Curves and Surfaces in CAGD

Dipl.-Math. Katja Bühler

March 20, 2000

1 Parametric Representations

A parametric curve in $\mathbb{E}^3$ is given by

$$c: \qquad \boldsymbol{c}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}; \quad t \in I = [a,b] \subset \mathbb{R}$$

where $x(t), y(t)$ and $z(t)$ are differentiable functions in t. The vector $\dot{\boldsymbol{c}}(t)$ is called *tangent vector*. A curve point $\boldsymbol{c}(t_0), t_0 \in I$ is called *regular* if $\dot{\boldsymbol{c}}(t_0) \neq \boldsymbol{o}$. A parametrization is called regular, if $\dot{\boldsymbol{c}}(t) \neq \boldsymbol{o}$ for all $t \in I$. Any differentable change of the parameter $\tau = \tau(t)$ does not change the curve. Moreover, if $\dot{\tau} \neq 0$ in I then $\boldsymbol{d}(t) = c(\tau(t)$ ist also a regular parametrization. A curve which admits a regular parametrization ist called regular.

A parametric surface is given by

$$s: \quad \boldsymbol{s}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \quad (u,v) \in [a,b] \times [c,d] = I \times J \subset \mathbb{R}^2$$

where $x(u,v), y(u,v)$ and $z(u,v)$ are differentiable functions of the parameters u and v.

The lines $\boldsymbol{s}(u, v_0)$ with $v_0 \in J$ fixed and $\boldsymbol{s}(u_0, v)$ with $u_0 \in I$ fixed are called *isoparametric lines* of the surface. The tangent plane of a surface point is defined by the two tangent vectors $\boldsymbol{s}_u$ and $\boldsymbol{s}_v$, the surface normal vector at this point is $\boldsymbol{n} = \boldsymbol{s}_u \times \boldsymbol{s}_v$. A surface point is called regular if $\boldsymbol{n} \neq 0$.

For a good introduction to differential geometry see e.g. [Aumann, Spitzmueller '93]. A detailed discussion of the following topics can be found in the literature listed at the end of this summary. Almost all pictures are generated with the help of a collection of **CAGD-Java applets** written by the Geometric Design Group at the University of Karlsruhe:

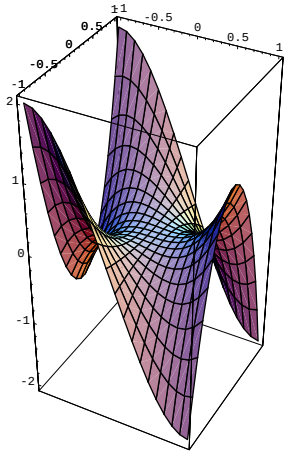


Figure 1: Surface with Isolines

http : *//i33www.ira.uka.de/applets/mocca/html/noplugin/inhalt.html*

2 Bézier Curves

2.1 The Constructive Approach: The de Casteljau Algorithm

First described by de Casteljau it is probably the most fundamental algorithm in the field of curve and surface design, not least because it is so easy to understand. " Its main attraction is the beautiful interplay between geometry and algebra: a very intuitive geometric construction leads to a powerful theory" [Farin '90].

The Algorithm

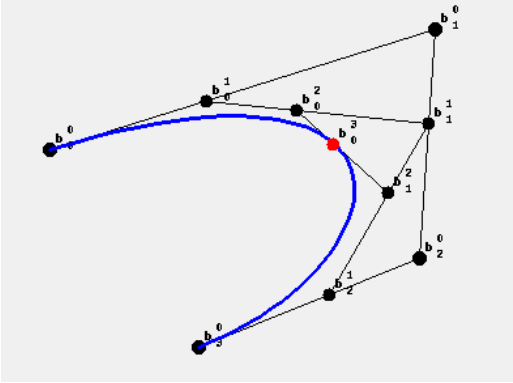
Given $n+1$ points $\boldsymbol{b}_0, ..., \boldsymbol{b}_n \in \mathbb{E}^3$ and an arbitrary $t \in \mathbb{R}$.
Then

$$\boldsymbol{b}_i^r := (1-t)\boldsymbol{b}_i^{r-1} + t\boldsymbol{b}_{i+1}^{r-1}, \qquad i = r, ..., n \tag{2.1}$$

and $\boldsymbol{b}_i^0 := \boldsymbol{b}_i$, defines the point $\boldsymbol{b}_0^n$ with parameter value t on the so called *Bézier curve* $\boldsymbol{b}^n$. The points $\boldsymbol{b}_i$ are called *Bézier points*.

Equation (2.1) describes a repeated linear interpolation whose intermediate coefficients can be written into the triangular de Casteljau scheme:

$$\begin{array}{ccccc}
\boldsymbol{b}_0^0 & & & & \\
& \boldsymbol{b}_0^1 & & & \\
\boldsymbol{b}_1^0 & & \boldsymbol{b}_0^2 & & \\
& \boldsymbol{b}_1^1 & & \boldsymbol{b}_0^3 & \\
\boldsymbol{b}_2^0 & & \boldsymbol{b}_1^2 & & \\
& \boldsymbol{b}_2^1 & & & \\
\boldsymbol{b}_3^0 & & & &
\end{array}$$



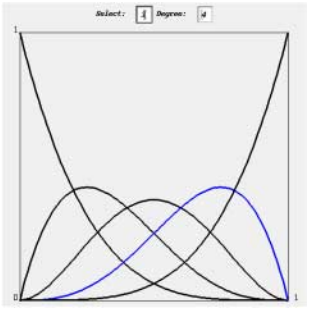
2.2 The Analytical Approach: Bézier Curves and Bernstein Polynomials

The de Casteljau algorithm gives a recursive definition of Bézier curves in terms of an algorithm. For further theoretical development it is also necessary to have an explicit parametric representation for them. Based on the de Casteljau recursion it can be shown by mathematical induction, that a Bézier curve $\boldsymbol{b}(t)$ with respect to the Bézier points $\boldsymbol{b}_i, i = 0, ..., n$ is given by

$$\boldsymbol{b}(t) = \sum_{i=0}^{n} \boldsymbol{b}_i B_i^n(t)$$

where $B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$. are the *Bernstein Polynomials* of n-th degree with the following important properties

- partition of unity: $\sum_{i=0}^{n} B_i^n(t) \equiv 1$
- positivity: $B_i^n(t) > 0$
- recursion: $B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i+1}^{n-1}(t)$



2.3 Some Properties of Bézier Curves

- Affine invariance: An affine transformation of the Bézier points is equivalent to an affine transformation of the whole curve.
- Convex hull property: A Bézier curve lies inside the convex hull of its Bézier points.
- Endpoint interpolation: The first and last Bézier points lie on the curve.
- Linear precision: If all Bézier points lie on a straight line, the corresponding Bézier curve is identical to this line.
- Variation diminishing property: A Bézier curve has no more intersections with a plane than its Bézier polygon

Disadvantages of Bézier curves:

- Pseudo local control: changing one of the control points changes the shape of the whole curve, although, if for instance $\boldsymbol{b}_i$ is changed, it is mostly affected around the point corresponding to the parameter value i/n, if n denotes the algebraic degree of the parametrization.
- The degree of a Bézier curve depends directly on the number of control points. Thus, higher flexibility through more control points is equivalent to a higher degree of the curve, which means higher computational cost and less control on the behaviour of the curve.

2.4 Derivatives

It can be easily shown, that

- the lines $\boldsymbol{b}_0\boldsymbol{b}_1$ and $\boldsymbol{b}_{n-1}\boldsymbol{b}_n$ are the tangent lines at the curve points $\boldsymbol{b}_0$ and $\boldsymbol{b}_n$.
- the intermediate points $\boldsymbol{b}_0^{n-1}$ and $\boldsymbol{b}_1^{n-1}$ of the de Casteljau algorithm detremine the tangent line at the position $\boldsymbol{b}_0^n(t)$.

2.5 Important Algorithms

- **Degree elevation:** Adding new control points to increase flexibility without changing the shape of the curve.

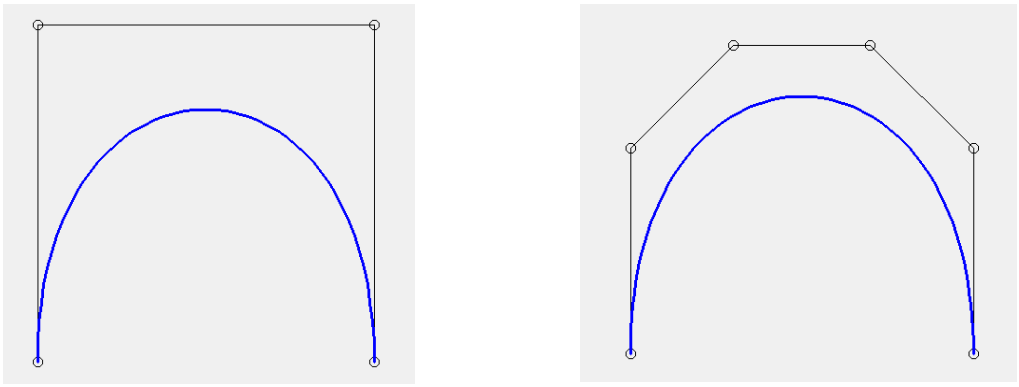
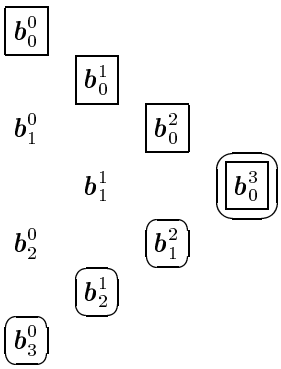
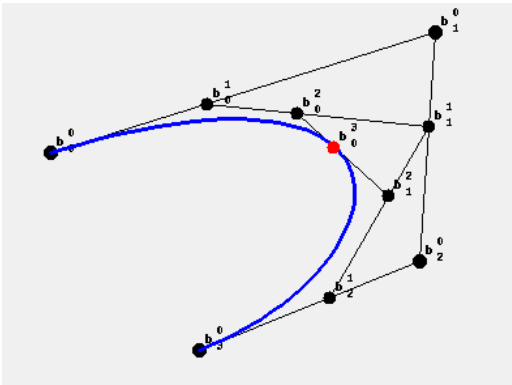


Figure 2: A curve of degree 3 displayed as curve of degree 5

- **Subdivision:** Subdivision of a Bézier curve increases its flexibility without increasing its degree and repeated subdivision converges very fast towards the curve.

  The subdivision algorithm is a byproduct of the de Casteljau algorithm: The two new generated sides of the triangle contain the Bézier points of the two parts of the subdivided curve.





For a detailed description of the algorithms see e.g. [Hoschek, Lasser '92].

2.6 Algorithms to Evaluate Bézier Curves

The de Casteljau algorithm is numerical stable, but not efficient.

Following the same idea like the Horner scheme, a Bézier curve can be written in the nested form

$$\boldsymbol{b}(t) = (...((\binom{n}{0} s\boldsymbol{b}_0 + \binom{n}{1} t\boldsymbol{b}_1)s + \binom{n}{2} t^2\boldsymbol{b}_2)s + ...)s + \binom{n}{n} t^n\boldsymbol{b}_n$$

with $s := 1 - t$, which leads to a more efficient algorithm.

Like mentioned above, repeated subdivision gives a good approximation for the curve.

3 Splines

The use of B-Splines to define curves and surface for CAGD was first proposed by Gordon and Riesenfeld in the early 70´s, but its theory goes back to the early 19´th century. B-Splines are a very common tool in CAD-Systems for the design of curves and surfaces and have two advantages over Bézier techniques:

1. The degree of a B-spline polynomial can be set independently of the number of control points (with certain limitations).
2. B-splines allow local control over the shape of a spline curve or surface.

3.1 B-Spline Curves

Given a set of $n+1$ control points (*de Boor points*) $\boldsymbol{d}_i, i = 0, ..., n$ and a knot vector $U = (u_0, ..., u_{n+k})$. The corresponding B-spline curve $\boldsymbol{s}(u)$ is a piecewise polynomial curve of order k (of degree $k-1$) with $1 \leq k \leq n+1$ of the form

$$\boldsymbol{s}(u) = \sum_{i=0}^{n} \boldsymbol{d}_i N_i^k(u)$$

The $N_i^k(u)$ are the normalized B-spline basis functions of order k with respect to U with the following recursive definition:

$$N_i^0(u) = \begin{cases} 1 & u \in [u_i, u_{i+1}) \\ 0 & \text{sonst} \end{cases} \tag{3.2}$$

$$N_i^r(u) = \frac{u - u_i}{u_{i+r-1} - u_i} N_i^{r-1}(u) + \frac{u_{i+r} - u}{u_{i+r} - u_{i+1}} N_{i+1}^{r-1}(u); \qquad r = 1, ..., k \tag{3.3}$$

Properties of the basis functions:

- partition of unity: $\sum_{i=0}^{n} N_i^k(u) \equiv 1$
- positivity: $N_i^k(u) > 0$
- local support $N_i^k(u) = 0$ if $u \notin [u_i, u_{i+k})$

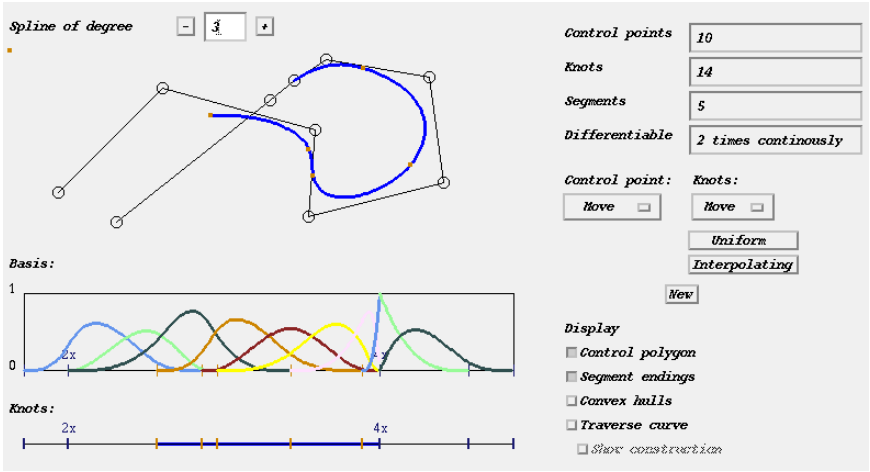


Figure 3: A B-spline curve with all related information

3.2 Some Properties of B-Spline Curves

- Affine invariance
- Strong convex hull property: The curve segment corresponding to the parameter values $[u_i, u_{i+1})$ lies inside the convex hull of the control points $\boldsymbol{d}_{i-k}, ..., \boldsymbol{d}_i$.
- Variation diminishing property.
- Local support: Moving $\boldsymbol{d}_i$ changes $\boldsymbol{s}(u)$ only in the parameter interval $[u_i, u_{i+k})$.
- Multiple knot points $u_i = ... = u_{i+s}$ are possible. If $s \geq k$ the curve goes through the control point $\boldsymbol{d}_i$. Furthermore if $n = k - 1$ and $U = (u_0, ..., u_0, u_1, ..., u_1)$ then $\boldsymbol{s}(u)$ is a Bézier curve.
- Differentiability: $\boldsymbol{s}(u)$ is $k - l - 1$ times differentiable at a knot u_i, if u_i is of multiplicity $l \geq 1$.

3.3 Some Special Types of B-Spline Curves

The form of the knot vector determines three special cases of B-spline curves:

1. open: $u_0 = ... = u_{k-1}$ and $u_{n+1} = ... = u_{n+k}$
2. closed: $\boldsymbol{d}_{n+1} := \boldsymbol{d}_0, \boldsymbol{d}_{n+2} := \boldsymbol{d}_1, ...$
3. uniform: If the spacing between knot values is constant, the resulting curve is called a uniform B-spline curve. Uniform B-spline curves have periodic basis functions and therefore many algorithms have a simpler and more effective implementation.

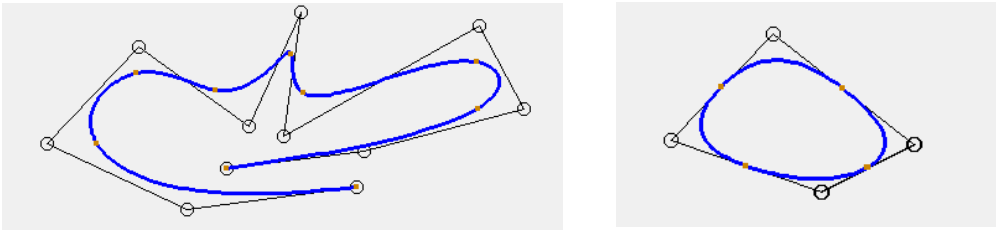


Figure 4: An open and a closed B-spline curve

3.4 Evaluating B-Spline Curves

The de Boor algorithm

The de Boor algorithm is a generalized de Casteljau algorithm and works following the same principles of linear interpolation. To evaluate $\boldsymbol{s}(u)$ at $u = x_0$, $x_0 \in [u_l, u_{l+1})$ the following recusion has to be done:

$$\boldsymbol{d}_i^r := (1 - \alpha_i^r)\boldsymbol{d}_{i-1}^{r-1} + \alpha_i^r \boldsymbol{d}_i^{r-1}, \quad i = l - k + 1, ..., l; \quad r = 0, ..., k - 1$$

with

$$\alpha_i^r := \frac{x_0 - u_i}{u_{i+k-r} - u_i}$$

where $\boldsymbol{d}_i^0 := \boldsymbol{d}_i$ and $\boldsymbol{s}(x_0) = \boldsymbol{d}_l^{k-1}$.

The de Boor scheme has the same form like the de Casteljau scheme.

The de Boor algorithm allows the evaluation of the curve, without any knowledge of the basis functions and proposes an effective methode.

The direct evaluation

A second possibility to evaluate a B-Spline function for s parameter value x is given by the following algorithm.

1. Find the knot span $[u_i, u_{i+1})$ in which x lies.
2. Compute the non zero basis functions.
3. multiply the values of the nonzero basis functions with the corresponding control points.

3.5 Algorithms

- **Knot insertion and knot refinement:**
  The *knot insertion* algorithm inserts a knot one or multiple times into the knot vector. Knot insertion does not change the shape of the curve, but refines its segmentation. This algorithm is used to increase the flexibility of a curve, to compute derivatives, to split curves and to evaluate a curve for a certain parameter value: The de Boor algorithm is a repeated knot insertion algorithm that inserts this parameter value $k + 1$ times into the knot vector. There exist special algorithms that insert several different knots simultaneously (*knot refinement*).

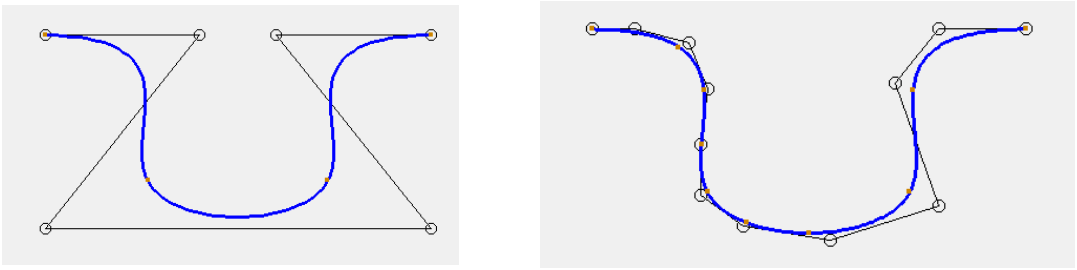


Figure 5: An example for knot refinement

- **Degree elevation:**
  Adapts curve degrees whithout changing the shape to build combined structures, like tensor product surfaces or to connect curves and surfaces.

For a detailed description of the algorithms see e.g. [Piegl, Tiller '95].

4 Rational Curves

Although polynomials offer a lot of advantages, there exist a number of important curve and surface types which cannot be represented precisely using polynomials, e.g. conic sections and quadrics that have a

rational parametrization.
In general a rational parametrized curve has the form:

$$\boldsymbol{c}(u) = \begin{pmatrix} \frac{x(u)}{w(u)} \\ \frac{y(u)}{w(u)} \\ \frac{z(u)}{w(u)} \end{pmatrix}$$

A more elegant and very useful representation is the one using homogeneous coordinates: the curve $\boldsymbol{c}$ is represented as a polynomial curve in $\mathbb{E}^4$.

$$\mathbb{c}(u) = \begin{pmatrix} w(u) \\ x(u) \\ y(u) \\ z(u) \end{pmatrix}$$

The original curve has to be interpreted as a projection of this curve onto the hyperplane $w(u) = 1$ of $\mathbb{E}^4$.

A homogeneous represenation $\mathbb{p} = (w, x, y, z)^T$ of a point can be converted back to the euclidean representation in the following way: $\boldsymbol{p} = (x/w, y/w, z/w)^T$.

4.1 Rational Bézier Curves

A rational Bézier curve is defined as

$$\boldsymbol{b}(u) = \frac{\sum_{i=0}^{n} w_i \boldsymbol{b}_i B_i^n(u)}{\sum_{i=0}^{n} w_i B_i^n(u)}$$

The $w_i, i = 0, ..., n$ are called weights and are assumed to be positive. If all $w_i = 1$, $\boldsymbol{b}(u)$ denotes a polynomial Bézier curve.

Writing $\boldsymbol{b}(u)$ in terms of homogeneous coordinates yields the following represenation:

$$\mathbb{b}(u) = \sum_{i=0}^{n} \mathbb{b}_i B_i^n(u)$$

with the *homogeneous Bézier points* $\mathbb{b}_i = (w_i, w_i \boldsymbol{b}_i^T)^T$.

Properties

Rational Bézier curves have the same properties as non-rational ones, but they

- are even projective invariant
- do not lie inside the control polygon if negative weights are allowed and
- have the weights as additional design parameter: increasing the weight w_i causes an attraction of the curve towards the Bézier point $\boldsymbol{b}_i$.

Algorithms

All algorithms for polynomial Bézier curves can be applied in the same way onto the homogeneous representation of a rational Bézier curve.

4.2 Non Uniform Rational B-Spline Curves (NURBS)

NURBS are the most important and flexible design elements provided in CAD systems. Polynomial and rational Bézier curves and B-spline curves are subsets of NURBS. A NURBS with respect to the control points $\boldsymbol{d}_0, ..., \boldsymbol{d}_n$ and the knot vector $U = (u_0, ..., u_{n+k})$ is defined as

$$\boldsymbol{n}(u) = \frac{\sum_{i=0}^{n} w_i \boldsymbol{d}_i N_i^k(u)}{\sum_{i=0}^{n} w_i N_i^k(u)}$$

The $w_i, i = 0, ..., n$ are called weights and are assumed to be positive. If all $w_i = 1$, $\boldsymbol{b}(u)$ denotes a polynomial B-Spline curve.

Writing $\boldsymbol{n}(u)$ in terms of homogeneous coordinates yields the following represenation:

$$\mathbb{n}(u) = \sum_{i=0}^{n} \mathbb{d}_i N_i^k(u)$$

with the *homogeneous control points* $\mathbb{d}_i = (w_i, w_i \boldsymbol{n}_i^T)^T$.

Properties

NURBS have the same properties as polynomial B-spline curves, but they

- are even projective invariant
- do not lie inside the control polygon if negative weights are allowed and
- have the weights as additional design parameter: changing the weight w_i affects only the interval $[u_i, u_{i+k})$

Algorithms

All algorithms for polynomial Bézier curves can be applied without any change to the homogeneous representation of a NURBS curve.

5 Surfaces

5.1 Tensorproduct Surfaces

"A surface is the locus of a curve that is moving through space and thereby changing its shape"

Let

$$\boldsymbol{f}(u) = \sum_{i=0}^{n} \boldsymbol{c}_i F_i(u)$$

be a curve in $\mathbb{E}^3$ with $F_i(u), i = 0, ..., n$ as Basis Functions (e.g. Bernstein polynomials or B-spline basis functions). Moving $\boldsymbol{f}$ through space while deforming it, is equivalent to continuously changing the control points $\boldsymbol{c}_i$ which can be described by

$$\boldsymbol{c}_i(v) = \sum_{j=0}^{m} \boldsymbol{a}_{ij} G_j(v)$$

where the $G_j(v), j = 0, ..., m$ are Basis Functions too.

Combining both equations yields the definition of a tensor product surface:

$$\boldsymbol{s}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \boldsymbol{a}_{ij} F_i(u) G_j(v).$$

Bézier Surfaces

A tensorproduct Bézier surface is given by

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

The Bézier points $\boldsymbol{b}_{ij}$ form the *control net* of the surface.

Tensorproduct Bézier surfaces have properties analogue to that of Bézier curves.

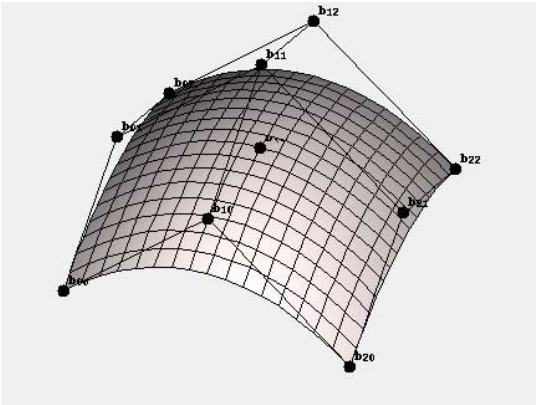


Figure 6: A Bézier tensor product surface

All algorithms for Bézier curves can be applied in two steps to the surface $\boldsymbol{b}(u,v)$:

1. Apply algorithm on the curves $\boldsymbol{b}_i(v) = \sum_{j=0}^{n} \boldsymbol{b}_{ij} B_j^n(v)$.
2. Apply algorithm on the curves $\boldsymbol{b}(u,v) = \sum_{i=0}^{m} \boldsymbol{b}_i(v) B_i^m(u)$.

B-Spline Surfaces

A tensorproduct B-spline surface with respect to the control points $\boldsymbol{d}_{ij}$ and the knot vector $U = (u_0, ..., u_{m+k})$ and $V = (v_0, ..., v_{n+l})$ is given by

$$\boldsymbol{s}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \boldsymbol{d}_{ij} N_i^k(u) B_j^l(v).$$

Tensorproduct B-spline surfaces have properties analogue to that of Bézier curves.

All algorithms for B-spline curves can be applied in in the same two steps to the surface $\boldsymbol{s}(u,v)$ like the algorithms for Bézier curves on tensorproduct Bézier surfaces.

5.2 Bézier Triangles

A triangular Bézier patch is defined by

$$bmx(u,v) = \sum_{i+j+k=n} \boldsymbol{b}_{i,j,k} B^n_{i,j,k}(u,v,w)$$

where $i,j,k \geq 0$ and u,v,w are baryzentric coordinates of the triangular parameter domain. The $B^n_{i,j,k}$ are generalized Bernstein polynomials of degree n

$$B^n_{i,j,k}(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k$$

. The Bézier net of the surfaces is formed by the $\frac{1}{2}(n+1)(n+2)$ Bézier points $\boldsymbol{b}_{i,j,k}$.

The properties of that the triangular Bézier patch inherits are the same as described for the univariat case.

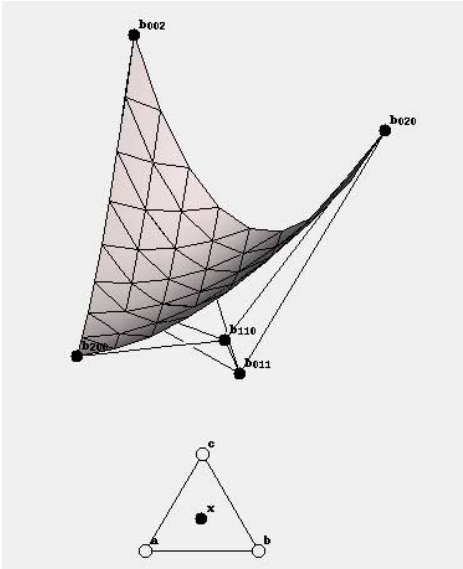


Figure 7: An elliptic paraboloid as Bézier triangle

5.2.1 Algorithms (see [Hoschek, Lasser '92])

- The de Casteljau algorithm for triangular patches produces a tetrahedral scheme. The recursion formula for the computation is:

  $$\boldsymbol{b}^l_{ijk} = u\boldsymbol{b}^{l-1}_{i-1jk} + v\boldsymbol{b}^{l-1}_{ij-1k} + w\boldsymbol{b}^{l-1}_{ijk-1}$$

  where $i+j+k = n-l, (i,j,k \geq 0)$ and $b^0_{ijk} = b_{ijk}$. It is easy to show that $\boldsymbol{x}(u,v,w) = b^n_{000}$.

- Degree elevation

- Subdivision: the subdivision of the patch into three subpatches can be derived from the de Calstjau algorithm, like in the univariat case.

References

Guenter Aumann, Klaus Spitzmueller (1993). *Computerorientierte Geometrie.* Reihe Informatik 89. B. I. Wissenschaftsverlag.

Jules Bloomenthal, Jon Rokne (1994). Homogeneous coordinates. *The Visual Computer*, 11:15–26.

Wolfgang Boehm (1984). A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60.

Gisela Engeln-Müllges, Fritz Reutter (1996). *Numerik Algorithmen.* VDI Verlag, 8 Auflage.

Gerald Farin (1990). *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, INC, 2. Auflage.

Gerald Farin (1992). From Conics to NURBS: A Tutorial and Survey. *IEEE Computer Graphics and Applications*, 12(9):78–86.

Josef Hoschek, Dieter Lasser (1992). *Grundlagen der geometrischen Datenverarbeitung.* B.G. Teubner Stuttgart, 2 Auflage.

Les Piegl, Wayne Tiller (1995). *The NURBS book.* Springer, Berlin, Heidelberg, New York.

Les Piegl (1991). On NURBS: A Survey. *IEEE Computer Graphics and Applications*, 11(1):55–71.