Curves and Surface II

Angel Ch.10

Curves and Surfaces I

Surface representation

- explicit
- implicit
- parametric

parametric forms are widely used in computer graphics

Parametric forms

- cubic polynomial
- local defininition
- Interpolating

This lecture: other parametric forms of surfaces

- Hermite
- Bezier
- B-Spline, NURBS

Hermite Curves and Surfaces

Rather than interpolating points we interpolate between endpoints + tangents at end points

- ensures continuity between curve/surface segments

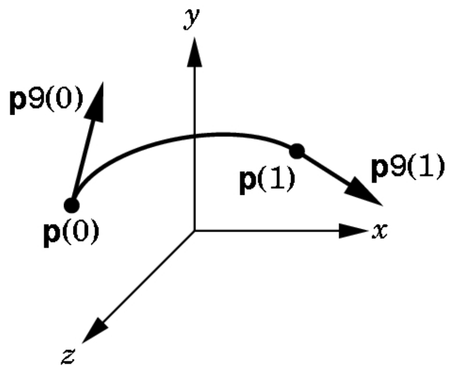
Hermite Form of a Curve define constraints as :

Curve intersects end - points

$$p(0) = p_0 = c_0 \qquad p(1) = p_3 = c_0 + c_1 + c_2 + c_3$$

Constrain the tangent at the end - points

$$p_u(0) = c_1 \qquad p_u(1) = c_1 + 2c_2 + 3c_3$$



In matrix form :

$$q = \begin{bmatrix} \mathrm{p}(0) \\ \mathrm{p}(1) \\ \mathrm{p_u}(0) \\ \mathrm{p_u}(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} c$$

Solve equations to find:

$c = M_H q$ Gives 'Hermite geometry' matrix $M_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$

Resulting polynomial is given by:

$$p(u) = u^T M_H q$$

This can be represented as a set of blending functions on the points:

$$p(u) = b(u)^T q$$

$$b(u) = M_H^T u = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix}$$

The four blending functions have none of their zero's in [0,1]

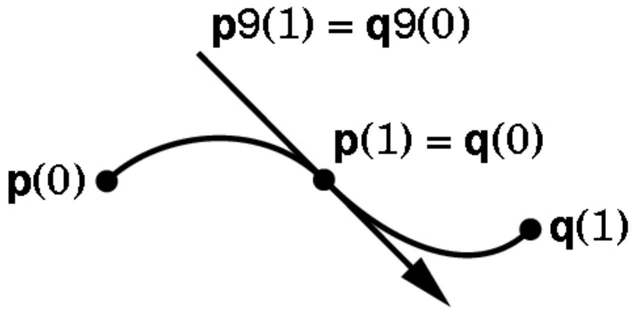
- smoother than interpolating blending function

Hermite polynomials can be used to represent a curve with continuous derivates
- such that the end point of one curve has the same derivative as the start point
  of the adjacent curve

$$p(1) = q(0)$$

$$p_u(1) = q_u(0)$$

where p(u) and q(u) are adjacent section of the curve with u = [0,1] for both
giving a C^1 continous curve



This overcomes the problem with interpolating cubics
where the end-points were only continuous in position

Parametric Cubic Polynomial Curves

Cubic polynomial curves are widely used:

$$p(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3 = \sum_{k=0}^{3} c_k u^k = u^T c$$

$$c = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad u = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \qquad c_k = \begin{bmatrix} c_{kx} \\ c_{ky} \\ c_{kz} \end{bmatrix}$$

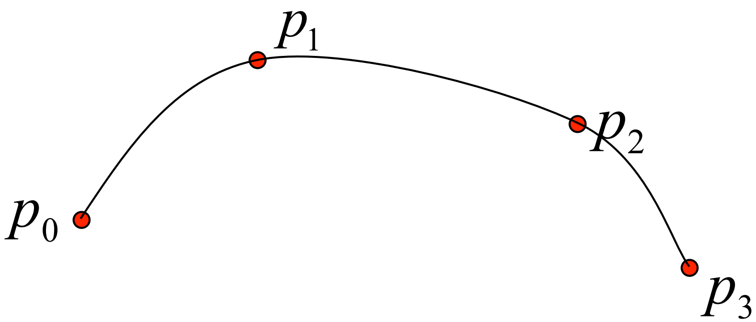
12 equations in 12 unknowns c

Want methods of deriving parameters c for a desired curve!

Cubic Polynomial Interpolation

Given a set of 4 - points (ie 12dof) derive curve that interpolates between them and exactly passes through them:

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad i = 0 \ldots 3$$



What are the coefficients c such that the curve $p(u) = u^T c$ interpolates the points p_i

Let the points be at equally spaced intervals along the curve $u = 0, \frac{1}{3}, \frac{2}{3}, 1$

This gives the four conditions:

$$p_0 = p(0) = c_0$$

$$p_1 = p(\tfrac{1}{3}) = c_0 + \tfrac{1}{3} c_1 + \left(\tfrac{1}{3}\right)^2 c_2 + \left(\tfrac{1}{3}\right)^3 c_3$$

$$p_2 = p(\tfrac{2}{3}) = c_0 + \tfrac{2}{3} c_1 + \left(\tfrac{2}{3}\right)^2 c_2 + \left(\tfrac{2}{3}\right)^3 c_3$$

$$p_3 = p(1) = c_0 + c_1 + c_2 + c_3$$

Hermite Curves and Surfaces

Rather than interpolating points we interpolate between endpoints + tangents at end points

- ensures continuity between curve/surface segments

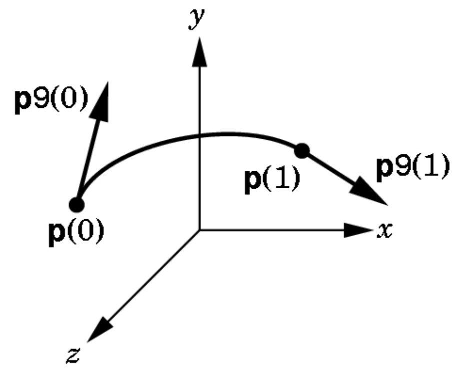
Hermite Form of a Curve define constraints as :

Curve intersects end - points

$$p(0) = p_0 = c_0 \qquad p(1) = p_3 = c_0 + c_1 + c_2 + c_3$$

Constrain the tangent at the end - points

$$p_u(0) = c_1 \qquad p_u(1) = c_1 + 2c_2 + 3c_3$$



In matrix form :

$$q = \begin{bmatrix} p(0) \\ p(1) \\ p_u(0) \\ p_u(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} c$$

Solve equations to find:

$$c = M_H q \quad \text{Gives 'Hermite geometry' matrix} \quad M_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

Resulting polynomial is given by:

$$p(u) = u^T M_H q$$

This can be represented as a set of blending functions on the points:

$$p(u) = b(u)^T q$$

$$b(u) = M_H^T u = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix}$$

The four blending functions have none of there zero's in [0,1]

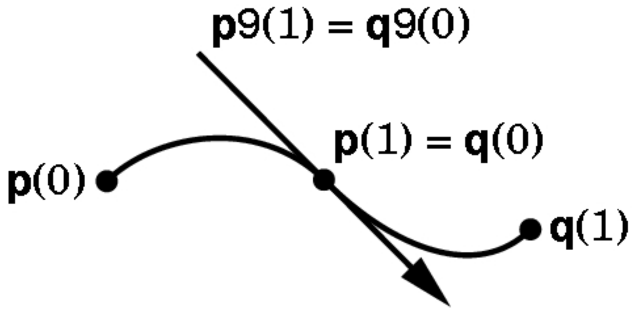
- smoother than interpolating blending function

Hermite polynomials can be used to represent a curve with continous derivates
- such that the end point of one curve has the same derivative as the start point of the adjacent curve

$$p(1) = q(0)$$

$$p_u(1) = q_u(0)$$

where p(u) and q(u) are adjacent section of the curve with u = [0,1] for both giving a C^1 continous curve



This overcomes the problem with interpolating cubics where the end-points were only continuous in position

Hermite surface patch :

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u) b_j(u) q_{ij}$$

is defined to interpolate the 4 corner points and their derivatives

At corner (0,0) we define :

$$p(0,0) = c_{00} \quad p_u(0,0) = c_{01} \quad p_v(0,0) = c_{10} \quad p_{uv}(0,0) = c_{11}$$

Solving gives a surface patch which is continous in position and 1st derivative between adjacent patches.

Therefore, Hermite surface patch has advantages over the direct interpolation.

Derivatives can be defined from the input control points

ie $p_u = p_{00} - p_{01}$

Hermite surface patch :

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u) b_j(u) q_{ij}$$

is defined to interpolate the 4 corner points and their derivatives

At corner (0,0) we define :

$$p(0,0) = c_{00} \quad p_u(0,0) = c_{01} \quad p_v(0,0) = c_{10} \quad p_{uv}(0,0) = c_{11}$$

Solving gives a surface patch which is continous in position and 1st derivative between adjacent patches.

Therefore, Hermite surface patch has advantages over the direct interpolation.

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} u^i v^j c_{ij} = u^T C v$$

$$C = \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \qquad u = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \quad v = \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

$$c_{ij} = \begin{bmatrix} c_{xij} & c_{yij} & c_{zij} \end{bmatrix}$$

Bezier Curves and Surfaces

Interpolating - interpolate 4 points along the curve
Hermite - interpolate 2 points (start/end) + derivatives 2 derivatives

Can use the 4 control points of the interpolating curve to define the derivatives in the Hermite curve: **'Bezier Curves'**

Given 4 control points : p_0, p_1, p_2, p_3

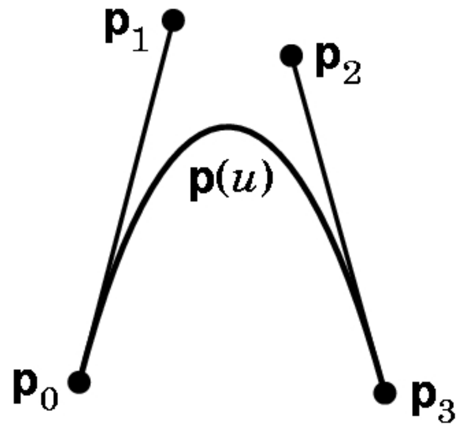
$p(0) = p_0$

$p(1) = p_3$

Bezier defined the derivatives by linear combinations of the control points as :

$$p_u(0) = \frac{p_1 - p_0}{\frac{1}{3}} = 3(p_1 - p_0)$$

$$p_u(1) = \frac{p_3 - p_2}{\frac{1}{3}} = 3(p_3 - p_2)$$



This gives 12 constraints on our cubic polynomial as :

$$p_0 = c_0$$

$$p_3 = c_0 + c_1 + c_2 + c_3$$

$$3(p_1 - p_0) = c_1$$

$$3(p_3 - p_2) = c_1 + 2c_2 + 3c_3$$

As for interpolating and Hermite cases we have 12 equations in 12 unknowns which can be soved to find the cubic parameters :

$$c = M_B p$$

$$M_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & 6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

The resulting cubic Bezier polynomial is :

$$p(u) = u^T M_B p$$

Given a set of control points $p_0 \ldots . p_n$ we interpolate the Bezier curve in sections (as for the interpolating curve) : $\{p_0 \ldots . p_3\}, \{p_3 \ldots . p_6\} \ldots \{p_{n-3} \ldots . p_n\}$

This curve is C^0 as different control points are used on either side of section

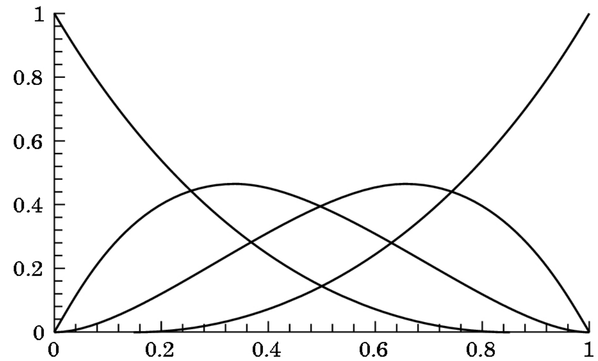
Bezier blending functions :

$$p(u) = b(u)^T p$$

$$b(u) = M_B^T u = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u) \\ u^3 \end{bmatrix}$$

Blending functions are a case of the 'Bernstein Polynomials':

$$b_{kd}(u) = \frac{d!}{k!(d-k)!} u^k (1-u)^{d-k}$$



Properties :

(1) All zeros of the polynomial are at 0 or 1

$$0 < b_{id}(u) \text{ for } 0 \le u \le 1$$

therefore, blending functions are smooth in this interval

(2) Sum of blending function for any u equals 1

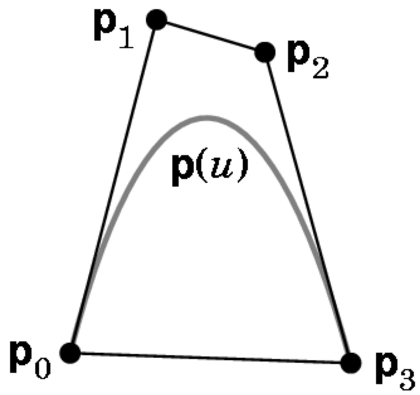
$$\sum_{i=0}^{d} b_{id}(u) = 1$$

therefore, cubic Bezier polynomial is a convex sum

$$p(u) = \sum_{i=0}^{3} b_i(u) p_i$$

All points p(u) must be inside the convex hull of the control points p_i

- Bezier curve is near the control points
- stable for interactive design (small change in control points gives a small change in curve)



Bezier Surface Patches

Bezier surface patch defined by 16 control points (as for interpolation)

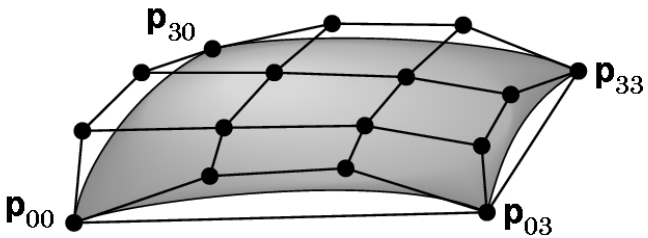
- Patch is constrained to pass through four corners

$$p(0,0) = p_{00} \quad p(1,0) = p_{30} \quad p(0,1) = p_{03} \quad p(1,1) = p_{33}$$

- Partial derivatives at corners are determined from control points

$$\frac{\partial \mathrm{p}(0,0)}{\partial \mathrm{u}} = 3(p_{10} - p_{00}) \qquad \frac{\partial \mathrm{p}(0,0)}{\partial \mathrm{v}} = 3(p_{01} - p_{00})$$

$$\frac{\partial^2 \mathrm{p}(0,0)}{\partial \mathrm{u} \partial \mathrm{v}} = 9(p_{00} - p_{01} + p_{10} - p_{11})$$



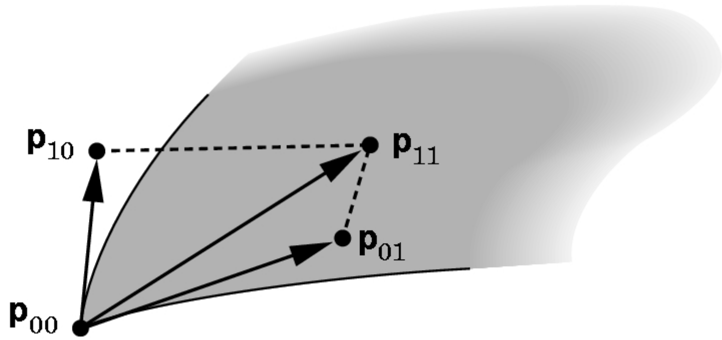
Bezier patch is given by blending functions as :

$$p(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} b_i(u) b_j(v) p_{ij} = u^T M_B P M_B^T v$$

The 2nd order partial derivative with respect to u and v constrains the twist

- tendency to deviate from being flat
- points lie in the same plane if twist is zero

$$(p_{00} - p_{01} + p_{10} - p_{11}) = 0$$



Bezier surface patches provide a means of smooth and intuitive control of surface shape from control points

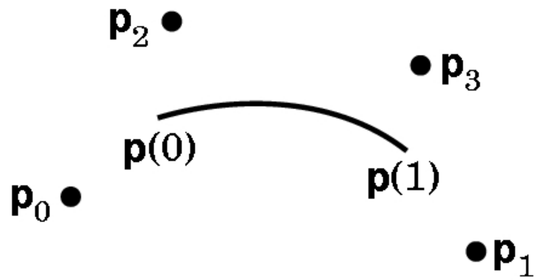
- surface is constrained to lie in convex hull of control points
- C^0 continuity between adjacent patches defined by adjacent control points

How can we define curves/surfaces with higher order continuity between patches

Cubic B-Spline

Ensure joins between patches are continuous
Options:

(1) use higher order polynomials
(2) shorten the interval & use more polynomial segments
(3) use the same control points but don't require the curve to interpolate (pass through) any of the control points



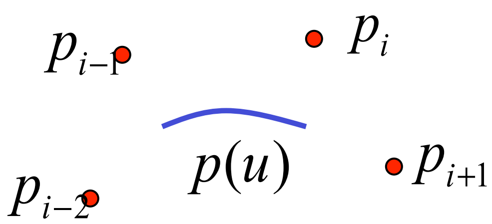
B-splines use option (3) the curve is controlled by sets of 4 control points

- use overlapping sets of control points to achieve continuity between patches

Cubic B-Spline Curves

For a set of control points : $\{p_{i-2}, p_{i-1}, p_i, p_{i+1}\}$

we define the curve *p(u)* between points p_{i-1}, p_i for $0 \le u \le 1$



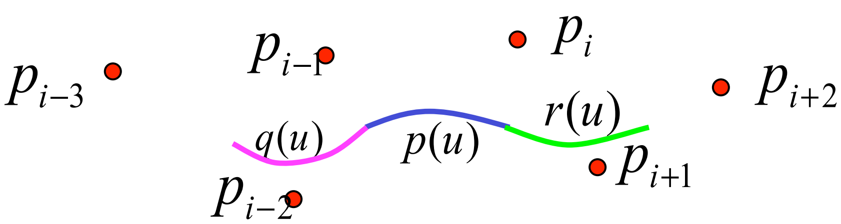
Similarly,

for $\{p_{i-3}, p_{i-2}, p_{i-1}, p_i\}$ we define the curve *q(u)* between points p_{i-2}, p_{i-1} for $0 \le u \le 1$

for $\{p_{i-1}, p_i, p_{i+1}, p_{i+2}\}$ we define the curve *r(u)* between points p_i, p_{i+1} for $0 \le u \le 1$

This provides sufficient degrees of freedom for C^2 contininuity between segments

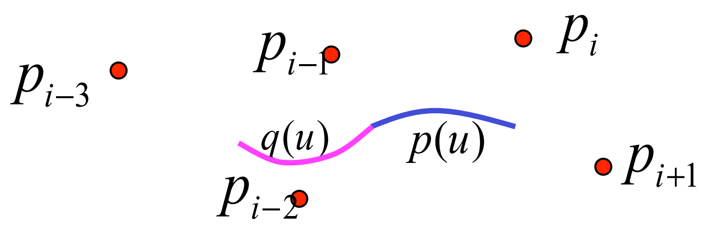
Note : none of the control points are interpolated



Consider the two segment

$q(u)$ controlled by $\{p_{i-3}, p_{i-2}, p_{i-1}, p_i\}$

$p(u)$ controlled by $\{p_{i-2}, p_{i-1}, p_i, p_{i+1}\}$



We have:

$$q(u) = u^T Mq \quad \text{with } q = [p_{i-3} \quad p_{i-2} \quad p_{i-1} \quad p_i]$$

$$p(u) = u^T Mp \quad \text{with } p = [p_{i-2} \quad p_{i-1} \quad p_i \quad p_{i+1}]$$

Could impose constraints

$$p(0) = q(1) \qquad p_u(0) = q_u(1)$$

& derive the corresponding shape matrix

- many possible conditions for relating constraint values to control points

Consider most common B-spline curve definition:

Let:

$$p(0) = q(1) = \tfrac{1}{6}(p_{i-2} + 4p_{i-1} + p_i) \qquad p_u(0) = q_u(1) = \tfrac{1}{2}(p_i - p_{i-2})$$

We have the relation to the coefficient array u :

$$p(u) = u^T c$$

appling the constraints at u = 0 gives

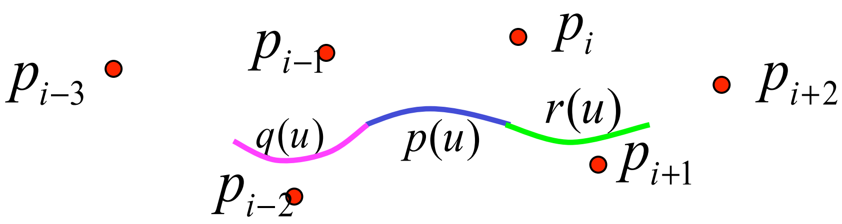
$$c_0 = \tfrac{1}{6}(p_{i-2} + 4p_{i-1} + p_i) \qquad c_1 = \tfrac{1}{2}(p_i - p_{i-2})$$

Applying symmetric constraints at p(1) :

$$\mathrm{p}(1) = \mathrm{r}(0) = c_0 + c_1 + c_2 + c_3 = \tfrac{1}{6}(p_{i-1} + 4p_i + p_{i+1})$$

$$\mathrm{p_u}(1) = \mathrm{r_u}(0) = c_1 + 2c_2 + 3c_3 = \tfrac{1}{2}(p_{i+1} - p_{i-1})$$

This gives the B-spline shape matrix



$$M_S = \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

where

$$p(u) = u^T M_S p$$

The B-spline blending functions

$$p(u) = b(u)^T p$$

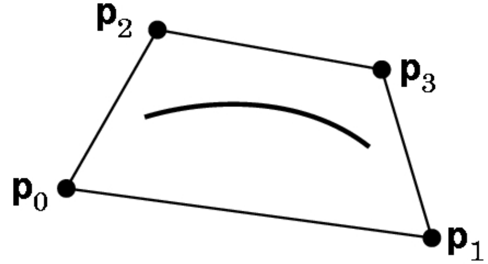
$$b(u) = M_S^T u = \frac{1}{6}\begin{bmatrix} (1-u)^3 \\ 4-6u^2+3u^3 \\ 1+3u+3u^2-3u^3 \\ u^3 \end{bmatrix}$$

As in the case of Bezier curves we have

$$0 \le b_i(u) \le 1 \quad \text{for} \quad 0 \le u \le 1$$

- curve varies slowly over the interval

$$\sum_{i=1}^{3} b_i(u) = 1$$



The blending functions are a convex sum of the points

- curve is always inside the convex hull of the points

The B-spline curve was constrained to be C^1 continuous

- resulting curve is C^2 continuous

$$p_{uu}(0) = q_{uu}(1) \qquad p_{uu}(1) = r_{uu}(0)$$

Due to the C^2 continuity B-spline curves are widely used

- physical processes such as bending of metal are continous in the 2nd derivative
- C^2 continuous curve will appear to be smooth even at the join points

Downside:

For each set of 4 control points we only define the section of the curve between the central control points (1/3 of the Bezier curve)

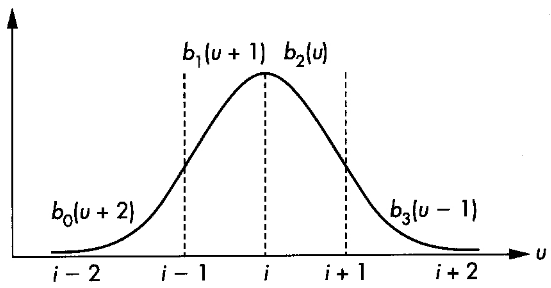
- require 3 times as many control points as Bezier
- requires 3 times as much computation to compute the complete curve for a given set of points

B-Splines and Bases

Each control points p_i contributes to the curve in four adjacent intervals

The total contribution of a single control point can be written as $B_i(u)p_i$

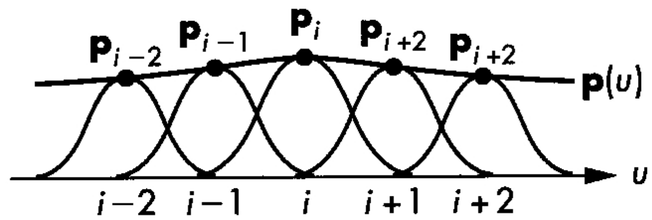
$$B_i(u)=\begin{cases} 0 & u<i-2 \\ b_0(u+2) & i-2\le u\le i-1 \\ b_1(u+1) & i-1\le u\le i \\ b_2(u) & i\le u\le i+1 \\ b_3(u-1) & i+1\le u\le i+2 \\ 0 & i+2\le u \end{cases}$$



Given a set of control points $p_0.....p_n$

The entire spline curve is defined as :

$$p(u)=\sum_{i=1}^{m-1} B_i(u-i)p_i$$



Each function B(u - i) is a shifted version of the same function

- same function forms the basis for the all B - spline curve segments
- curve over the whole interval is a linear combination of basis functions

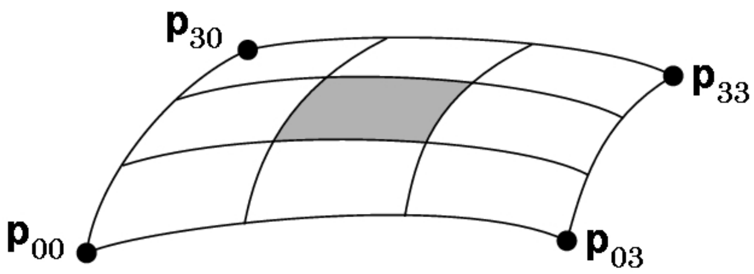
B-Spline Surfaces

Defined as for B-spline curves:

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u) b_j(v) p_{ij}$$

p_{ij} are the 16 control points which define the surface

for the central region $p_{11} - p_{22}$



B-spline surface patch is inside the convex hull of the control points

- C^2 continuity
- smooth control of surface
- appears much smoother than Bezier patch
- Requires 9 times more computation than Bezier

Generalised B-Splines

The generalised approximation problem can be stated as

Given a set of control points $p_0 \ldots. p_m$

find a function $p(u) = [x(u), y(u), z(u)]^T$ over $u_{\min} \le u \le u_{\max}$

that is smooth and close to the control points (in some sense)

Suppose we have a set of 'knots' $\{u_K\}$

$$u_{\min} \le u_0 \le u_1 \ldots\ldots \le u_n \le u_{\max}$$

$[u_0 u_1 \ldots\ldots u_n]$ is the knot array

A general spline is defined as the d order polynomial between the knots

$$p(u) = \sum_{j=0}^{d} c_{jk} u^j \qquad u_k \le u \le u_{k+1}$$

$n(d+1)$ parameters c_{jk}

Continuity between segments is enforced by applying conditions at the knots based on the control points

Example : Cubic splines $d = 3$

$n + 1$ control points $\Rightarrow$ $n - 1$ internal knots $+ 2$ ends

$4n$ parameter coefficients

To ensure C^2 continuity at knots we have $3n - 3$ conditions

$+$ Interpolation of $n + 1$ control $\Rightarrow$ $4n - 2$ conditions

Additional two conditions obtained by constraints on 2 ends ie slope

However, general spline is global solve $4n$ equations in $4n$ unknowns

- no local solution
- difficult to use for computer graphics

For a generalised B-splines the curve is defined as a set of blending or basis functions:

$$p(u) = \sum_{i=0}^{m} B_{id}(u)\, p_i$$

$B_{id}(u)$ is a polynomial of degree d except at the knots over interval $(u_{i\min}, u_{i\max})$ and 0 elsewhere

Note: 'B-Spline' comes from 'Basis Spline' as $B_{id}(u)$ form a basis for the given knot sequences and degree

Many possible choices of basis functions

Want to choose a set which give local smoothness and control

Cox - deBoor recursion :

$$B_{K0}(u) = \begin{cases} 1 & u_k \le u \le u_{k+1} \\ 0 & otherwise \end{cases}$$

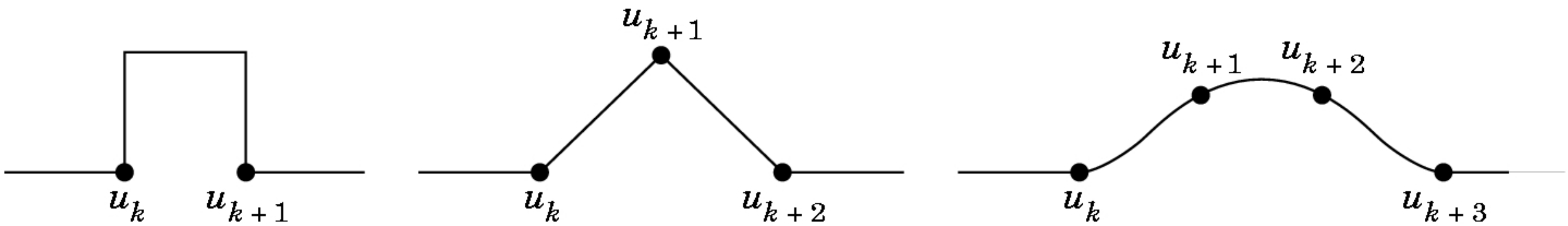
$$B_{kd}(u) = \frac{u - u_k}{u_{k+d} - u_k} B_{K,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d+1} - u_{k+1}} B_{K+1,d-1}(u)$$

B_{K0} is constant over one interval and zero elsewhere

B_{K1} is linear over 2 intervals and zero elsewhere

B_{K2} is quadratic over 3 intervals and zero elsewhere

B_{Kd} is order d polynomial nonzero over d + 1 intervals between $u_k \le u \le u_{k+d+1}$



Generalised B-spline using Cox-deBoor basis functions

- C^{d-1} continuity at knots
- spline is inside the convex hull of the points

$$0 \leq B_{id}(u) \leq 1$$

$$\sum_{i=0}^{m} B_{id}(u) = 1$$

- each control points p_i affects only $d+1$ intervals therefore the curve segment is within the convex hull defined by $d+1$ points

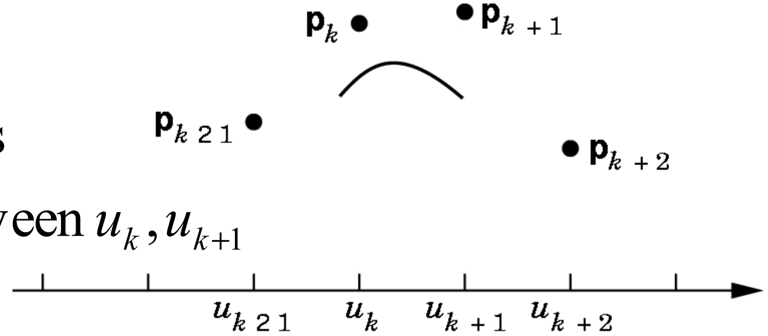
Knot Values

Thus far we have only constrained knot values such that $u_k \leq u_{k+1}$

- if knots are equally spaced we have 'uniform spline'
- greater flexibility can be achieved with non - uniform spacing
- we can have multiple repeated knots $u_k = u_{k+1}$ by defining : $\frac{0}{0} = 1$ in recursion

Uniform B - Splines

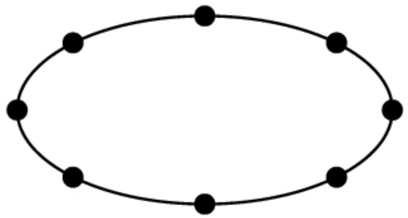
- using 3rd order spline with Cox - deBoor basis
- points $\{p_{k-1}, \ p_k, p_{k+1}, p_{k+2}\}$ control curve between u_k, u_{k+1}



Uniform Periodic B - Spline

- repeat start and end control points to form a closed curve

$$p_0 = p_{m-1} \quad p_1 = p_m$$



Non - Uniform B - Splines

- Repeating knots pulls the spline closer to the control points
  use to introduce discontinuities in the spline
- Repeating knots at the ends forces interpolation of the end points
  a common knot sequence for open splines [0,0,0,0,1,2,...., n - 1, n, n, n, n]
- Knot sequence [0,0,0,0,1,1,1,1] gives the cubic Bezier curve

NURBS: Non-uniform Rational B-Spline

Further generalisation of B - Splines to rational functions

2 additional properties

(1) B - splines are distorted under perspective transforms (not Affine)

NURBS ensure curves/surfaces are handled correctly under perspective

(2) Quadrics (elipse, circle....) can only be approximated by B - splines

Quadrics are a special case of Quadratic NURBS

Represent p(u) in homogenous coordinates where

The weighted homogenous coordinate for a control point is

$$q_i = w_i \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

Use weights to increase/decrease the importance of a control point

In homogenous coordinates the spline is defined by four functions for the first three components we have a set of basis functions with weighted control points

$$q(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \\ w(u) \end{bmatrix} = \sum_{i=0}^{n} B_{id}(u) q_i = \sum_{i=0}^{n} B_{id}(u) w_i \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

We transform q(u) to p(u) by perspective division with the function w(u)

$$p(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \frac{1}{w(u)} q(u) = \frac{\sum_{i=0}^{n} B_{id}(u) w_i p_i}{\sum_{i=0}^{n} B_{id}(u) w_i}$$

Each component of p(u) is a rational function

Perspective division results in a representation which can obtain the same curve/surface under perspective viewing conditions

The knot points are not restricted in any way

'Non - uniform Rational B - splines' NURBS

Summary

Derived a set of curve/surface representation who's shape
is controlled by a set of control points:

Cubic curves

- Interpolating: pass through control points (rough)
- Hermite: interpolate end-points+end-point derivatives (smooth)
- Bezier: special case of Hermite defined from control points

All have problems of continuity between adjacent segments

Cubic B-spline curves:

- continuity between adjacent segments
- 4-control points define central part of curve
- gives C^2 continuity
- represent as a set of basis functions acting on control points

NURBS: Non-uniform rational B-splines

- preserve shape under perspective transforms
- widely used in CAD/graphics