Kontrollpunkte

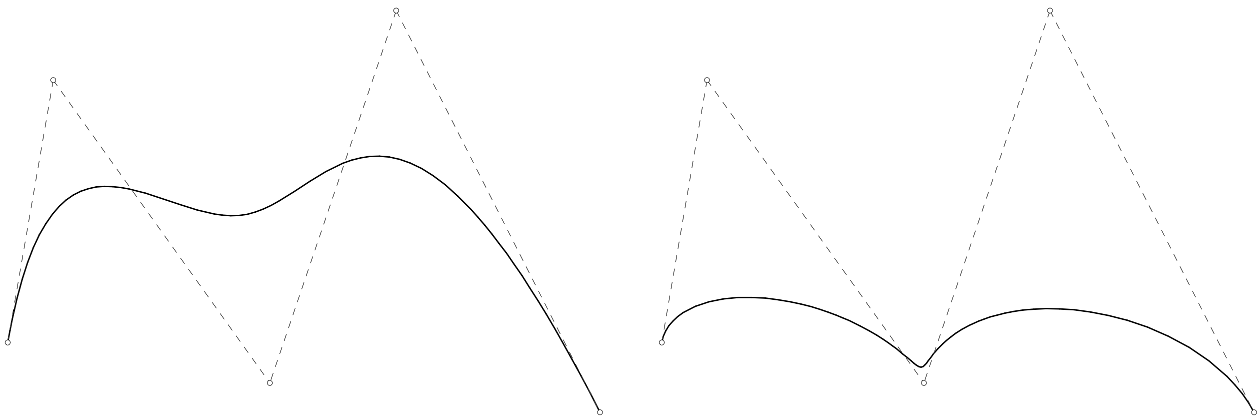


Abbildung 22: Gewichte von Kontrollpunkten

einbauen zu können, ohne die Ordnung und die Form der Kurve zu ändern. Dies wird in den Abbildungen 23 und 24 gezeigt. Wir gehen aus von einem Kreis mit seinen Kontrollpunkten. Abbildung 23 zeigt links einen Kreis mit der minimalen Anzahl von Kontrollpunkten. Rechts wurde ein Kontrollpunkt verändert. In Abbildung 24 wurden Kontrollpunkte hinzugefügt und die rechte Kurve zeigt die Veränderung durch Ziehen an einem Kontrollpunkt. Man beachte die geänderte Auswirkung der Veränderung des Kontrollpunktes auf die Form der neuen Kurve.
Analog zur Konstruktion der Bezierflächen als Tensorprodukte von Bezierkurven können auch B-Splineflächen und NURBS-Flächen konstruiert werden. Da die mathematische Darstellung dieser Freiformflächen schon einigermaßen kompliziert ist wird darauf hier verzichtet. In das Bauwesen und in die Architektur haben die Freiformflächen aber bereits seit einger Zeit Einzug gehalten. Dies mögen die beiden abschließenden Abbildungen zeigen:

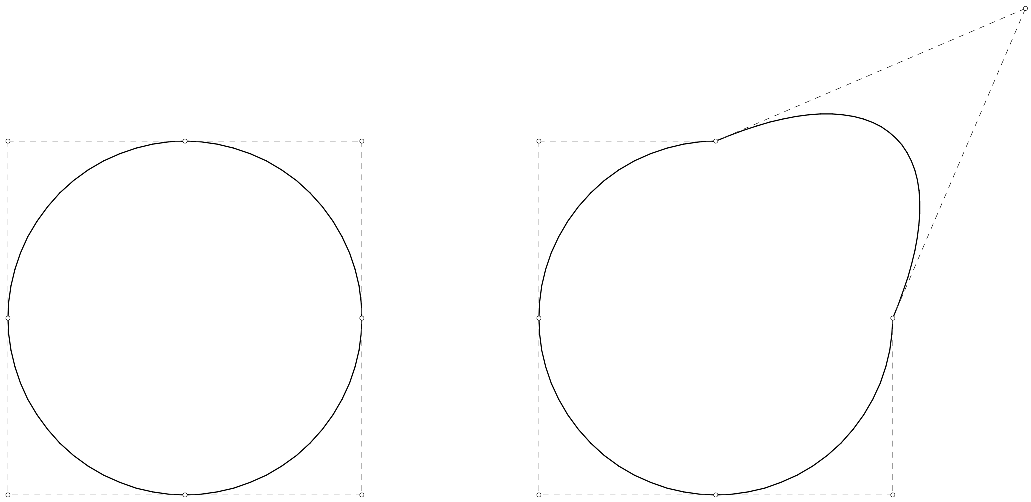


Abbildung 23: Minimale Kontrollpunkte eines KreisNURBS

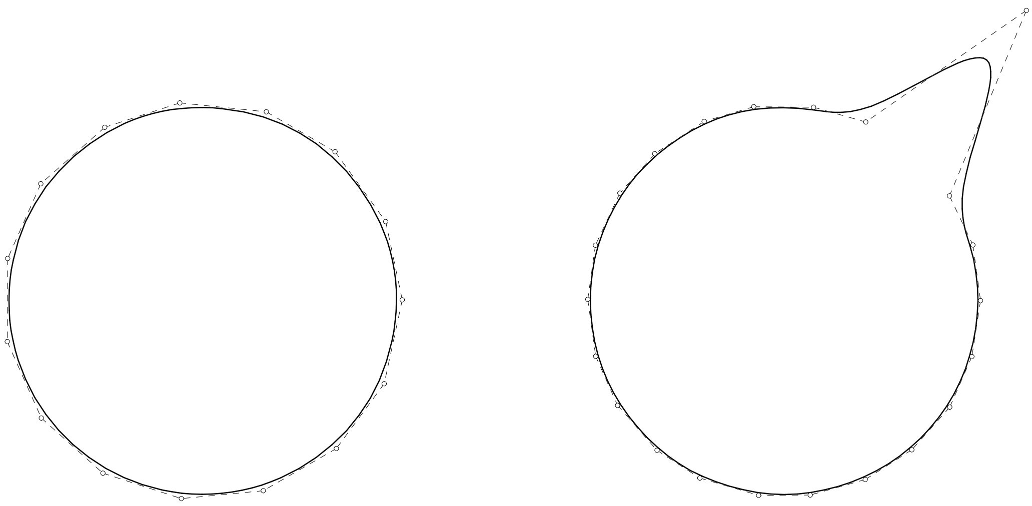


Abbildung 24: Hinzufügen von Kontrollpunkten

## 4.4 Übersichtsfragen

1. Welche Darstellungsformen gibt es für Kurven, Flächen?
2. Welche Darstellungsforme sind für die Computergraphik günstig?
3. Was versteht man unter linearer Interpolation?
4. Was ist eine Bezierkurve?
5. Welche geometrische Eigenschaften haben Bezierkurven?
6. Wie funktioniert der De Casteleau Algorithmus?
7. Wozu braucht man ein Kontrollpolygon und Kontrollpunkte?
8. Was ist ein B-Spline?
9. Welche zusätzlichen Eigenschaften haben B-Splines?
10. Welche Eigenschaften haben NURBS?

Abbildung 25: Freiformfläche zur Überdachung der Ankunftshalle Inchon Airport, Seoul Korea

Abbildung 26: Konstruktive Ausführung des Daches Inchon Airport, Seoul Korea

11. Was wird durch Veränderung von Gewichten bewirkt?

12. Was bewirkt das Einfügen von zusätzlichen Kontrollpunkten?