CS 445 / 645
Introduction to Computer Graphics

Lecture 23

Bézier Curves

UNIVERSITY of VIRGINIA

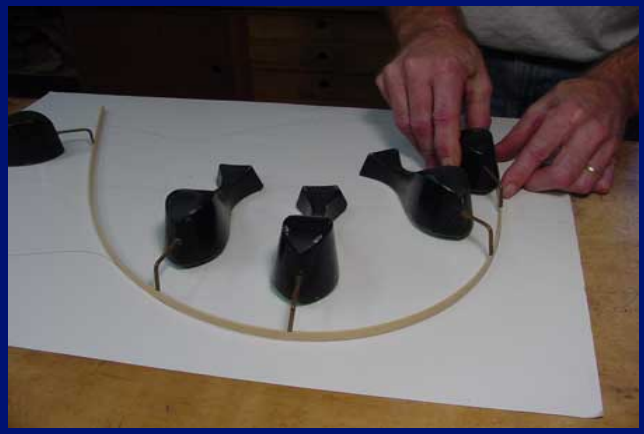
Splines - History

Draftsman use 'ducks' and strips of wood (splines) to draw curves

Wood splines have second-order continuity

And pass through the control points

A Duck (weight)



Ducks trace out curve

Representations of Curves

Problems with series of points used to model a curve

- Piecewise linear - Does not accurately model a smooth line
- It's tedious
- Expensive to manipulate curve because all points must be repositioned

Instead, model curve as piecewise-polynomial

- x = x(t), y = y(t), z = z(t)
  - where x(), y(), z() are polynomials

Specifying Curves (hyperlink)

Control Points

- A set of points that influence the curve's shape

Knots

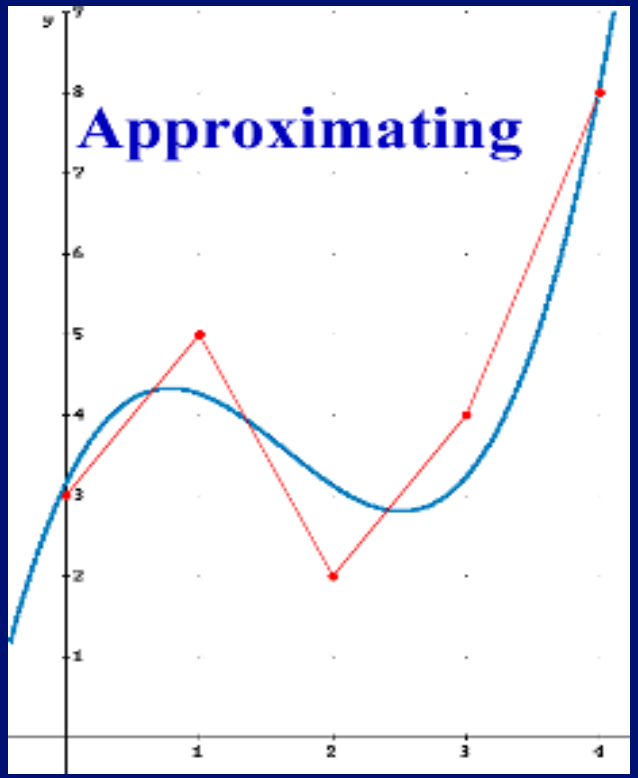
- Control points that lie on the curve

Interpolating Splines

- Curves that pass through the control points (knots)

Approximating Splines

- Control points merely influence shape

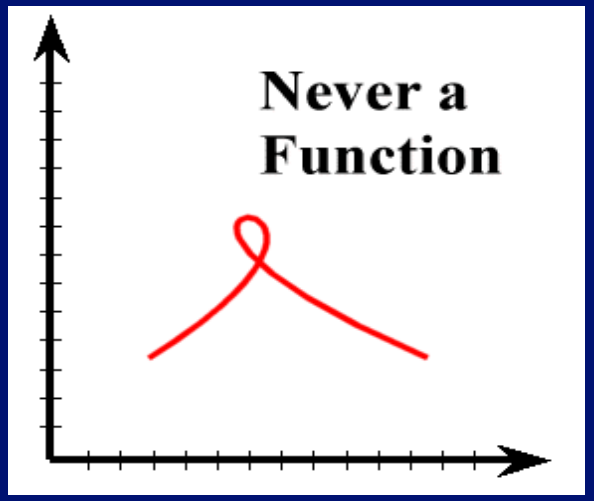


Parametric Curves

Very flexible representation

They are not required to be functions

- They can be multivalued with respect to any dimension



Cubic Polynomials

$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$

- Similarly for y(t) and z(t)

Let t: (0 <= t <= 1)

Let T = [t^3 t^2 t 1]

Coefficient Matrix C

$$\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} * \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Curve: Q(t) = T*C

Parametric Curves

How do we find the tangent to a curve?

- If $f(x) = x^2 - 4$
  - tangent at (x=3) is $f'(x) = 2\,(x) - 4 = 2\,(3) - 4$

Derivative of Q(t) is the tangent vector at t:

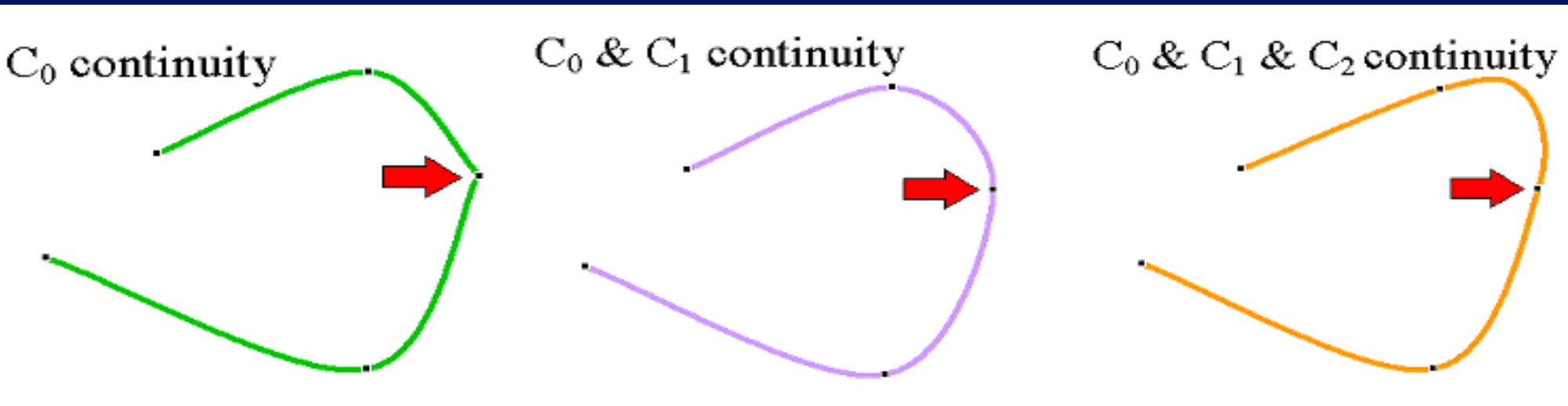
- $d/dt\ Q(t) = Q'(t) = d/dt\ T * C = [3t^2\ 2t\ 1\ 0] * C$

Piecewise Curve Segments

One curve constructed by connecting many smaller segments end-to-end

Continuity describes the joint

- C_1 is tangent continuity (velocity)
- C_2 is 2^{nd} derivative continuity (acceleration)



Continuity of Curves

If direction (but not necessarily magnitude) of tangent matches

- G^1 geometric continuity
- The tangent value at the end of one curve is proportional to the tangent value of the beginning of the next curve

Matching direction and magnitude of d^n / dt^n

- C^n continous

Parametric Cubic Curves

In order to assure C_2 continuity, curves must be of at least degree 3

Here is the parametric definition of a cubic (degree 3) spline in two dimensions

How do we extend it to three dimensions?

$$x = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y = a_y t^3 + b_y t^2 + c_y t + d_y$$

Parametric Cubic Splines

Can represent this as a matrix too

$$x = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$\begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix}$$

Coefficients

So how do we select the coefficients?

- $[a_x\ b_x\ c_x\ d_x]$ and $[a_y\ b_y\ c_y\ d_y]$ must satisfy the constraints defined by the knots and the continuity conditions

Parametric Curves

Difficult to conceptualize curve as

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

(artists don't think in terms of coefficients of cubics)

Instead, define curve as weighted combination of 4 well-defined cubic polynomials

(wait a second! Artists don't think this way either!)

Each curve type defines different cubic polynomials and weighting schemes

Parametric Curves

Hermite – two endpoints and two endpoint tangent vectors

Bezier - two endpoints and two other points that define the endpoint tangent vectors

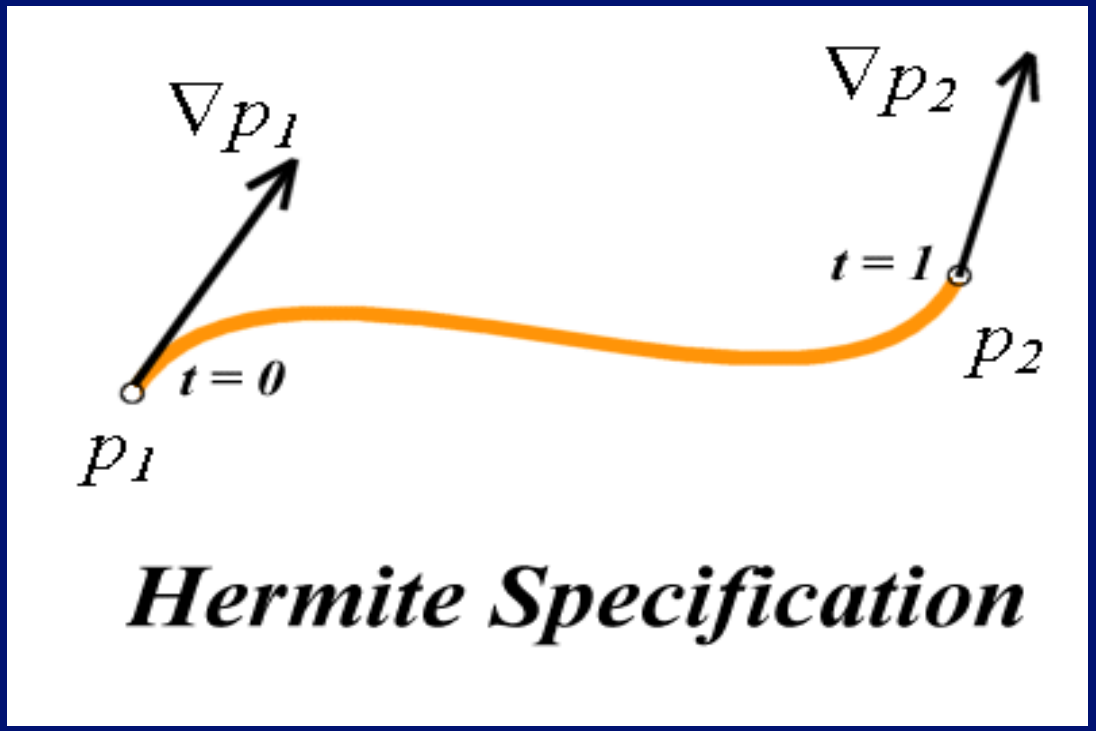
Splines – four control points

- C1 and C2 continuity at the join points
- Come close to their control points, but not guaranteed to touch them

Examples of Splines

Hermite Cubic Splines

An example of knot and continuity constraints



Hermite Cubic Splines

One cubic curve for each dimension

A curve constrained to x/y-plane has two curves:

$$f_x(t) = at^3 + bt^2 + ct + d$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$f_y(t) = et^3 + ft^2 + gt + h$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix}$$

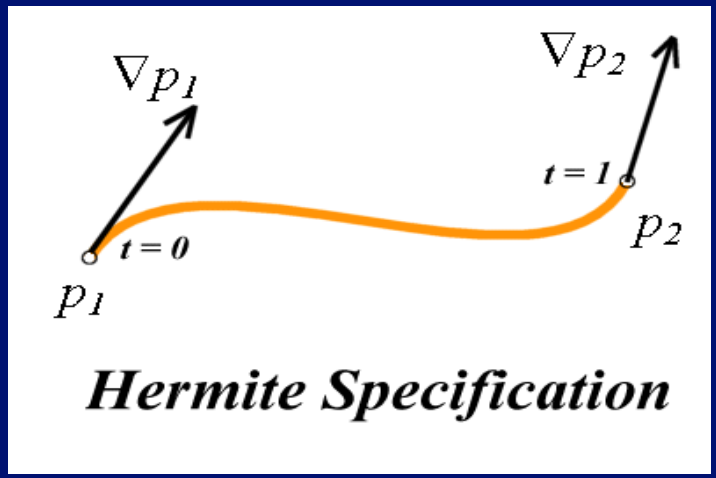
Hermite Cubic Splines

A 2-D Hermite Cubic Spline is defined by eight parameters: a, b, c, d, e, f, g, h

How do we convert the intuitive endpoint constraints into these (relatively) unintuitive eight parameters?

We know:

- (x, y) position at t = 0, p_1
- (x, y) position at t = 1, p_2
- (x, y) derivative at t = 0, dp/dt
- (x, y) derivative at t = 1, dp/dt



Hermite Specification

Hermite Cubic Spline

We know:

- (x, y) position at t = 0, p_1

$$f_x(0) = a0^3 + b0^2 + c0 + d$$

$$= \begin{bmatrix} 0^3 & 0^2 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$f_x(0) = d = p_{1_x}$$

$$f_y(0) = e0^3 + f0^2 + g0 + h$$

$$= \begin{bmatrix} 0^3 & 0^2 & 0 & 1 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix}$$

$$f_y(0) = h = p_{1_y}$$

Hermite Cubic Spline

We know:

- (x, y) position at t = 1, p_2

$$f_x(1) = a1^3 + b1^2 + c1 + d$$

$$= \begin{bmatrix} 1^3 & 1^2 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$f_x(1) = a + b + c + d = p_{2_x}$$

$$f_y(1) = e1^3 + f1^2 + g1 + h$$

$$= \begin{bmatrix} 1^3 & 1^2 & 1 & 1 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix}$$

$$f_y(1) = e + f + g + h = p_{2_y}$$

Hermite Cubic Splines

So far we have four equations, but we have eight unknowns

Use the derivatives

$$f_x(t) = at^3 + bt^2 + ct + d$$

$$f_x'(t) = 3at^2 + 2bt + c$$

$$f_x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$f_y(t) = et^3 + ft^2 + gt + h$$

$$f_y'(t) = 3et^2 + 2ft + g$$

$$f_y'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix}$$

Hermite Cubic Spline

We know:

- (x, y) derivative at t = 0, dp/dt

$$f_x'(0) = 3a0^2 + 2b0 + c$$

$$= \begin{bmatrix} 3\cdot 0^2 & 2\cdot 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$f_x'(0) = c = {dp_{1_x}}/{dt}$$

$$f_y'(0) = 3e0^2 + 2f0 + g$$

$$= \begin{bmatrix} 3\cdot 0^2 & 2\cdot 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix}$$

$$f_y'(0) = g = {dp_{1_y}}/{dt}$$

Hermite Cubic Spline

We know:

- (x, y) derivative at t = 1, dp/dt

$$f_x'(1) = 3a1^2 + 2b1 + c$$

$$= \begin{bmatrix} 3\cdot 1^2 & 2\cdot 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$f_x'(1) = 3a + 2b + c = {dp_{1_x}}/{dt}$$

$$f_y'(1) = 3e1^2 + 2f1 + g$$

$$= \begin{bmatrix} 3\cdot 1^2 & 2\cdot 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix}$$

$$f_y'(1) = 3e + 2f + g = {dp_{1_y}}/{dt}$$

Hermite Specification

Matrix equation for Hermite Curve

$$
\begin{array}{r}
 \\
p_1 \\
p_2 \\
r\,p_1 \\
r\,p_2
\end{array}
\begin{array}{c}
\begin{array}{cccc} t^3 & t^2 & t^1 & t^0 \end{array} \\
\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}
\end{array}
\begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix}
=
\begin{bmatrix} p_{1_x} & p_{1_y} \\ p_{2_x} & p_{2_y} \\ dp_{1_x}/dt & dp_{1_y}/dt \\ dp_{1_x}/dt & dp_{2_y}/dt \end{bmatrix}
\begin{array}{l}
 \\
t = 0 \\
t = 1 \\
t = 0 \\
t = 1
\end{array}
$$

Solve Hermite Matrix

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} p_{1_x} & p_{1_y} \\ p_{2_x} & p_{2_y} \\ dp_{1_x}/dt & dp_{1_y}/dt \\ dp_{1_x}/dt & dp_{2_y}/dt \end{bmatrix} = \begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix}$$

Spline and Geometry Matrices

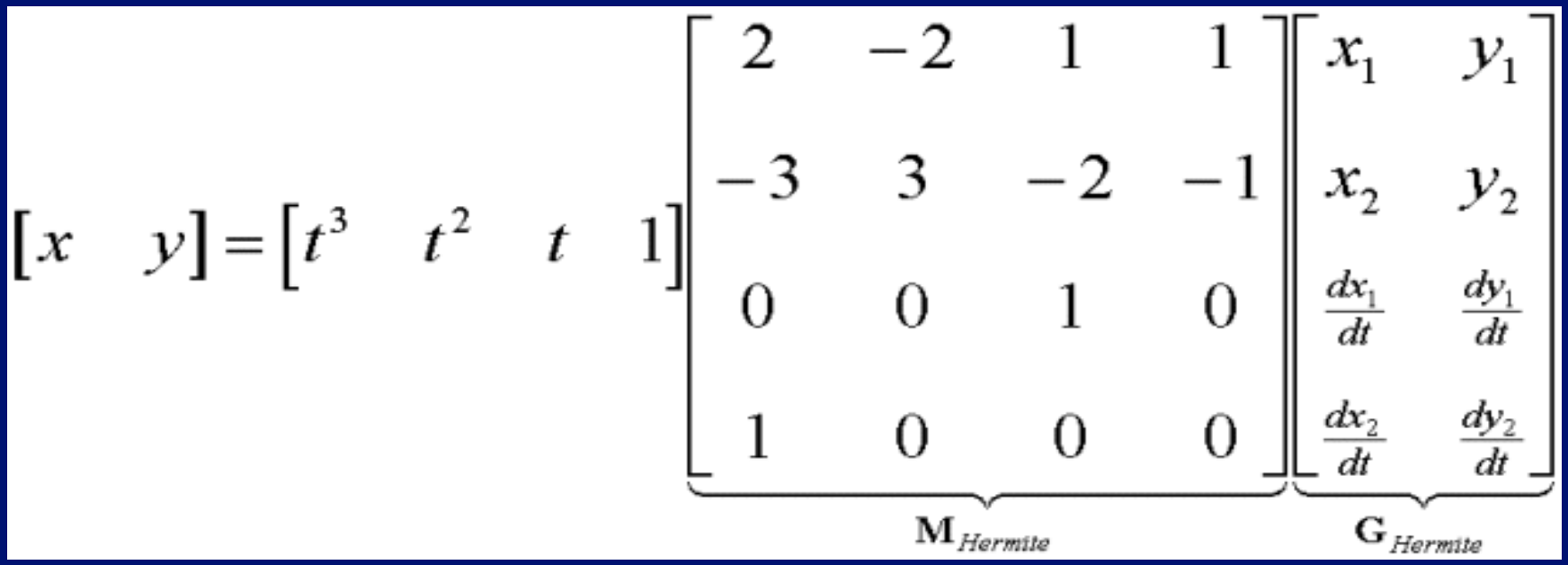
$$\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{1_x} & p_{1_y} \\ p_{2_x} & p_{2_y} \\ dp_{1_x}/dt & dp_{1_y}/dt \\ dp_{1_x}/dt & dp_{2_y}/dt \end{bmatrix} = \begin{bmatrix} a & e \\ b & f \\ c & g \\ d & h \end{bmatrix}$$

$M_{Hermite}$

$G_{Hermite}$

UNIVERSITY of VIRGINIA

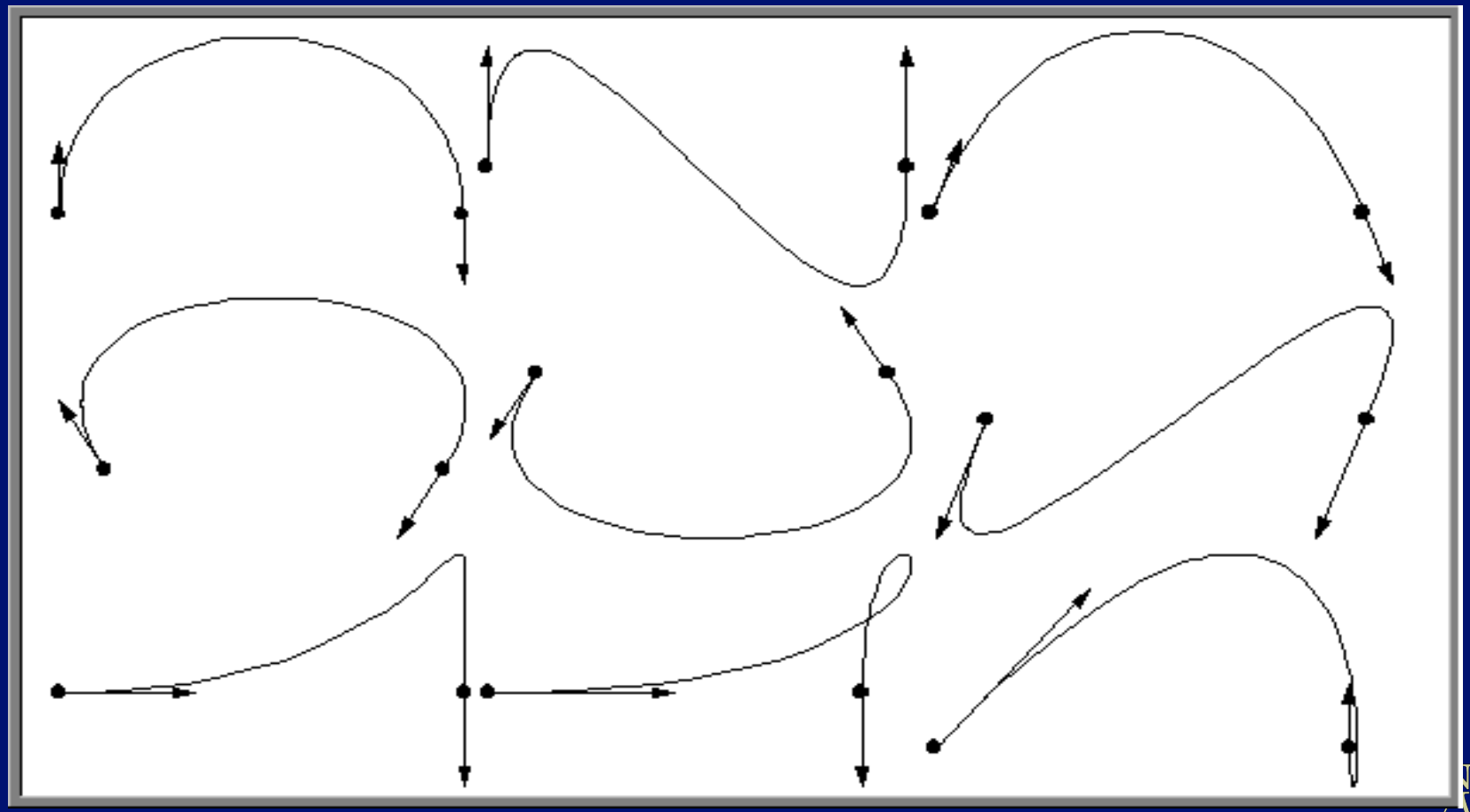
Resulting Hermite Spline Equation



Demonstration

Hermite

Sample Hermite Curves



Blending Functions

By multiplying first two matrices in lower-left equation, you have four functions of 't' that blend the four control parameters

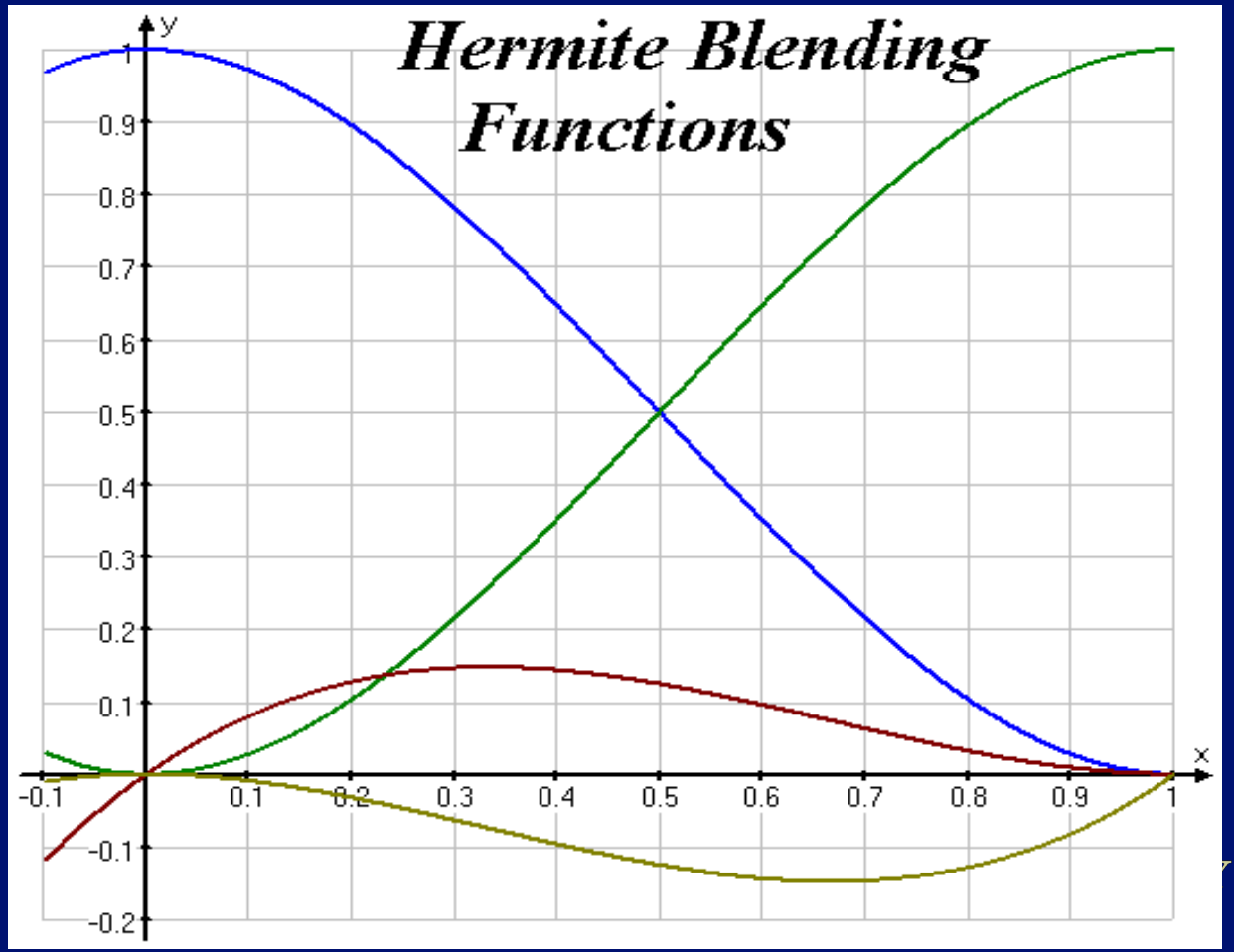
These are blending functions

$$[x \quad y] = [t^3 \quad t^2 \quad t \quad 1] \underbrace{\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{M}_{Hermite}} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \frac{dx_1}{dt} & \frac{dy_1}{dt} \\ \frac{dx_2}{dt} & \frac{dy_2}{dt} \end{bmatrix}}_{\mathbf{G}_{Hermite}}$$

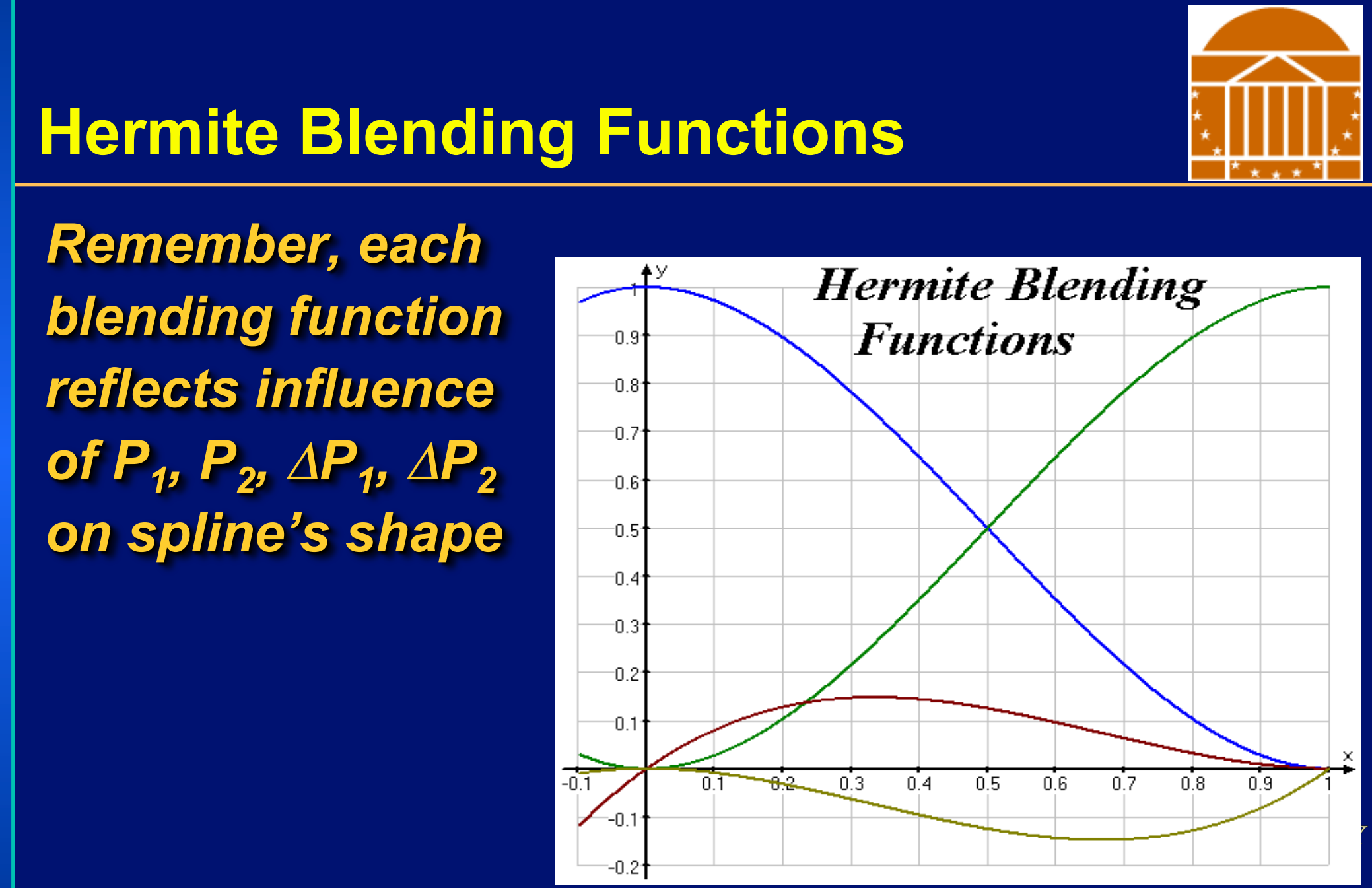
$$p(t) = \begin{bmatrix} 2t^3 - 3t^2 + 1 \\ -2t^3 + 3t^2 \\ t^3 - 2t^2 + t \\ t^3 - t^2 \end{bmatrix}^{\mathbf{T}} \begin{bmatrix} p_1 \\ p_2 \\ \nabla p_1 \\ \nabla p_2 \end{bmatrix}$$

Hermite Blending Functions

If you plot the blending functions on the parameter 't'



Hermite Blending Functions

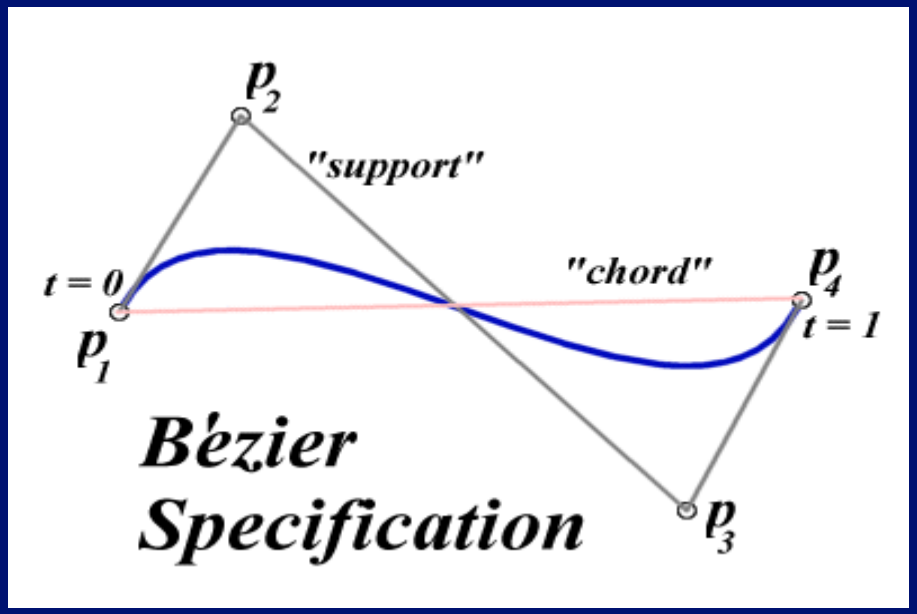


Remember, each blending function reflects influence of P_1, P_2, ΔP_1, ΔP_2 on spline's shape

Bézier Curves

Similar to Hermite, but more intuitive definition of endpoint derivatives

Four control points, two of which are knots



Bézier Curves

The derivative values of the Bezier Curve at the knots are dependent on the adjacent points

$$\nabla p_1 = 3(p_2 - p_1)$$
$$\nabla p_4 = 3(p_4 - p_3)$$

The scalar 3 was selected just for this curve

Bézier vs. Hermite

We can write our Bezier in terms of Hermite

- Note this is just matrix form of previous equations

$$\underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \frac{dx_1}{dt} & \frac{dy_1}{dt} \\ \frac{dx_2}{dt} & \frac{dy_2}{dt} \end{bmatrix}}_{\mathbf{G}_{Hermite}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{\mathbf{G}_{Bezier}}$$

Bézier vs. Hermite

Now substitute this in for previous Hermite

$$\begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} = \underbrace{\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{M}_{Hermite}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{\mathbf{G}_{Bezier}}$$

Bézier Basis and Geometry Matrices

Matrix Form

$$\begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} = \underbrace{\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{M}_{Bezier}} \underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{\mathbf{G}_{Bezier}}$$

But why is M_{Bezier} a good basis matrix?

Bézier Blending Functions

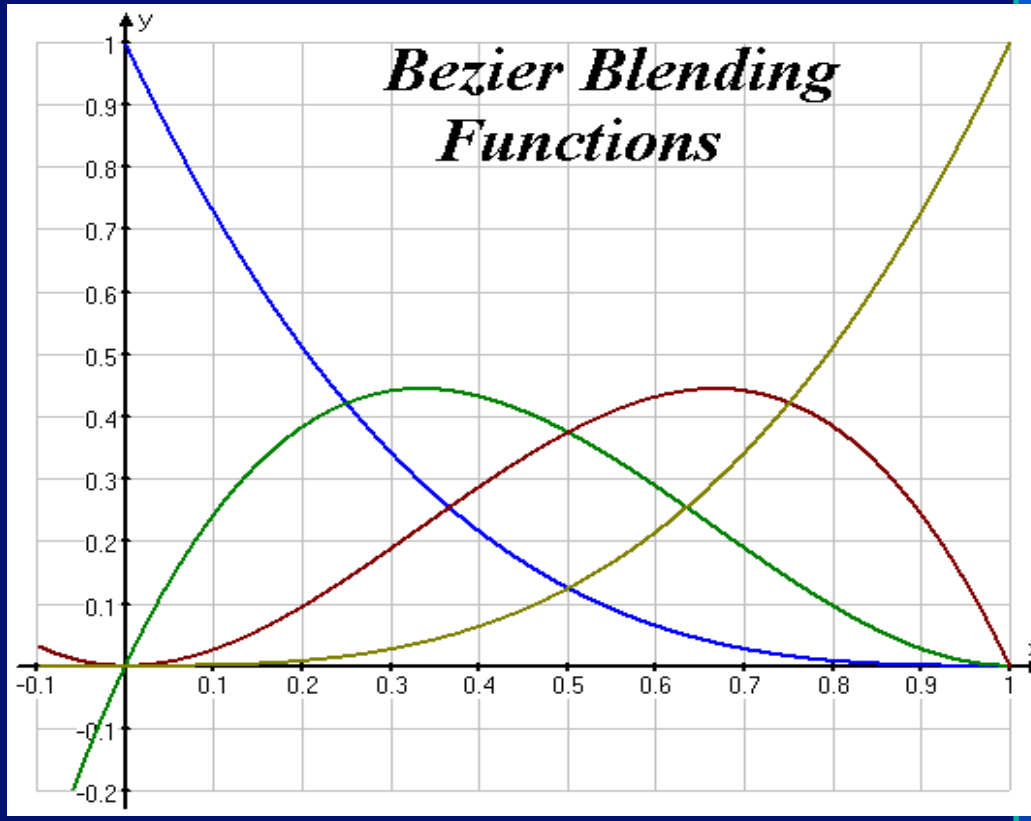
Look at the blending functions

This family of polynomials is called order-3 Bernstein Polynomials

- $C(3, k)\ t^k\ (1-t)^{3-k};\ 0 <= k <= 3$
- They are all positive in interval [0,1]
- Their sum is equal to 1

$$p(t) = \begin{bmatrix} (1-t)^3 \\ 3t(1-t)^2 \\ 3t^2(1-t) \\ t^3 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

Bézier Blending Functions



Thus, every point on curve is linear combination of the control points

The weights of the combination are all positive

The sum of the weights is 1

Therefore, the curve is a convex combination of the control points

Bézier Curves

Will always remain within bounding region defined by control points

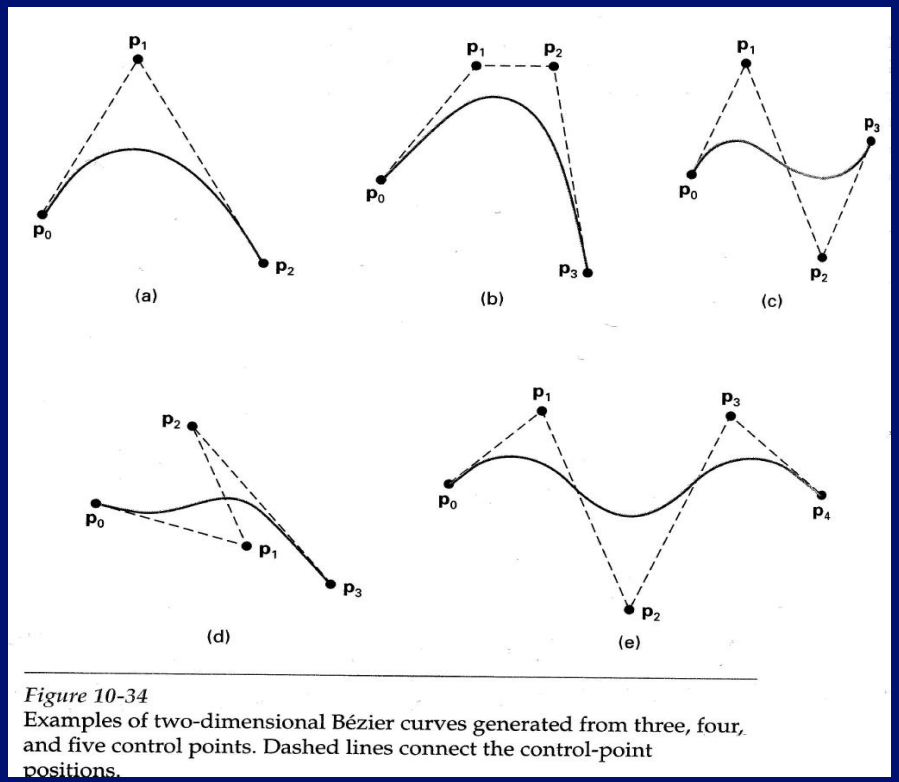


Figure 10-34
Examples of two-dimensional Bézier curves generated from three, four, and five control points. Dashed lines connect the control-point positions.

UNIVERSITY of VIRGINIA

Bézier Curves

Bezier

Why more spline slides?

Bezier and Hermite splines have global influence

- Piecewise Bezier or Hermite don't enforce derivative continuity at join points
- Moving one control point affects the entire curve

B-splines consist of curve segments whose polynomial coefficients depend on just a few control points

- Local control

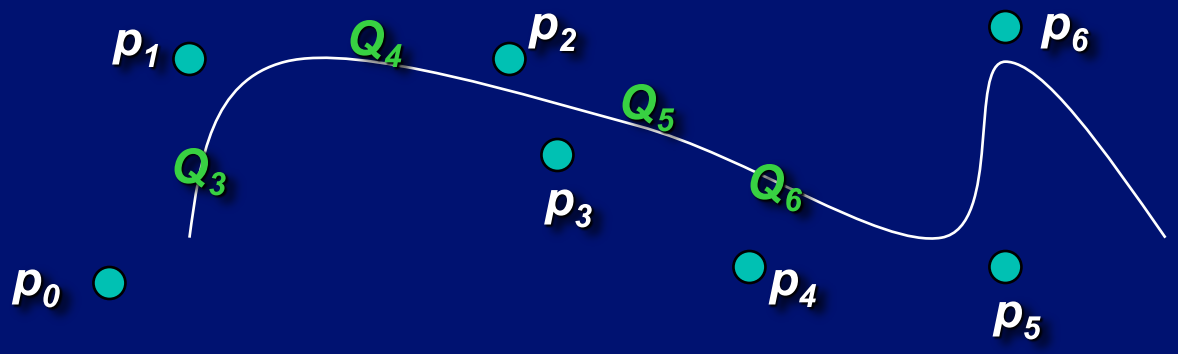
Examples of Splines

B-Spline Curve

Start with a sequence of control points

Select four from middle of sequence ***(p_{i-2}, p_{i-1}, p_i, p_{i+1}) d***

- Bezier and Hermite goes between p_{i-2} and p_{i+1}
- B-Spline doesn't interpolate (touch) any of them but approximates the going through p_{i-1} and p_i



Uniform B-Splines

Approximating ***Splines***

Approximates n+1 control points

- $P_0, P_1, \ldots, P_n$, n ¸ 3

Curve consists of n –2 cubic polynomial segments

- $Q_3, Q_4, \ldots Q_n$

t varies along B-spline as Q_i: t_i <= t < t_{i+1}

t_i (i = integer) are knot points that join segment Q_{i-1} to Q_i

Curve is* uniform *because knots are spaced at equal intervals of parameter, t

Uniform B-Splines

First curve segment, Q_3, is defined by first four control points

Last curve segment, Q_m, is defined by last four control points, P_{m-3}, P_{m-2}, P_{m-1}, P_m

Each control point affects four curve segments

B-spline Basis Matrix

Formulate 16 equations to solve the 16 unknowns

The 16 equations enforce the C_0, C_1, and C_2 continuity between adjoining segments, Q

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

B-Spline Basis Matrix

Note the order of the rows in my $M_{B\text{-}Spline}$ is different from in the book

- Observe also that the order in which I number the points is different
- Therefore my matrix aligns with the book's matrix if you reorder the points, and thus reorder the rows of the matrix

B-Spline

Points along B-Spline are computed just as with Bezier Curves

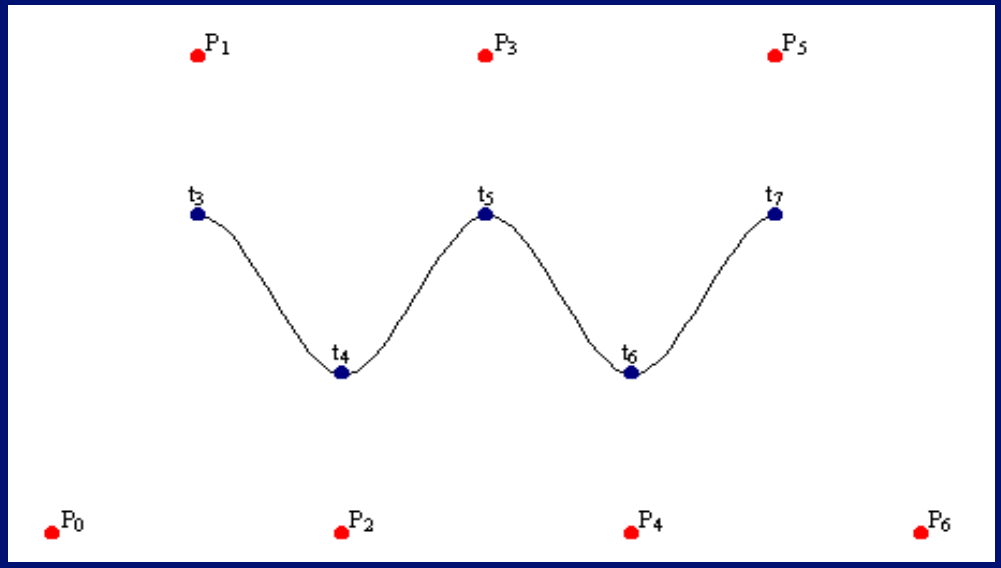
$$Q_i(t) = UM_{B-Spline}P$$

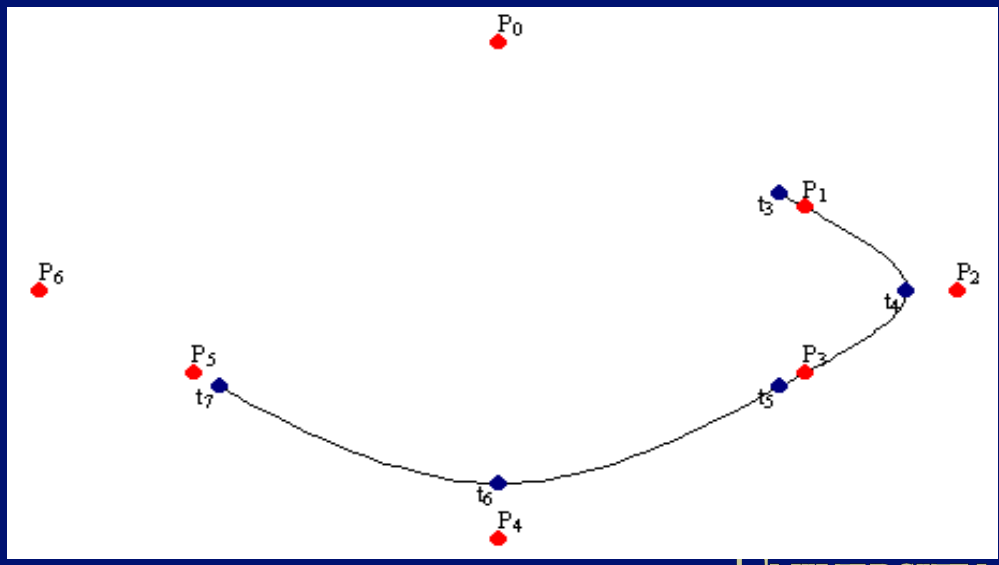
$$Q_i(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{bmatrix}$$

B-Spline

By far the most popular spline used

C_0, C_1, and C_2 continuous





B-Spline

Locality of points

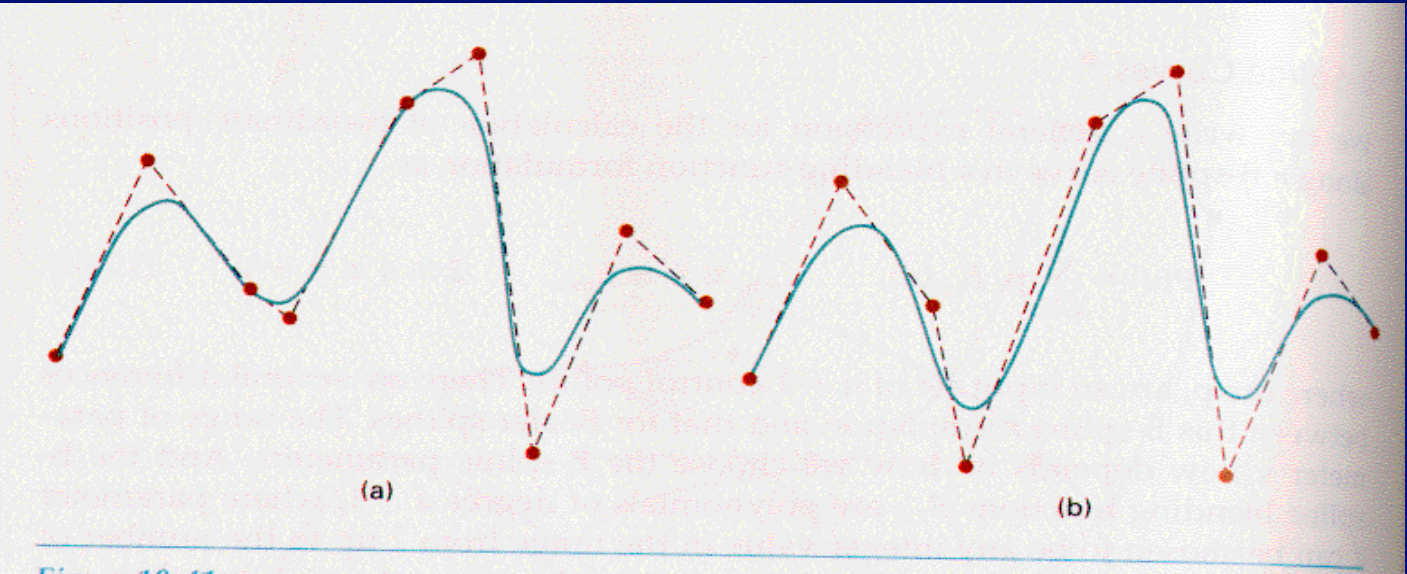


Figure 10-41
Local modification of a B-spline curve. Changing one of the control points in (a) produces curve (b), which is modified only in the neighborhood of the altered control point.

Nonuniform, Rational B-Splines (NURBS)

The native geometry element in Maya

Models are composed of surfaces defined by NURBS, not polygons

NURBS are smooth

NURBS require effort to make non-smooth

NURBs

To generate three-dimension points

- Four B-Splines are used
  - Three define the x, y, and z position
  - One defines a weighting term
- The 3-D point output by a NURB is the x, y, z coordinates divided by the weighting term (like homogeneous coordinates)

What is a NURB?

Nonuniform: The amount of parameter, t, that is used to model each curve segment varies

- Nonuniformity permits either C^2, C^1, or C^0 continuity at join points between curve segments
- Nonuniformity permits control points to be added to middle of curve

What do we get?

NURBs are invariant under rotation, scaling, translation, and perspective transformations of the control points (nonrational curves are not preserved under perspective projection)

- This means you can transform the control points and redraw the curve using the transformed points
- If this weren't true you'd have to sample curve to many points and transform each point individually
- B-spline is preserved under affine transformations, but that is all

Converting Between Splines

Consider two spline basis formulations for two spline types

$$P = T \times M_{spline_1} \times G_{spline_1}$$

$$P = T \times M_{spline_2} \times G_{spline_2}$$

$$T \times M_{spline_1} \times G_{spline_1} = T \times M_{spline_2} \times G_{spline_2}$$

Converting Between Splines

We can transform the control points from one spline basis to another

$$P = T \times M_{spline_1} \times G_{spline_1}$$

$$P = T \times M_{spline_2} \times G_{spline_2}$$

$$T \times M_{spline_1} \times G_{spline_1} = T \times M_{spline_2} \times G_{spline_2}$$

$$M_{spline_1} \times G_{spline_1} = M_{spline_2} \times G_{spline_2}$$

$$G_{spline_1} = M^{-1}_{spline_1} \times M_{spline_2} \times G_{spline_2}$$

Converting Between Splines

With this conversion, we can convert a B-Spline into a Bezier Spline

Bezier Splines are easy to render

Rendering Splines

Horner's Method

Incremental (Forward Difference) Method

Subdivision Methods

Horner's Method

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$x(u) = [(a_x u + b_x)u + c_x]u + d_x$$

Three multiplications

Three additions

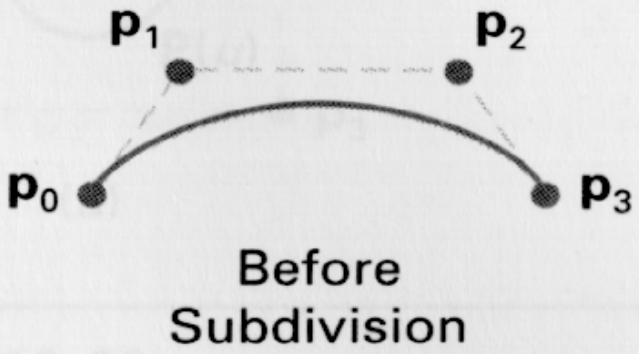
Forward Difference

$$x_{k+1} = x_k + \Delta x_k$$
$$x_k = a_x u^3 + b_x u^2 + c_x u + d$$
$$x_{k+1} = a_x (u_k + \delta)^3 + b_x (u_k + \delta)^2 + c_x (u_k + \delta) + d_x$$
$$x_{k+1} - x_k = \Delta x_k = 3a_x \delta u_k^{\ 2} + (3a_x \delta^2 + 2b_x \delta) u_k + (a_x \delta^3 + b_x \delta^2 + c_x \delta)$$

But this still is expensive to compute

- Solve for change at k (Δ_k) and change at k+1 (Δ_{k+1})
- Boot strap with initial values for x_0, Δ_0, and Δ_1
- Compute x_3 by adding $x_0 + \Delta_0 + \Delta_1$

Subdivision Methods



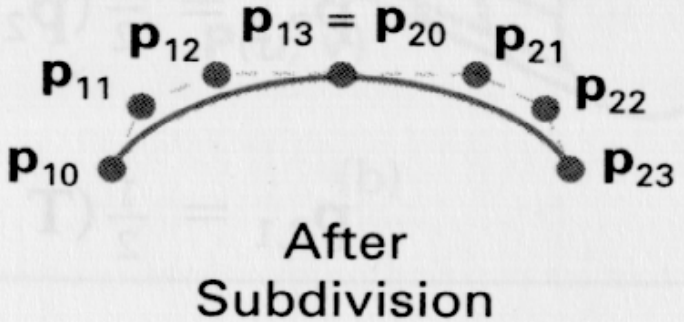


Figure 10-52
Subdividing a cubic Bézier curve section into two sections, each with four control points.

Bezier

Rendering Bezier Spline

```
public void spline(ControlPoint p0, ControlPoint p1,
                   ControlPoint p2, ControlPoint p3, int pix) {
    float len = ControlPoint.dist(p0,p1) + ControlPoint.dist(p1,p2)
              + ControlPoint.dist(p2,p3);
    float chord = ControlPoint.dist(p0,p3);
    if (Math.abs(len - chord) < 0.25f) return;
    fatPixel(pix, p0.x, p0.y);
    ControlPoint p11 = ControlPoint.midpoint(p0, p1);
    ControlPoint tmp = ControlPoint.midpoint(p1, p2);
    ControlPoint p12 = ControlPoint.midpoint(p11, tmp);
    ControlPoint p22 = ControlPoint.midpoint(p2, p3);
    ControlPoint p21 = ControlPoint.midpoint(p22, tmp);
    ControlPoint p20 = ControlPoint.midpoint(p12, p21);
    spline(p20, p12, p11, p0, pix);
    spline(p3, p22, p21, p20, pix);
}
```

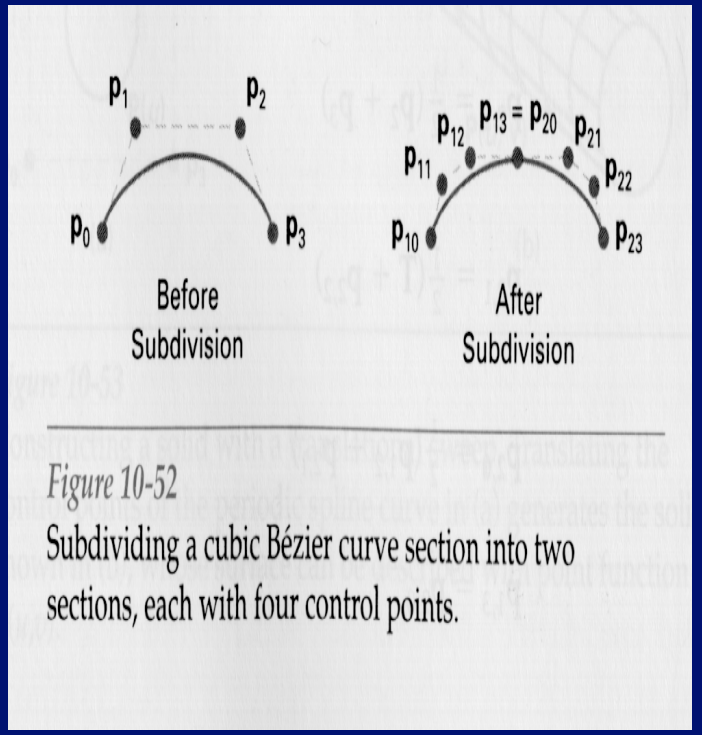


Figure 10-52
Subdividing a cubic Bézier curve section into two sections, each with four control points.

UNIVERSITY of VIRGINIA