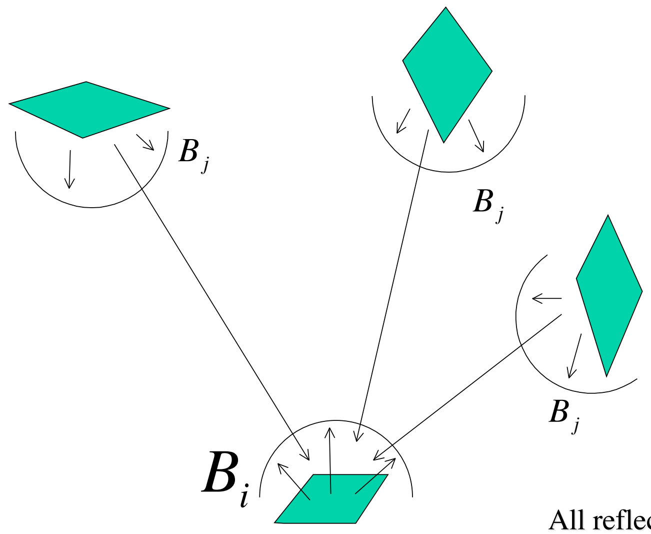
Radiosity (review)



All reflectors are Lambertian

Radiosity equation

$$B_i = E_i + \rho_i \sum_j F_{j\to i} B_j \frac{A_j}{A_i}$$

The form factor $F_{j\to i}$

is the fraction of light leaving dA_j arriving at dA_i taking into account orientation and obstructions

Useful relation

$$A_i F_{i\to j} = A_j F_{j\to i}$$

The equation now becomes

$$B_i = E_i + \rho_i \sum_j F_{i\to j} B_j$$

Rearrange to get

$$B_i - \rho_i \sum_j F_{i\to j} B_j = E_i$$

Radiosity (review)

In matrix form

$$\begin{bmatrix} 1-\rho_1 F_{1\to 1} & -\rho_1 F_{1\to 2} & \cdots & -\rho_1 F_{1\to n} \\ -\rho_2 F_{2\to 1} & 1-\rho_2 F_{2\to 2} & & -\rho_2 F_{2\to n} \\ & & & \\ -\rho_n F_{n\to 1} & -\rho_n F_{n\to 2} & & 1-\rho_n F_{n\to n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \\ E_n \end{bmatrix}$$

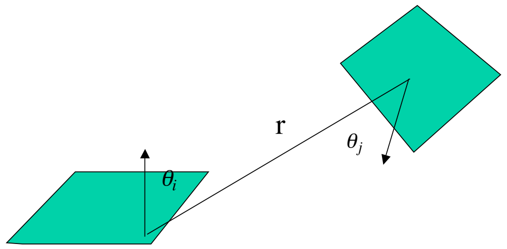
So, in theory, we just compute the Bi's by solving this (large!) matrix equation.

Optional

Optional

The fun part: Computing the $F_{i\to j}$

Without obstruction $dF_{dj\to di} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j$



Fancy methods exist for of computing and/or approximating storing form factors (e.g. hemisphere and hemicube methods)

Not covered in 2006

Fancy methods exist for of computing and/or approximating storing form factors (e.g. hemisphere and hemicube methods)

Hemicube: For a given patch, project other patches onto a cube surrounding it, which is an appropriately scaled version of the projection onto the hemisphere (use cube trick because planar projection is faster).

Hemicube projections can be tagged for distance (like Z-buffer) to deal with for occlusion.

Iterative Solution (gather)

The matrix equation can be solved iteratively (Gauss-Siedel method) starting with a crude estimate of the solution.

More intuitively, consider an estimate $\hat{B}_j$ for the B_j and plug them into our equation to re-estimated them.

$$B_i = E_i + \rho_i \sum_j F_{i\to j} \hat{B}_j$$

This is sometimes referred to as “gather”, as we update the brightness of each patch in turn by gathering light from the other patches.

Iterative Solution (cast)

Iteration has the additional benefit that we can provide intermediate, approximate, solutions while the user waits.

But, in the previous version, each patch gets better in turn. Better to have all patches get a little better on each iteration.

This leads to the alternative approach where energy is cast from each patch in turn to update the others with:

$$B_j \quad due \quad to \quad B_i \quad is \quad p_j B_i F_{i\to j} \frac{A_i}{A_j} \qquad (\text{do } \forall \ B_j)$$

A second advantage is that the casting can be done in order of brightness, which is obviously helpful.

Modeling

- Need to usefully represent objects in the world
- Need to provide for easy interaction
  - manual modeling
    - user would like to "fiddle" until it is right (e.g. CAD)
    - user has an idea what an object is like
  - fitting to measurements
    - laser range finder data
- Support rendering/geometric computations

Modeling tools

- Polygon meshes
- Fitting curves to points (from data)
- Fitting curves to points (user interaction)
- Generating shapes with sweeps
- Constructive solid geometry

Polygon Meshes

- Common, straightforward, often built in (e.g. torus mesh)
- Ready to render (many of the representations discussed soon are often be reduced to polygon meshes for rendering)
- Problems
  - Awkward to provide user editing
  - The number of polygons can be very large
    - Some kind of adaptive process makes sense
    - More polygons at high curvature points
    - More polygons where the object is larger
    - Extra care then needs to be taken to avoid temporal aliasing

Explicit curve representation

- Usual representation learned first
- Generally less useful in graphics, but know the term
- Explicit curve is a function of one variable. Examples
  - line, $y=m*x + b$
  - circle (need to glue two together) $y = \pm\mathrm{sqrt}(r*r - x*x)$
- Explicit surface is a function of two variables. Examples
  - plane $z = m*x+ n*y + b$

Implicit representation

- Also less useful for this section, but again, know the term
- An implicit curve is given by the vanishing of some functions
  - circle on the plane (2D), x*x + y*y - r*r = 0
  - twisted cubic in space,
    - x*y - z = 0, x*z - y*y=0, x*x - y = 0
- Similarly, an implicit surface is given by the vanishing of some functions
  - sphere in space x*x + y*y + z*z - r*r = 0
  - plane a*x + b*y + c*z + d = 0

Parametric representation

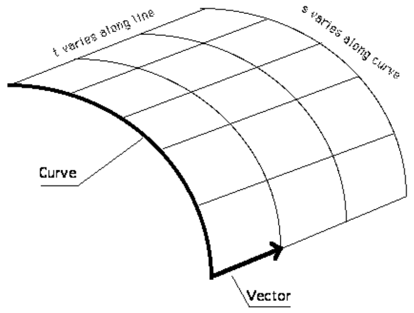
- A parametric **curve** is given as a function of one parameter. Examples:
  - circle as (cos(t), sin(t))
  - twisted cubic as (t, t*t, t*t*t)
- A parametric **surface** is given as a function of two parameters. Examples:
  - sphere as (cos(s) cos(t), sin(s)cos(t), sin(t))
- Advantage - easy to compute normal, easy to render, easy to put patches together, ranges can be easy (e.g. half circle)
- Disadvantage - intersecting with rays for ray tracing can be hard

Generating Surfaces

- We can construct surfaces from curves in a variety of user intuitive ways
  - Extruded surfaces
  - Generalized cones
  - Surfaces of revolution
  - Sweeping (generalized cylinders)
- In many the examples that follow, we will assume that we know how to generate a 3D parameteric curve (studied later)
  - e.g. twisted cubic as (t, t*t, t*t*t)

Extruded surfaces

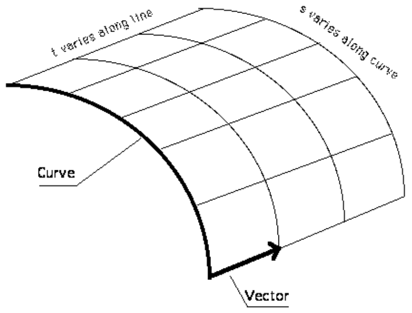
- Geometrical model - Pasta machine
- Take curve and “extrude” surface along vector
- Many human artifacts have this form - rolled steel, etc.



Parametric formula?

Extruded surfaces

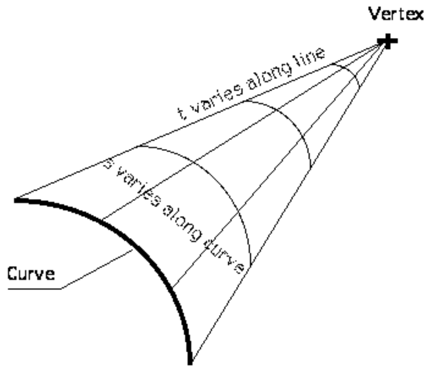
- Geometrical model - Pasta machine
- Take curve and "extrude" surface along vector
- Many human artifacts have this form - rolled steel, etc.



$$(x(s,t), y(s,t), z(s,t)) = (x_c(s), y_c(s), z_c(s)) + t(v_0, v_1, v_2)$$

Cones

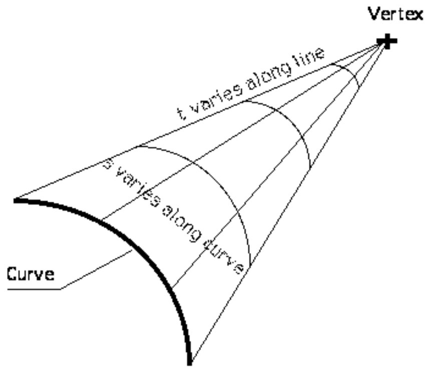
- From every point on a curve, construct a line segment through a single fixed point in space - the vertex
- Curve can be space or plane curve, but shouldn't pass through the vertex



Parametric formula?

Cones

- From every point on a curve, construct a line segment through a single fixed point in space - the vertex
- Curve can be space or plane curve, but shouldn't pass through the vertex

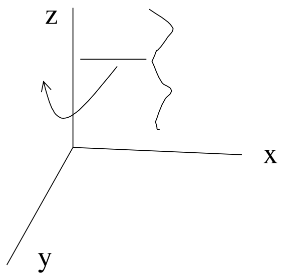


$$(x(s,t), y(s,t), z(s,t)) = (1-t)(x_c(s), y_c(s), z_c(s)) + t(v_0, v_1, v_2)$$

Surfaces of revolution

- Plane curve + axis
- "spin" plane curve around axis to get surface
- Choice of plane is arbitrary, choice of axis affects surface
- In the example to the right, curve is on x-z plane, axis is z axis.
- So curve is $(x_c(s), z_c(s))$

Parametric formula?

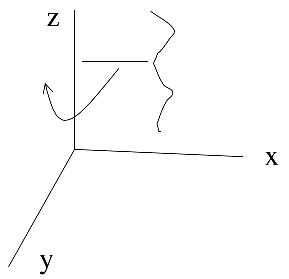


Surfaces of revolution

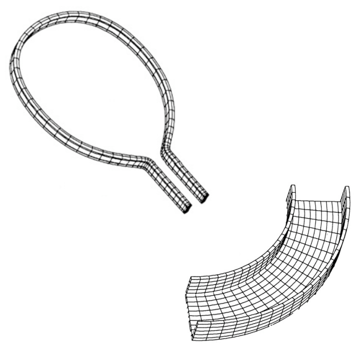
- Plane curve + axis
- "spin" plane curve around axis to get surface
- Choice of plane is arbitrary, choice of axis affects surface
- In the example to the right, curve is on x-z plane, axis is z axis. (Think of $x_c(s)$ as a radius)

$$(x(s,t), y(s,t), z(s,t)) =$$

$$(x_c(s)\cos(t), x_c(s)\sin(t), z_c(s))$$



Sweeps/Generalized Cylinders



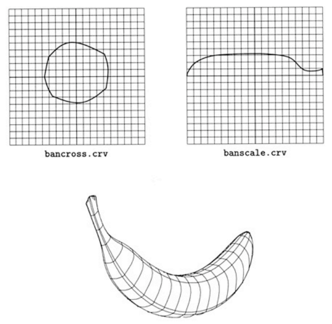
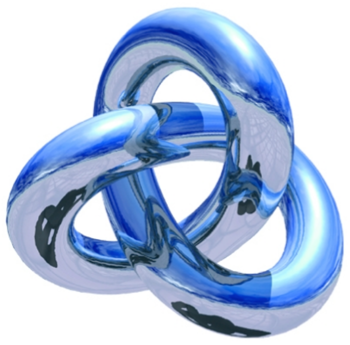


Figure 3.8: Banana example. A banana is represented by an affine transformation surface. The cross section is scaled, translated along z from -1 to 1, and rotated around the y axis.
□

[Synder 92, via CMU course page]

Sweeps/Generalized Cylinders



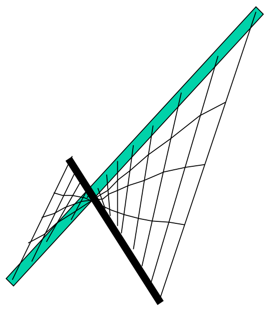
MetaCreations, via CMU course page

Ruled surfaces -1

- Popular, because it's easy to build a curved surface out of straight segments - e.g. pavilions, etc.
- Take two space curves, and join corresponding points—same *s* parameter value—with line segment.
- Even if space curves are lines, the surface is usually curved.

Ruled Surfaces - 2

Easy to explain,
hard to draw!



Ruled surfaces -3

Parameterized form

$$\begin{aligned}(x(s,t), y(s,t), z(s,t)) = \\ (1-t)(x_1(s), y_1(s), z_1(s)) + \\ t(x_2(s), y_2(s), z_2(s))\end{aligned}$$

Normals

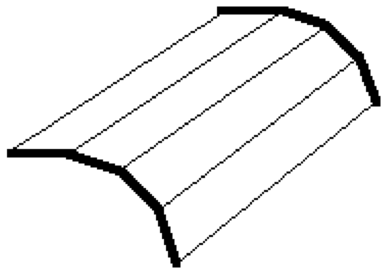
- Normal is cross product of tangent in t direction and s direction.

$$\left(\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}, \frac{\delta z}{\delta t}\right) \times \left(\frac{\delta x}{\delta s}, \frac{\delta y}{\delta s}, \frac{\delta z}{\delta s}\right)$$

- Examples
  - Cylinder: normal is cross-product of curve tangent and direction vector
  - Surface of revolution: take curve normal and spin round axis

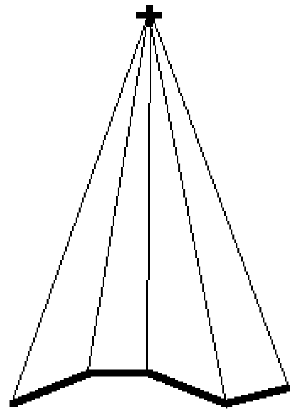
Rendering

- Cylinders: small steps along curve, straight segments along t generate polygons; exact normal is known.



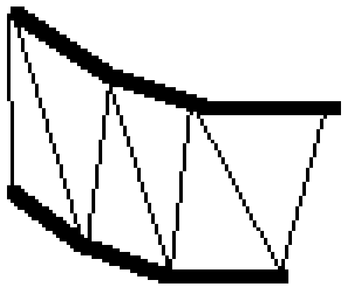
Rendering

- Cone: small steps in s generate straight edges, join with vertex to get triangles, normals known exactly except at vertex.



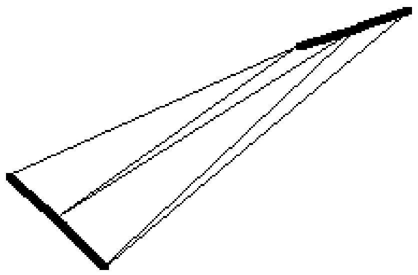
Rendering

- Surface of revolution: small steps in s generate strips, small steps in t along the strip generate edges; join up to form triangles. Normals known exactly.



Rendering

- Ruled surface: steps in s generate polygons, join opposite sides to make triangles - otherwise “non planar polygons” result. Normals known exactly.
- **Must** understand why rectangular sections do not work!



Specifying Curves from Points

- Want to modulate curves via “control” points.
- Strategy depends on application. Possibilities:
  - Force a polynomial of degree N-1 through N points (Lagrange interpolate)
  - Specify a combination of “anchor” points and derivatives (Hermite interpolate)
  - Other “blends” (Bezier, B-splines)--more useful than Lagrange/Hermite

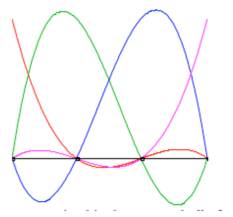
Specifying Curves from Points-II

- Issues:
  - Continuity of curve and derivatives (geometric, parametric)
  - Local versus global control
  - Polynomials verses other forms
  - Higher polynomial degree versus stitching lower order polynomials together
  - Polynomial degree (usually 3--fewer is not flexible enough, and higher gives hard to control wiggles).
  - It is relatively easy to fit a curve through points in explicit form, but we will use parametric form as it more useful in graphics.

Lagrange Interpolate (degree 3)

- Want a parametric curve that passes through (interpolates) four points.
- Use the points to combine four Lagrange polynomials (blending functions)
- As the parameter goes through each 4 particular values, one blending function is 1, and the other 3 are zero.

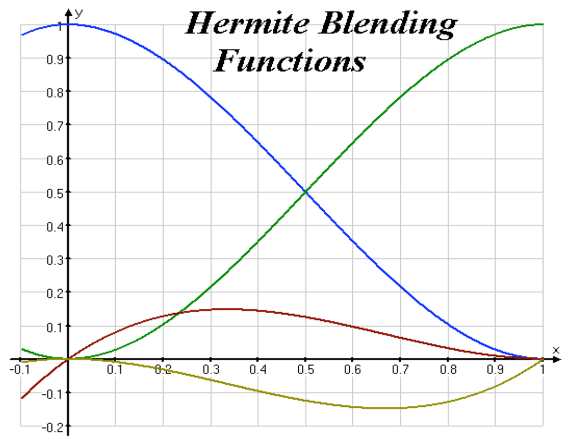
$$\sum_{i \in \text{points}} p_i \phi_i^{(l)}(t)$$



Hermite curves (degree 3)

- Hermite interpolate
  - Curve passes through specified points **and** has specified derivatives at those points.
- Standard degree 3 case: 2 points, 2 derrivatives at those points
- 4 functions of degree 3, two each of two kinds
  - one at an endpoint, zero at the other, AND derrivative is zero at both
  - derivative is one at an endpoint and zero at others, AND value is zero at the endpoints.

$$\sum_{i \in \text{points}} p_i \phi_i^{(h)}(t) + \sum_{i \in \text{points}} v_i \phi_i^{(hd)}(t)$$



From www.cs.virginia.edu/~gfx/Courses/ 2002/Intro.fall.02

Blended curves

- Assume degree 3
- Includes Hermite, Bézier and others

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = C * T(t) = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Blended curves

- General pattern: Decompose the matrix C into two factors
  - One factor encodes the "control" or data points
  - The second factor is the blending functions

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = [G_1 \quad G_2 \quad G_3 \quad G_4] \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Hermite

- Geometry matrix
  - First two columns are endpoints
  - Next two columns are derivatives at those points

$$M_H = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Where does this come from?

Hermite

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = [G_1 \quad G_2 \quad G_3 \quad G_4] M_H \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Can solve for M_H using x(t) only (or y(t), or z(t))

$$x(t) = [G_{1x} \quad G_{2x} \quad G_{3x} \quad G_{4x}] * M_H * T(t)$$

Hermite

$$x(t) \;=\; [G_{1x} \quad G_{2x} \quad G_{3x} \quad G_{4x}]\; M_H * \mathbf{T}(t)$$

We have

$x(0) = G_{1x}$ so $M_H * T(0) = [1\ 0\ 0\ 0\ ]$
$x(1) = G_{2x}$ so $M_H * T(1) = [0\ 1\ 0\ 0\ ]$
$x'(0) = G_{3x}$ so $M_H * T'(0) = [0\ 0\ 1\ 0\ ]$
$x'(1) = G_{4x}$ so $M_H * T'(1) = [0\ 0\ 0\ 1\ ]$

Hermite

$x(0) = G_{1x}$ so $M_H * \mathbf{T}(0) = [1\ 0\ 0\ 0\ ]$
$x(1) = G_{2x}$ so $M_H * \mathbf{T}(1) = [0\ 1\ 0\ 0\ ]$
$x'(0) = G_{3x}$ so $M_H * \mathbf{T}'(0) = [0\ 0\ 1\ 0\ ]$
$x'(1) = G_{4x}$ so $M_H * \mathbf{T}'(1) = [0\ 0\ 0\ 1\ ]$

So

$$M_H * \begin{bmatrix}\mathbf{T(0)} & \mathbf{T(1)} & \mathbf{T'(0)} & \mathbf{T'(1)}\end{bmatrix} = I$$

So

$$M_H = \begin{bmatrix}\mathbf{T(0)} & \mathbf{T(1)} & \mathbf{T'(0)} & \mathbf{T'(1)}\end{bmatrix}^{-1}$$

Hermite

$$\mathbf{T}(t) = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \qquad \mathbf{T}'(t) = \begin{bmatrix} 3t^2 \\ 2t^2 \\ 1 \\ 0 \end{bmatrix}$$

So, **T**(0), **T**(1), **T’**(0), and **T’**(1) are?

Hermite

$$\mathbf{T}(t) = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \qquad \mathbf{T}'(t) = \begin{bmatrix} 3t^2 \\ 2t^2 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{T(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{T}(1) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{T'(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{T}'(1) = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

Hermite

Recall that

$$M_H * \begin{bmatrix} \mathbf{T(0)} & \mathbf{T(1)} & \mathbf{T'(0)} & \mathbf{T'(1)} \end{bmatrix} = I$$

And thus we seek

$$M_H = \begin{bmatrix} \mathbf{T(0)} & \mathbf{T(1)} & \mathbf{T'(0)} & \mathbf{T'(1)} \end{bmatrix}^{-1}$$

$$M_H = \begin{vmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{vmatrix}^{-1} = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Hermite

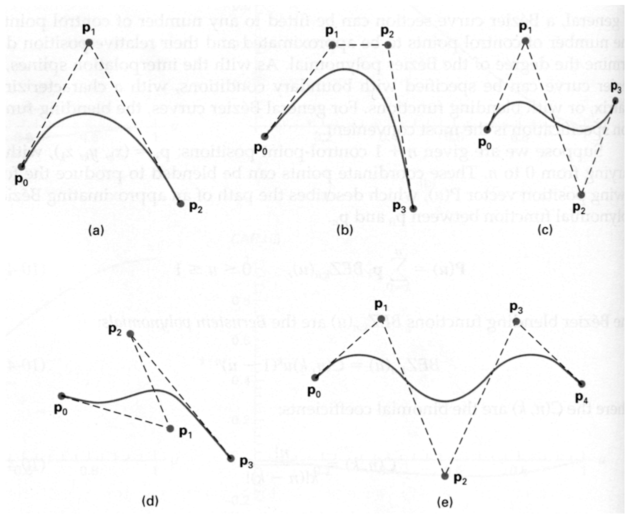
Finally

$$M_H = \begin{vmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{vmatrix}^{-1} = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Bézier

- Curve goes through two of the control points
- Curve is adjusted by moving two (cubic case) **other** control points
- Tangent at endpoints is in direction of adjacent control point
- Curve lies in convex hull of all 4 (cubic case) control points.

Example Bézier Curves



Bézier

- Geometry matrix based on Hermite case where
  - First two columns are endpoints
  - Next two are like derivatives from the Hermite case, but are now defined by
    $$R_1 = 3(P_2 - P_1)$$
    $$R_2 = 3(P_4 - P_3)$$
  - Note that this gives our condition on endpoint tangents
  - Factor of 3 gives good “balance” in control point effect, and is needed to be consistent with other derivations (e.g., Bernstein polynomials, subdivision, etc).

$$R_1 = 3(P_2 - P_1)$$
$$R_2 = 3(P_4 - P_3)$$

Means that

$$\begin{bmatrix} P_1 & P_4 & R_1 & R_2 \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{bmatrix}$$

Translation to Hermite case

M_{HB}

Recall Hermite

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} P_1 & P_4 & R_1 & R_2 \end{bmatrix} M_H \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

From previous slide

$$\begin{bmatrix} P_1 & P_4 & R_1 & R_2 \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \end{bmatrix} M_{HB}$$

So, for Bézier

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \end{bmatrix} M_{HB} M_H \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Want M_B in $Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \end{bmatrix} M_B \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$

$$M_B = M_{HB} M_H$$

$$= \begin{bmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

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

Bézier in standard form (summary)

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \end{bmatrix} M_B \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

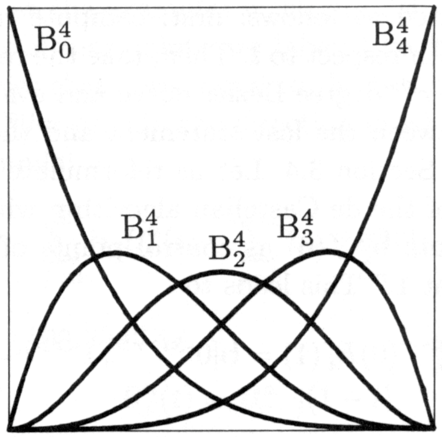
$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Bézier curves - II

- The blending functions are the Bernstein polynomials

$$c(t) = \sum_{i=0}^{n} p_i B_i^n(t)$$

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

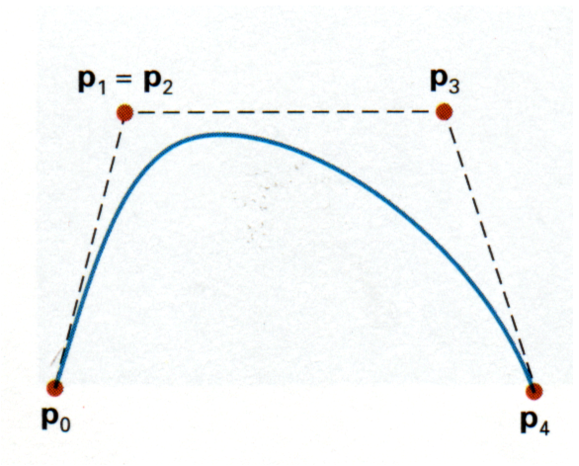


Bézier curves - III

- Bernstein polynomials have several important properties
  - they sum to 1, hence curve lies within convex hull of control points
  - curve interpolates its endpoints
  - curve's tangent at start lies along the vector from p_0 to p_1
  - tangent at end lies along vector from p_{n-1} to p_n

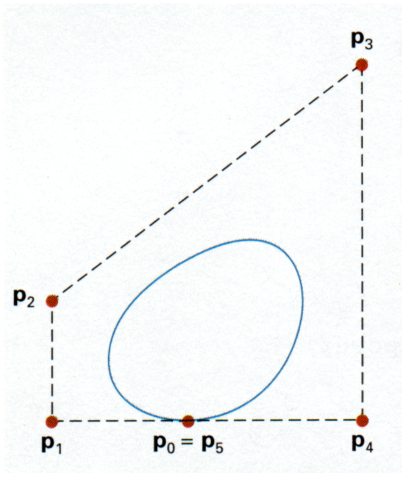
Bézier curve tricks - I

- "Pull" a curve toward a control point by doubling the control point



Bézier curve tricks-II

- Close the curve by making last point and first point coincident
  - curve has continuous tangent if first segment and last segment are collinear



Interpolating Splines

- Key idea:
  - high degree interpolates are badly behaved->
  - construct curves out of low degree segments

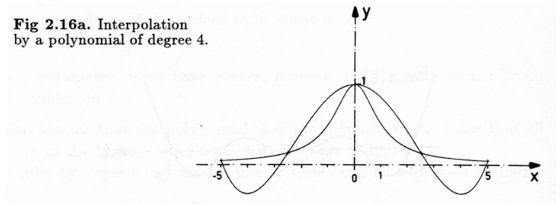


Fig 2.16a. Interpolation by a polynomial of degree 4.

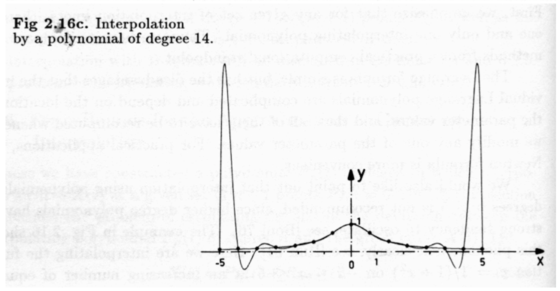


Fig 2.16c. Interpolation by a polynomial of degree 14.

Interpolating Splines - II

- n+1 points;
- write derivatives X'
- X_i is spline for interval between P_i and P_{i+1}

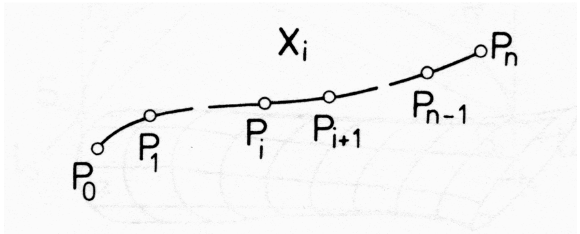


Fig. 3.11. The spline segment X_i.

Interpolating Splines - II

- Bolt together a series of Hermite curves with derivatives matching at joints (Knots).
- But where are the derivative values to come from?
  - Measurements
  - Combination of points (see cardinal splines--next topic)
  - Continuity considerations
  - Conventions for enpoints

- Cardinal splines

Equation Optional

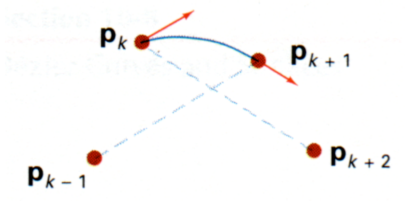
$$P_k' = \left(\frac{1}{2}\right)(1-t)\left(P_{k+1} - P_{k-1}\right)$$

  - t is "tension"
  - still need to specify endpoint tangents
    - or use difference between first two, last two points

(Don't confuse t with parameter!)

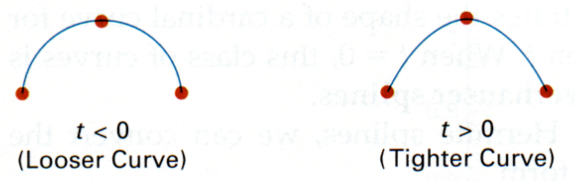
Details Optional

Tension



- larger values of tension give tighter curves (limit (as t-->1) is linear interpolate).

(Don't confuse t with parameter!)



Interpolating Splines

- Intervals:

$$a = t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N = b.$$

$$\Delta t_i := t_{i+1} - t_i.$$

  - t values often called "knots"

- Spline form:

$$\boldsymbol{X}_i(t) := \boldsymbol{A}_i(t-t_i)^3 + \boldsymbol{B}_i(t-t_i)^2 + \boldsymbol{C}_i(t-t_i) + \boldsymbol{D}_i,$$
$$t \in [t_i, t_{i+1}], \qquad i = 0(1)N\text{–}1,$$

Continuity

- Require at endpoints:
  - endpoints equal
  - 1'st derivatives equal
  - 2'nd derivatives equal

- Now we get extra information from continuity (instead of tension equation, tangent measurements, etc)

$$\begin{aligned}
\boldsymbol{X}_i(t_i) &= \boldsymbol{X}_{i-1}(t_i) &\text{or}\quad \boldsymbol{X}_i(t_{i+1}) &= \boldsymbol{X}_{i+1}(t_{i+1}),\\
\boldsymbol{X}'_i(t_i) &= \boldsymbol{X}'_{i-1}(t_i) &\text{or}\quad \boldsymbol{X}'_i(t_{i+1}) &= \boldsymbol{X}'_{i+1}(t_{i+1}),\\
\boldsymbol{X}''_i(t_i) &= \boldsymbol{X}''_{i-1}(t_i) &\text{or}\quad \boldsymbol{X}''_i(t_{i+1}) &= \boldsymbol{X}''_{i+1}(t_{i+1}).
\end{aligned}$$

Details Optional

- From endpoint and 1'st derivative:

$$\boldsymbol{X}_i(t_i) = \boldsymbol{P}_i = \boldsymbol{D}_i, \qquad \boldsymbol{X}_i(t_{i+1}) = \boldsymbol{P}_{i+1} = \boldsymbol{A}_i\Delta t_i^3 + \boldsymbol{B}_i\Delta t_i^2 + \boldsymbol{C}_i\Delta t_i + \boldsymbol{D}_i,$$
$$\boldsymbol{X}'_i(t_i) = \boldsymbol{P}'_i = \boldsymbol{C}_i, \qquad \boldsymbol{X}'_i(t_{i+1}) = \boldsymbol{P}'_{i+1} = 3\boldsymbol{A}_i\Delta t_i^2 + 2\boldsymbol{B}_i\Delta t_i + \boldsymbol{C}_i,$$

- So that

$$\boldsymbol{A}_i = \frac{1}{(\Delta t_i)^3}[2(\boldsymbol{P}_i - \boldsymbol{P}_{i+1}) + \Delta t_i(\boldsymbol{P}'_i + \boldsymbol{P}'_{i+1})],$$
$$\boldsymbol{B}_i = \frac{1}{(\Delta t_i)^2}[3(\boldsymbol{P}_{i+1} - \boldsymbol{P}_i) - \Delta t_i(2\boldsymbol{P}'_i + \boldsymbol{P}'_{i+1})].$$

- Yielding:

$$\begin{aligned}
\boldsymbol{X}_i(t) =\ & \boldsymbol{P}_i\left(2\frac{(t-t_i)^3}{(\Delta t_i)^3} - 3\frac{(t-t_i)^2}{(\Delta t_i)^2} + 1\right) + \boldsymbol{P}_{i+1}\left(-2\frac{(t-t_i)^3}{(\Delta t_i)^3} + 3\frac{(t-t_i)^2}{(\Delta t_i)^2}\right)\\
& + \boldsymbol{P}'_i\left(\frac{(t-t_i)^3}{(\Delta t_i)^2} - 2\frac{(t-t_i)^2}{\Delta t_i} + (t-t_i)\right) + \boldsymbol{P}'_{i+1}\left(\frac{(t-t_i)^3}{(\Delta t_i)^2} - \frac{(t-t_i)^2}{\Delta t_i}\right)
\end{aligned}$$

Details Optional

- Second Derivative:

$$X_i''(t) = 6P_i\left(\frac{2(t-t_i)}{(\Delta t_i)^3} - \frac{1}{(\Delta t_i)^2}\right) + 6P_{i+1}\left(-2\frac{(t-t_i)}{(\Delta t_i)^3} + \frac{1}{(\Delta t_i)^2}\right) + 2P_i'\left(3\frac{(t-t_i)}{(\Delta t_i)^2} - \frac{2}{\Delta t_i}\right) + 2P_{i+1}'\left(\frac{3(t-t_i)}{(\Delta t_i)^2} - \frac{1}{\Delta t_i}\right).$$

- Want:

$$X_{i-1}''(t_i) = X_i''(t_i)$$

- Yielding:

$$\Delta t_i P_{i-1}' + 2(\Delta t_{i-1} + \Delta t_i)P_i' + \Delta t_{i-1}P_{i+1}' = 3\frac{\Delta t_{i-1}}{\Delta t_i}(P_{i+1} - P_i) + 3\frac{\Delta t_i}{\Delta t_{i-1}}(P_i - P_{i-1}).$$

Missing equations

- Recurrence relations represent d(n-1) equations in d(n+1) unknowns (d is dimension)
- We need to supply the derivative at the start and at the finish (or two equivalent constraints)
- Options:
  - second derivatives vanish at each end (natural spline)
  - give slopes at the boundary
    - vector from first to second, second last to last
    - parabola through first three, last three points
  - third derivative is the same at first, last knot

B-splines - I

- Now consider stitching together curves which do not necessarily pass through the control points.
- Local control
- Blending functions are non-zero over limited range--thus they are like “switches”
- In the simplest case of uniformly spaced control points, the blending functions will be shifted versions of the same function.

B-splines - II

- Curve (general case):

$$X(t) = \sum_{k=0}^{n} P_k B_{k,d}(t)$$

- The “degree parameter” d is:

$$2 \le d \le n+1$$

- Note the “degree” of the polynomial is d-1 (not d).
- We have n+1 control points, and n+1 blending functions. The most common case is d=4.

B-Spline Blending Functions

- Knots
  - parameter values where curve segments meet
- There are n+d+1 knots for B-spline

  $(t_0, t_1, \ldots, t_{n+d})$

  where $t_0 \le t_1 \le \ldots \le t_{n+d}$
- B-spline is defined for the range $(t_{d-1}, \ldots, t_{n+1})$
  - So all curve points have contributions from d control points
  - All sections of curves behave the same (later we will see how to interpolate the endpoints.
- Blending functions for degree parameter d have support (are non-zero) for d subintervals.

B-Spline Blending Functions

- Blending functions

$$B_{k,1}(t) = \begin{cases} 1 & t_k \le t \le t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{k,d}(t) = \left(\frac{t - t_k}{t_{k+d-1} - t_k}\right) B_{k,d-1}(t) + \left(\frac{t_{k+d} - t}{t_{k+d} - t_{k+1}}\right) B_{k+1,d-1}(t)$$

- If knots are repeated we use 0/0=0

B-Spline Blending Functions

$$B_{k,d}(t) = \left(\frac{t - t_k}{t_{k+d-1} - t_k}\right) B_{k,d-1}(t) + \left(\frac{t_{k+d} - t}{t_{k+d} - t_{k+1}}\right) B_{k+1,d-1}(t) \qquad B_{k,1}(t) = \begin{cases} 1 & t_k \le t \le t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

So, assuming uniformly spaced knots,

$$B_{k,2}(t) = \quad ?$$

B-Spline Blending Functions

$$B_{k,d}(t) = \left(\frac{t - t_k}{t_{k+d-1} - t_k}\right) B_{k,d-1}(t) + \left(\frac{t_{k+d} - t}{t_{k+d} - t_{k+1}}\right) B_{k+1,d-1}(t) \qquad B_{k,1}(t) = \begin{cases} 1 & t_k \le t \le t_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

So, assuming uniformly spaced knots,

$$B_{k,2}(t) = \begin{cases} \left(\frac{t - t_k}{\Delta t}\right) & t_k \le t \le t_{k+1} \\ \left(\frac{t_k - t}{\Delta t}\right) & t_{k+1} \le t \le t_{k+2} \\ 0 & \text{otherwise} \end{cases}$$

B-Spline Blending Functions

$$B_{k,d}(t) = \left(\frac{t - t_k}{t_{k+d-1} - t_k}\right) B_{k,d-1}(t) + \left(\frac{t_{k+d} - t}{t_{k+d} - t_{k+1}}\right) B_{k+1,d-1}(t)$$

$$B_{k,2}(t) = \begin{cases} \left(\frac{t - t_k}{\Delta t}\right) & t_k \le t \le t_{k+1} \\ \left(\frac{t_k - t}{\Delta t}\right) & t_{k+1} \le t \le t_{k+2} \\ 0 & \text{otherwise} \end{cases}$$

So, assuming uniformly spaced knots,

$$B_{k,3}(t) = \quad ?$$

B-Spline Blending Functions

$$B_{k,d}(t) = \left(\frac{t - t_k}{t_{k+d-1} - t_k}\right) B_{k,d-1}(t) + \left(\frac{t_{k+d} - t}{t_{k+d} - t_{k+1}}\right) B_{k+1,d-1}(t)$$

$$B_{k,2}(t) = \begin{cases} \left(\frac{t - t_k}{\Delta t}\right) & t_k \le t \le t_{k+1} \\ \left(\frac{t_k - t}{\Delta t}\right) & t_{k+1} \le t \le t_{k+2} \\ 0 & \text{otherwise} \end{cases}$$

So, assuming uniformly spaced knots,

$$B_{k,3}(t) = \begin{cases} \text{quadratic function} & t_k \le t \le t_{k+1} \\ \text{quadratic function} & t_{k+1} \le t \le t_{k+2} \\ \text{quadratic function} & t_{k+2} \le t \le t_{k+3} \\ 0 & \text{otherwise} \end{cases}$$

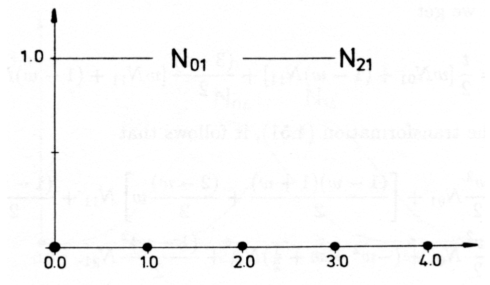


Fig. 4.22c. The B-splines N_{01}, N_{21}.

These figures show blending functions with a uniform knot vector, knots at 0, 1, 2, etc. Note that N is the same as our B

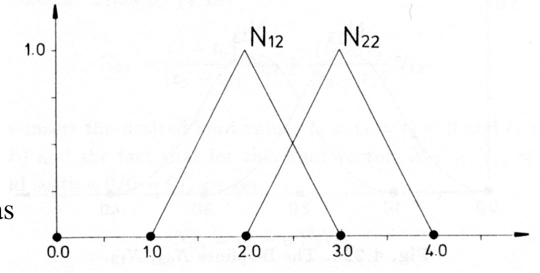


Fig. 4.22d. The B-splines N_{12}, N_{22}.

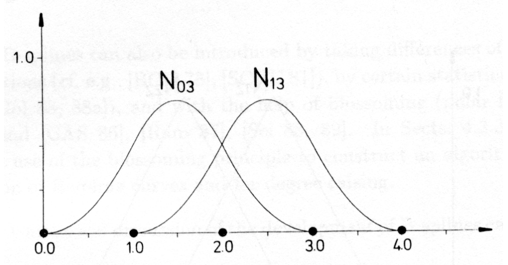
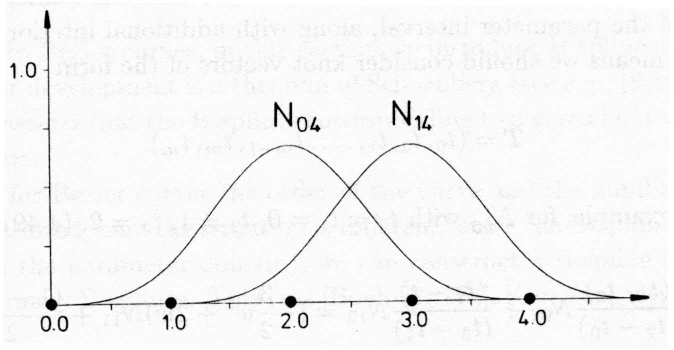


Fig. 4.22e. The B-splines N_{03}, N_{13}.



Matrix form of Uniform Cubic B-Spline Blending Functions

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

Closed B-Splines

- Periodically extend the control points and the knots

$$P_{n+1} = P_0$$
$$t_{n+1} = t_0$$

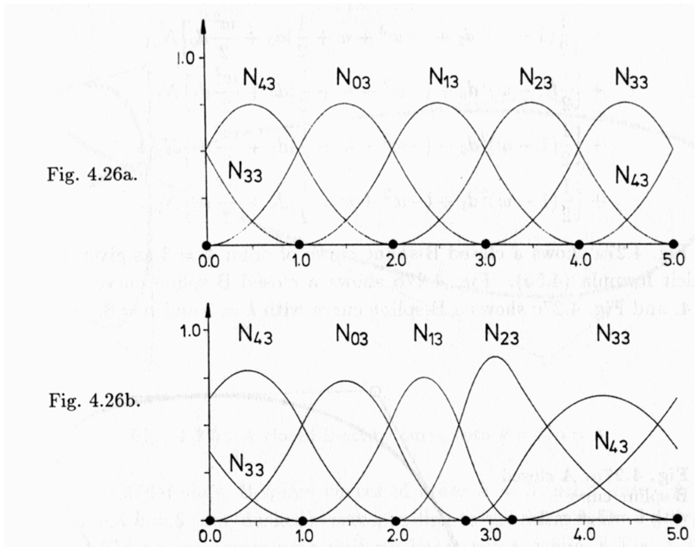


Fig. 4.26a.

Fig. 4.26b.

Fig. 4.26. B-splines with uniform and non-uniform knot vectors for a closed B-spline curve.

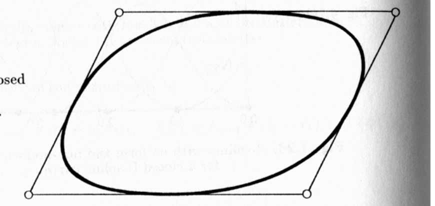


Fig. 4.27a. A closed B-spline curve with $k = 3, n = 3$.

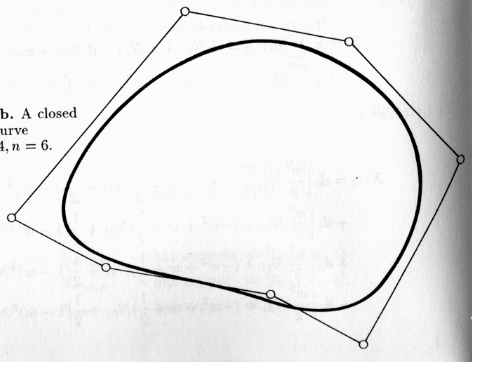


Fig. 4.27b. A closed B-spline curve with $k = 4, n = 6$.

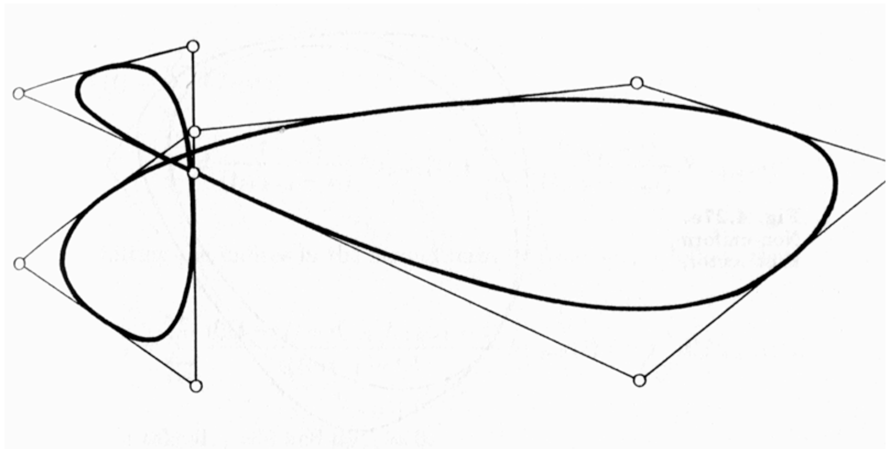
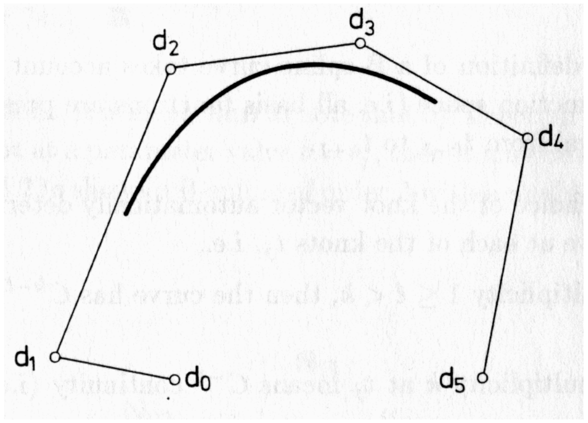


Fig. 4.27c. A closed B-spline curve with $k = 3, n = 8$.



Recall that each curve section is a blend of d control points.

What if we want to interpolate the endpoints?

Repeated knots

- Definition works for repeated knots (if we are understanding about 0/0)
- Repeated knot reduces continuity. A B-spline blending function has continuity C^{d-2}; if the knot is repeated m times, continuity is now C^{d-m-1}
- e.g. -> quadratic B-spline (i.e. order 3) with a double knot

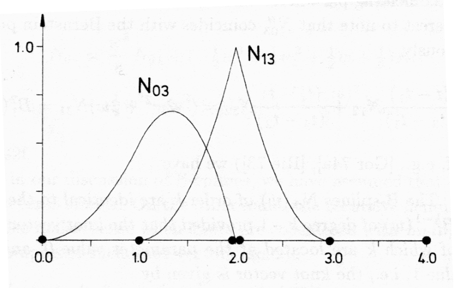


Fig. 4.22g. A quadratic B-spline with a double knot.

Most useful case

- Select the first d and the last d knots to be the same
  - we then get the first and last points lying on the curve
  - also, the curve is tangent to the first and last segment
- E.g. cubic case below
- Notice that a control point influences at most d parameter intervals - **local control**

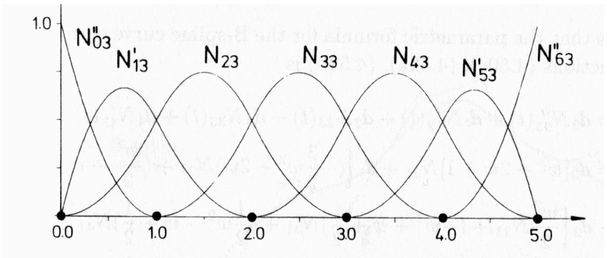


Fig. 4.24a. B-splines for an open B-spline curve with uniform knot vector.

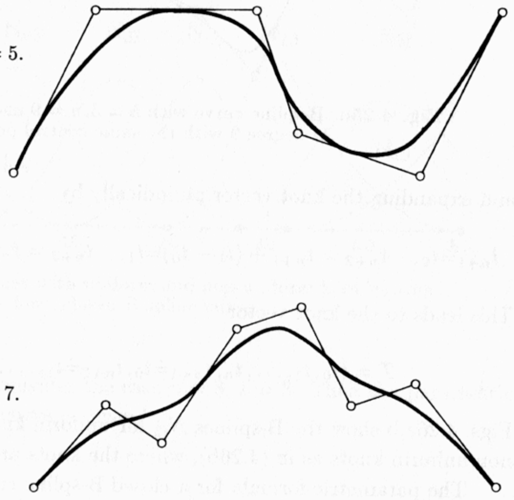


Fig. 4.25a. B-spline curve with $k = 3$, $n = 5$.

Fig. 4.25b. B-spline curve with $k = 4$, $n = 7$.

k is our d - top curve has order 3, bottom order 4

Example of blending function with repeated knots at the endpoints and non-uniform spacing of interior knots

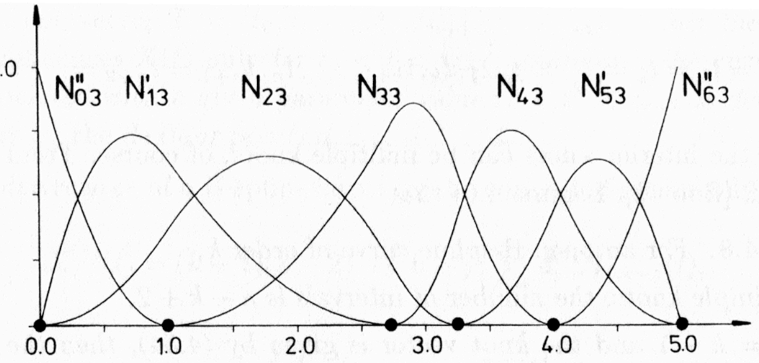


Fig. 4.24b. B-splines for an open B-spline curve with non-uniform knot vector.

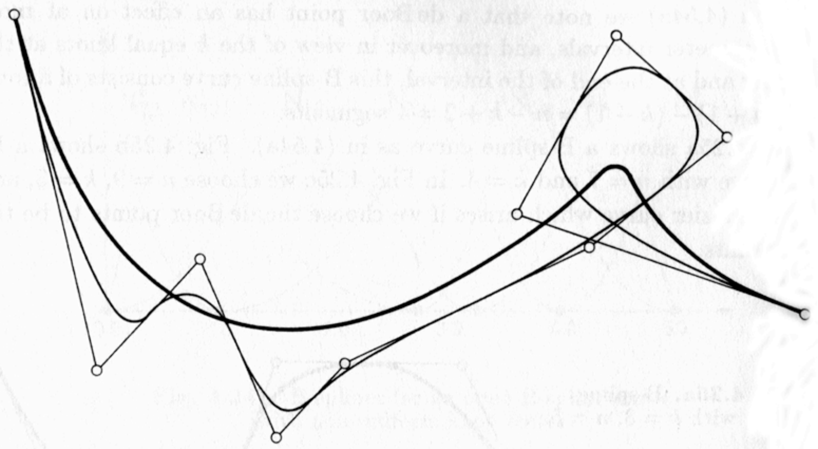


Fig. 4.25c. B-spline curve with $k = 3, n = 9$ and the Bézier curve of degree 9 with the same control polygon.

Bézier curve is the heavy curve

B-Spline properties

- For a B-spline curve of order d
  - if m knots coincide, the curve is C^{d-m-1} at the corresponding point
  - if d-1 consecutive* points of the control polygon are collinear, then the curve is tangent to the polygon
  - if d consecutive* points of the control polygon are collinear, then the curve and the polygon have a common segment
  - if d-1 points coincide, then the curve interpolates the common point and the two adjacent sides of the polygon are tangent to the curve
  - each segment of the curve lies in the convex hull of the associated d points

*The fish shaped curve a few slides back have 4 collinear points (d=3), but they are not consecutive so the condition does not hold.