Computer Graphics

- Splines -

Hendrik Lensch

Overview

- **Last Time**
  - Image-Based Rendering

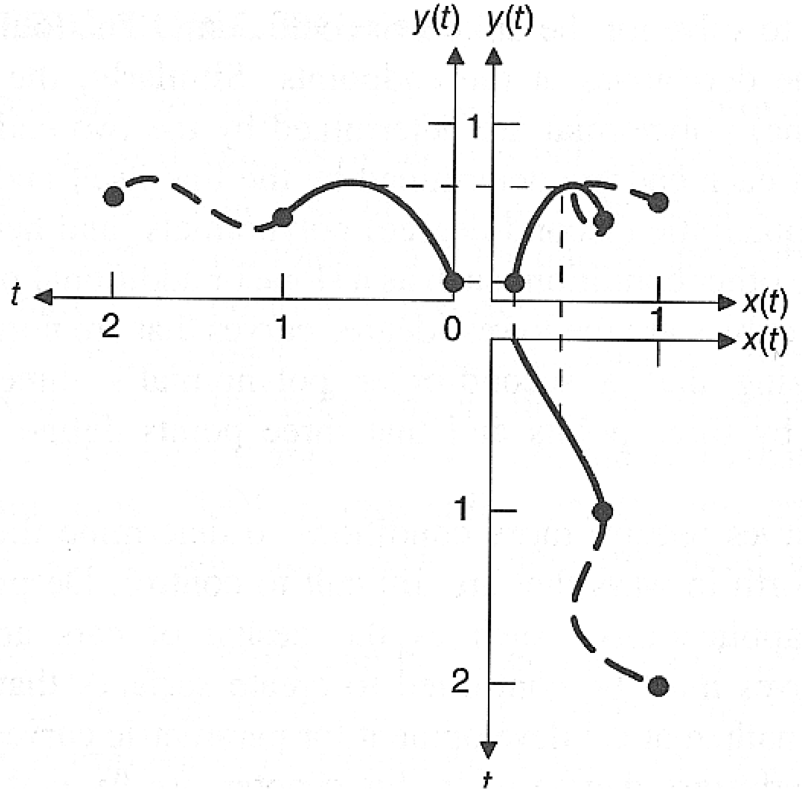
- **Today**
  - Parametric Curves
  - Lagrange Interpolation
  - Hermite Splines
  - Bezier Splines
  - DeCasteljau Algorithm
  - Parameterization

Curves

- **Curve descriptions**
  - Explicit
    - y(x)= ± sqrt(r^2 - x^2), restricted domain
  - Implicit:
    - $x^{2\ +}\ y^2 = r^2$ unknown solution set
  - Parametric:
    - x(t)= r cos(t), y(t)= r sin(t), $t \in [0, 2\pi]$
    - Flexibility and ease of use
- **Polynomials**
  - Avoids complicated functions (z.B. pow, exp, sin, sqrt)
  - Use simple polynomials of low degree

Parametric curves

- **Separate function in each coordinate**
  - 3D: f(t)= (x(t), y(t), z(t))



Monomials

- **Monomial basis**
  - Simple basis: 1, t, t^2, ... (t usually in [0 .. 1])
- **Polynomial representation**

Degree (= Order – 1)

$$\underline{P}(t) = \left(\underline{x}(t) \quad \underline{y}(t) \quad \underline{z}(t)\right) = \sum_{i=0}^{n} t^i \, \underline{A}_i$$

Coefficients ∈R³

Monomials

  - Coefficients can be determined from a sufficient number of constraints (e.g. interpolation of given points)
    - Given (n+1) parameter values t_i and points P_i
    - Solution of a linear system in the A_i − possible, but inconvenient
- **Matrix representation**

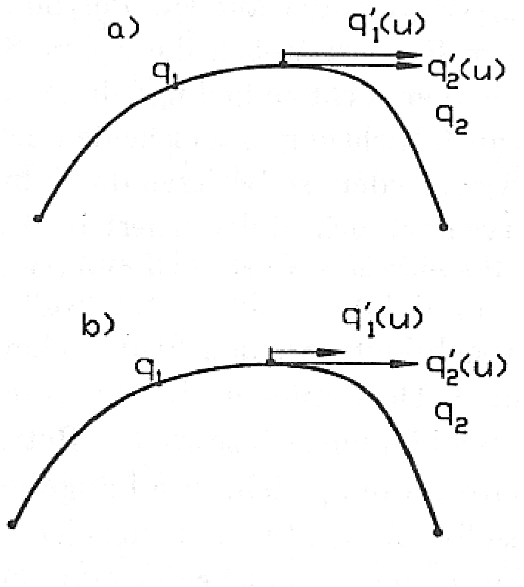
$$P(t) = \left(x(t) \quad y(t) \quad z(t)\right) = T(t)\,\mathbf{A} = \begin{bmatrix} t^n & t^{n-1} & \cdots & 1 \end{bmatrix} \begin{bmatrix} A_{x,n} & A_{y,n} & A_{z,n} \\ A_{x,n-1} & A_{y,n-1} & A_{z,n-1} \\ & \vdots & \\ A_{x,0} & A_{y,0} & A_{z,0} \end{bmatrix}$$

Derivatives

- **Derivative = tangent vector**
  - Polynomial of degree (n-1)

$$P'(t) = \begin{pmatrix} x'(t) & y'(t) & z'(t) \end{pmatrix} = T'(t)\,\mathbf{A} = \begin{bmatrix} nt^{n-1} & (n\text{-}1)t^{n\text{-}1} & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} A_{x,n} & A_{y,n} & A_{z,n} \\ A_{x,n-1} & A_{y,n-1} & A_{z,n-1} \\ & \vdots & \\ A_{x,0} & A_{y,0} & A_{z,0} \end{bmatrix}$$

- **Continuity and smoothness between parametric curves**
  - $C^0 = G^0$ = same point
  - Parametric continuity C^1
    - Tangent vectors are identical
  - Geometric continuity G^1
    - Same direction of tangent vectors
  - Similar for higher derivatives



More on Continuity

- **at one point:**

- **Geometric Continuity:**
  - G0: curves are joined
  - G1: first derivatives are proportional at joint point, same direction but not necessarily same length
  - G2: first and second derivatives are proportional

- **Parametric Continuity:**
  - C0: curves are joined
  - C1: first derivative equal
  - C2: first and second derivatives are equal. If t is the time, this implies the acceleration is continuous.
  - Cn: all derivatives up to and including the nth are equal.

Lagrange Interpolation

- **Interpolating basis functions**
  - Lagrange polynomials for a set of parameters T={t_0, ..., t_n}

$$L_i^n(t) = \prod_{\substack{j=o \\ i \neq j}}^{n} \frac{t - t_j}{t_i - t_j}, \quad \text{with} \quad L_i^n(t_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

- **Properties**
  - Good for interpolation at given parameter values
    - At each t_i: One basis function = 1, all others = 0
  - Polynomial of degree n (n factors linear in t)
- **Lagrange Curves**
  - Use Lagrange Polynomials with point coefficients

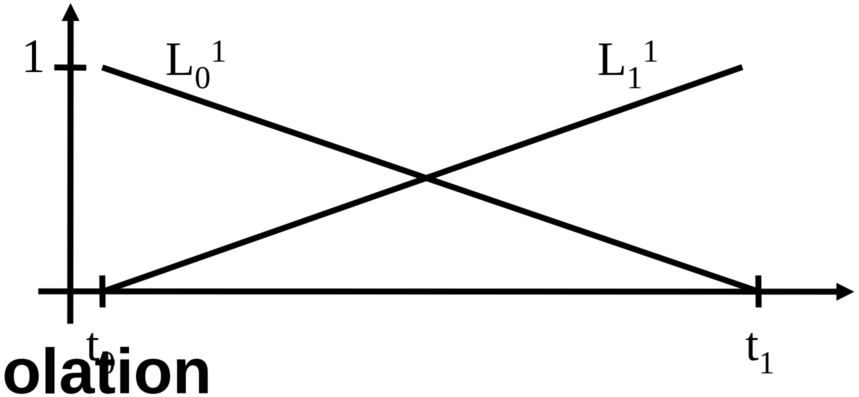
$$\underline{P}(t) = \sum_{i=0}^{n} L_i^n(t) \underline{P}_i$$

Lagrange Interpolation

- **Simple Linear Interpolation**
  - T={t$_0$, t$_1$}

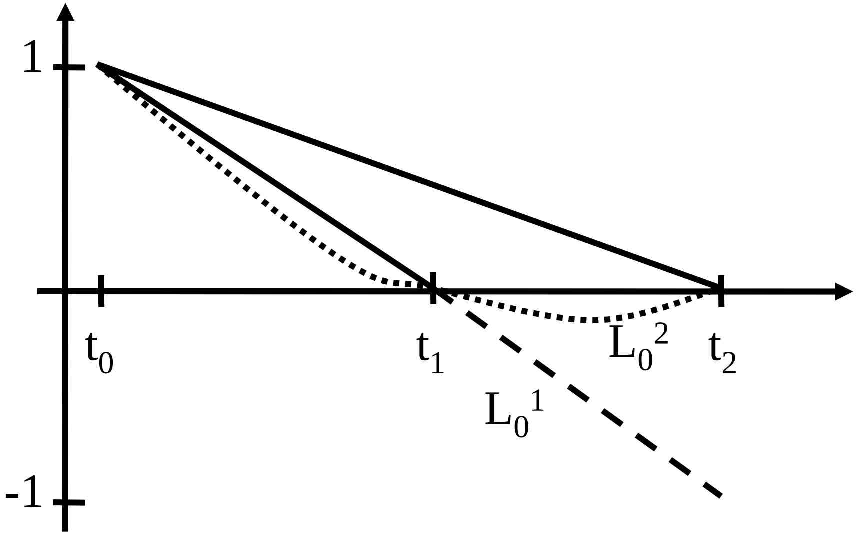
$$L_0^1(t) = \frac{t - t_1}{t_0 - t_1}$$

$$L_1^1(t) = \frac{t - t_0}{t_1 - t_0}$$



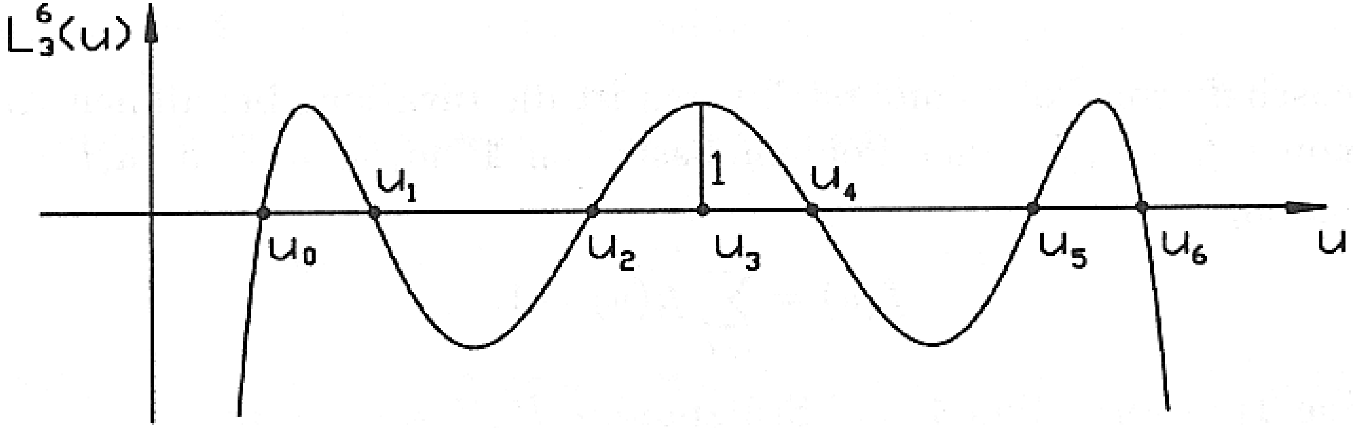
- **Simple Quadratic Interpolation**
  - T={t$_0$, t$_{1,}$ t$_2$}

$$L_0^2(t) = \frac{t - t_1}{t_0 - t_1}\,\frac{t - t_2}{t_0 - t_2}$$



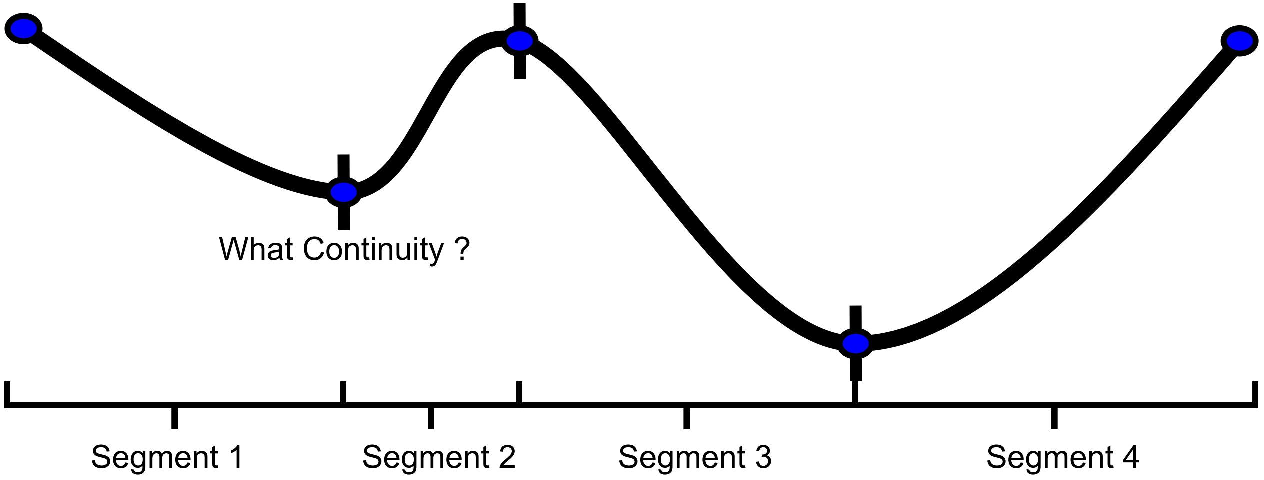
Problems

- **Problems with a single polynomial**
  - Degree depends on the number of interpolation constraints
  - Strong overshooting for high degree (n > 7)
  - Problems with smooth joints
  - Numerically unstable
  - No local changes



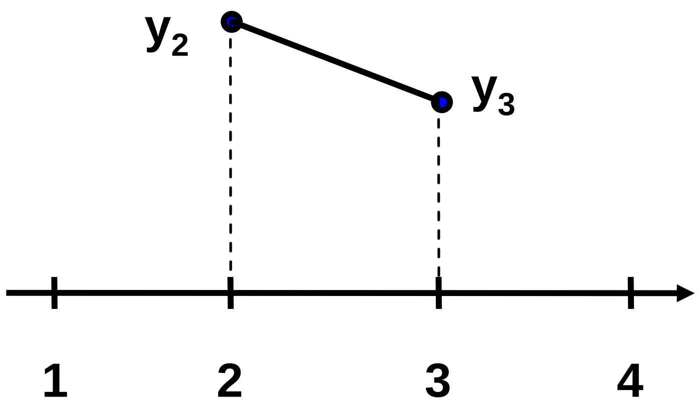
Splines

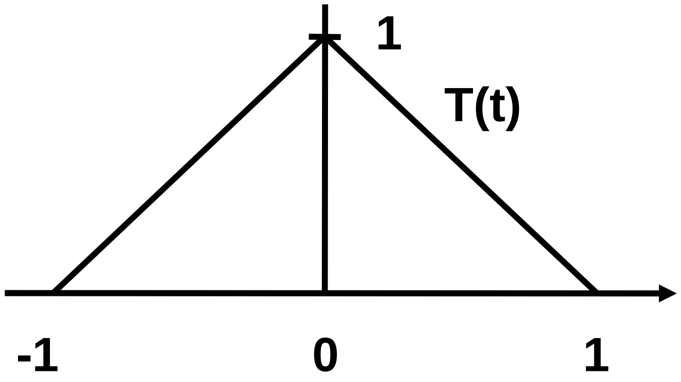
- **Functions for interpolation & approximation**
  - Standard curve and surface primitives in geometric modeling
  - Key frame and in-betweens in animations
  - Filtering and reconstruction of images
- **Historically**
  - Name for a tool in ship building
    - Flexible metal strip that tries to stay straight
  - Within computer graphics:
    - Piecewise polynomial function

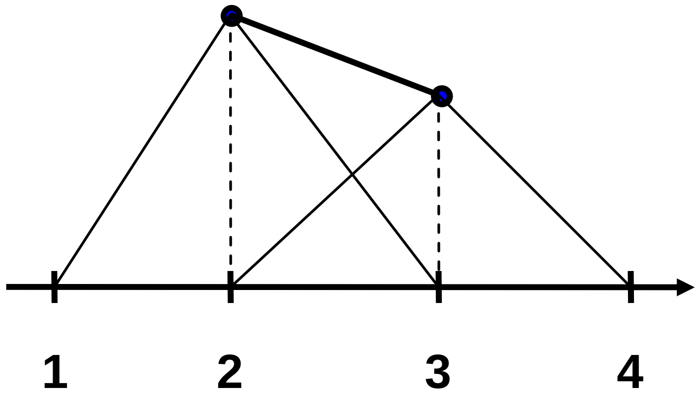


Linear Interpolation

- **Hat Functions and Linear Splines**







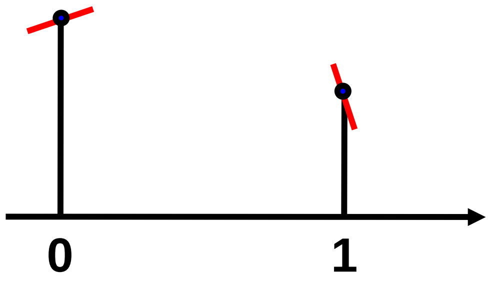
$$T(\mathrm{t}) = \begin{cases} 0 & t < -1 \\ 1+t & -1 \le t < 0 \\ 1-t & 0 \le t < 1 \\ 0 & t \ge 1 \end{cases}$$

$$\mathrm{P(t)} = \sum \mathrm{P}_i \mathrm{T}_i(t) = \mathrm{y}_2\mathrm{T}_2(t) + \mathrm{y}_3\mathrm{T}_3(t)$$

$$T_i(t) = T(t-i)$$

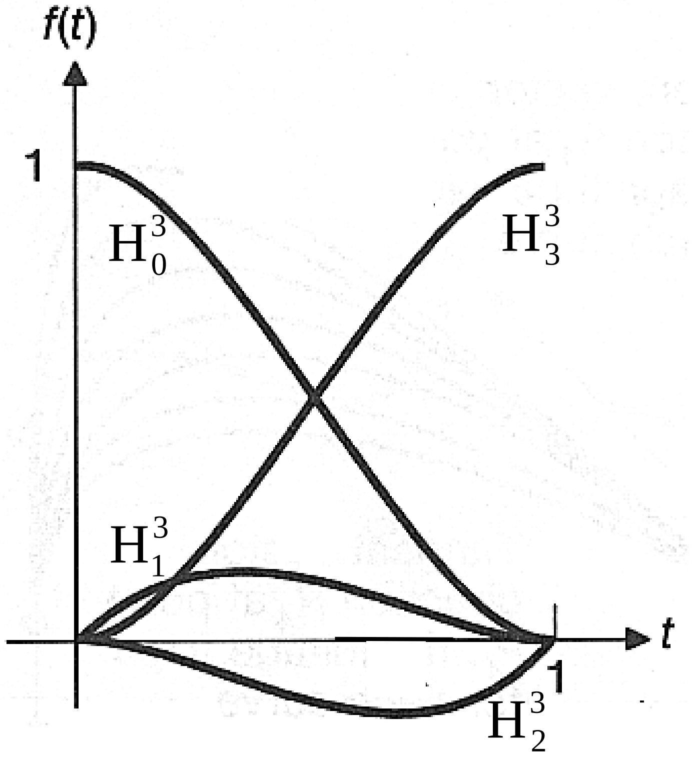
Hermite Interpolation

- **Hermite Basis (cubic)**
  - Interpolation of position P and tangent P´ information for t= {0, 1}



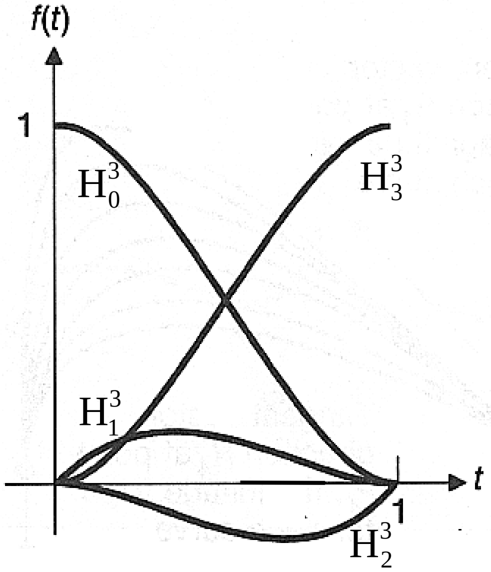
  - Basis functions

$$H_0^3(t) = (1-t)^2(1+2t)$$
$$H_1^3(t) = t(1-t)^2$$
$$H_2^3(t) = -t^2(1-t)$$
$$H_3^3(t) = (3-2t)t^2$$



Hermite Interpolation

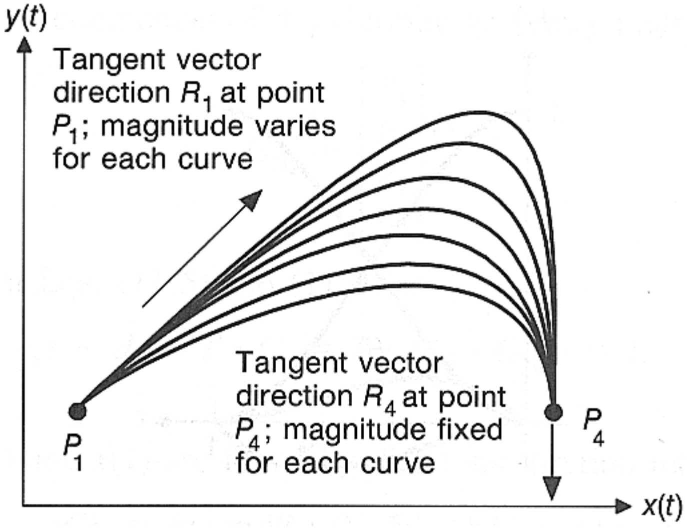
- **Properties of Hermite Basis Functions**
  - H_0 (H_3) interpolates smoothly from 1 to 0 (1 to 0)
  - H_0 and H_3 have zero derivative at t= 0 and t= 1
    - No contribution to derivative (H_1, H_2)
  - H_1 and H_2 are zero at t= 0 and t= 1
    - No contribution to position (H_0, H_3)
  - H_1 (H_2) has slope 1 at t= 0 (t= 1)
    - Unit factor for specified derivative vector

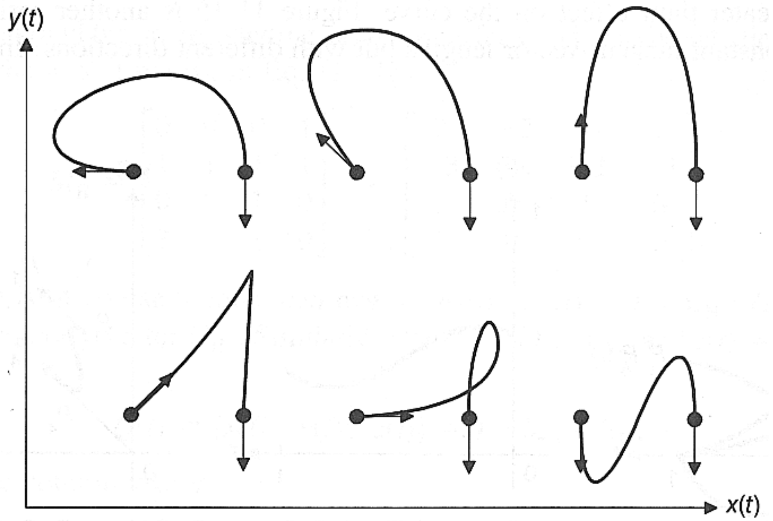


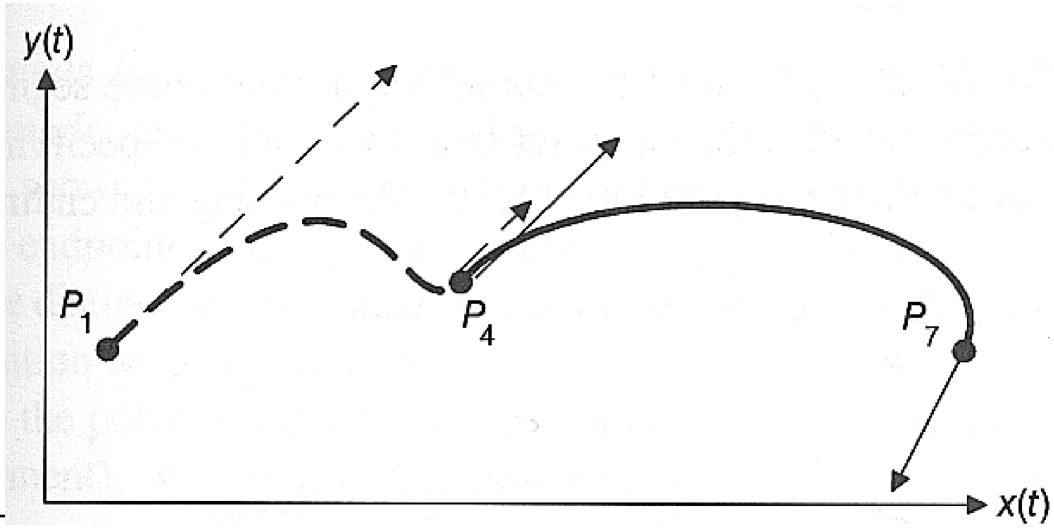
- **Hermite polynomials**
  - P_0, $P`_1$ are positions $\in R^3$
  - $P`_0$, P_1 are derivatives (tangent vectors) $\in R^3$

$$\underline{P}(t) = P_0 H_0^3(t) + P_0{}' H_1^3(t) + P_1{}' H_2^3(t) + P_1 H_3^3(t)$$

Examples: Hermite Interpolation


y(t)
Tangent vector direction R_1 at point P_1; magnitude varies for each curve
Tangent vector direction R_4 at point P_4; magnitude fixed for each curve
P_1
P_4
x(t)





Matrix Representation

- **Matrix representation**

$$P(t) = \begin{bmatrix} t^3 & t^2 & \cdots & 1 \end{bmatrix} \begin{bmatrix} A_{x,n} & A_{y,n} & A_{z,n} \\ A_{x,n-1} & A_{y,n-1} & A_{z,n-1} \\ & \vdots & \\ A_{x,0} & A_{y,0} & A_{z,0} \end{bmatrix} =$$

$$\underbrace{\begin{bmatrix} t^3 & t^2 & \cdots & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} M_{11} & M_{12} & M_{13} & \\ M_{21} & \ddots & & \\ & & & \\ & & & \end{bmatrix}}_{\text{Basis Matrix M (4x4)}} \underbrace{\begin{bmatrix} G_{x,3} & G_{y,3} & G_{z,3} \\ G_{x,2} & G_{y,2} & G_{z,2} \\ G_{x,1} & G_{y,1} & G_{y,1} \\ G_{x,0} & G_{y,0} & G_{z,0} \end{bmatrix}}_{\text{Geometry Matrix G (4x3)}} =$$

$$\underbrace{\begin{bmatrix} t^3 & t^2 & \cdots & 1 \end{bmatrix} \underbrace{\begin{bmatrix} M_{11} & M_{12} & M_{13} & \\ M_{21} & \ddots & & \\ & & & \\ & & & \end{bmatrix}}_{M_H}}_{\text{Basis Functions}} \underbrace{\begin{bmatrix} P_0^T \\ P_1^T \\ P_0'^T \\ P_1'^T \end{bmatrix}}_{G_H}$$

Matrix Representation

- **For cubic Hermite interpolation we obtain:**

$$
\begin{aligned}
P_0^T &= (0 \quad 0 \quad 0 \quad 1)\mathbf{M}_H\,\mathbf{G}_H \\
P_1^T &= (1 \quad 1 \quad 1 \quad 1)\mathbf{M}_H\,\mathbf{G}_H \\
P'^T_0 &= (0 \quad 0 \quad 1 \quad 0)\mathbf{M}_H\,\mathbf{G}_H \\
P'^T_1 &= (3 \quad 2 \quad 1 \quad 0)\mathbf{M}_H\mathbf{G}_H
\end{aligned}
$$

or

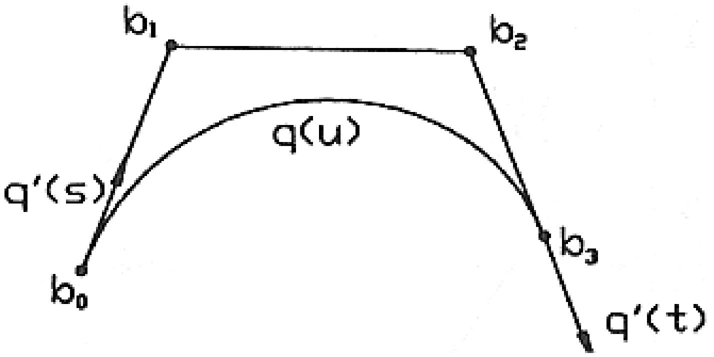
$$
\begin{pmatrix} P_0^T \\ P_1^T \\ P_0'^T \\ P_1'^T \end{pmatrix} = \mathbf{G}_H = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \mathbf{M}_H \mathbf{G}_H
$$

- **Solution:**
  - Two matrices must multiply to unit matrix

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

Bézier

- **Bézier Basis [deCasteljau´59, Bézier´62]**
  - Different curve representation
  - Start and end point
  - 2 point that are approximated by the curve (cubics)
  - P´$_0$= 3(b$_1$-b$_0$) and P´$_1$= 3(b$_3$-b$_2$)
    - Factor 3 due to derivative of t^3



$$G_H = \begin{bmatrix} P_0^T \\ P_1^T \\ P'^{\,T}_0 \\ P'^{\,T}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} b_0^T \\ b_1^T \\ b_2^T \\ b_3^T \end{bmatrix} = M_{HB} G_B$$

Basis transformation

- **Transformation**
  - P(t)=T M_H G_H = T M_H (M_{HB} G_B) = T ($M_H M_{HB}$) G_B = T M_B G_B

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

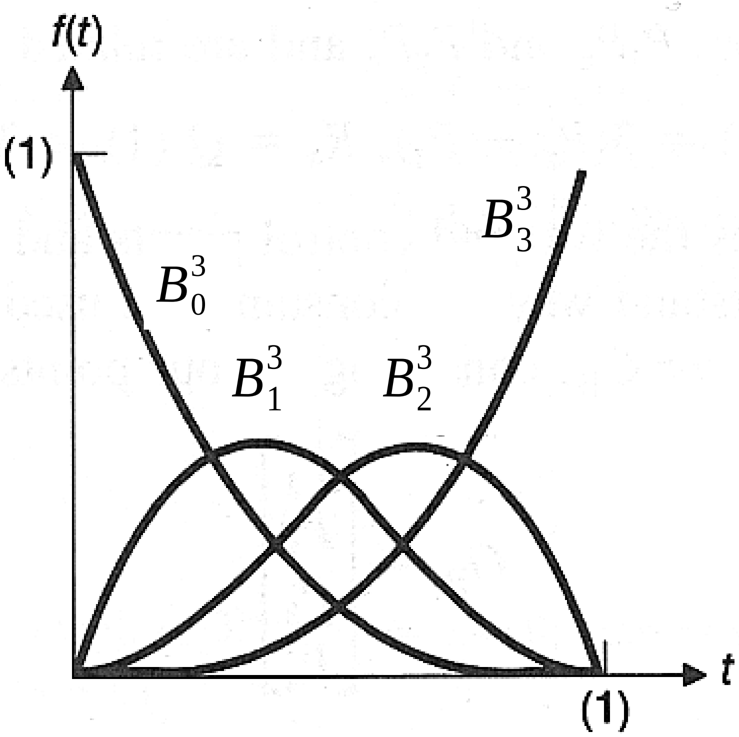
- **Bézier Curves & Basis Function**

$$\mathrm{P(t)} = \sum_{i=0}^{3} B_i^3(t) b_i =$$

$$(1\text{-}t)^3 b_0 + 3t(1\text{-}t)^2\ b_1 + 3t^2(1\text{-}t) b_2 + t^3 b_3$$

$$\mathrm{P(t)} = \sum_{i=0}^{n} B_i^n(t) b_i$$

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

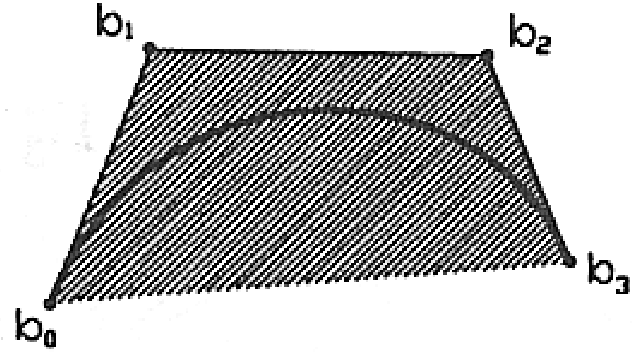


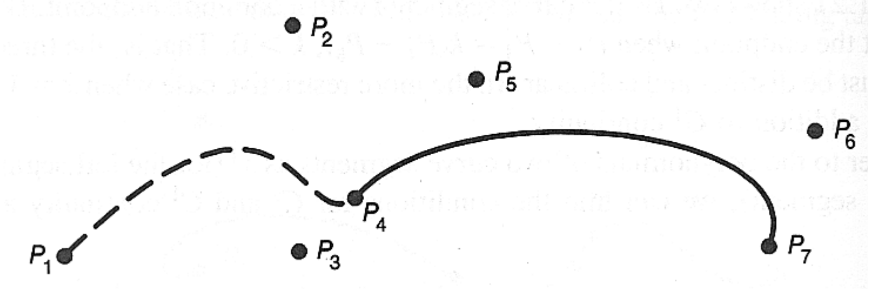
  - Basis functions: Bernstein polynomials

Properties: Bézier

- **Advantages:**
  - End point interpolation
  - Tangents explicitly specified
  - Smooth joints are simple
    - P_3, P_4, P_5 collinear → G^1 continuous
  - Geometric meaning of control points
  - Affine invariance

    $\forall \sum B_i(t) = 1$
  - Convex hull property
    - For $0<t<1$: $B_i(t) \geq 0$
  - Symmetry: $B_i(t) = B_{n-i}(1-t)$





- **Disadvantages**
  - Smooth joints need to be maintained explicitly
    - Automatic in B-Splines (and NURBS)

DeCasteljau Algorithm

- **Direct evaluation of the basis functions**
  - Simple but expensive
- **Use recursion**
  - Recursive definition of the basis functions

$$B_i^n(t) = tB_{i-1}^{n-1}(t) + (1-t)B_i^{n-1}(t)$$

  - Inserting this once yields:

$$P(t) = \sum_{i=0}^{n} b_i^0 B_i^n(t) = \sum_{i=0}^{n-1} b_i^1(t) B_i^{n-1}(t)$$

  - with the new Bézier points given by the recursion

$$b_i^k(t) = tb_{i+1}^{k-1}(t) + (1-t)b_i^{k-1}(t) \text{ and } b_i^0(t) = b_i$$

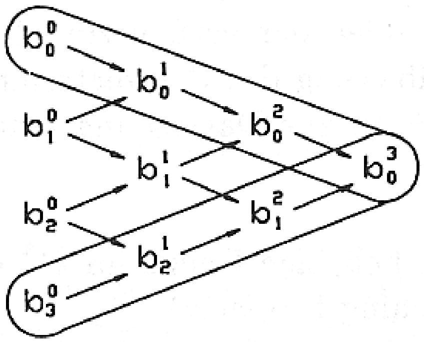
DeCasteljau Algorithm

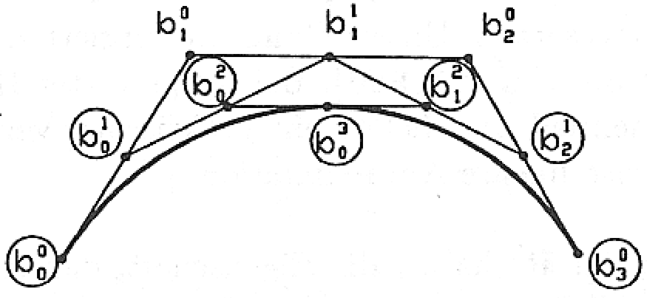
- **DeCasteljau-Algorithm:**
  - Recursive degree reduction of the Bezier curve by using the recursion formula for the Bernstein polynomials

$$P(t)=\sum_{i=0}^{n} b_i^0 B_i^n(t)=\sum_{i=0}^{n-1} b_i^1(t) B_i^{n-1}(t)=\cdots=b_i^n(t)\cdot 1$$

$$b_i^k(t)=t b_{i+1}^{k-1}(t)+(1-t) b_i^{k-1}(t)$$

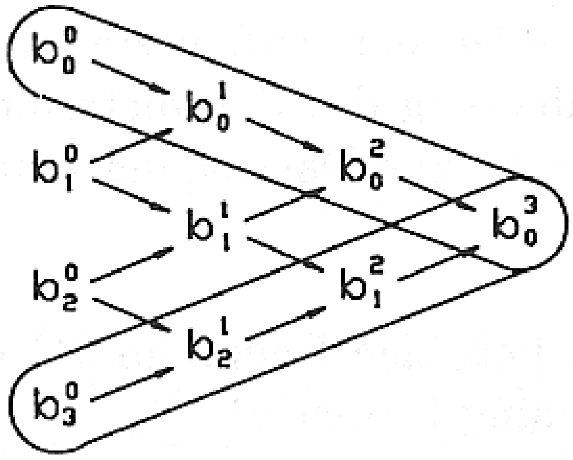
- **Example:**
  - t= 0.5

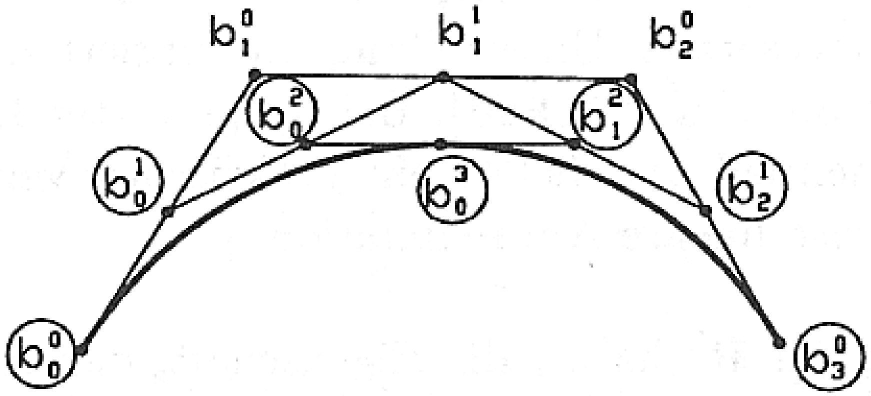




DeCasteljau Algorithm

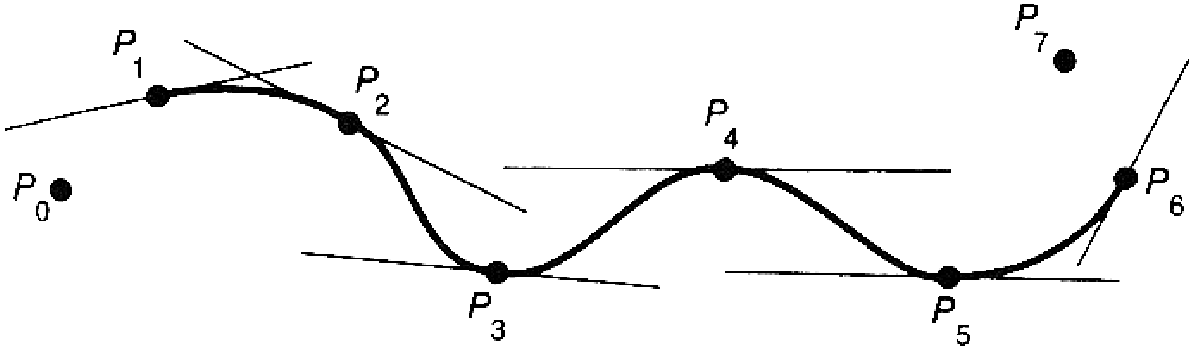
- **Subdivision using the deCasteljau-Algorithm**
  - Take boundaries of the deCasteljau triangle as new control points for left/right portion of the curve
- **Extrapolation**
  - Backwards subdivision
    - Reconstruct triangle from one side





Catmull-Rom-Splines

- **Goal**
  - Smooth (C^1)-joints between (cubic) spline segments
- **Algorithm**
  - Tangents given by neighboring points P_{i-1} P_{i+1}
  - Construct (cubic) Hermite segments
- **Advantage**
  - Arbitrary number of control points
  - Interpolation without overshooting
  - Local control



Matrix Representation

- **Catmull-Rom-Spline**
  - Piecewise polynomial curve
  - Four control points per segment
  - For n control points we obtain (n-3) polynomial segments

$$\underline{P}^{i}(t) = T\mathbf{M}_{CR}G_{CR} = T\,\frac{1}{2}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}\begin{bmatrix} \underline{P}_{i}^{T} \\ \underline{P}_{i+1}^{T} \\ \underline{P}_{i+2}^{T} \\ \underline{P}_{i+3}^{T} \end{bmatrix}$$

- **Application**
  - Smooth interpolation of a given sequence of points
  - Key frame animation, camera movement, etc.
  - Only G^1-continuity
  - Control points should be equidistant in time

Choice of Parameterization

- **Problem**
  - Often only the control points are given
  - How to obtain a suitable parameterization t_i ?
- **Example: Chord-Length Parameterization**

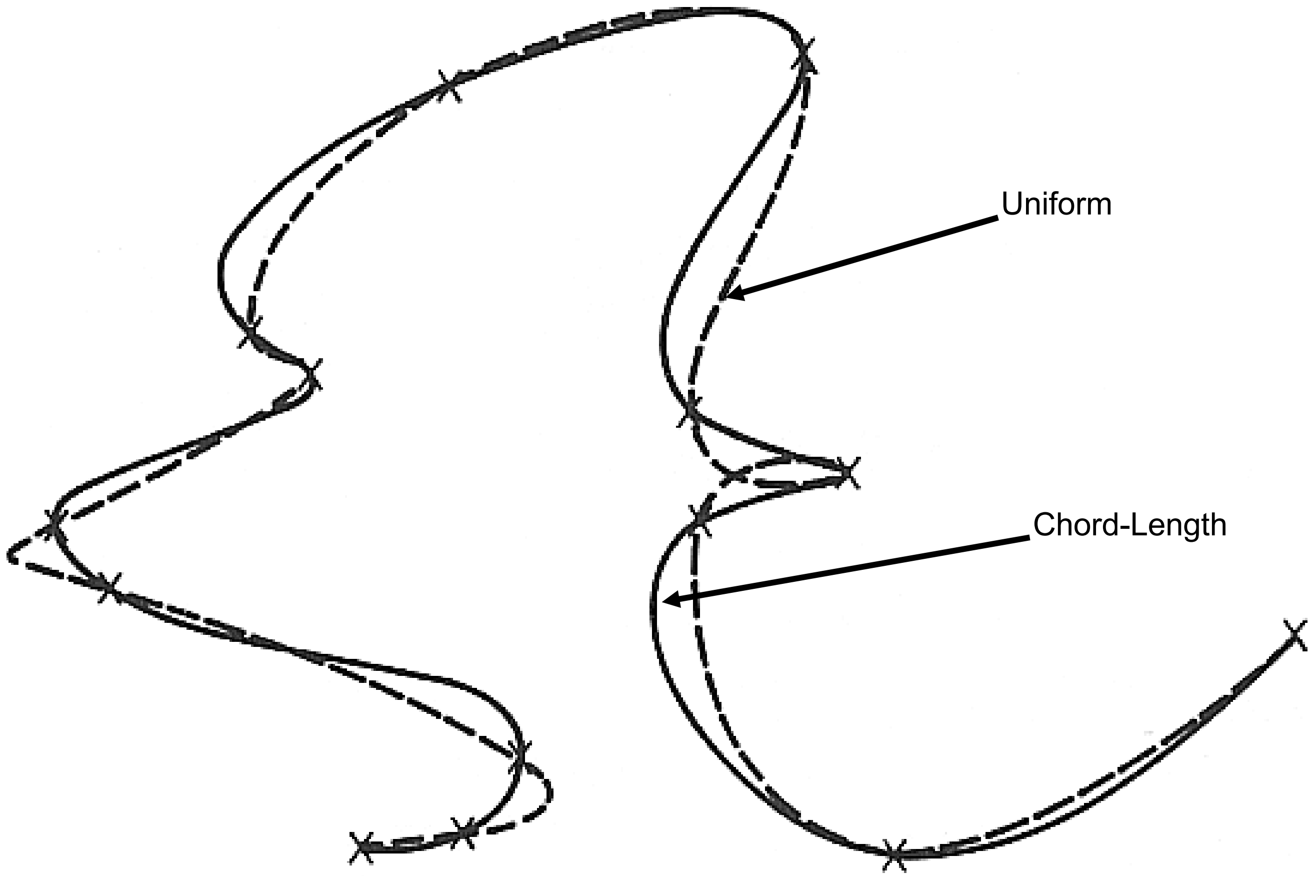
$$t_0 = 0$$

$$t_i = \sum_{j=1}^{i} dist(P_i - P_{i-1})$$

  - Arbitrary up to a constant factor
- **Warning**
  - Distances are not affine invariant !
  - Shape of curves changes under transformations !!

Parameterization

- **Chord-Length versus uniform Parameterization**
  - Analog: Think P(t) as a moving object with mass that may overshoot



B-Splines

- **Goal**
  - Spline curve with local control and high continuity
- **Given**
  - Degree: n
  - Control points: $P_0, \ldots, P_m$ (Control polygon, $m \geq n+1$)
  - Knots: $t_0, \ldots, t_{m+n+1}$ (Knot vector, weakly monotonic)
  - The knot vector defines the parametric locations where segments join
- **B-Spline Curve**

$$\underline{P}(t) = \sum_{i=0}^{m} N_i^n(t) \underline{P}_i$$

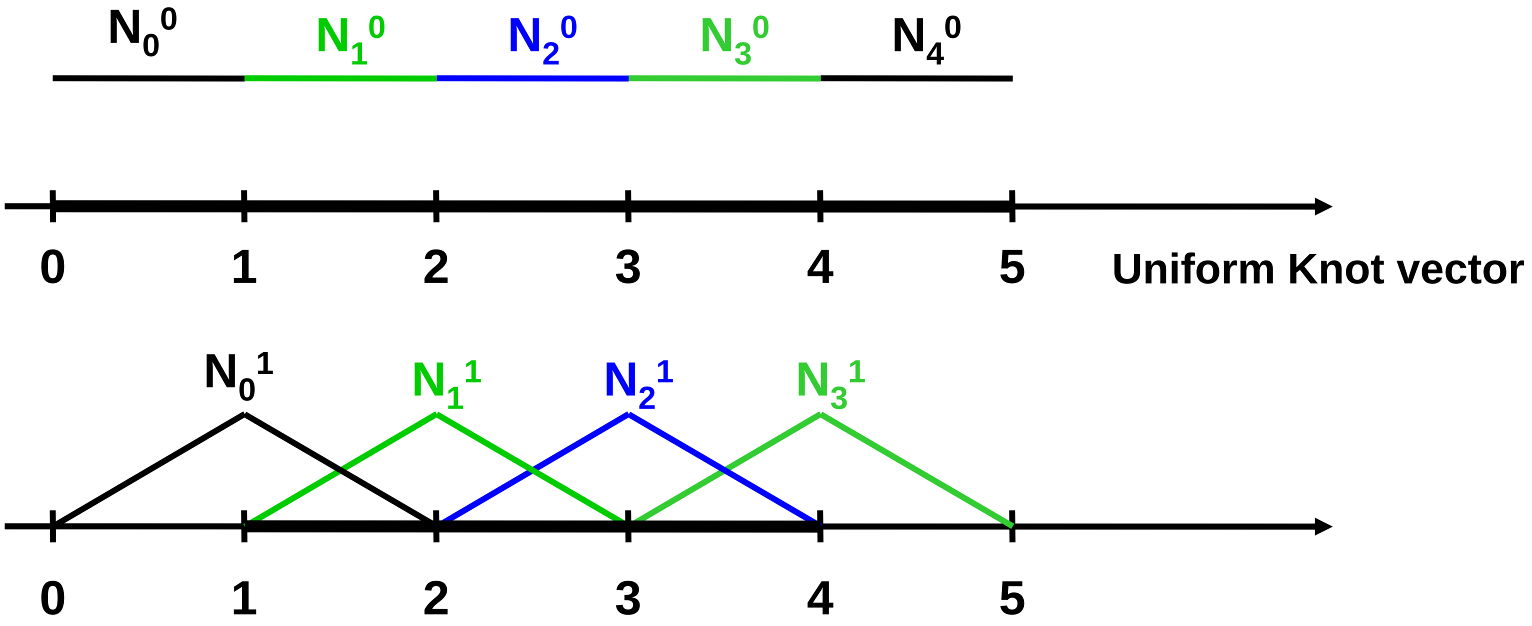
  - Continuity:
    - C_{n-1} at simple knots
    - C_{n-k} at knot with multiplicity k

B-Spline Basis Functions

- **Recursive Definition**

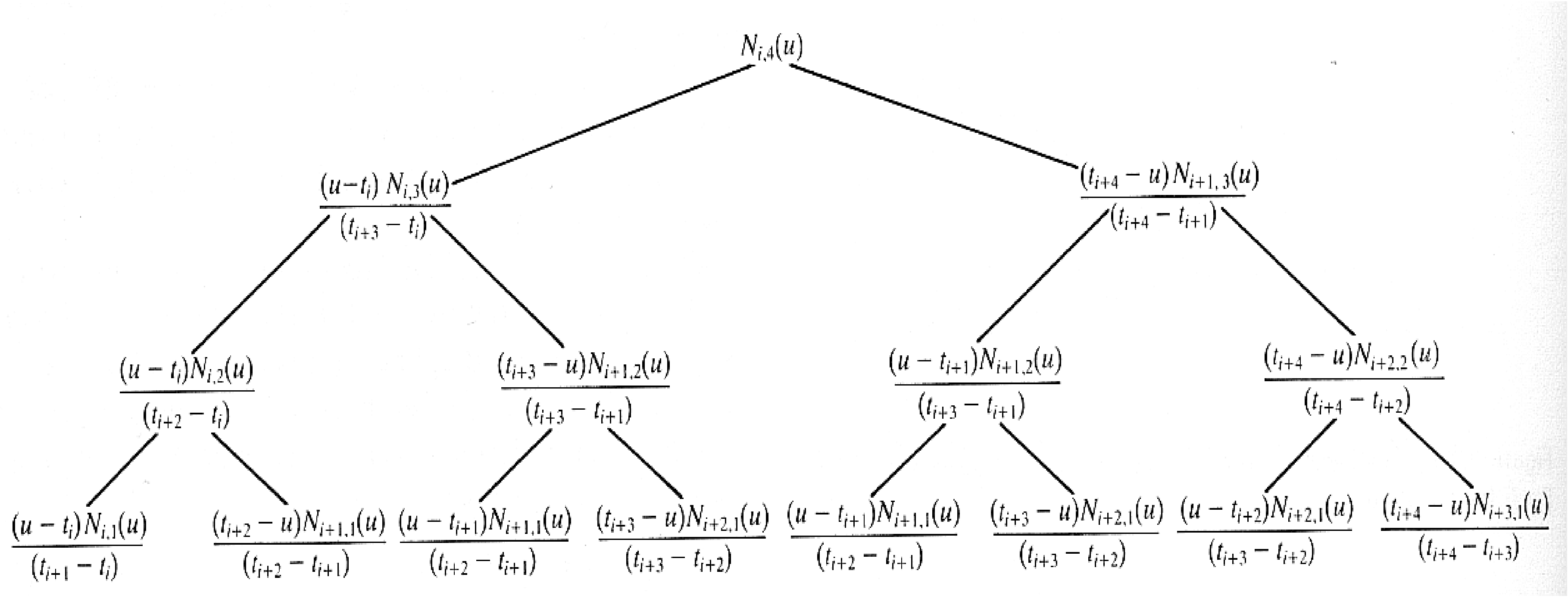
$$N_i^0(t) = \begin{cases} 1 & \text{if } \; t_i < t < t_{i+1} \\ 0 & \quad \text{otherwise} \end{cases}$$

$$N_i^n(t) = \frac{t - t_i}{t_{i+n} - t_i} N_i^{n-1}(t) - \frac{t - t_{i+n+1}}{t_{i+n+1} - t_{i+1}} N_{i+1}^{n-1}(t)$$



B-Spline Basis Functions

- **Recursive Definition**
  - Degree increases in every step
  - Support increases by one knot interval



$N_{i,4}(u)$

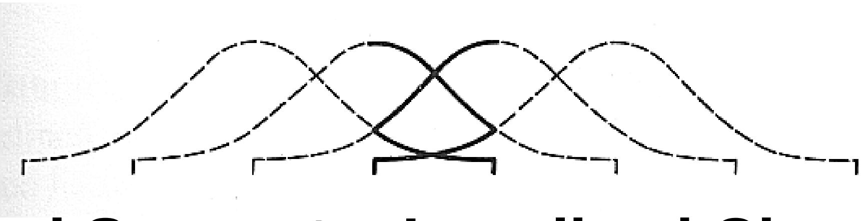
$\frac{(u-t_i)\, N_{i,3}(u)}{(t_{i+3}-t_i)}$ $\frac{(t_{i+4}-u)\, N_{i+1,3}(u)}{(t_{i+4}-t_{i+1})}$

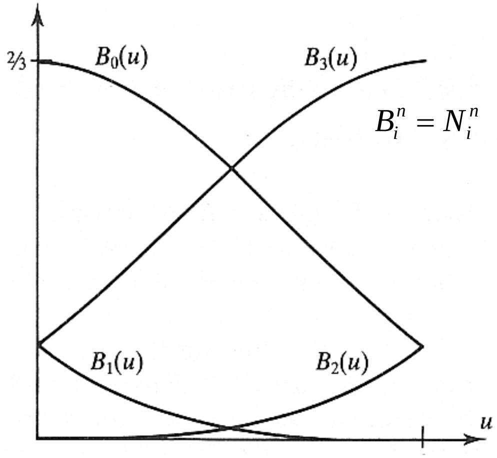
$\frac{(u-t_i)N_{i,2}(u)}{(t_{i+2}-t_i)}$ $\frac{(t_{i+3}-u)N_{i+1,2}(u)}{(t_{i+3}-t_{i+1})}$ $\frac{(u-t_{i+1})N_{i+1,2}(u)}{(t_{i+3}-t_{i+1})}$ $\frac{(t_{i+4}-u)N_{i+2,2}(u)}{(t_{i+4}-t_{i+2})}$

$\frac{(u-t_i)N_{i,1}(u)}{(t_{i+1}-t_i)}$ $\frac{(t_{i+2}-u)N_{i+1,1}(u)}{(t_{i+2}-t_{i+1})}$ $\frac{(u-t_{i+1})N_{i+1,1}(u)}{(t_{i+2}-t_{i+1})}$ $\frac{(t_{i+3}-u)N_{i+2,1}(u)}{(t_{i+3}-t_{i+2})}$ $\frac{(u-t_{i+1})N_{i+1,1}(u)}{(t_{i+2}-t_{i+1})}$ $\frac{(t_{i+3}-u)N_{i+2,1}(u)}{(t_{i+3}-t_{i+2})}$ $\frac{(u-t_{i+2})N_{i+2,1}(u)}{(t_{i+3}-t_{i+2})}$ $\frac{(t_{i+4}-u)N_{i+3,1}(u)}{(t_{i+4}-t_{i+3})}$

B-Spline Basis Functions

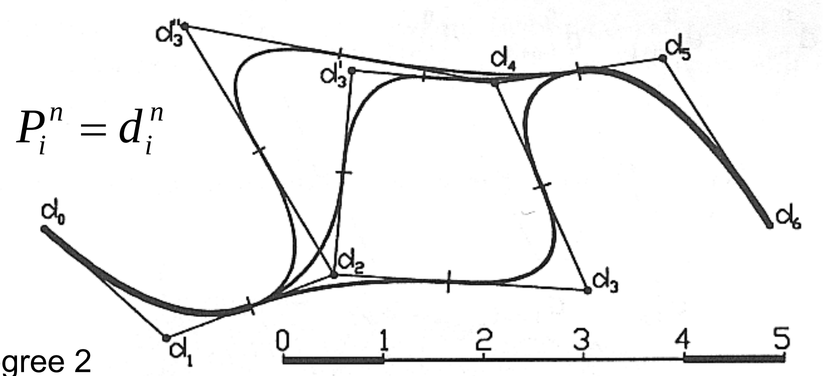
- **Uniform Knot Vector**
  - All knots at integer locations
    - UBS: Uniform B-Spline
  - Example: cubic B-Splines





- **Local Support = Localized Changes**
  - Basis functions affect only (n+1) Spline segments
  - Changes are localized

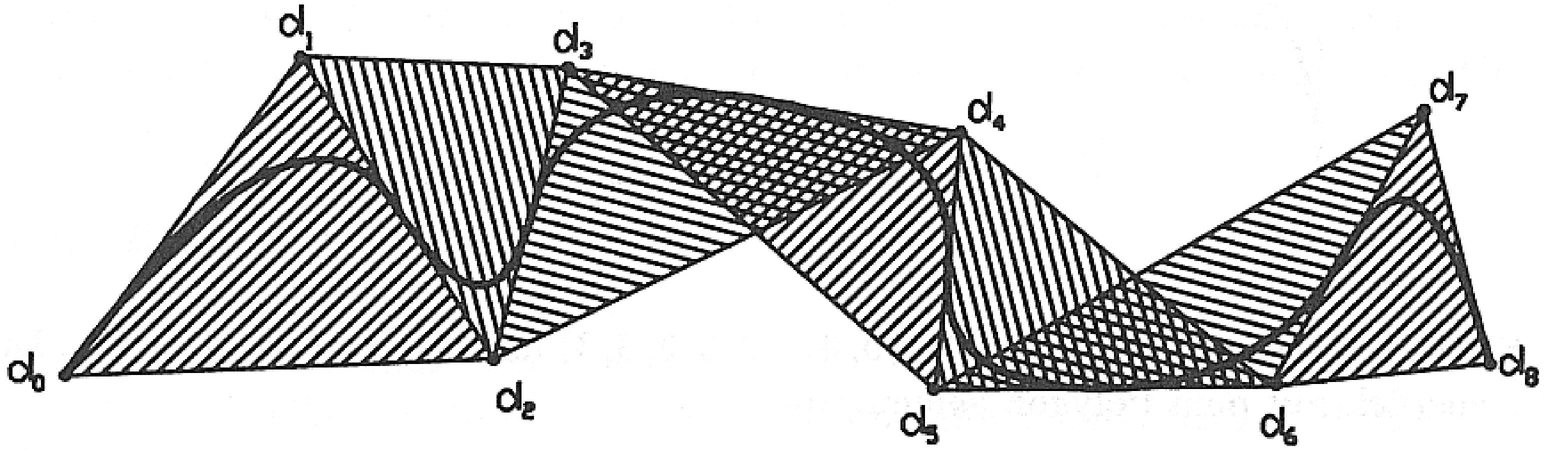
$$P_i^n = d_i^n$$



Degree 2

B-Spline Basis Functions

- **Convex Hull Property**
  - Spline segment lies in convex hull of (n+1) control points

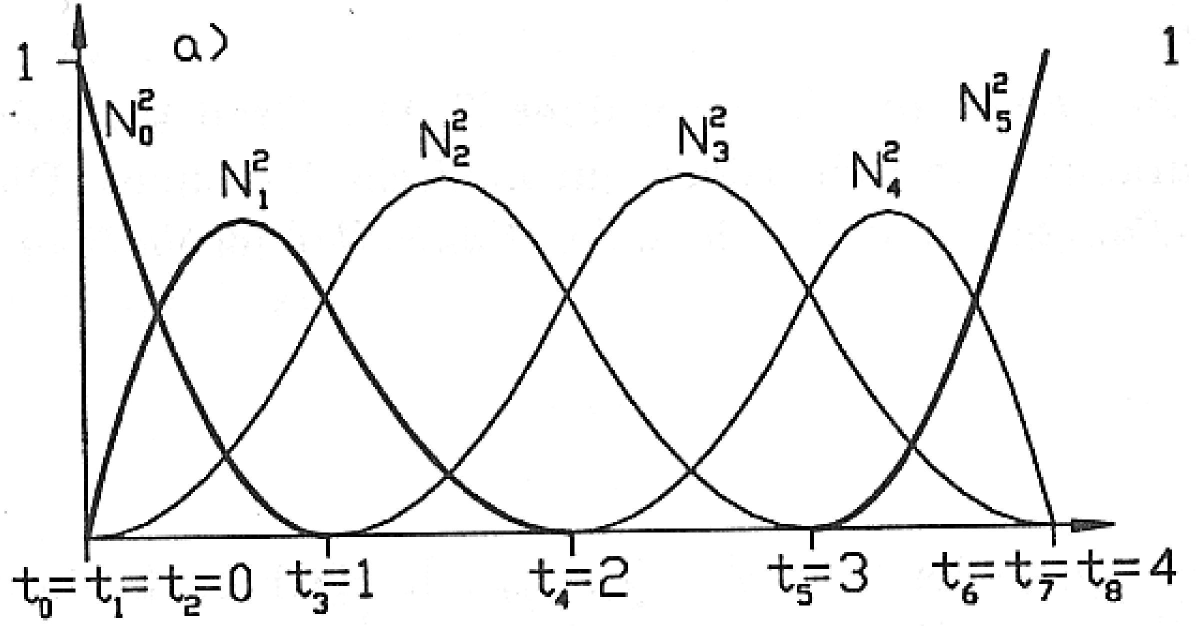


Degree 2

  - (n+1) control points lie on a straight line ➔ curve touches this line
  - n control points coincide ➔ curve interpolates this point and is tangential to the control polygon

Normalized Basis Functions

- **Basis Functions on an Interval**
  - Knots at beginning and end with multiplicity
    - NUBS: Non-uniform B-Splines
  - Interpolation of end points and tangents there
  - Conversion to Bézier segments via knot insertion



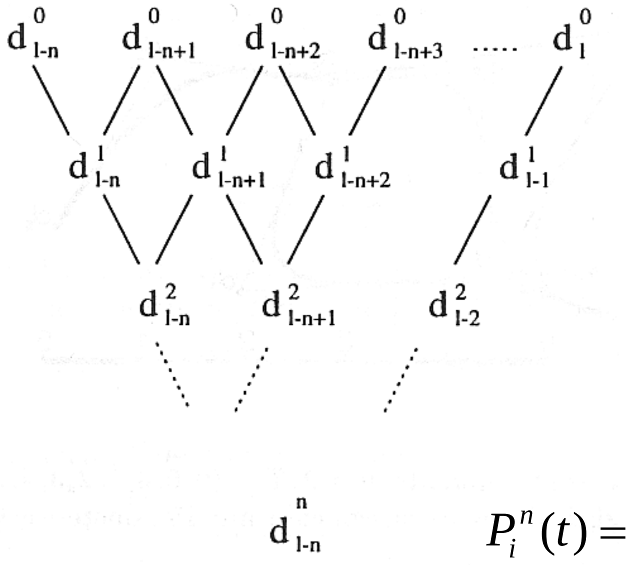
deBoor-Algorithm

- **Recursive Definition of Control Points**
  - Evaluation at t: $t_l < t < t_{l+1}$: $i \in \{l\text{-}n, ..., l\}$
    - Due to local support only affected by (n+1) control points

$$\underline{P}_i^r(t) = (1 - \frac{t - t_{i+r}}{t_{i+n+1} - t_{i+r}})\underline{P}_i^{r-1}(t) + \frac{t - t_{i+r}}{t_{i+n+1} - t_{i+r}}\underline{P}_{i+1}^{r-1}(t)$$

$$\underline{P}_i^0(t) = \underline{P}_i$$

- **Properties**
  - Affine invariance
  - Stable numerical evaluation
    - All coefficients > 0



$$P_i^n(t) = d_i^n$$