Topic 3: Geometric Modeling and Computer Graphics – Curves and Surfaces

Jessica Zhang
Department of Mechanical Engineering
Courtesy Appointment in Biomedical Engineering
Carnegie Mellon University
jessicaz@andrew.cmu.edu
http://www.andrew.cmu.edu/user/jessicaz

Representation of Curves and Surfaces

- Three major types of object representation:
  - Explicit $y = f(x)$
  - Implicit $f(x, y) = 0$
  - Parametric

$$x = x(u)$$
$$y = y(u)$$
$$z = z(u)$$

Explicit Representation

- The explicit form of a curve in 2D gives the value of one variable, the dependent variable, in terms of the other, the independent variable. In x, y space, we might write

$$y = f(x)$$

- We may also be able to express x as a function of y

$$x = g(y)$$

Explicit Representation

- The equation of a line can be written as

$$y = mx + b$$

in terms of its slope m and y-intercept b, even though this equation doesn't hold for vertical lines.

- Consider a circle of radius r centered at the origin. A circle has constant **curvature** – a measure of how rapidly a curve is bending at a point. Using an explicit representation, we can only write one equation for half of it,

$$y = \sqrt{r^2 - x^2} \qquad 0 \leq |x| \leq r$$

The other half can be written as

$$y = -\sqrt{r^2 - x^2} \qquad 0 \leq |x| \leq r$$

Explicit Representations in 3D

- In 3D, the explicit representation of a curve requires two equations. For example, if x is again the independent variable, we have two dependent variables, y and z.

$$y = f(x)$$

$$z = g(x)$$

- A surface requires two independent variables

$$z = f(x, y)$$

- A curve or surface may not have an explicit representation. For example, the equations

$$y = ax + b$$

$$z = cx + d,$$

describe a line in 3D, but these equations can't represent a line in a plane of constant x. Likewise, a surface represented by an equation of the form $z = f(x, y)$ can't represent a sphere, because a given x and y can generate zero, one or two points on the sphere.

Implicit Representation

- Most of the curves and surfaces with which we work have implicit representation. In 2D, an implicit curve can be represented by the equation:

$$f(x, y) = 0.$$

- Our two examples – the line and the circle centered at the origin – have the respective representations

$$ax + by + c = 0,$$

$$x^2 + y^2 - r^2 = 0.$$

- The function *f*, is really a testing, or membership, function that divides space into those points that belong to the curve and those that do not.
- The implicit form is less coordinate-system-dependent than is the explicit form, however, in that it does represent all lines and circles.

Implicit Representations in 3D

- In 3D, the implicit form

$$f(x, y, z) = 0$$

describes a surface. For example, any plane can be written as

$$ax + by + cz + d = 0$$

for constants a, b, c, and d. A sphere of radius r centered at the origin can be described by

$$x^2 + y^2 + z^2 - r^2 = 0$$

- Curves in 3D are not as easily represented in implicit form. We can represent a curve as the intersection, if it exists, of the two surfaces:

$$f(x, y, z) = 0,$$
$$g(x, y, z) = 0.$$

Algebraic Surfaces

- Algebraic surfaces are those for which the function $f(x, y, z)$ is the sum of polynomials in the three variables.

- Of particular importance are the quadric surfaces, where each term in f can have degree up to 2. Quadric surfaces can represent useful objects (such as spheres, disks and cones), and those objects intersect with lines/curves.

Parametric Form

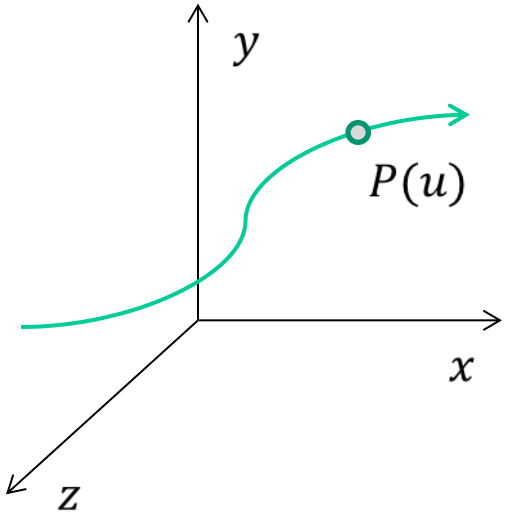
- The parametric form of a curve expresses the value of each spatial variable for points on the curve in terms of an independent variable, u: the parameter. In 3D, we have three explicit functions:

$$x = x(u),$$
$$y = y(u),$$
$$z = z(u).$$

- One of the advantages of the parametric form is that it is the same in 2D and 3D.

Parametric Form

- A useful interpretation of the parametric form is to visualize the locus of points $\mathbf{p}(u) = [\, x(u) \quad y(u) \quad z(u) \,]^T$ being drawn as u varies.
- The derivative can be thought as the velocity with which the curve is traced out and points in the direction tangent to the curve.



Parametric Curve

$$\frac{d\mathbf{p}(u)}{du} = \begin{bmatrix} \frac{dx(u)}{du} \\ \frac{dy(u)}{du} \\ \frac{dz(u)}{du} \end{bmatrix}$$

Parametric Form

- Parametric surfaces require two parameters. We can describe a surface by three equations of the form

$$x = x(u, v),$$
$$y = y(u, v),$$
$$z = z(u, v),$$

or we can use the column matrix

$$\mathbf{p}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix}.$$

As u and v vary over some interval, we generate all the points $\mathrm{p}(u, v)$ on the surface.

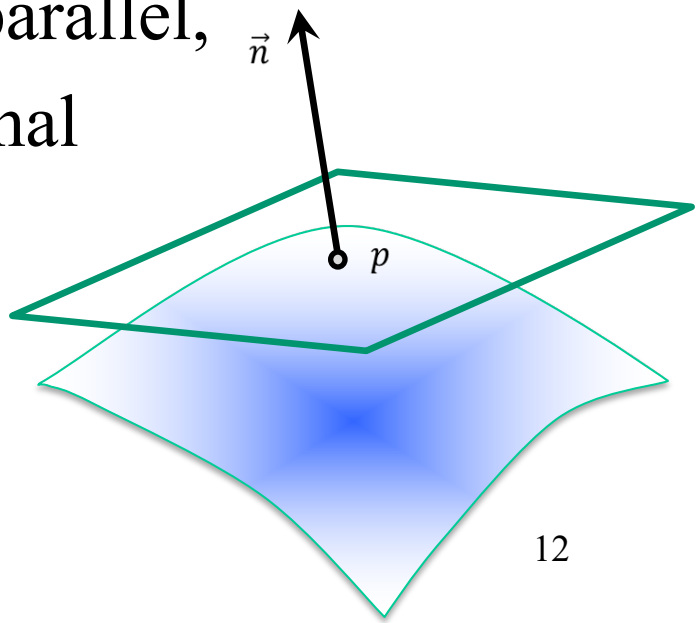
Parametric Form

- The derivative vectors determine the tangent plane at each point on the surface.

$$\frac{\partial \mathbf{p}}{\partial u} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial u} \\ \frac{\partial y(u,v)}{\partial u} \\ \frac{\partial z(u,v)}{\partial u} \end{bmatrix} \qquad \frac{\partial \mathbf{p}}{\partial v} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial v} \\ \frac{\partial y(u,v)}{\partial v} \\ \frac{\partial z(u,v)}{\partial v} \end{bmatrix}$$

- As long as these vectors are not parallel, their cross product gives the normal at each point

$$\mathbf{n} = \partial \mathbf{p}/\partial u \times \partial \mathbf{p}/\partial v.$$



Tangent plane and normal at a point on a parametric surface.

Parametric Form

- The parametric form of curves and surfaces is the most flexible and robust for computer graphics.
- We could still argue that we have not fully removed all dependencies on a particular coordinate system or frame, because we are still using the x, y, and z for a particular representation.
- It is possible to develop a system solely on the basis of $\mathrm{p}(u)$ for curves and $\mathrm{p}(u, v)$ for surfaces, however, the frame changes for each point on the curve.

Parametric Polynomial Curves

- Parametric forms are not unique. A given curve or surface can be represented in many ways.
- The parametric forms in which the functions are polynomials in u for curves, and polynomials in u and v for surfaces, are of most use in computer graphics.
- Consider a curve of the form

$$\mathbf{p}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}.$$

A polynomial parametric curve of degree n is of the form

$$\mathbf{p}(u) = \sum_{k=0}^{n} u^k \mathbf{c}_k,$$

Parametric Polynomial Curves

- Each c_k has independent x, y, and z components.

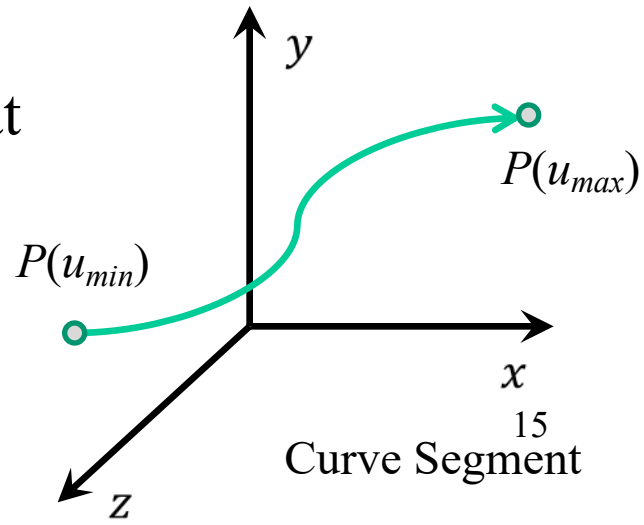
$$\mathbf{c}_k = \begin{bmatrix} c_{xk} \\ c_{yk} \\ c_{zk} \end{bmatrix}$$

- The n+1 column matrices $\{\mathbf{c}_k\}$ are the coefficients of p. They give us 3(n+1) degrees of freedom in how we choose the coefficients of a particular p.
- The curves are defined for any range interval of u

$$u_{\min} \leq u \leq u_{\max}$$

With no loss of generality, we assume that

$0 \leq u \leq 1$.



Curve Segment

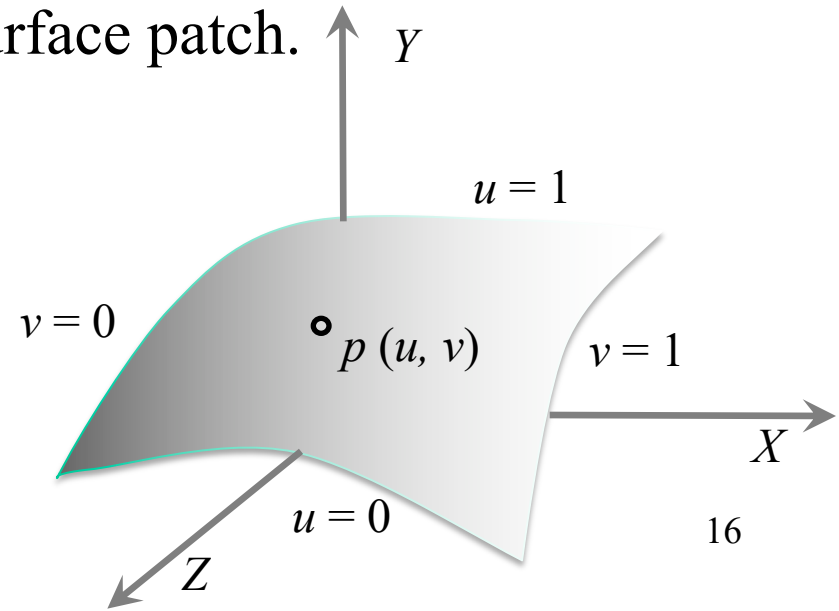
Parametric Polynomial Surfaces

- We can define a parametric polynomial surface as

$$\mathbf{p}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix} = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{c}_{ij} u^i v^j .$$

We must specify $3(n+1)(m+1)$ coefficients to determine a particular surface $\mathbf{p}(u, v)$.

- We shall always take $n = m$, and let u and v vary over the rectangle $0 \le u, v \le 1$, defining a surface patch.



Parametric Polynomial Surfaces

- Any surface patch can be viewed as the limit of a collection of curves that we generate by holding either u or v constant and varying the other.

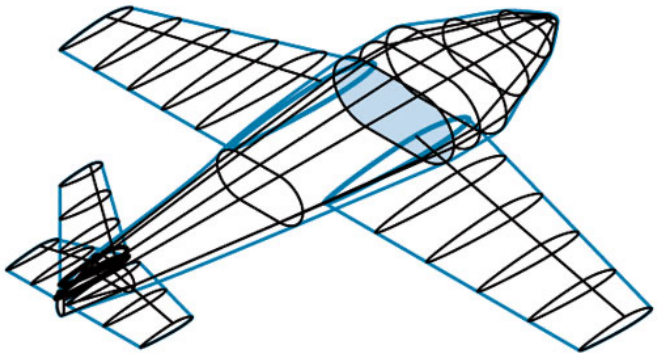
- Our strategy shall be to define parametric polynomial curves and to use the curves to generate surfaces with similar characteristics.

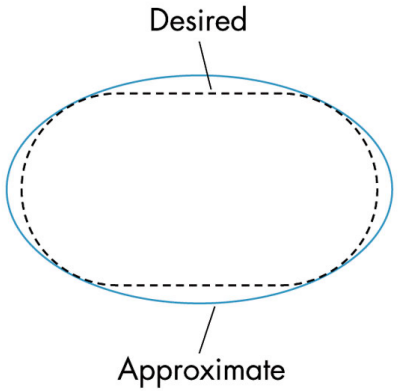
Design Criteria

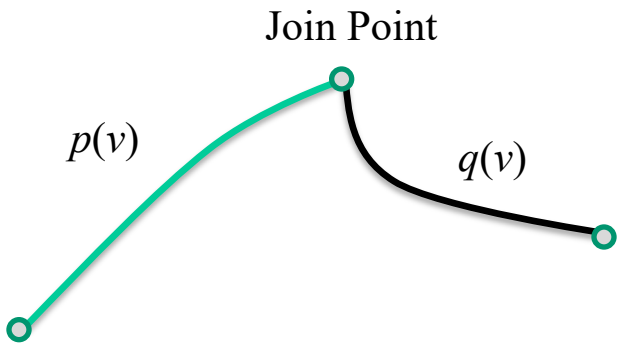
- There are many considerations that determine why we prefer to use parametric polynomials of low degree in computer graphics and computer-aided design:
  - Local control of shape
  - Smoothness and continuity
  - Ability to evaluate derivatives
  - Stability
  - Ease of rendering

Design Criteria

- Suppose that we want to build a model airplane, using flexible strips of wood for the structure. We can build the body of the model by constructing a set of cross sections, and then connecting them with longer pieces.
- To design our cross sections, we might start with a picture of a real airplane, or sketch a desired curve. In practice, we probably will make our cross section out of a number of wood strips, each of which will become a curve segment for the cross section.
- Each segment should be smooth, we also want a degree of smoothness where the segments meet at joint points.







Design Criteria

- Although we might be able to ensure that a curve segment is smooth, we have to be particularly careful at the joint points.
- The usual definition of smoothness is given in terms of the derivatives along the curve. It should be clear that, for a polynomial curve

$$\mathbf{p}(u) = \sum_{k=0}^{3} \mathbf{c}_k u^k$$

all derivatives exist and can be computed analytically.

Consequently, the only places where we can encounter continuity difficulties are at the joint points.

Design Criteria

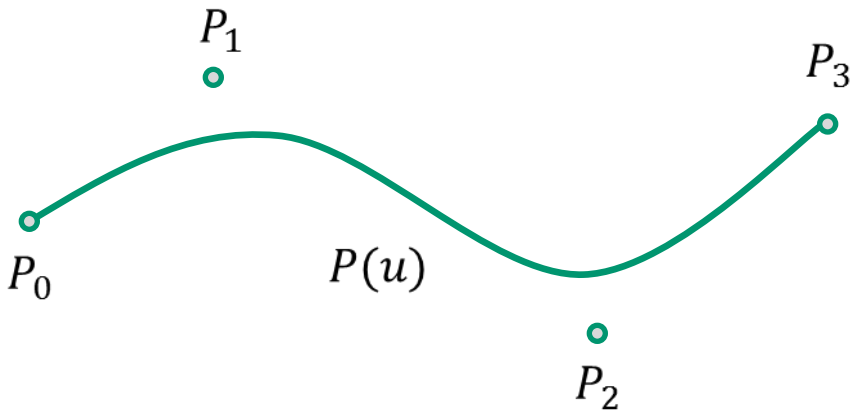
- We would like to design each segment individually, rather than designing all the segments by a single global calculation.
- When we make a change, this change will affect the shape in only the area where we are working. This sort of local control is one aspect of a more general stability principle:

Small changes in the values of input parameters should cause only small changes in output variables.

Small changes in independent variables should cause only small changes in dependent variables.

Design Criteria

- More likely, we would consider data at a small number of control or data points, and would use only those data to design our shape.
- The curve passes through, or interpolates, some of the control points, but only comes close to others.
- We are usually satisfied if the curve passes close to the control-point data, as long as it is smooth.



Curve segment and control points

Parametric Cubic Polynomial Curves

- Once we have decided to use parametric polynomial curves, we must choose the degree of the curve.
- If we choose a high degree, we will have many parameters, and evaluation of points on the curve will be costly.
- If we pick too low a degree, we may not have enough parameters with which to work.
- Most designers, at least initially, work with cubic polynomial curves.

Parametric Cubic Polynomial Curves

- A cubic parametric polynomial

$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3 = \sum_{k=0}^{3} \mathbf{c}_k u^k = \mathbf{u}^T \mathbf{c},$$

where

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}, \qquad c_k = \begin{bmatrix} c_{kx} \\ c_{ky} \\ c_{kz} \end{bmatrix}.$$

- c is a column matrix containing the coefficients of the polynomial. It is what we wish to determine from the control-point data.
- We seek to find 12 equations in 12 unknowns for each type, but, because x, y, and z are independent, we can group these equations into three independent sets of four equations in four unknowns.

Parametric Cubic Polynomial Curves

- The design of a particular type of cubic will be based on data given at some values of the parameter u.
  - Satisfy interpolating conditions in which the polynomial must agree with the data at some points.
  - Require the polynomial to interpolate some derivatives at certain values of the parameter.
  - Satisfy smoothness conditions that enforce various continuity conditions at the join points that are shared by two curve segments.
  - Satisfy conditions that are not as strict, requiring only that the curve passes close to several known data points. The same data can define more than one single curve.

Interpolation

- Cubic interpolating polynomial: suppose that we have four control points in 3D: $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$.

$$\mathbf{p}_k = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix}$$

- We seek the coefficients c such that the polynomial $\mathbf{p}(u) = \mathbf{u}^T\mathbf{c}$ passes through, or interpolates, the four control points.
- We have four 3D interpolating points, so we have 12 conditions and 12 unknowns.
- Generally we choose the equally spaced values

$$u = 0, \frac{1}{3}, \frac{2}{3}, 1$$

Interpolation

- The four conditions are:

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

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

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

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

We can write these equations in matrix form as

$$\mathbf{p} = \mathbf{A}\mathbf{c},$$

where

$$\mathbf{p} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \\ 1 & \frac{2}{3} & \left(\frac{2}{3}\right)^2 & \left(\frac{2}{3}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

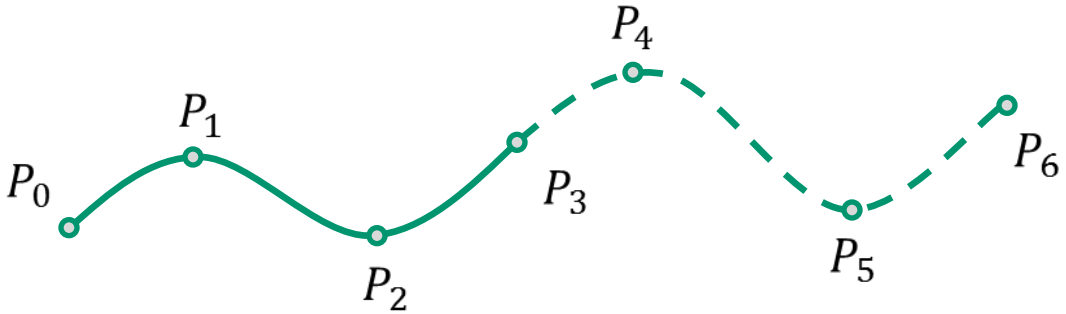
Interpolation

- A is nonsingular, and we invert it to obtain the interpolating geometry matrix

$$\mathbf{M}_I = \mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -5.5 & 9 & -4.5 & 1 \\ 9 & -22.5 & 18 & -4.5 \\ -4.5 & 13.5 & -13.5 & 4.5 \end{bmatrix}$$

$$\mathbf{c} = \mathbf{M}_I \mathbf{p}$$

- Suppose that we have a sequence of control points. Rather than defining a single interpolating curve of degree *m* for all the points, we can define a set of cubic interpolating curves, each defined by a group of four control points. Using this method, we can achieve continuity for the sequence of segments, but derivatives at the join points may not be continuous.



Joining of interpolating segments

Blending Functions

- We can substitute the interpolating coefficients into our polynomial, then we obtain

$$\mathbf{p}(u) = \mathbf{u}^T\mathbf{c} = \mathbf{u}^T\mathbf{M}_I\mathbf{p}$$

which we can write as

$$\mathbf{p}(u) = \mathbf{b}(u)^T\mathbf{p} \qquad \text{where} \qquad \mathbf{b}(u) = \mathbf{M}_I^T\mathbf{u}$$

b(u) is a column matrix of the four **blending polynomials**

$$\mathbf{b}(u) = \begin{bmatrix} b_0(u) \\ b_1(u) \\ b_2(u) \\ b_3(u) \end{bmatrix}$$

- Each blending polynomial is a cubic. If we express p(u) in terms of these blending polynomials as

$$\mathbf{p}(u) = b_0(u)\mathbf{p}_0 + b_1(u)\mathbf{p}_1 + b_2(u)\mathbf{p}_2 + b_3(u)\mathbf{p}_3 = \sum_{i=0}^{3} b_i(u)\mathbf{p}_i$$

Blending Functions

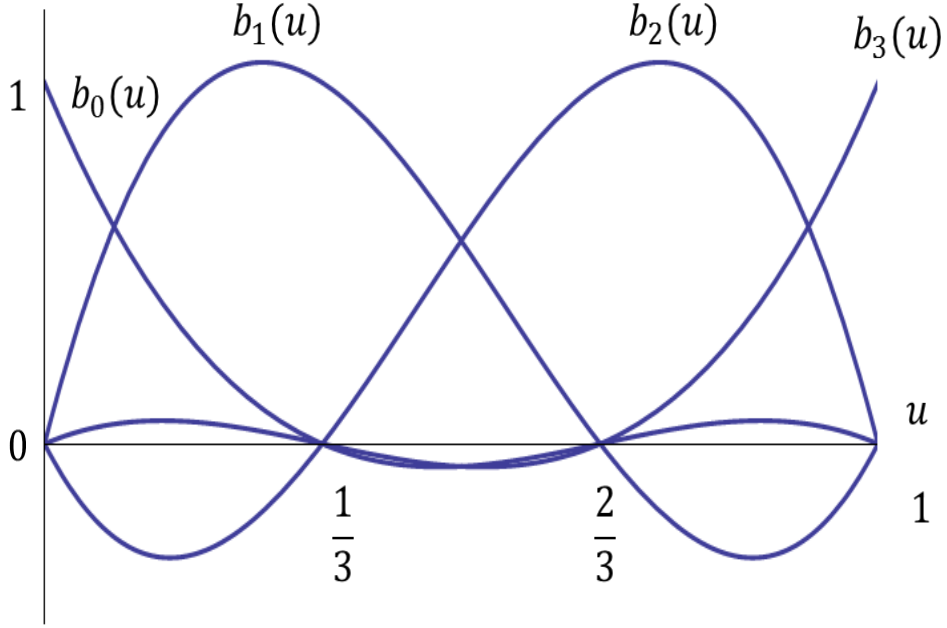
- The polynomials blend together the individual contributions of each control point and enable us to see the effect of a given control point on the entire curve.

$$b_0(u) = -\frac{9}{2}\left(u - \frac{1}{3}\right)\left(u - \frac{2}{3}\right)(u - 1),$$

$$b_1(u) = \frac{27}{2}u\left(u - \frac{2}{3}\right)(u - 1),$$

$$b_2(u) = -\frac{27}{2}u\left(u - \frac{1}{3}\right)(u - 1),$$

$$b_3(u) = \frac{9}{2}u\left(u - \frac{1}{3}\right)\left(u - \frac{2}{3}\right).$$



- The blending functions lack the derivative continuity at the join points.

The Cubic Interpolating Patch

- There is a natural extension of the interpolating curve to an interpolating patch. A bicubic surface patch can be written as

$$\mathbf{p}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} u^i v^j \mathbf{c}_{ij}$$

where $\mathbf{c}_{ij}$ is a three-element column matrix of the x, y, and z coefficients for the ijth term in the polynomial. If we define a 4×4 matrix whose elements are three-element column matrices,

$$\mathbf{C} = [\, \mathbf{c}_{ij} \,]$$

then we can write the surface patch as

$$\mathbf{p}(u, v) = \mathbf{u}^T \mathbf{C} \mathbf{v},$$

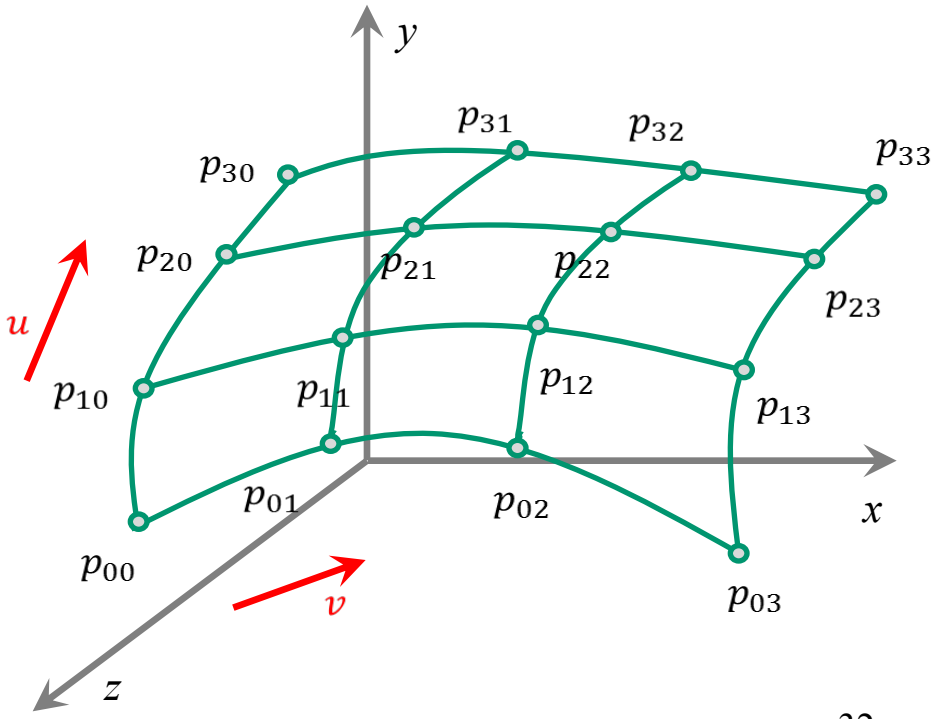
$$\mathbf{v} = \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

A particular bicubic polynomial patch is defined by the 48 elements of C—16 three-element vectors.

The Cubic Interpolating Patch

- Suppose that we have 16 3D control points $\mathbf{p}_{ij}$, $i = 0, \ldots, 3$, $j = 0, \ldots, 3$. We can use these points to define an interpolating surface patch. If we assume that these data are used for interpolation at the equally spaced values of both u and v of $0, \frac{1}{3}, \frac{2}{3}$, and 1, then we get three sets of 16 equations in 16 unknowns. For example, for $u = v = 0$, we get the three independent equations

$$\mathbf{p}_{00} = [\,1 \quad 0 \quad 0 \quad 0\,]\,\mathbf{C}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \mathbf{c}_{00}$$



Interpolating surface patch

The Cubic Interpolating Patch

Rather than writing down and solving all these equations, we can proceed in a more direct fashion. If we consider $v = 0$, we get a curve in u that must interpolate $\mathbf{p}_{00}$, $\mathbf{p}_{10}$, $\mathbf{p}_{20}$, and $\mathbf{p}_{30}$. Using our results on interpolating curves, we write this curve as

$$\mathbf{p}(u, 0) = \mathbf{u}^T \mathbf{M}_I \begin{bmatrix} \mathbf{p}_{00} \\ \mathbf{p}_{10} \\ \mathbf{p}_{20} \\ \mathbf{p}_{30} \end{bmatrix} = \mathbf{u}^T \mathbf{C} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Likewise, the values of $v = \frac{1}{3}, \frac{2}{3}, 1$ define three other interpolating curves, each of which has a similar form. Putting these curves together, we can write all 16 equations as

$$\mathbf{u}^T \mathbf{M}_I \mathbf{P} = \mathbf{u}^T \mathbf{C} \mathbf{A}^T,$$

where $\mathbf{A}$ is the inverse of $\mathbf{M}_I$. We can solve this equation for the desired coefficient matrix

$$\mathbf{C} = \mathbf{M}_I \mathbf{P} \mathbf{M}_I^T,$$

and, substituting into the equation for the surface, we have

$$\mathbf{p}(u, v) = \mathbf{u}^T \mathbf{M}_I \mathbf{P} \mathbf{M}_I^T \mathbf{v}.$$

The Cubic Interpolating Patch

We can interpret this result in several ways. First, the interpolating surface can be derived from our understanding of interpolating curves—a technique that will enable us to extend other types of curves to surfaces. Second, we can extend our use of blending polynomials to surfaces. By noting that $\mathbf{M}_I^T\mathbf{u}$ describes the interpolating blending functions, we can rewrite our surface patch as

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

Each term $b_i(u)b_j(v)$ describes a **blending patch**. We form a surface by blending together 16 simple patches, each weighted by the data at a control point. The basic properties of the blending patches are determined by the same blending polynomials that arose for interpolating curves; thus, most of the characteristics of surfaces are similar to those of the curves. In particular, the blending patches are not particularly smooth, because the zeros of the functions $b_i(u)b_j(v)$ lie inside the unit square in u, v space. Surfaces formed from curves using this technique are known as **tensor-product surfaces**. Bicubic tensor-product surfaces are a subset of all surface patches that contain up to cubic terms in both parameters. They are an example of **separable surfaces**, which can be written as

$$\mathbf{p}(u, v) = \mathbf{f}(\mathbf{u})\mathbf{g}(v)$$

where *f* and *g* are suitably chosen row and column matrices, respectively.

Hermite Curves and Surfaces

Suppose that we start with only the control points $\mathbf{p}_0$ and $\mathbf{p}_3$, and again insist that our curve interpolate these points at the parameter values $u = 0$ and $u = 1$, respectively. Using our previous notation, we have the two conditions

$$\mathbf{p}(0) = \mathbf{p}_0 = \mathbf{c}_0,$$

$$\mathbf{p}(1) = \mathbf{p}_3 = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3.$$

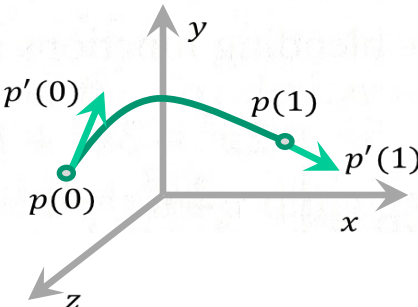
We can get two other conditions if we assume that we know the derivatives of the function at $u = 0$ and $u = 1$. The derivative of the polynomial is simply the parametric quadratic polynomial

$$\mathbf{p}'(u) = \begin{bmatrix} \frac{dx}{du} \\ \frac{dy}{du} \\ \frac{dz}{du} \end{bmatrix} = \mathbf{c}_1 + 2u\mathbf{c}_2 + 3u^2\mathbf{c}_3.$$

If we denote the given values of the two derivatives as $\mathbf{p}'_0$ and $\mathbf{p}'_3$, then our two additional conditions

$$\mathbf{p}'_0 = \mathbf{p}'(0) = \mathbf{c}_1,$$

$$\mathbf{p}'_3 = \mathbf{p}'(1) = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3.$$



Definition of the Hermite cubic

Hermite Curves and Surfaces

We can write these equations in matrix form as

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_3 \\ \mathbf{p}'_0 \\ \mathbf{p}'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c}.$$

Letting $\mathbf{q}$ denote the data matrix

$$\mathbf{q} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_3 \\ \mathbf{p}'_0 \\ \mathbf{p}'_3 \end{bmatrix},$$

we can solve the equations to find

$$\mathbf{c} = \mathbf{M}_H \mathbf{q},$$

where $\mathbf{M}_H$ is the **Hermite geometry** matrix

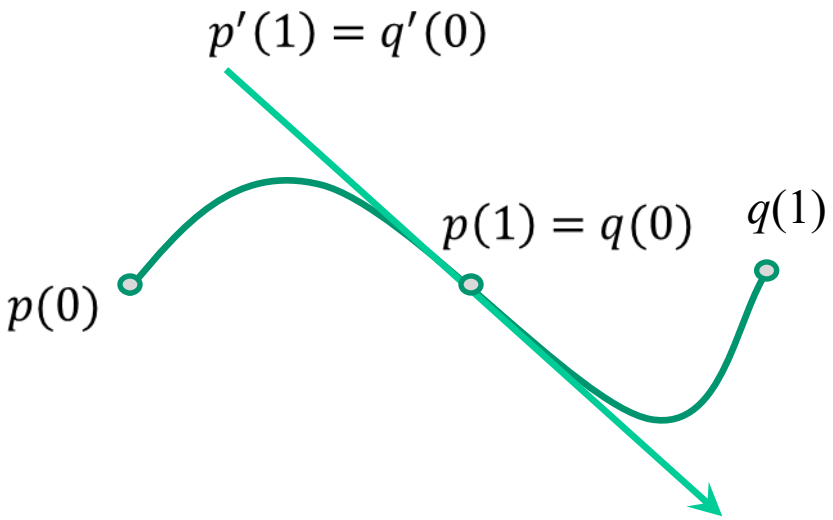
$$\mathbf{M}_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}.$$

The resulting polynomial is given by

$$\mathbf{p}(u) = \mathbf{u}^T \mathbf{M}_H \mathbf{q}.$$

Hermite Curves and Surfaces

- Both the interpolated value and the derivative are shared by the curve segments on the two sides of a join point, thus both the resulting function and the first derivative are continuous over all segments.



Hermite form at join point

Hermite Curves and Surfaces

- We can get a more accurate idea of the increased smoothness of the Hermite form by rewriting the polynomial in the form

$$\mathbf{p}(u) = \mathbf{b}(u)^T \mathbf{q},$$

where the new blending functions are given by

$$\mathbf{b}(u) = \mathbf{M}_H^T \mathbf{u} = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix}.$$

- These four polynomials have none of their zeros inside the interval (0, 1) and are much smoother than are the interpolating polynomial blending functions.

Hermite Curves and Surfaces

- We can go on and define a bicubic Hermite surface patch through these blending functions

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

where $\mathbf{Q} = [\mathbf{q}_{ij}]$ is the extension of q to surface data.

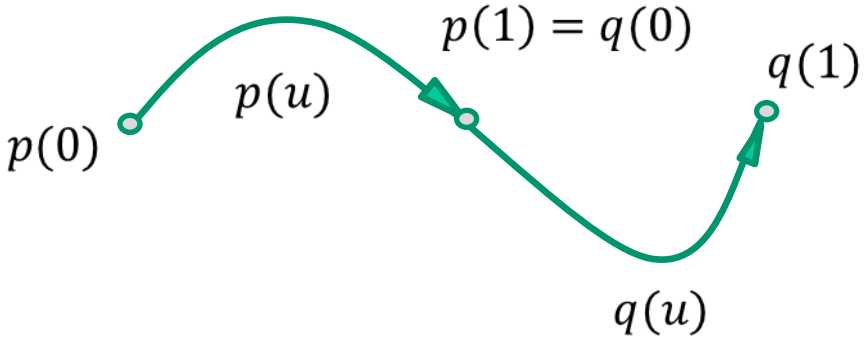
- Four of the elements of Q are chosen to interpolate the corners of the patch, whereas the others are chosen to match certain derivatives at the corners of the patch.

Geometric and Parametric Continuity

- Suppose that the polynomial on the left is p(u) and the one on the right is q(u). We enforce various continuity conditions by matching the polynomials and their derivatives at $u = 1$ for p(u), with the corresponding values for q(u) at $u = 0$. If we want the function to be continuous, we must have

$$\mathbf{p}(1) = \begin{bmatrix} p_x(1) \\ p_y(1) \\ p_z(1) \end{bmatrix} = \mathbf{q}(0) = \begin{bmatrix} q_x(0) \\ q_y(0) \\ q_z(0) \end{bmatrix}$$

- All three parametric components must be equal at the join point. We call this property C^0 parametric continuity.



Geometric and Parametric Continuity

- When we consider derivatives, we can require, as we did with the Hermite curve, that

$$\mathbf{p}'(1) = \begin{bmatrix} p'_x(1) \\ p'_y(1) \\ p'_z(1) \end{bmatrix} = \mathbf{q}'(0) = \begin{bmatrix} q'_x(0) \\ q'_y(0) \\ q'_z(0) \end{bmatrix}$$

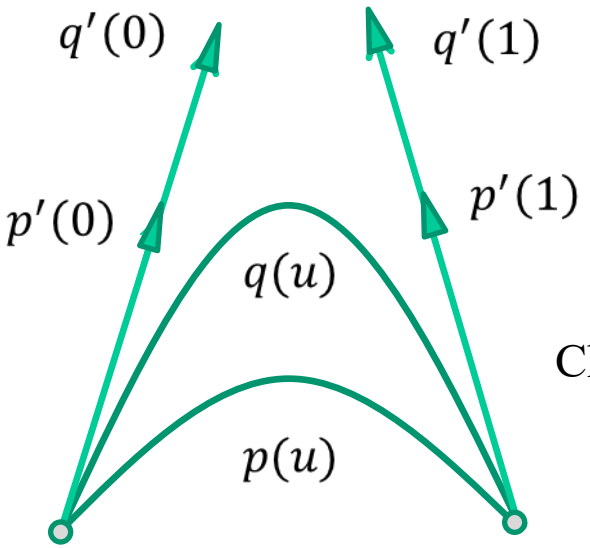
  If we match all three parametric equations and the first derivative, we have C^1 parametric continuity.

- In 3D, the derivative at a point on a curve defines the tangent line at that point. Suppose that instead of requiring matching of the derivatives for the two segments at the join point, we require only that their derivatives be proportional:

$$\mathbf{p}'(1) \propto \mathbf{q}'(0)$$

Geometric Continuity

- If the tangent of the two curves are proportional, then they point in the same direction, but they may have different magnitudes, we call this type of continuity ***G*^{1} geometric continuity**.
- This idea can be extended to higher derivatives, such as C^n and G^n continuity.
- The curves p(u) and q(u) share the same endpoints, and the tangents at the endpoints in the same direction, but the curves are different.



Change of magnitude in G^1 continuity

Bézier Curves and Surfaces

- Consider the four control points: p_0, p_1, p_2 and p_3. Suppose that we still insist on interpolating known values at the endpoints with a cubic polynomial $p(u)$:

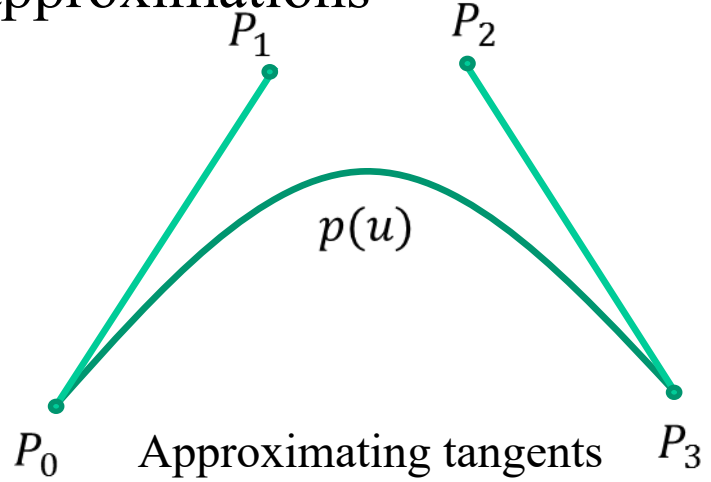
$$p_0 = p(0),$$

$$p_3 = p(1).$$

- Bézier proposed that rather than using the other two control points, p_1 and p_2, for interpolation, we use them to approximate the tangents at u=0 and u=1. In parameter space, we can use the linear approximations

$$p'(0) = \frac{p_1 - p_0}{\frac{1}{3}} = 3(p_1 - p_0),$$

$$p'(1) = \frac{p_3 - p_2}{\frac{1}{3}} = 3(p_3 - p_2),$$



Approximating tangents

Bézier Curves and Surfaces

- Applying these approximations to the derivatives of our parametric polynomial, $\mathbf{p}(u) = \mathbf{u}^T\mathbf{c}$, at the two endpoints, we have the two conditions

$$3\mathbf{p}_1 - 3\mathbf{p}_0 = \mathbf{c}_1,$$
$$3\mathbf{p}_3 - 3\mathbf{p}_2 = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3,$$

To add to our interpolation conditions

$$\mathbf{p}_0 = \mathbf{c}_0,$$
$$\mathbf{p}_3 = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3.$$

At this point, we again have three sets of four equations in four unknowns that we can solve to find

$$\mathbf{c} = \mathbf{M}_B\mathbf{p},$$

where $\mathbf{M}_B$ is the Bézier geometry matrix

$$\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}.$$

Bézier Curves and Surfaces

- The cubic Bézier polynomial is thus

$$\mathbf{p}(u) = \mathbf{u}^T \mathbf{M}_B \mathbf{p}.$$

- If we have a set of control points, p_0, …, p_n, we use p_0, p_1, p_2, and p_3 for the first curve; p_3, p_4, p_5, and p_6 for the second; and so on. It should be clear that we have C^0 continuity.
- We have given up the C^1 continuity of the Hermite polynomial because we use different approximations on the left and right of a join point.

Bézier Curves and Surfaces

- We can see important advantages to the Bézier curve by examining the blending functions. We write the curve as

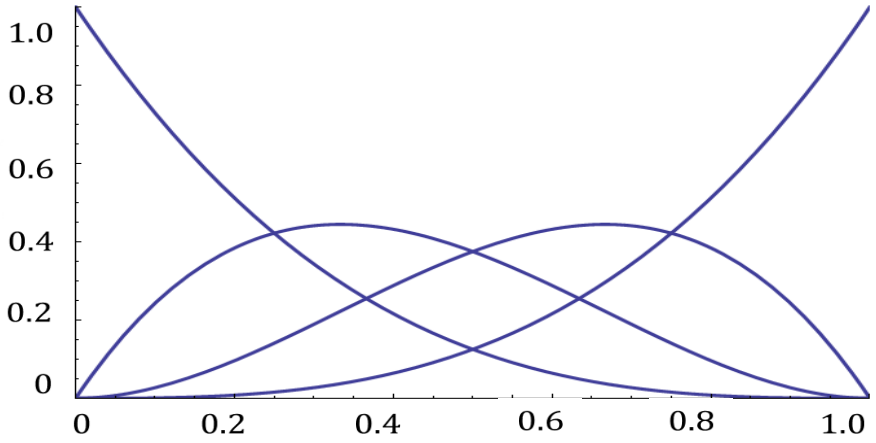
$$\mathbf{p}(u) = \mathbf{b}(u)^T \mathbf{p},$$

where

$$\mathbf{b}(u) = \mathbf{M}_B^T \mathbf{u} = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u) \\ u^3 \end{bmatrix}.$$

These four polynomials are one case of the Bernstein polynomials

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

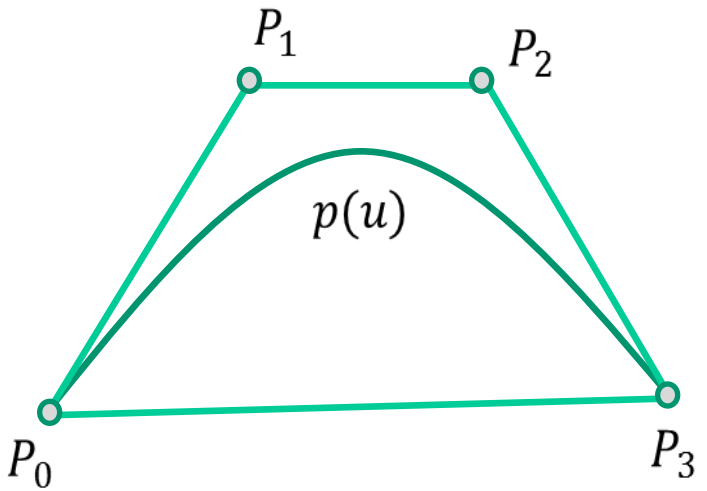


Blending polynomials for Bézier cubic

Bézier Curves and Surfaces

- Remarkable properties of the Bernstein polynomials:
  - All the zeros of the polynomials are either at $u = 0$ or at $u = 1$. Consequently, for each blending polynomial $0 < b_{id}(u), \text{ for } 0 < u < 1.$ Without any zeros in the interval, each blending polynomial must be smooth.
  - In the interval $b_{id}(u) \leq 1$ and $\sum_{i=0}^{d} b_{id}(u) = 1$. The representation of our cubic Bézier polynomial in terms of its blending polynomials is a convex sum. Consequently, p(u) must lie in the convex hull of the four control points.

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



Convex hull and Bézier polynomial

Bézier Surface Patches

- We can generate the Bézier surface patches through the blending functions. If P is a 4 x 4 array of control points,

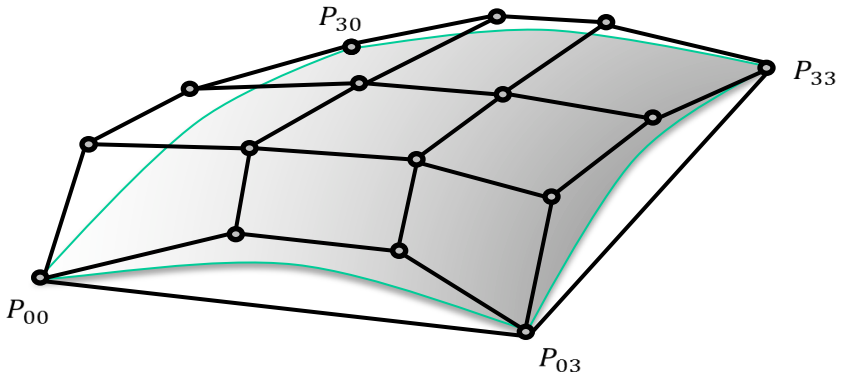
$$\mathbf{P} = \left[\mathbf{p}_{ij}\right],$$

then the corresponding Bézier patch is

$$\mathbf{p}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} b_i(u) b_j(v) \mathbf{p}_{ij} = \mathbf{u}^T \mathbf{M}_B \mathbf{P} \mathbf{M}_B^T \mathbf{v}.$$

- The patch is fully contained in the convex hull of the control points and interpolates $\mathbf{p}_{00}$, $\mathbf{p}_{03}$, $\mathbf{p}_{30}$, and $\mathbf{p}_{33}$. We can interpret the other conditions as approximations to various derivatives at the corners of the patch.

Bézier patch

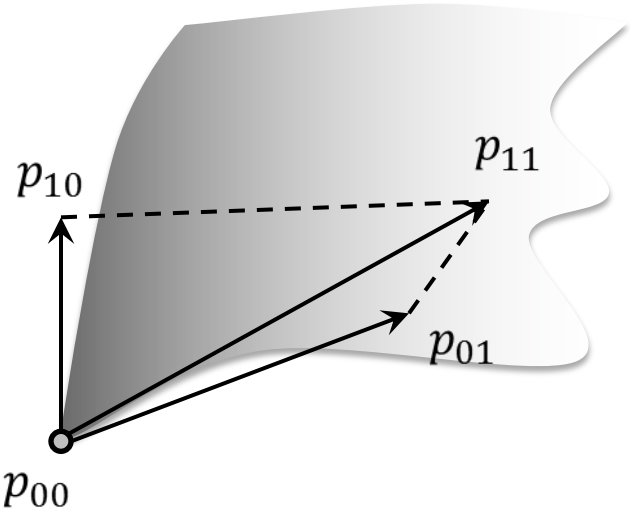


Bézier Surface Patches

- Consider the corner for $u = v = 0$. We can evaluate $\mathbf{p}(u, v)$ and the first partial derivatives to find

$$\mathbf{p}(0,0) = \mathbf{p}_{00},$$
$$\frac{\partial \mathbf{p}}{\partial u}(0,0) = 3(\mathbf{p}_{10} - \mathbf{p}_{00}),$$
$$\frac{\partial \mathbf{p}}{\partial v}(0,0) = 3(\mathbf{p}_{01} - \mathbf{p}_{00}),$$
$$\frac{\partial^2 \mathbf{p}}{\partial u \partial v}(0,0) = 9(\mathbf{p}_{00} - \mathbf{p}_{01} + \mathbf{p}_{10} - \mathbf{p}_{11}).$$

Twist at corner of Bézier patch



- The first three conditions are clearly extensions of our results for the Bézier curve. The fourth can be seen as a measure of the tendency of the patch to divert from being flat, or to twist, at the corner.

Cubic B-Splines

- Cubic Bézier curves and surface patches have one fundamental limitation: at the join points (or patch edges for surfaces), we have only C^0 continuity.

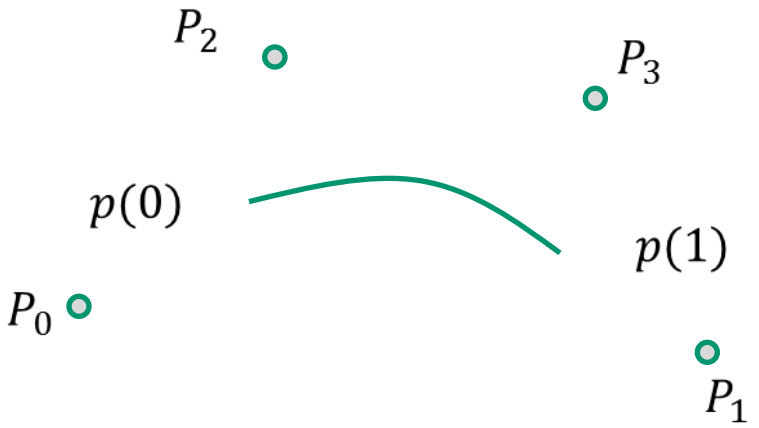
- Possible ways to solve this problem:
  - High-degree polynomials
  - Shorten the interval and use more polynomial segments.
  - Use the same control-point data but not require the polynomials to interpolate any of these points.

Cubic B-Spline Curve

- Consider four control points in the middle of a sequence of control points: $\{\mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}\}$. Our previous approach was to use these four points to define a cubic curve such that, as the parameter u varied from 0 to 1, the curve spanned the distance from $\mathbf{p}_{i-2}$ to $\mathbf{p}_{i+1}$, interpolating $\mathbf{p}_{i-2}$ and $\mathbf{p}_{i+1}$.

Cubic B-Splines

- Suppose that as u goes from 0 to 1, we span only the distance between the middle two control points. Likewise, we use $\{\mathbf{p}_{i-3}, \mathbf{p}_{i-2}, \mathbf{p}_{i-1}, \mathbf{p}_i\}$ between $\mathbf{p}_{i-2}$ and $\mathbf{p}_{i-1}$, and $\{\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{p}_{i+2}\}$ between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. Suppose that p(u) is the curve we use between $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$, and q(u) is the curve to its left, used between $\mathbf{p}_{i-2}$ and $\mathbf{p}_{i-1}$. We can match conditions at p(0) with conditions at q(1).



Four points that define a curve between the middle two points.

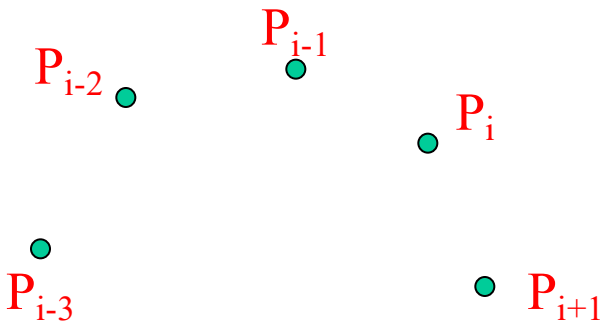
Cubic B-Spline Curve

- We are looking for a matrix M, such that the desired cubic polynomial is

$$\mathbf{p}(u) = \mathbf{u}^T \mathbf{M} \mathbf{p},$$

where p is the matrix of control points

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_{i-2} \\ \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \end{bmatrix}.$$



We can use the same matrix to write q(u) as

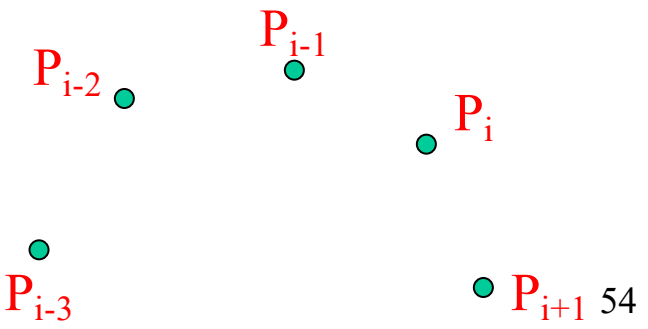
$$\mathbf{q}(u) = \mathbf{u}^T \mathbf{M} \mathbf{q}, \qquad \mathbf{q} = \begin{bmatrix} \mathbf{p}_{i-3} \\ \mathbf{p}_{i-2} \\ \mathbf{p}_{i-1} \\ \mathbf{p}_i \end{bmatrix}.$$

Cubic B-Spline Curve

- In principle, we could write down a set of conditions on p(0) that would match conditions for q(1), and we could write equivalent conditions matching various derivatives of p(1) with conditions for another polynomial that starts there.
- We can take a shortcut to deriving the most popular matrix, by noting that we must use symmetric approximations at the join point. Hence, any evaluation of conditions on q(1) can't use p_{i-3}, because this control point doesn't appear in the equation for p(u). Likewise, we can't use p_{i+1} in any condition on p(0).
- Two conditions that satisfy this symmetry condition are

$$\mathbf{p}(0) = \mathbf{q}(1) = \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i),$$

$$\mathbf{p}'(0) = \mathbf{q}'(1) = \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2}).$$



Cubic B-Spline Curve

- If we write p(u) in terms of the coefficient array c,

$$\mathbf{p}(u) = \mathbf{u}^T \mathbf{c},$$

these conditions are

$$\mathbf{c}_0 = \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i),$$

$$\mathbf{c}_1 = \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2}).$$

- We can apply the symmetric conditions at p(1):

$$\mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3 = \frac{1}{6}(\mathbf{p}_{i-1} + 4\mathbf{p}_i + \mathbf{p}_{i+1}),$$

$$\mathbf{p}'(1) = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3 = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}).$$

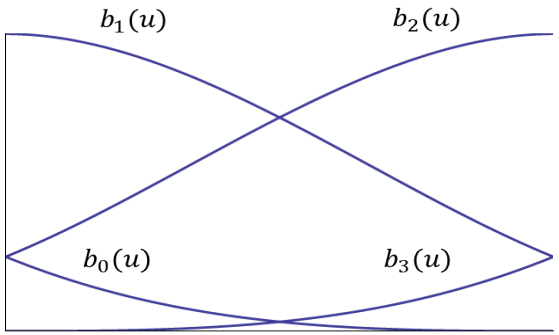
Cubic B-Spline Curve

- We now have four equations for the coefficients of c, which we can solve for a matrix M_s, the B-spline geometry matrix

$$\mathbf{M}_S = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}.$$

- This particular matrix yields a polynomial that has several important properties. We can see these properties by again examining the blending polynomials:

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



Spline-blending functions

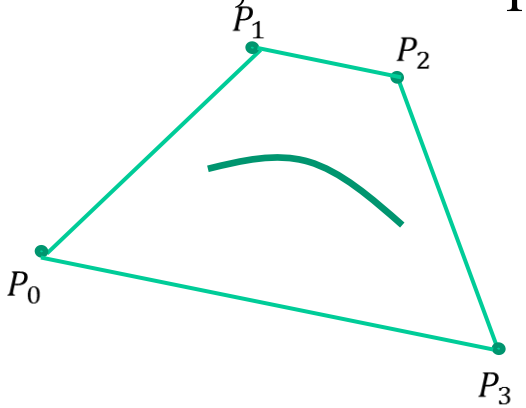
Cubic B-Spline Curve

- We can show that

$$\sum_{i=0}^{3} b_i(u) = 1,$$

$$0 \le b_i(u) \le 1 \quad \text{in the interval } 0 \le u \le 1$$

- The curve must lie in the convex hull of the control points. Note that the curve is used for only part of the range of the convex hull.
- We defined the curve to have C^1 continuity; in fact, however, it has C^2 continuity. We can verify it by computing p''(u) at $u = 0$ and $u = 1$.
- Compared to Bézier cubic, cubic B-splines are time consuming.

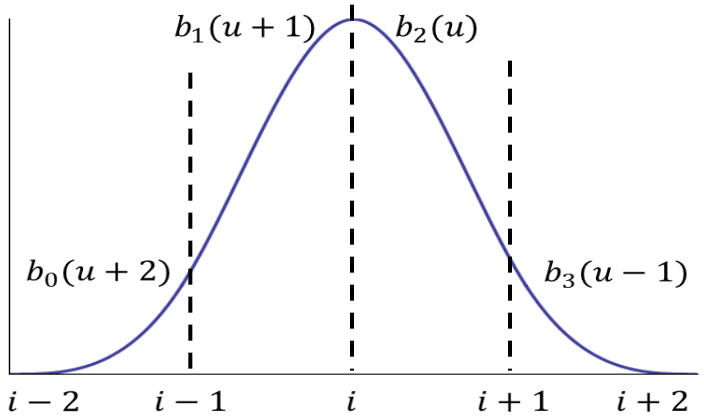


Convex hull for spline curve

B-Splines and Basis

- Each control point contributes to the spline in four adjacent intervals. This property guarantees the locality of the spline; that is, if we change a single control point, we can affect the resulting curve in only four adjacent intervals.
- The total contribution of a single control point can be written as $B_i(u)\mathbf{p}_i$, where B_i is the function

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



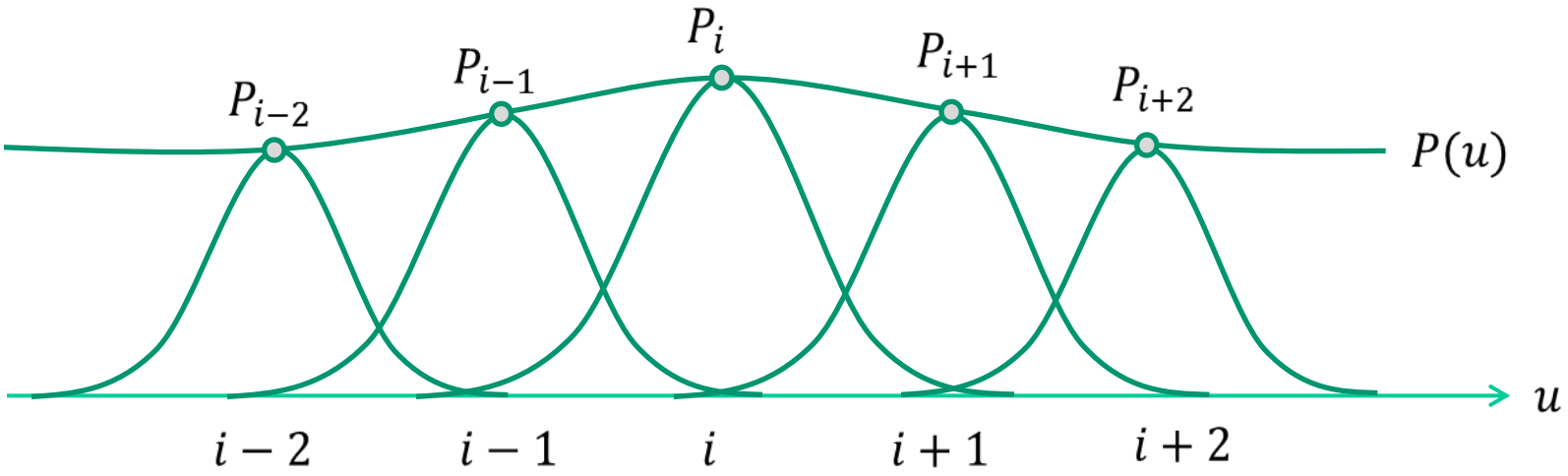
Spline basis function

- Given a set of control points $P_0, \ldots, P_m$, we can write the entire spline with the single expression

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

B-Spline and Basis

- This expression shows that, for the set of functions $B(u\text{-}i)$, each member is a shifted version of a single function, and the set forms a basis for all our cubic B-spline curves.
- Given a set of control points, we form a piecewise polynomial curve p(u) over the whole interval as a linear combination of basis functions.



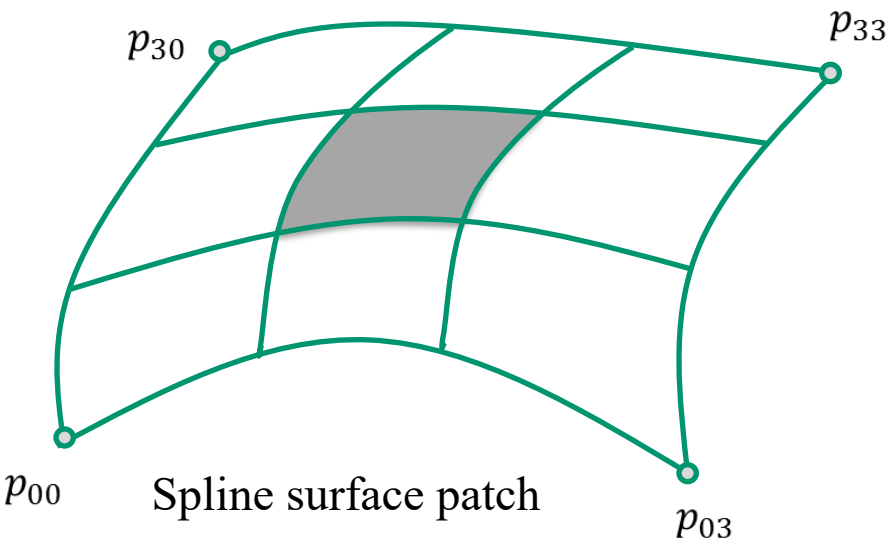
Approximating function over interval

Spline Surfaces

- B-spline surfaces can be defined in a similar way. If we start with the B-spline blending functions, the surface patch is given by

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

- We use the patch over only the central area, and we must do nine times the work that we would do with the Bézier patch.
- Because of inheritance of the convex hull property and the additional continuity at the edges from the B-spline curves, the B-spline patch is considerably smoother than a Bézier patch constructed from the same data would be.



Spline surface patch

General B-Splines

- Suppose that we have a set of control points, $\mathbf{p}_0, \ldots, \mathbf{p}_m$. The general approximation problem is to find a function $\mathbf{p}(u) = [\, x(u) \quad y(u) \quad z(u) \,]^T$ defined over an interval $u_{\min} \le u \le u_{\max}$, that is smooth and is close to the control points.
- Suppose we have a set of values $\{u_k\}$, called knots, s.t.

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

We call the sequence $u_0, u_1, \ldots, u_n$ the knot array. In splines, the function $\mathbf{p}(u)$ is a polynomial of degree d between the knots,

$$\mathbf{p}(u) = \sum_{j=0}^{d} \mathbf{c}_{jk} u^j, \qquad u_k < u < u_{k+1}.$$

- To define a spline of degree d, we must define the $n(d+1)$ 2D coefficients $\mathbf{c}_{jk}$. We get the required conditions by applying various continuity requirements at the knots and interpolation requirements at control points.

Recursively Defined B-Splines

- The approach taken in B-splines is to define the spline in terms of a set of basis or blending functions, each of which is nonzero over only the regions spanned by a few knots. Thus, we write the function p(u) as an expansion:

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

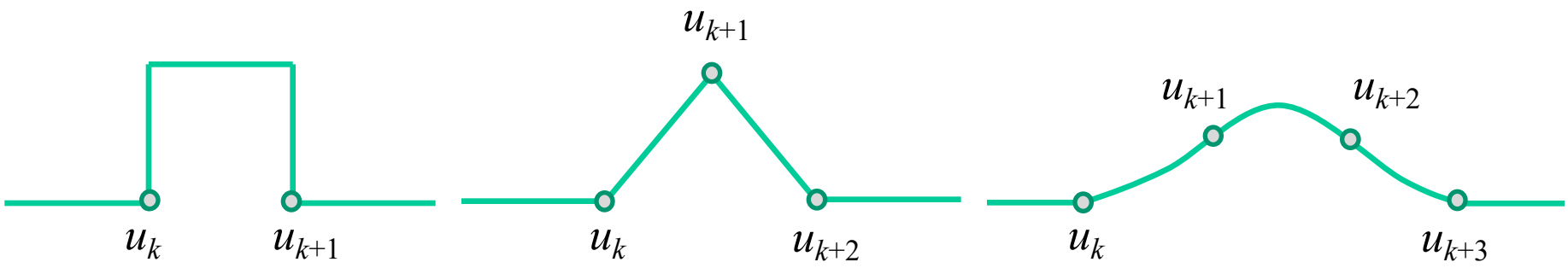
where each function $\mathrm{B}_{id}(u)$ is a polynomial of degree d, except at the knots, and is zero outside the interval $(u_{i_{\min}}, u_{i_{\max}})$.

Recursively Defined B-Splines

- The name *B-splines* comes from the term *basis splines*, in recognition that the set of functions $\{B_{id}(u)\}$ forms a basis for the given knot sequence and degree.
- The set of splines defined by the Cox-deBoor recursion:

$$B_{k0} = \begin{cases} 1, & u_k \le u \le u_{k+1}; \\ 0, & \text{otherwise} \end{cases}$$

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



First three basis functions

Recursively Defined B-Splines

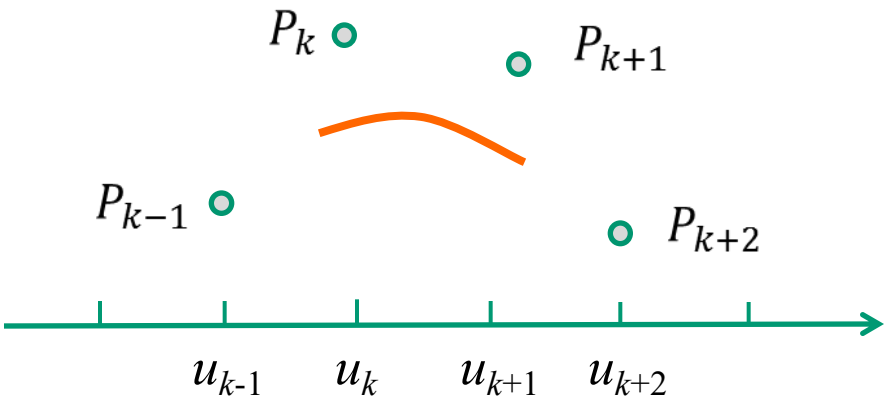
- At the knots, there is C^{d-1} continuity. The convex-hull property holds because

$$\sum_{i=0}^{m} B_{i,d}(u) = 1, \qquad \text{and} \qquad 0 \le B_{id}(u) \le 1.$$

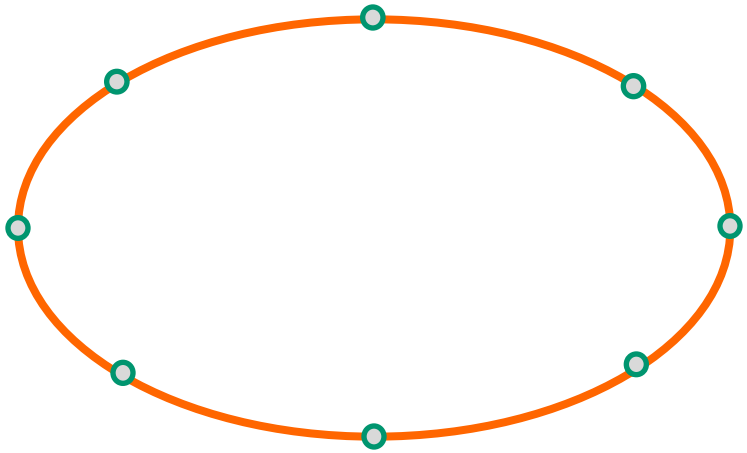
- However, each B_{id} is nonzero in only d+1 intervals, each control point can affect only d+1 intervals, and each point on the resulting curve is within the convex hull defined by these d+1 control points.
- Note that careful examination of the Cox-deBoor formula shows that each step of the recursion is a linear interpolation of functions produced on the previous step. Linear interpolation of polynomials of degree d produces polynomials of degree d+1.

Uniform Splines

- Consider the uniform knot sequence {0, 1, 2, …, n}. The cubic B-spline could be derived from the Cox-deBoor formula with equally spaced knots.
- In certain situations, we can use the periodic nature of the control-point data to define the spline over the entire knot sequence.
- These uniform periodic B-splines have the property that each spline basis function is a shifted version of a single function.



Uniform B-spline



Periodic uniform B-spline

Non-Uniform B-Splines

- Repeated knots have the effect of pulling the spline closer to the control point associate with the knot.
- If a knot at the end has multiplicity d+1, the B-spline of degree d must interpolate the point. Hence one solution to the problem of the spline not having sufficient data to span the desired interval is to repeat knots at the ends, forcing interpolation at the endpoints, and using uniform knots everywhere else. Such splines are called open splines.
- The knot sequence $\{0, 0, 0, 0, 1, 2, \ldots, n\text{-}1, n, n, n, n\}$ is often used for cubic B-spline. In $\{0, 0, 0, 0, 1, 1, 1, 1\}$, the cubic B-spline becomes the cubic Bézier curve. We can also repeat internal knots.

NURBS

In our development of B-splines, we have assumed that $\mathbf{p}(u)$ is the array $[\, x(u) \quad y(u) \quad z(u) \,]^T$. In two dimensions, however, we could have replaced it with simply $[\, x(u) \quad y(u) \,]^T$, and all our equations would be unchanged. Indeed, the equations remain unchanged if we go to four-dimensional B-splines. Consider a control point in three dimensions:

$$\mathbf{p}_i = [x_i \quad y_i \quad z_i].$$

The weighted homogeneous-coordinate representation of this point is

$$\mathbf{q}_i = w_i \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}.$$

The idea is to use the weights w_i to increase or decrease the importance of a particular control point. We can use these weighted points to form a four-dimensional B-spline. The first three components of the resulting spline are simply the B-spline representation of the weighted points,

$$\mathbf{q}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \sum_{i=0}^{n} B_{i,d}(u) w_i \mathbf{p}_i.$$

NURBS

The w component is the scalar B-spline polynomial derived from the set of weights:

$$w(u) = \sum_{i=0}^{n} B_{i,d}(u) w_i.$$

In homogeneous coordinates, this representation has a w component that may not be equal to 1; thus, we must do a perspective division to derive the three-dimensional points:

$$\mathbf{p}(u) = \frac{1}{w(u)}\mathbf{q}(u) = \frac{\sum_{i=0}^{n} B_{i,d}(u) w_i \mathbf{p}_i}{\sum_{i=0}^{n} B_{i,d}(u) w_i}.$$

Each component of $\mathbf{p}(u)$ is now a rational function in u, and, because we have not restricted the knots in any way, we have derived a **nonuniform rational B-spline (NURBS)** curve.

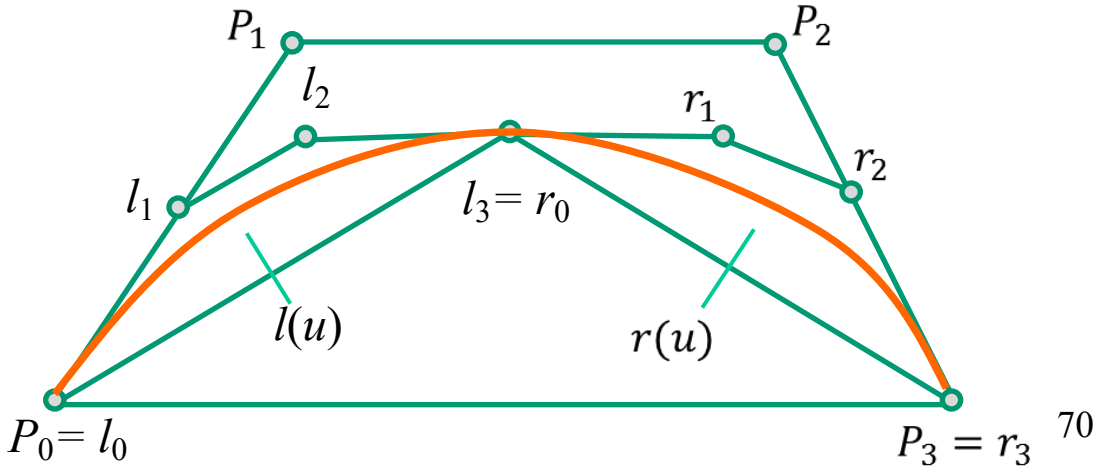
NURBS

- NURBS curves retain all the properties of our 3D B-splines, such as the convex-hull and continuity properties.
- NURBS curves can be handled correctly in perspective views, which is useful for computer graphics.
- Quadric surfaces are usually defined as algebraic implicit forms. Quadrics can be shown to be a special case of quadratic NURBS curve, therefore we can use a single modeling methods, NURBS curves, for the most widely used curves and surfaces.

Recursive Subdivision of Bézier Polynomials

- Suppose that we have a cubic Bézier polynomial. We know that the curve must lie inside the convex hull of the control points. We can break the curve into two separate polynomials, $l(u)$ and $r(u)$, each valid over one-half of the original interval.
- Because the original polynomial is a cubic, each of these polynomials is also a cubic. We need to rescale the parameter u for l and r.
- The convex hull for l and r must lie inside the convex hull for p, a result known as the variation-diminishing property of the Bézier curve.

Convex hulls and control points



Recursive Subdivision of Bézier Polynomials

We shall find the hull for $\mathbf{l}(u)$; the calculation for $\mathbf{r}(u)$ is symmetric. We can start with

$$\mathbf{p}(u) = \mathbf{u}^T \mathbf{M}_B \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix},$$

where

$$\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}.$$

The polynomial $\mathbf{l}(u)$ must interpolate $\mathbf{p}(0)$ and $\mathbf{p}\left(\frac{1}{2}\right)$; hence,

$$\mathbf{l}(0) = \mathbf{l}_0 = \mathbf{p}_0,$$

$$\mathbf{l}(1) = \mathbf{l}_3 = \mathbf{p}\left(\frac{1}{2}\right) = \frac{1}{8}(\mathbf{p}_0 + 3\mathbf{p}_1 + 3\mathbf{p}_2 + \mathbf{p}_3).$$

Recursive Subdivision of Bézier Polynomials

At $u = 0$, the slope of $\mathbf{l}$ must match the slope of $\mathbf{p}$, but, because the parameter for $\mathbf{p}$ covers only the range $\left(0, \frac{1}{2}\right)$ while u varies over $(0, 1)$, implicitly we have made the substitution $\bar{u} = 2u$. Consequently, derivatives for $\mathbf{l}$ and $\mathbf{p}$ are related by $d\bar{u} = 2du$, and

$$\mathbf{l}'(0) = 3(\mathbf{l}_1 - \mathbf{l}_0) = \mathbf{p}'(0) = \frac{3}{2}(\mathbf{p}_1 - \mathbf{p}_0).$$

Likewise, at the midpoint,

$$\mathbf{l}'(1) = (\mathbf{l}_3 - \mathbf{l}_2) = \mathbf{p}'\left(\frac{1}{2}\right) = \frac{3}{8}(-\mathbf{p}_0 - \mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3).$$

These four equations can be solved algebraically. Alternatively, this solution can be expressed geometrically. Here, we construct

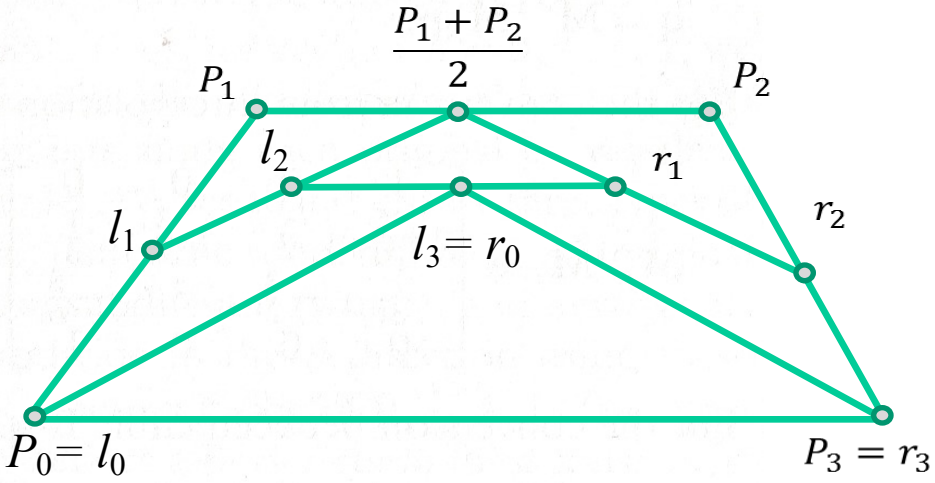
Recursive Subdivision of Bézier Polynomials

both the left and right sets of control points concurrently. First, we note that the interpolation condition requires that

$$\mathbf{l}_0 = \mathbf{p}_0,$$
$$\mathbf{r}_3 = \mathbf{p}_3.$$

We can verify by substitution in the four equations that the slopes on the left and right yield

$$\mathbf{l}_1 = \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1),$$
$$\mathbf{r}_2 = \frac{1}{2}(\mathbf{p}_2 + \mathbf{p}_3).$$



The interior points are given by

$$\mathbf{l}_2 = \frac{1}{2}\left(\mathbf{l}_1 + \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2)\right),$$
$$\mathbf{r}_1 = \frac{1}{2}\left(\mathbf{r}_2 + \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2)\right).$$

Finally, the shared middle point is given by

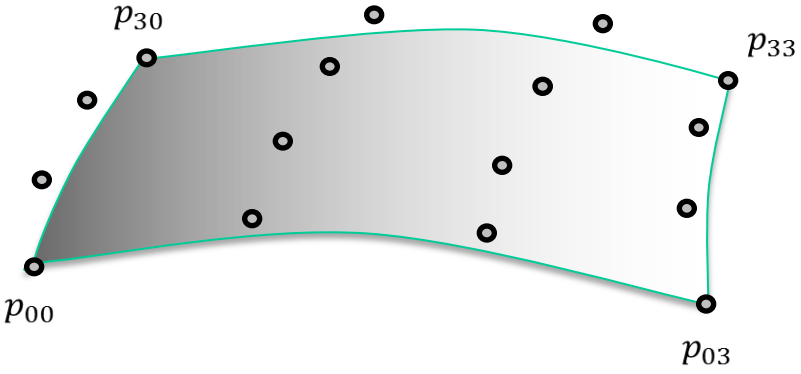
$$\mathbf{l}_3 = \mathbf{r}_0 = \frac{1}{2}(\mathbf{l}_2 + \mathbf{r}_1).$$

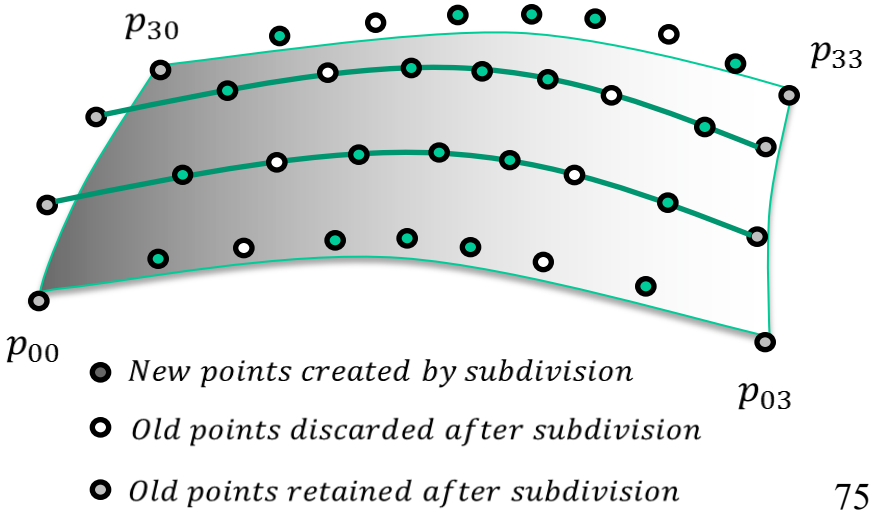
Recursive Subdivision of Bézier Polynomials

- One of the advantages of the subdivision approach is that it can be made adaptive, and only one of the sides may require subdivision.
- Recursive subdivision is used in the rendering, as well as geometric modeling.

Subdivision of Bézier Surfaces

- We can extend our subdivision algorithm to Bézier surfaces. Consider the cubic surface with the 16 control points. Each four points in a row or column determine a Bézier curve that can be subdivided.
- However, our subdivision algorithm should split the patch into four patches, and we calculate control points along the center of the patch.





Algebraic Surfaces – Quadrics

Quadric surfaces are described by implicit algebraic equations in which each term is a polynomial of the form $x^i y^j z^k$, with $i + j + k \leq 2$. Any quadric can be written in the form

$$q(x, y, z) = a_{11}x^2 + 2a_{12}xy + a_{22}y^2 + a_{33}z^2 + 2a_{23}yz + 2a_{13}xz + b_1x + b_2y + b_3z + c = 0.$$

This class of surfaces includes ellipsoids, parabaloids, and hyperboloids. We can write the general equation in matrix form in terms of the three-dimensional column matrix $\mathbf{p} = [\, x \quad y \quad z \,]^T$ as the **quadratic form**

$$\mathbf{p}^T\mathbf{A}\mathbf{p} + \mathbf{b}^T\mathbf{p} + c = 0,$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

The 10 independent coefficients in $\mathbf{A}$, $\mathbf{b}$, and c determine a given quadric.

Algebraic Surfaces – Quadrics

- We can apply a sequence of rotations and translations that reduces a quadric to a standard form without changing the type of surface. In 3D, we can write such a transformation as

$$\mathbf{p}' = \mathbf{M}\mathbf{p} + \mathbf{d}.$$

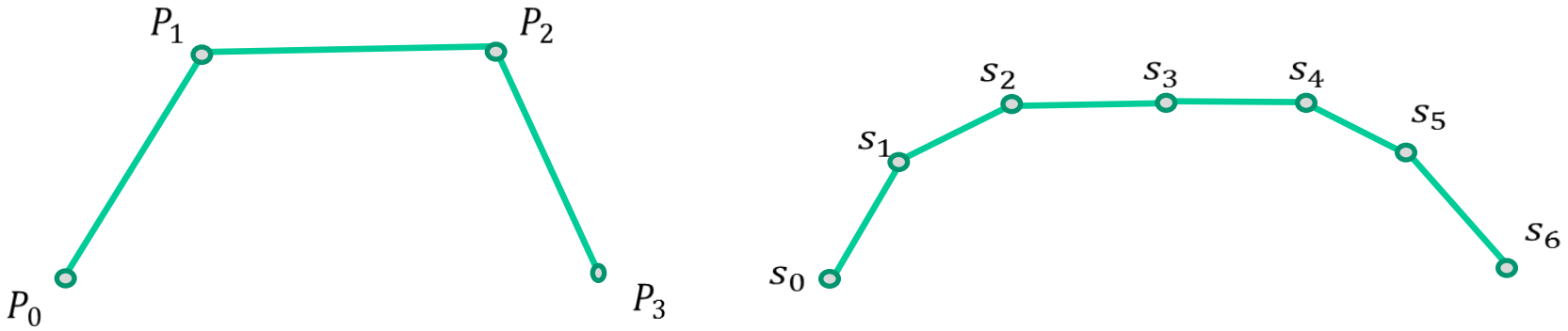
$$\mathbf{D} \leftarrow \mathbf{M}^T\mathbf{A}\mathbf{M}$$

- The matrix M can be chosen to be a rotation matrix such that D = M^TAM is a diagonal matrix. The diagonal elements of D can be used to determine the type of quadric.

- For example, $a'_{11}x'^2 + a'_{22}y'^2 + a_{33}z'^2 - c' = 0$, where all the coefficients are positive, represents an ellipsoid.

Subdivision Curves and Surfaces

- We start with four points – P_0, P_1, P_2, P_3 – and end up with seven points, s_0, …, s_6. We can view each of these sets of points as defining a piecewise-linear curve:



Left: Piecewise-linear curve determined by four points.

Right: Piecewise-linear curve after one subdivision step.

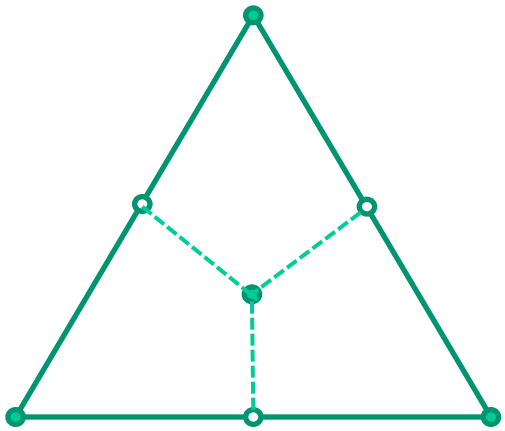
Subdivision Curves and Surfaces

- The second curve is said to be a refinement of the first. We can continue the process iteratively and in the limit converge to the B-spline.

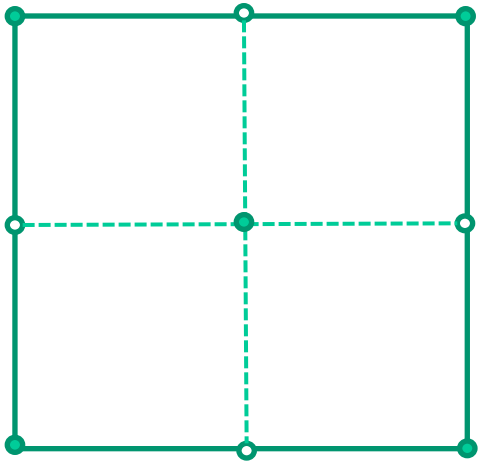
$$
\begin{aligned}
\mathbf{s}_0 &= \mathbf{p}_0 \\
\mathbf{s}_1 &= \tfrac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1), \\
\mathbf{s}_2 &= \tfrac{1}{4}(\mathbf{p}_0 + 2\mathbf{p}_1 + \mathbf{p}_2), \\
\mathbf{s}_3 &= \tfrac{1}{8}(\mathbf{p}_0 + 3\mathbf{p}_1 + 3\mathbf{p}_2 + \mathbf{p}_3), \\
\mathbf{s}_4 &= \tfrac{1}{4}(\mathbf{p}_1 + 2\mathbf{p}_2 + \mathbf{p}_3), \\
\mathbf{s}_5 &= \tfrac{1}{2}(\mathbf{p}_2 + \mathbf{p}_3), \\
\mathbf{s}_6 &= \mathbf{p}_3
\end{aligned}
$$

Subdivision Curves and Surfaces

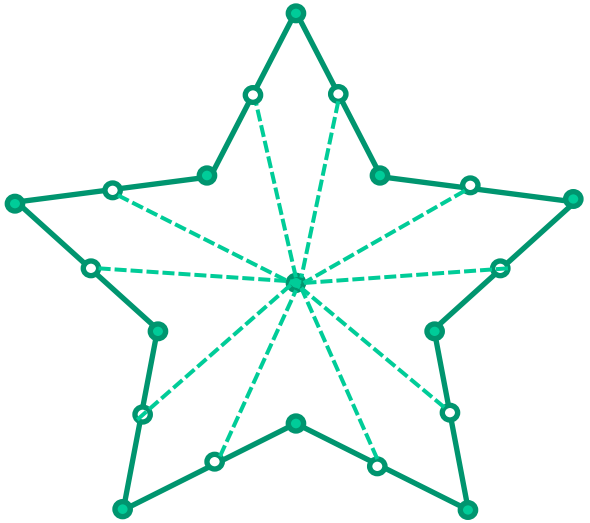
- Subdivision and its benefits are not limited to B-splines. Over the past few years, a variety of methods for generating these subdivision curves have appeared. In all cases, the refined curves converge to a smooth curve.
- Subdivision can also be used on surfaces.



triangle



rectangle



star-shaped polygon

Subdivision Surface

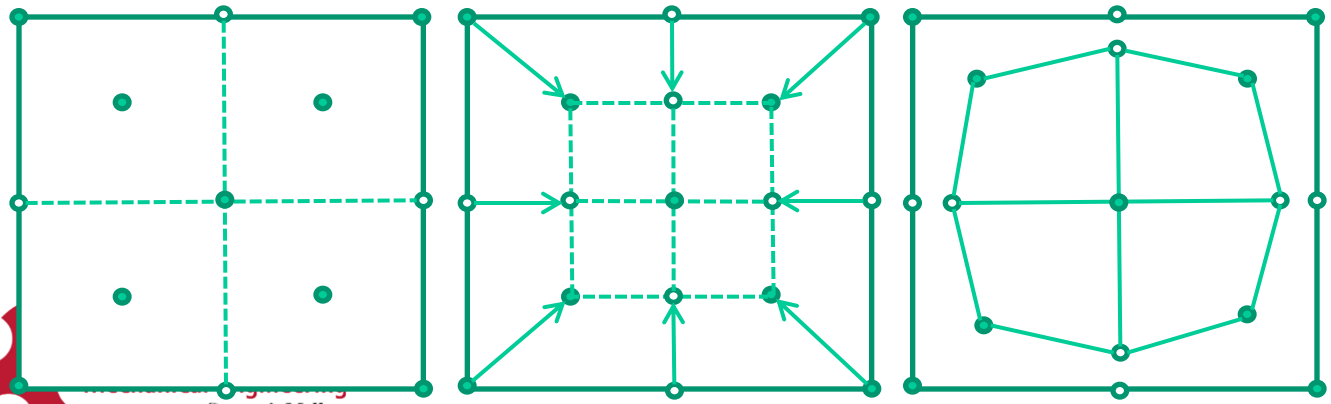
- A subdivision surface is a method of representing a smooth surface via the specification of a coarser piecewise linear polygon mesh.
- The smooth surface can be calculated from the coarse mesh as the limit of an iterative process of subdividing each polygonal face into smaller faces that better approximate the smooth surface.
- The subdivision surfaces algorithm is recursive in nature. Starting with a given polygonal mesh, a refinement scheme is applied to the mesh, creating new vertices and new faces.

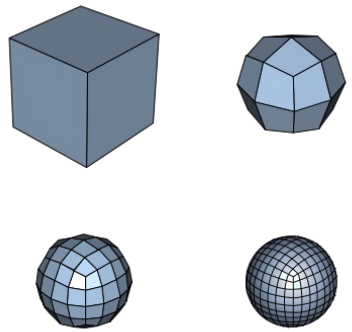
Approximating Refinement Schemes

- The limit surfaces approximate the initial meshes and that after subdivision, the newly generated control points may not be on the limit surfaces.
  - Catmull-Clark: bi-cubic uniform B-spline was generalized to produce their subdivision scheme. For arbitrary initial meshes, this scheme generates limit surfaces that are C^2 continuous everywhere except at extraordinary vertices where they are C^1 continuous.
  - Doo-Sabin: the analytical expression of bi-quadratic uniform B-spline surface was used to generate the subdivision procedure to produce C^1 limit surfaces with arbitrary topology for arbitrary initial meshes.
  - Loop (triangles): a quartic box-spline of 6 direction vectors was used to provide a rule to generate C^2 continuous limit surfaces everywhere except at extraordinary vertices where they are C^1 continuous.
  - Mid-Edge subdivision scheme: this scheme generates C^1 continuous limit surfaces on initial meshes with arbitrary topology.
  - Others

Catmull-Clark Subdivision

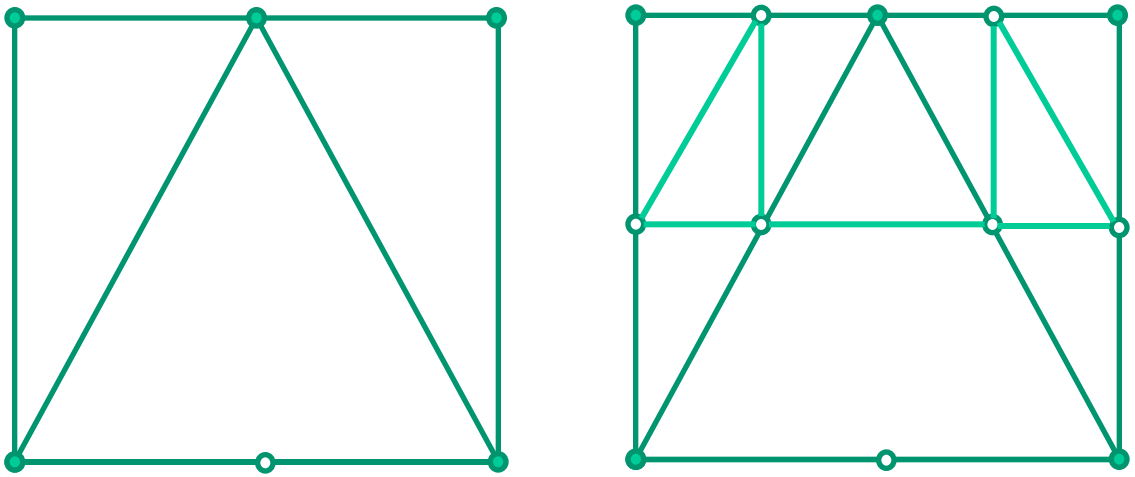
- Starting with a mesh of an arbitrary polyhedron
  - For each face, add a face point: (1) Set each face point to be the centroid of all original points for the respective face; (2) For each face point, add an edge for each edge of the face, connecting the face point to each edge point for the face.
  - For each edge, add an edge point: Set each edge point to be the average of all neighboring face points and original points.
  - For each original point P, take the average F of all n face points for faces touching P, and take the average R of all n edge midpoints touching P, where each edge midpoint is the average of its two endpoint vertices. Move each original point to the point $\frac{F + 2R + (n-3)P}{n}$.





Loop's Subdivision

- For triangular meshes: a quartic box-spline of 6 direction vectors was used to provide a rule to generate C^2 continuous limit surfaces everywhere except at extraordinary vertices where they are C^1 continuous.



Left: Triangular mesh. Right: Triangles after one subdivision.

http://en.wikipedia.org/wiki/Subdivision_surface

Successive Subdivisions of Polygonal Mesh

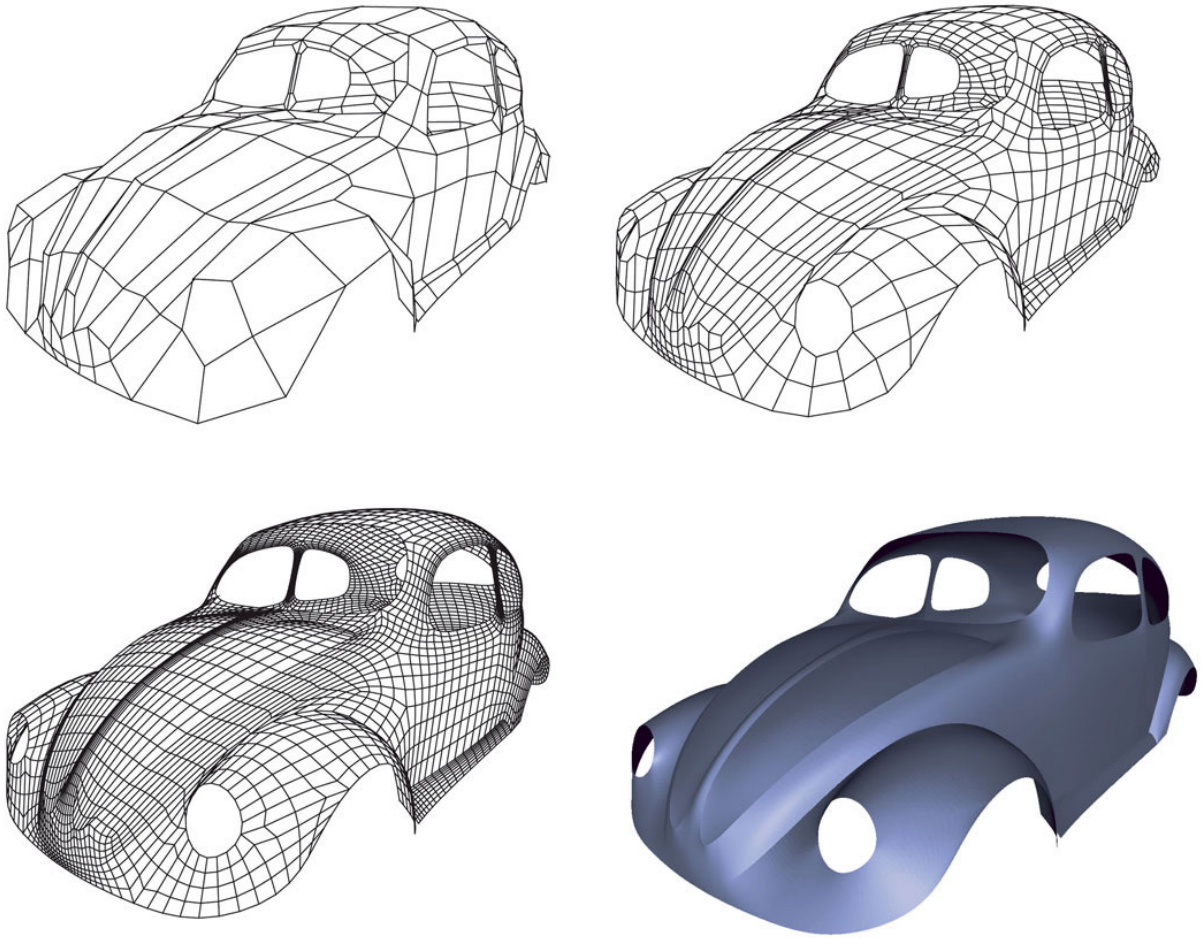


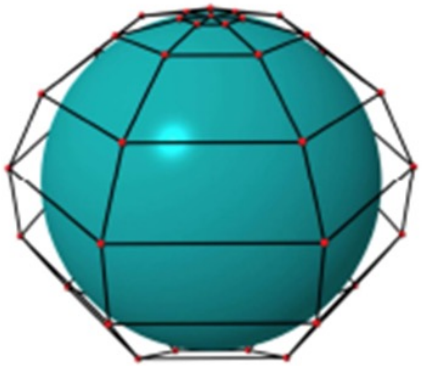
Image courtesy Caltech Multi-Res Modeling Group

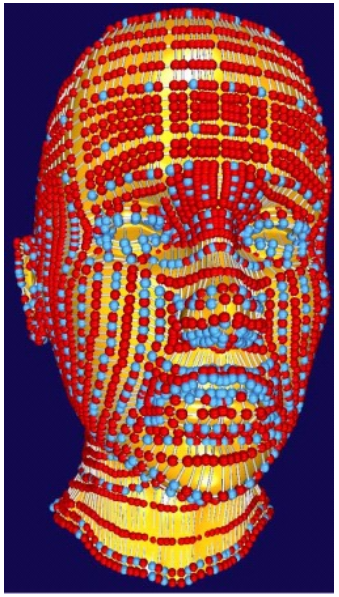
NURBS vs T-spline

NURBS (Non-Uniform Rational B-Spline) is a mathematical representation tool for free form curves and surfaces.

$$S(u,v)=\frac{\sum_i\sum_j P_{ij}w_iN_i^u(u)N_j^v(v)}{\sum_i\sum_j w_iN_i^u(u)N_j^v(v)}$$

- **Two global knot vectors**
- Control points with weight
- Degree

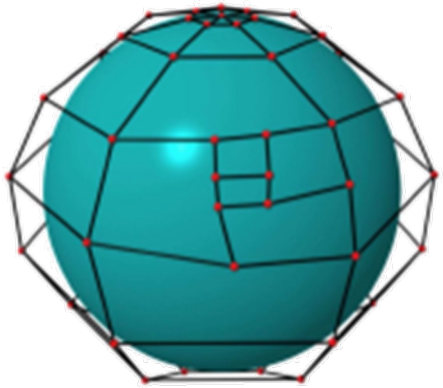




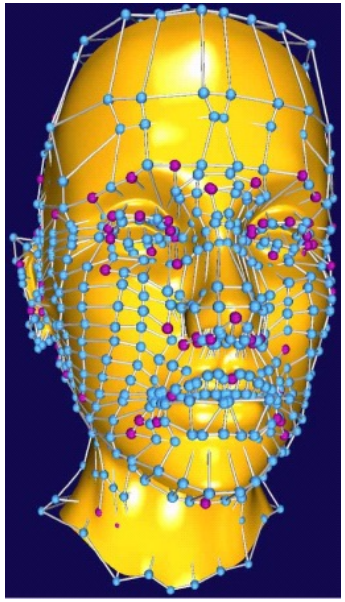
T-spline is a generalization of NURBS. It allows T-junctions in its control grid [T. Sederberg *et. al* 2003].

$$S(s,t)=\frac{\sum_i P_iw_iB_i(s,t)}{\sum_i w_iB_i(s,t)},\ B_i(s,t)=N_i^s(s)N_i^t(t)$$

- T-mesh
  - **Two local knot vectors** for each control point inferred from T-mesh
  - Control points with weight
- Degree

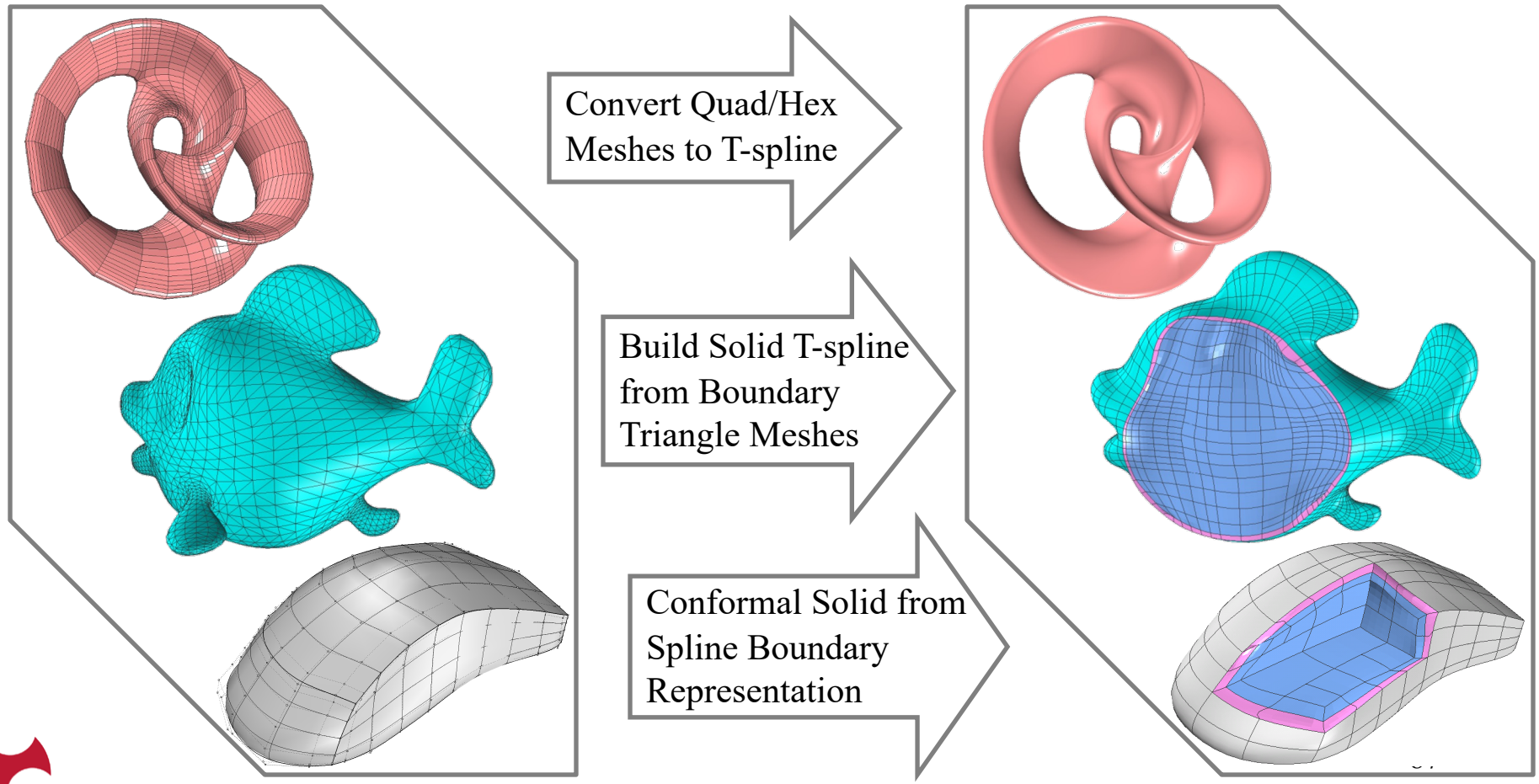


Allow local refinement



Our New Research: Volumetric T-spline Construction

To automatically construct solid T-spline models from spline boundary representations, which can be used directly in Isogeometric Analysis.



References

- Interactive Computer Graphics: A top-down approach using OpenGL. Edward Angel, 3rd Edition. Pearson Edition. Chapter 10.

- http://en.wikipedia.org/wiki/Subdivision_surface