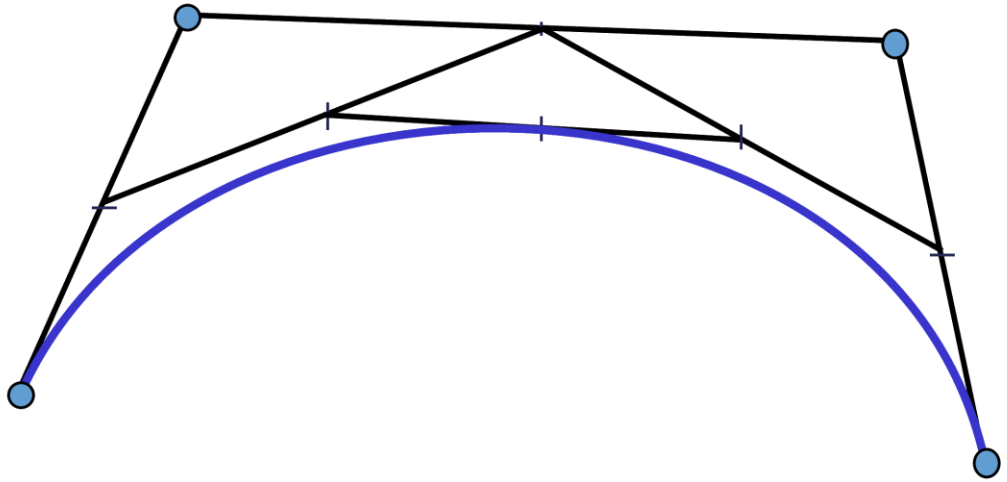
计算机辅助几何设计
2019秋学期

Bezier Splines

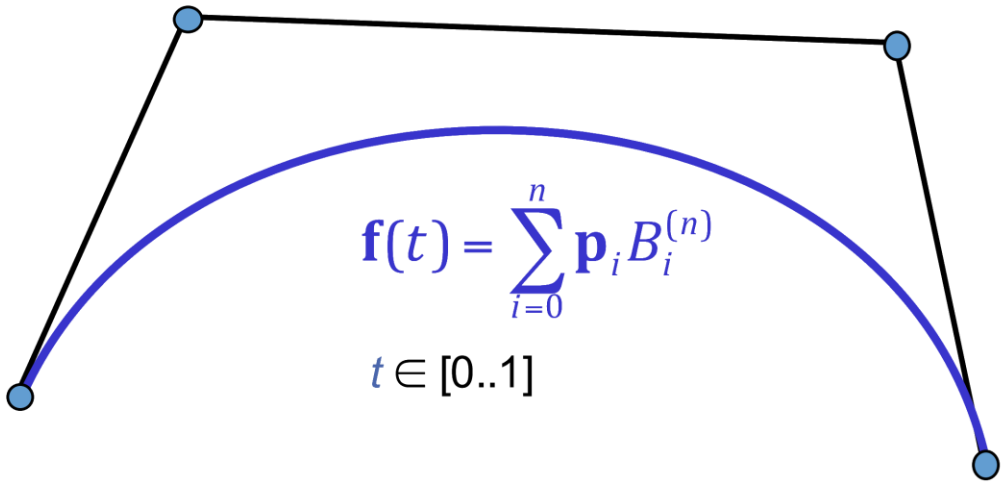
陈仁杰

中国科学技术大学

Recap



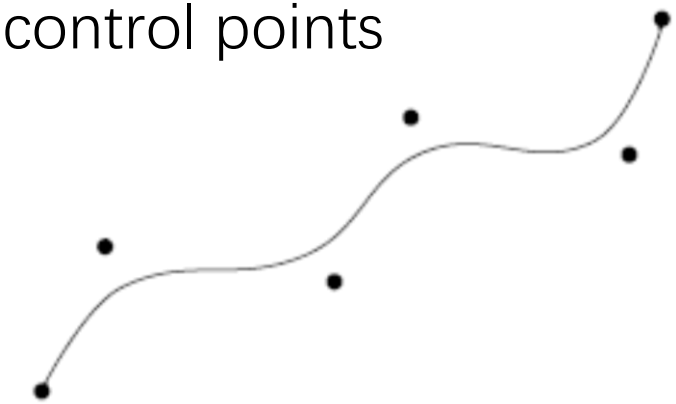
de Casteljau algorithm



Bernstein form

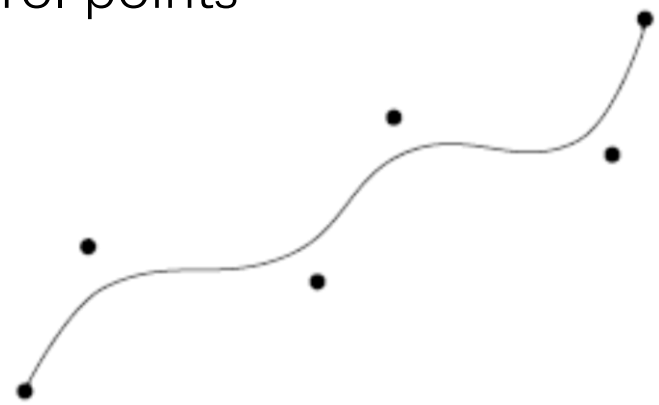
Recap

- **Bezier curves and curve design:**
  - The rough form is specified by the position of the control points
  - Results: smooth curve approximating the control points
  - Computation / Representation
    - de Casteljau algorithm
    - Bernstein form

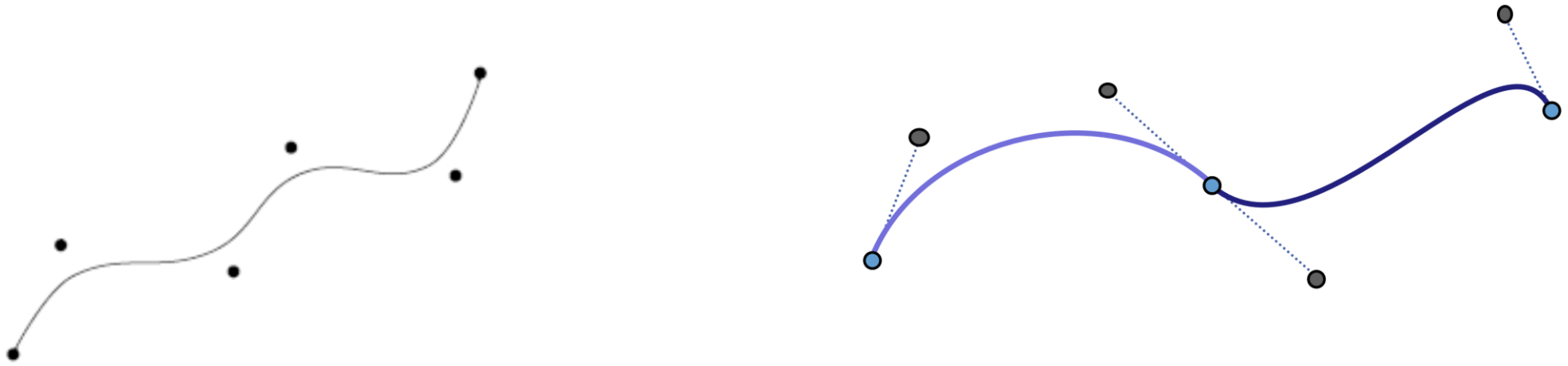


Recap

- **Bezier curves and curve design:**
  - The rough form is specified by the position of the control points
  - Results: smooth curve approximating the control points
  - Computation / Representation
    - de Casteljau algorithm
    - Bernstein form
  - Problems:
    - High polynomial degree
    - Moving a control point can change the whole curve
    - Interpolation of points
    - →**Bezier splines**



Recap



Approximation → Interpolation

Towards Bezier Splines

- **Interpolation problems:**
  - given:

    $\boldsymbol{k}_0, \ldots, \boldsymbol{k}_n \in \mathbb{R}^3$ control points

    $t_0, \ldots, t_n \in \mathbb{R}$ knot sequence

    $t_i < t_{i+1}$, for $i = 0, \ldots, n-1$
  - wanted
    - Interpolating curve $\boldsymbol{x}(i)$, i.e. $\boldsymbol{x}(t_i) = \boldsymbol{k}_i$ for $i = 0, \ldots, n$
  - Approach: “Joining” of n Bezier curves with certain intersection conditions

Towards Bezier Splines

- **The following issues arise when stitching together Bezier curves:**
  - Continuity
  - Parameterization
  - Degree

Bezier Splines

Parametric and Geometric Continuity

Parametric Continuity

- **Joining curves – continuity**
  - Given: 2 curves

    $\boldsymbol{x}_1(t)$ over $[t_0, t_1]$

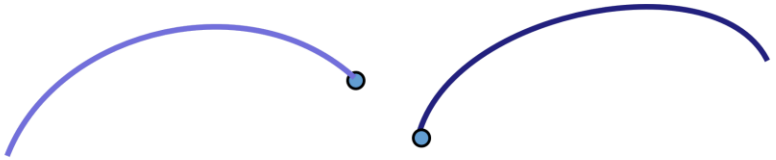
    $\boldsymbol{x}_2(t)$ over $[t_1, t_2]$

  - $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are C^r continuous at t_1, if all their 0^{th} to r^{th} derivative vectors coincides at t_1

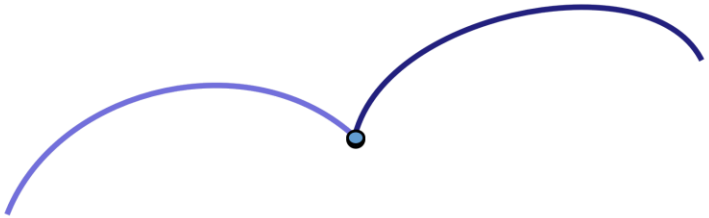
Parametric Continuity

- C^0: **position varies continuously**
- C^1: **First derivative is continuous across junction**
  - In other words: the velocity vector remains the same
- C^2: **Second derivative is continuous across junction**
  - The acceleration vector remains the same

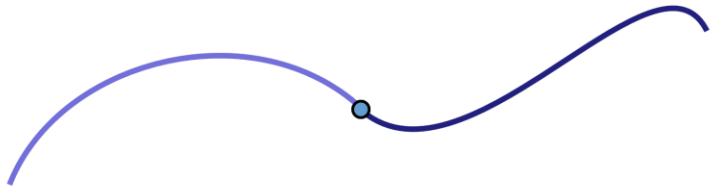
Parametric Continuity



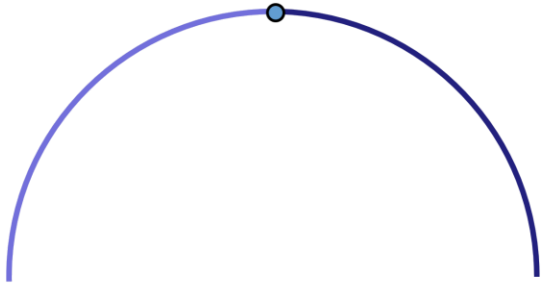
C^{-1} continuity



C^{0} continuity



C^{1} continuity



C^{2} continuity

Continuity

Parametric Continuity C^r:

- C^0, C^1, C^2 ⋯ continuity
- Does a particle moving on this curve have a smooth trajectory (position, velocity, acceleration, ⋯)?
- Depends on parameterization
- Useful for animation (object movement, camera paths)

Geometric Continuity G^r:

- Is the curve itself smooth?
- Independent of parameterization
- More relevant for modeling (curve design)

Geometric continuity:

- **Geometric continuity of curves**
  - Given: 2 curves

    $\boldsymbol{x}_1(t)$ over $[t_0, t_1]$

    $\boldsymbol{x}_2(t)$ over $[t_1, t_2]$

  - $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are G^r continuous in t_1, if they can be reparameterized in such a way that they are C^r continuous in t_1

Geometric continuity:

- $G^0 = C^0$: **position varies continuously (connected)**

- G^1: **tangent direction varies continuously (same tangent)**
  - In other words: the **normalized** tangent varies continuously
  - Equivalently: The curve can be reparameterzed so that it becomes C^1
  - Also equivalent: A unit speed parameterization would be C^1

- G^2: **curvature varies continuously (same tangent and curvature)**
  - Equivalently: The curve can be reparameterized so that it becomes C^2
  - Also equivalent: A unit speed parameterization would be C^2

$$\kappa = \|c''\|$$

Bezier Splines

Parameterization

Bezier spline curves

- **Local and global parameters:**
  - Given:
    - $\boldsymbol{b}_0,\cdots,\boldsymbol{b}_n$
    - $\boldsymbol{y}(u)$: Bezier curve in interval $[0,1]$
    - $\boldsymbol{x}(t)$: Bezier curve in interval $[t_i, t_{i+1}]$
  - Setting $u(t) = \frac{t-t_i}{t_{i+1}-t_i}$
  - Results in $\boldsymbol{x}(t) = \boldsymbol{y}\big(u(t)\big)$

  The local parameter u runs from 0 to 1,
  while the global parameter t runs from t_i to t_{i+1}

Bezier spline curves

$$u(t) = \frac{t - t_i}{t_{i+1} - t_i}$$

$$\boldsymbol{x}(t) = \boldsymbol{y}\big(u(t)\big)$$

- **Derivatives:**

$$\dot{\boldsymbol{x}}(t) = \quad \dot{\boldsymbol{y}}\big(u(t)\big) \cdot \dot{u}(t) = \frac{\dot{\boldsymbol{y}}\big(u(t)\big)}{t_{i+1} - t_i}$$

$$\ddot{\boldsymbol{x}}(t) = \quad \ddot{\boldsymbol{y}}\big(u(t)\big) \cdot \big(\dot{u}(t)\big)^2 + \dot{\boldsymbol{y}}\big(u(t)\big) \cdot \ddot{u}(t) = \frac{\ddot{\boldsymbol{y}}\big(u(t)\big)}{(t_{i+1} - t_i)^2}$$

...

$$\boldsymbol{x}^{[n]}(t) = \frac{\boldsymbol{y}^{[n]}\big(u(t)\big)}{(t_{i+1} - t_i)^n}$$

Bezier Curve

$$\boldsymbol{f}(t) = \sum_{i=0}^{n} B_i^n(t)\,\boldsymbol{p}_i$$

- Function value at $\{0,1\}$:

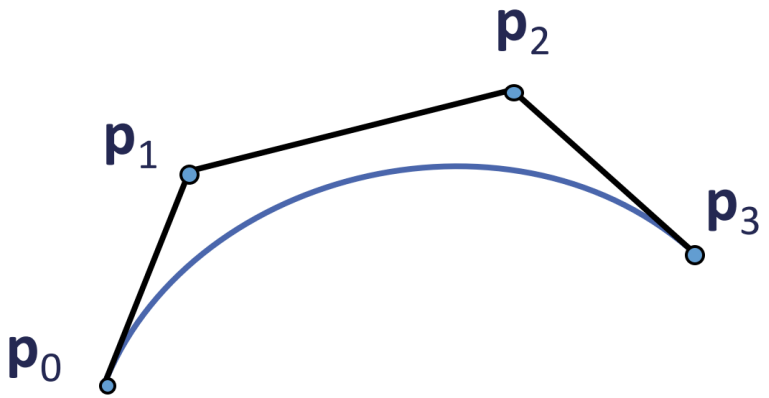
$$\boldsymbol{f}(0) = \boldsymbol{p}_0$$
$$\boldsymbol{f}(1) = \boldsymbol{p}_1$$

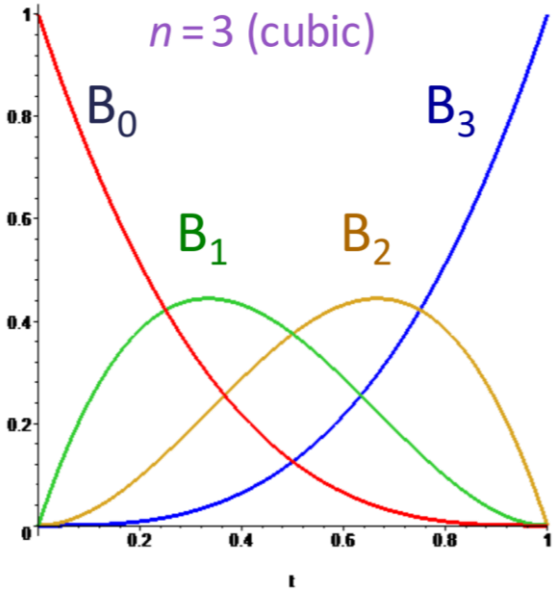
- First derivative vector at $\{0,1\}$

$$\boldsymbol{f}'(0) = n[\boldsymbol{p}_1 - \boldsymbol{p}_0]$$
$$\boldsymbol{f}'(1) = n[\boldsymbol{p}_n - \boldsymbol{p}_{n-1}]$$

- Second derivative vector at $\{0,1\}$

$$\boldsymbol{f}''(0) = n(n-1)[\boldsymbol{p}_2 - 2\boldsymbol{p}_1 + \boldsymbol{p}_0]$$
$$\boldsymbol{f}''(1) = n(n-1)[\boldsymbol{p}_n - 2\boldsymbol{p}_{n-1} + \boldsymbol{p}_{n-2}]$$





Bezier spline curves

- **Special cases:**

$$\dot{\boldsymbol{x}}(t_i) = \frac{n \cdot (\boldsymbol{b}_1 - \boldsymbol{b}_0)}{t_{i+1} - t_i}$$

$$\dot{\boldsymbol{x}}(t_{i+1}) = \frac{n \cdot (\boldsymbol{b}_n - \boldsymbol{b}_{n-1})}{t_{i+1} - t_i}$$

$$\ddot{\boldsymbol{x}}(t_i) = \frac{n \cdot (n-1) \cdot (\boldsymbol{b}_2 - 2\boldsymbol{b}_1 + \boldsymbol{b}_0)}{(t_{i+1} - t_i)^2}$$

$$\ddot{\boldsymbol{x}}(t_{i+1}) = \frac{n \cdot (n-1) \cdot (\boldsymbol{b}_n - 2\boldsymbol{b}_{n-1} + \boldsymbol{b}_{n-2})}{(t_{i+1} - t_i)^2}$$

Bezier Splines

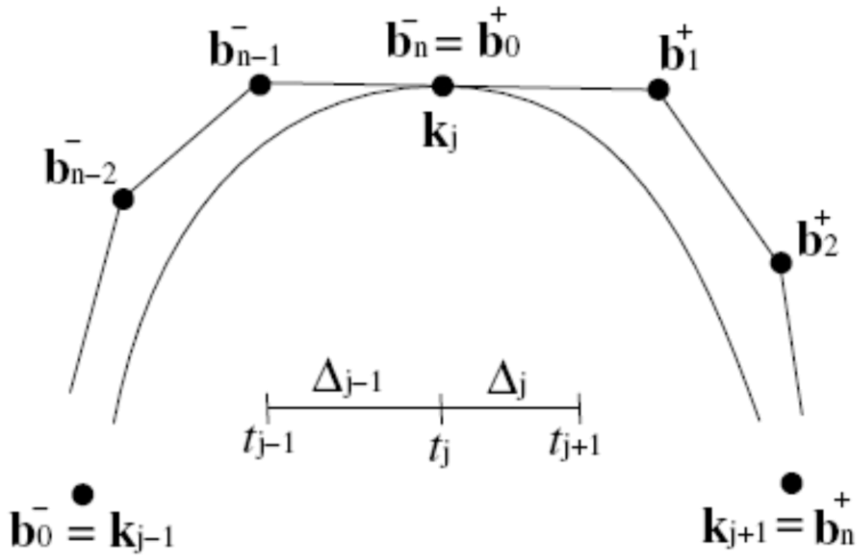
General Case

Bezier spline curves

- **Joining Bezier curves:**
  - Given: 2 Bezier curves of degree n through

$$k_{j-1} = b_0^-, b_1^-, \ldots, b_n^- = k_j$$

$$k_j = b_0^+, b_1^+, \ldots, b_n^+ = k_{j+1}$$

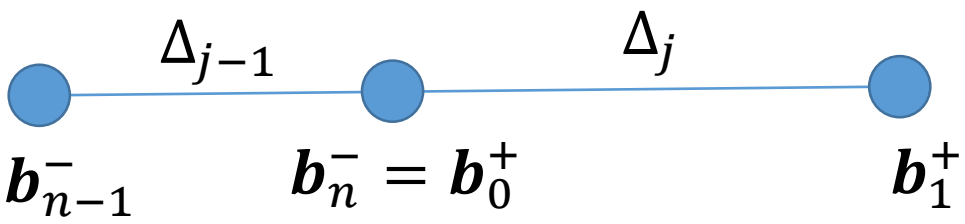


$$\dot{\boldsymbol{x}}(t_i) = \frac{n \cdot (\boldsymbol{b}_1 - \boldsymbol{b}_0)}{t_{i+1} - t_i}$$

Bezier spline curves

- Required: C^1-continuity at $\boldsymbol{k}_j$:
- $\boldsymbol{b}_{n-1}^-, \boldsymbol{k}_j, \boldsymbol{b}_1^+$ collinear and

$$\frac{\boldsymbol{b}_n^- - \boldsymbol{b}_{n-1}^-}{t_j - t_{j-1}} = \frac{\boldsymbol{b}_1^+ - \boldsymbol{b}_0^+}{t_{j+1} - t_j}$$



Bezier spline curves

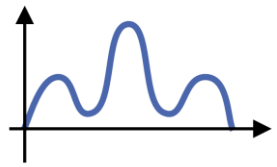
- Required: G^1-continuity at $\boldsymbol{k}_j$:
  - $\boldsymbol{b}_{n-1}^-$, $\boldsymbol{k}_j$, $\boldsymbol{b}_1^+$ collinear
- Less restrictive than C^1-continuity

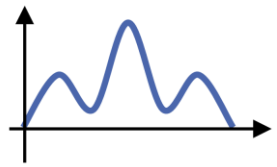
Bezier Splines

Choosing the degree

Choosing the Degree…

- **Candidates:**
  - $d = 0$ (piecewise constant) : not smooth
  - $d = 1$ (piecewise linear) : not smooth enough
  - $d = 2$ (piecewise quadratic) : constant 2nd derivative, still too inflexible
  - $d = 3$ (piecewise cubic): degree of choice for computer graphics applications





Cubic Splines

- **Cubic piecewise polynomials:**
  - We can attain C^2 continuity without fixing the second derivative throughout the curve

Cubic Splines

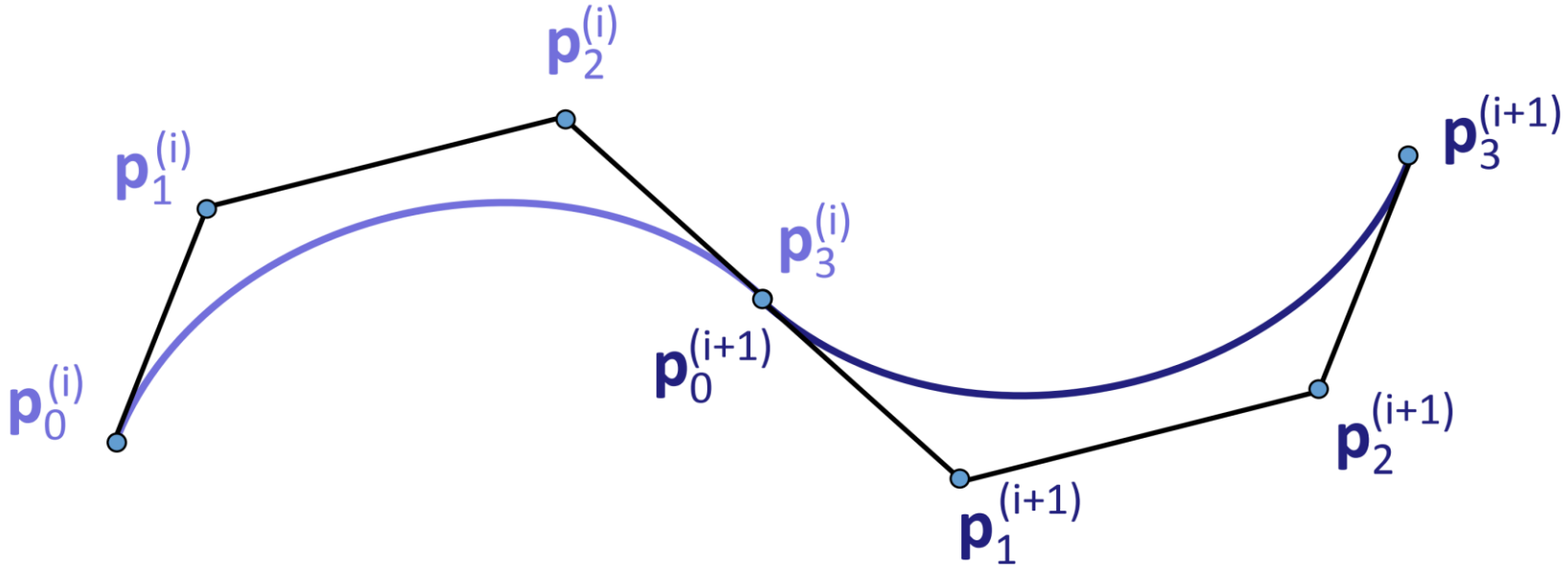
- **Cubic piecewise polynomials:**
  - We can attain C^2 continuity without fixing the second derivative throughout the curve
  - C^2 continuity is perceptually important
    - Motion: continuous position, velocity & acceleration
      Discontinuous acceleration noticeable (object/camera motion)
    - We can see second order shading discontinuities
      (esp.: reflective objects)

Cubic Splines

- **Cubic piecewise polynomials**
  - We can attain C^2 continuity without fixing the second derivative throughout the curve
  - C^2 continuity is perceptually important
    - We can see second order shading discontinuities
      (esp.: reflective objects)
    - Motion: continuous *position, velocity* & *acceleration*
      Discontinuous acceleration noticeable (object/camera motion)
  - One more argument for cubics:
    - Among all C^2 curves that interpolate a set of points (and obey to the same end condition), a piecewise cubic curve has the least integral acceleration ("smoothest curve you can get").

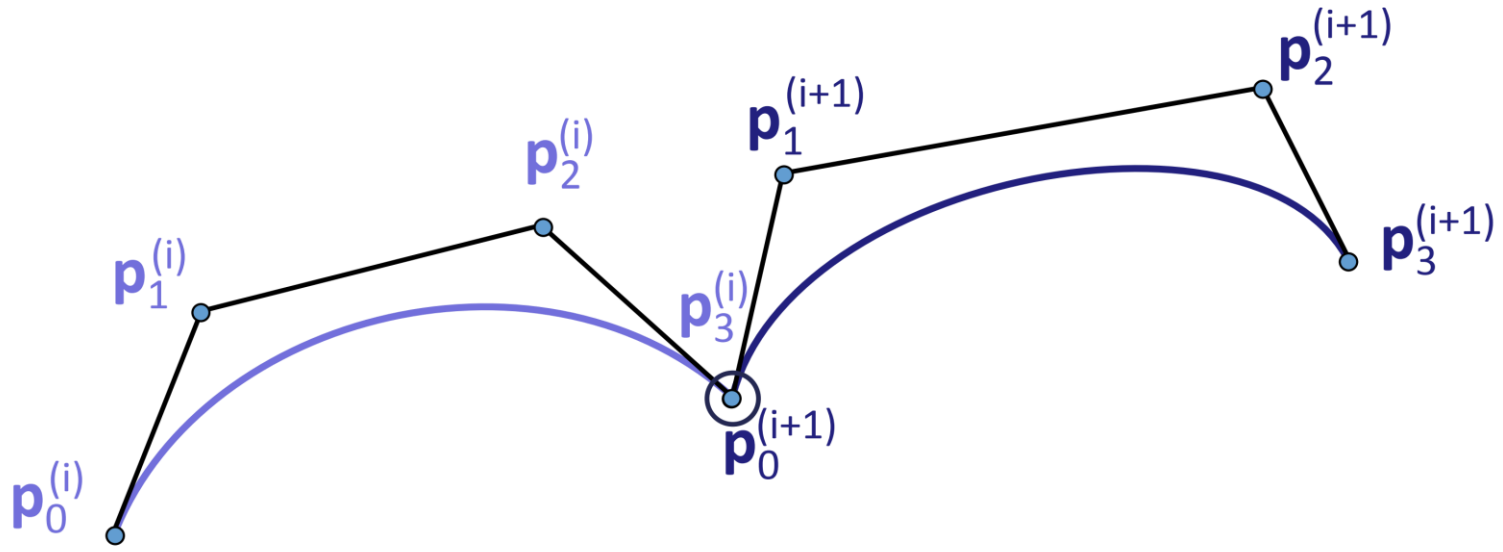
Bezier Splines

- **Local control: Bezier splines**
  - Concatenate several curve segments
  - Question: Which constraints to place upon the control points in order to get C^{-1}, C^{0}, C^{1}, C^{2} continuity?



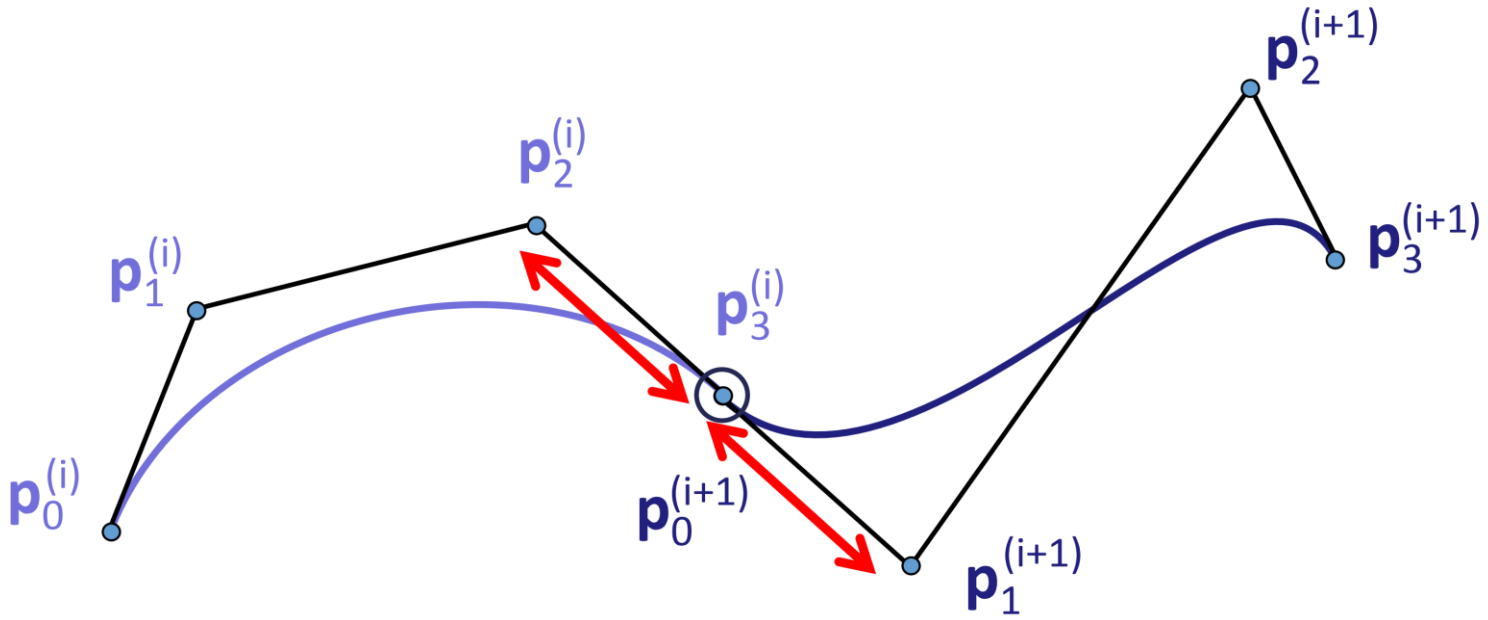
Bezier Spline Continuity

- **Rules for Bezier spline continuity:**
  - C^0 continuity:
    - Each spline segment interpolates the first and last control point
    - Therefore: Points of neighboring segments have to coincide for C^0 continuity



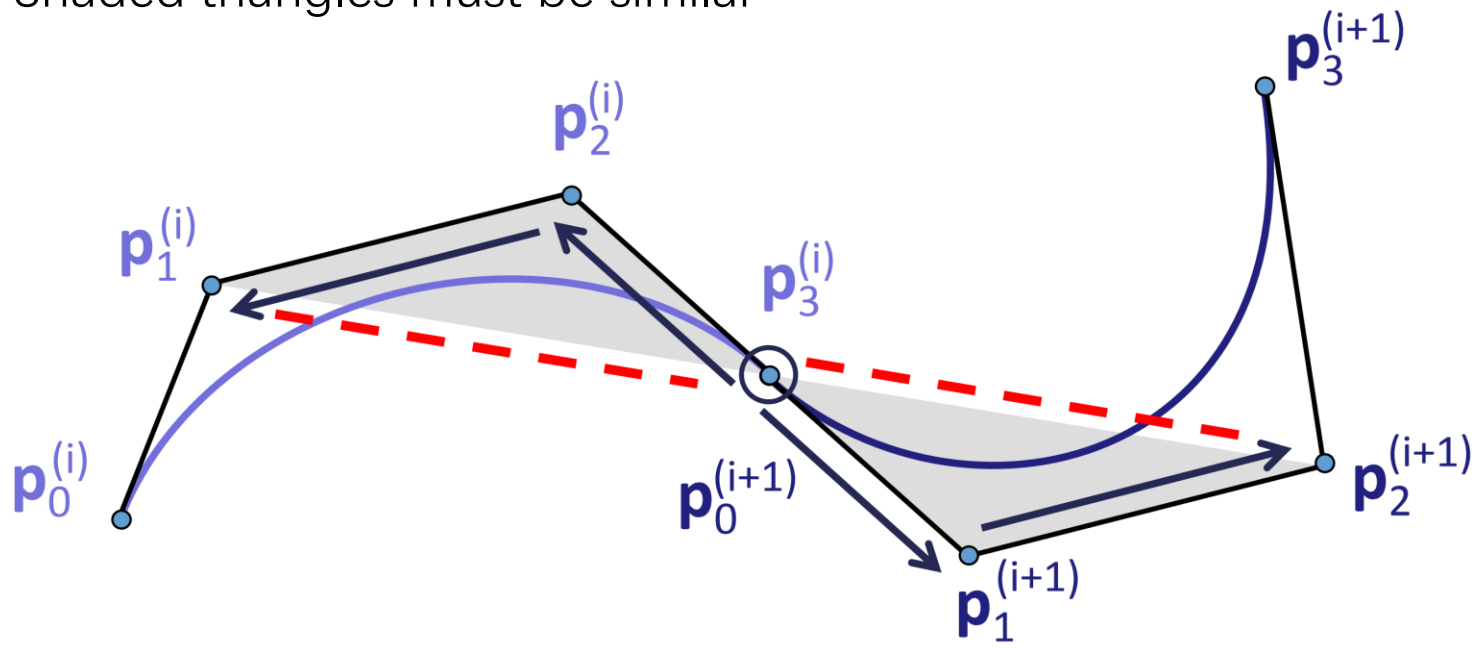
Bezier Spline Continuity

- **Rules for Bezier spline continuity:**
  - Additional requirement for C^1 continuity:
    - Tangent vectors are proportional to differences $\boldsymbol{p}_1 - \boldsymbol{p}_0$, $\boldsymbol{p}_n - \boldsymbol{p}_{n-1}$
    - Therefore: These vectors must be identical for C^1 continuity

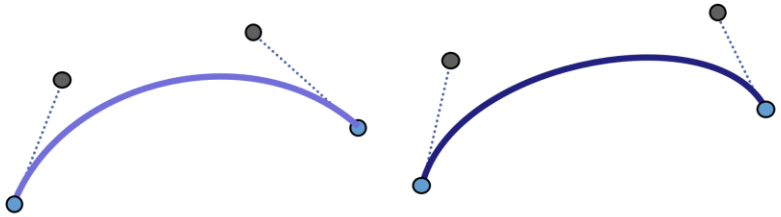


Bezier Spline Continuity

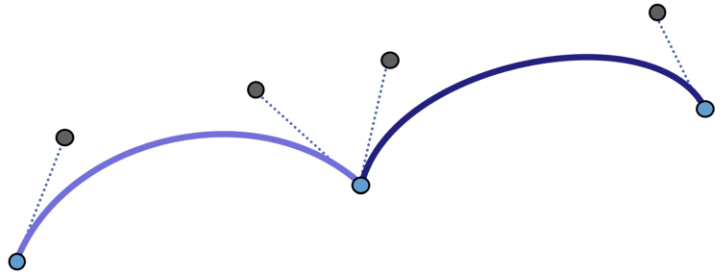
- **Rules for Bezier spline continuity**
  - Additional requirement for C^2 continuity:
    - d^2/dt^2 vectors are prop. to $(\boldsymbol{p}_2 - 2\boldsymbol{p}_1 + \boldsymbol{p}_0)$, $(\boldsymbol{p}_n - 2\boldsymbol{p}_{n-1} + \boldsymbol{p}_{n-2})$
    - Tangents must be the same (C^2 implies C^1)
    - Shaded triangles must be similar



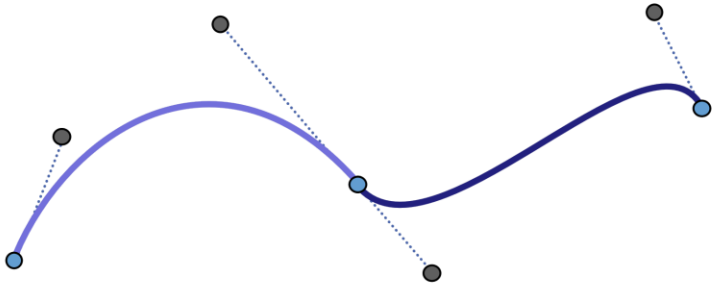
Continuity



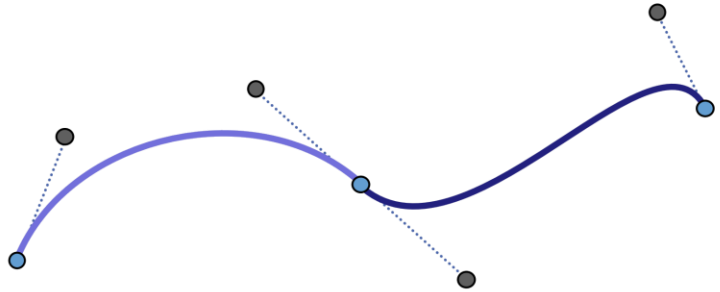
C^{-1} continuity



C^{0} continuity



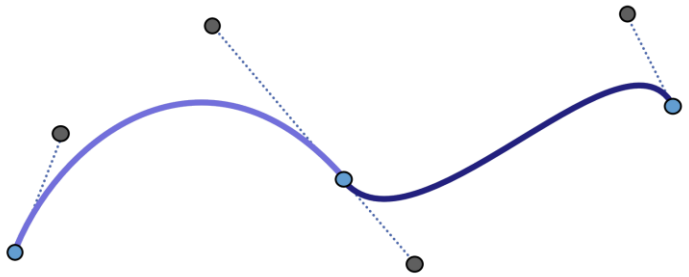
G^{1} continuity



C^{1} continuity

Continuity for Bezier Splines

- This means



G^1 continuity

This Bezier curve is G^1: It can be reparameterized to become C^1. (Just increase the speed for the second segment by ratio of tangent vector lengths)

In Practice

- Everyone is using cubic Bezier curves
- Higher degree are rarely used (some CAD/CAM applications)
- Typically: “points & handles” interface
- Four modes:
  - Discontinuous (two curves)
  - C^0 Continuous (points meet)
  - G^1 continuous: Tangent direction continuous
    - Handles point into the same direction, but different length
  - C^1 continuous
    - Handle points have symmetric vectors
- C^2 is more restrictive: control via k_i

Bezier spline curves

- Required: C^2-continuity at $\boldsymbol{k}_j$
- C^1 implies $$\frac{\boldsymbol{b}_n^- - \boldsymbol{b}_{n-1}^-}{t_j - t_{j-1}} = \frac{\boldsymbol{b}_1^+ - \boldsymbol{b}_0^+}{t_{j+1} - t_j}$$
- C^2 implies $$\frac{\boldsymbol{b}_n^- - 2\boldsymbol{b}_{n-1}^- + \boldsymbol{b}_{n-2}^-}{(t_j - t_{j-1})^2} = \frac{\boldsymbol{b}_2^+ - 2\boldsymbol{b}_1^+ + \boldsymbol{b}_0^+}{(t_{j+1} - t_j)^2}$$

$$\frac{t_{j+1} - t_j}{t_j - t_{j-1}} = \frac{\Delta_j}{\Delta_{j-1}}$$

Bezier spline curves

- Required: C^2-continuity at $\boldsymbol{k}_j$:

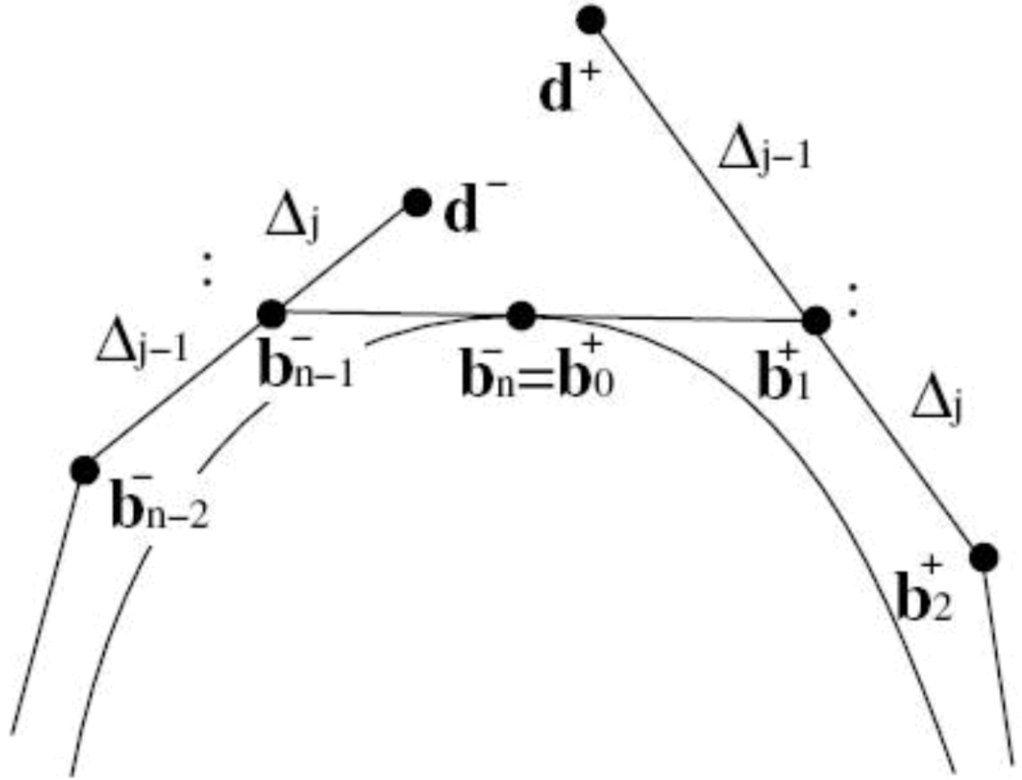
- Introduce $\boldsymbol{d}^- = \boldsymbol{b}^-_{n-1} + \frac{\Delta_j}{\Delta_{j-1}}(\boldsymbol{b}^-_{n-1} - \boldsymbol{b}^-_{n-2})$

  and $\boldsymbol{d}^+ = \boldsymbol{b}^+_1 \quad - \frac{\Delta_{j-1}}{\Delta_j}(\boldsymbol{b}^+_2 - \boldsymbol{b}^+_1)$

- By manipulating equation from the previous slides
- C^2-continuity $\Leftrightarrow$ C^1-continuity and $\boldsymbol{d}^- = \boldsymbol{d}^+$

Bezier spline curves

- C^2-continuity $\Leftrightarrow$ C^1-continuity and $\boldsymbol{d}^- = \boldsymbol{d}^+$

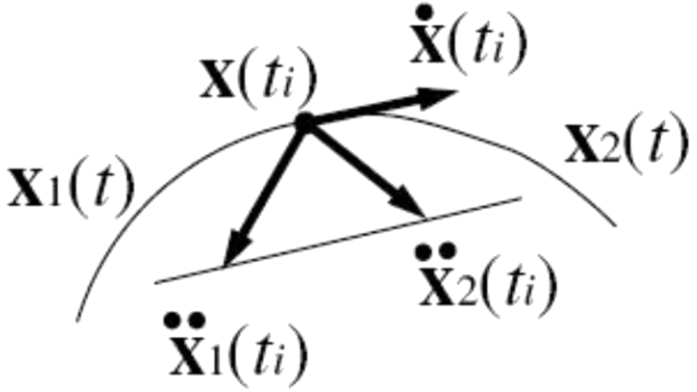


Bezier spline curves

- G^2-continuity in general (for all types of curves):
- Given:
  - $\boldsymbol{x}_1(t)$, $\boldsymbol{x}_2(t)$ with
  - $\boldsymbol{x}_1(t_i) = \boldsymbol{x}_2(t_i) = \boldsymbol{x}(t_i)$
  - $\dot{\boldsymbol{x}}_1(t_i) = \dot{\boldsymbol{x}}_2(t_i) = \dot{\boldsymbol{x}}(t_i)$
- Then the requirement for G^2-continuity at $t = t_i$:

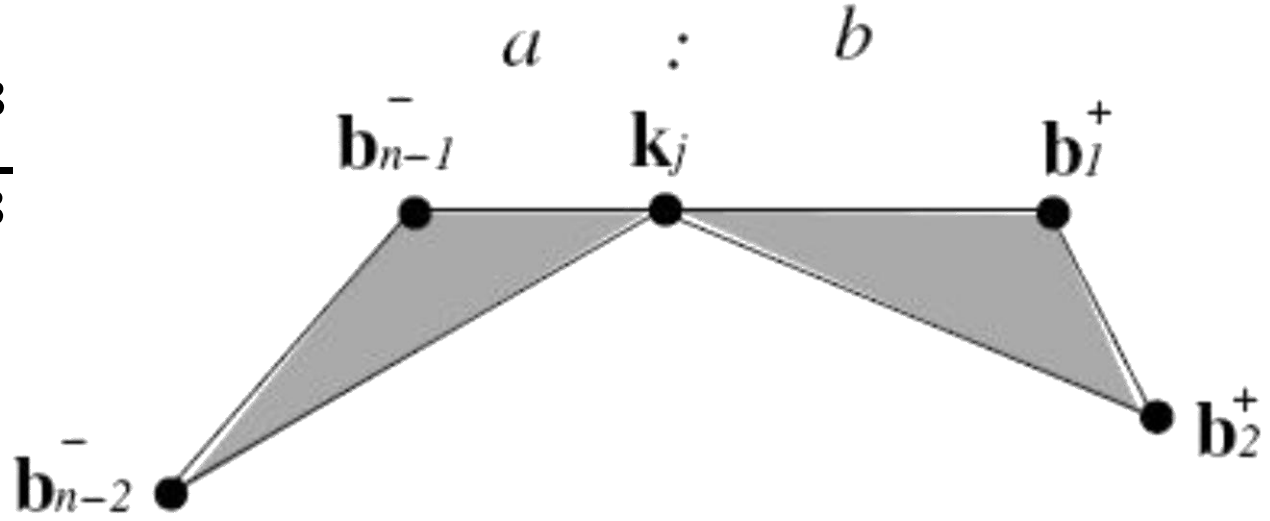
$$\ddot{\boldsymbol{x}}_2(t_i) - \ddot{\boldsymbol{x}}_1(t_i) \parallel \dot{\boldsymbol{x}}(t_i)$$

Parallel



Bezier spline curves

- Required: G^2-continuity at k_j:
- G^1-continuity
- Co-planarity for : $\boldsymbol{b}_{n-2}^{-}, \boldsymbol{b}_{n-1}^{-}, \boldsymbol{k}_j, \boldsymbol{b}_1^{+}, \boldsymbol{b}_2^{+}$
- And: $\frac{\text{area}\left(\boldsymbol{b}_{n-2}^{-}, \boldsymbol{b}_{n-1}^{-}, \boldsymbol{k}_j\right)}{\text{area}\left(\boldsymbol{k}_j, \boldsymbol{b}_1^{+}, \boldsymbol{b}_2^{+}\right)} = \frac{a^3}{b^3}$



Bezier Splines

C^2 Cubic Bezier Splines

Cubic Bezier Splines

- **Cubic Bezier spline curves**
  - Given:

$$\boldsymbol{k}_0, \ldots, \boldsymbol{k}_n \in \mathbb{R}^3 \qquad \text{control points}$$

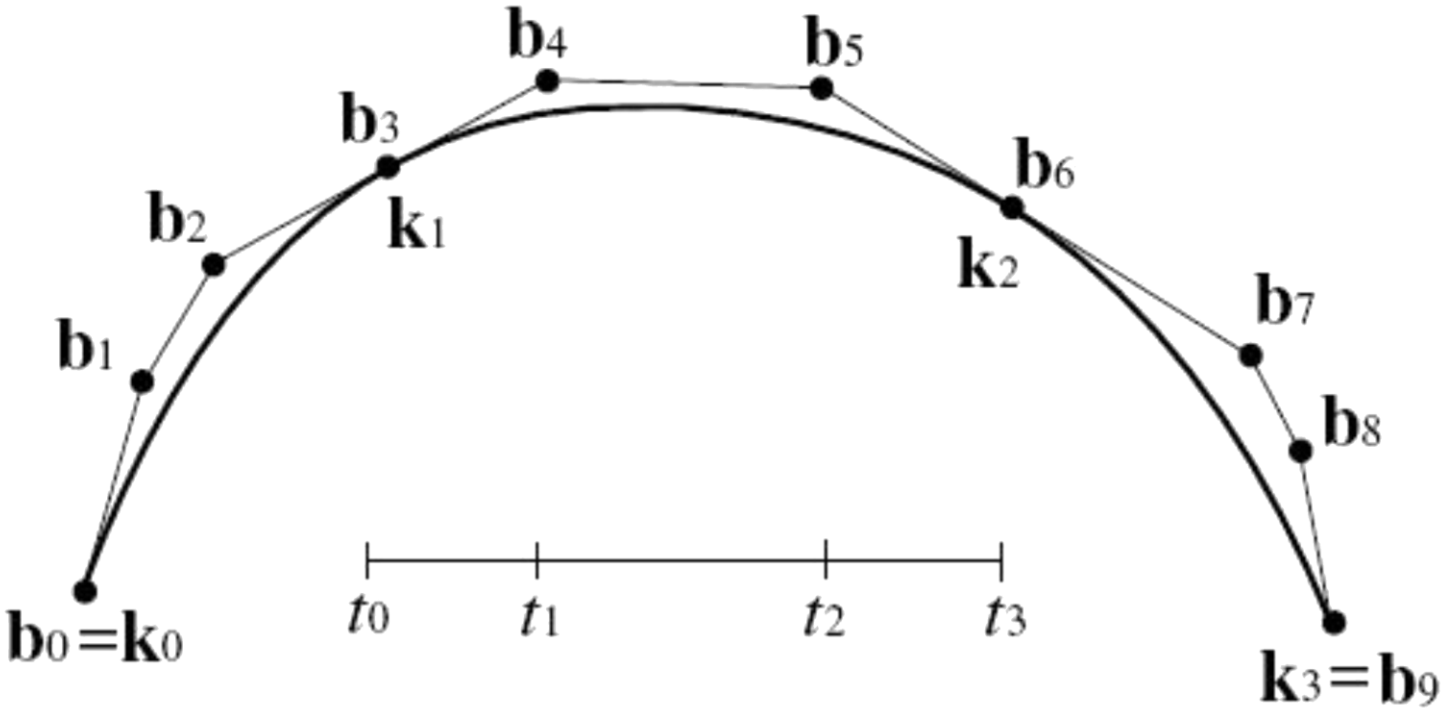
$$t_0, \ldots, t_n \in \mathbb{R} \qquad \text{knot sequence}$$

$$t_i < t_{i+1}, \text{ for } i = 0, \ldots, n_1$$

  - Wanted: Bezier points $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{3n}$ for an interpolating C^2-continuous piecewise cubic Bezier spline curve

Cubic Bezier Splines

- Examples: $n = 3$:



Cubic Bezier Splines

- $3n + 1$ unknown points
- $b_{3i} = k_i$ for $i = 0, \dots, n$

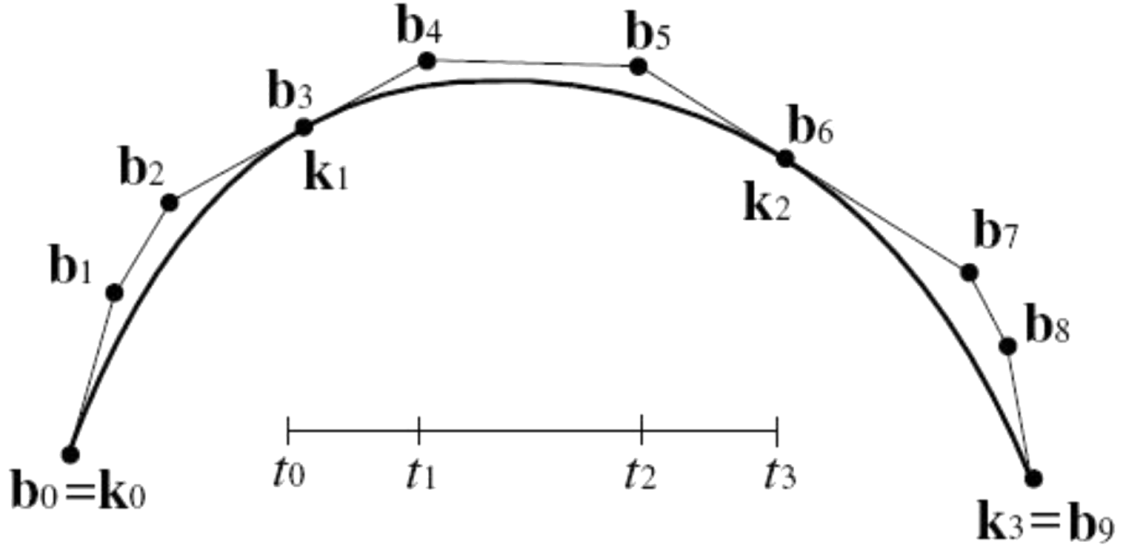
  $n + 1$ equations
- C^1 in points k_i for $i = 1, \dots, n - 1$

  $n - 1$ equations
- C^2 in points k_i for $i = 1, \dots, n - 1$

  $n - 1$ equations

$3n - 1$ equations

⇒ **2 additional conditions necessary: end conditions**

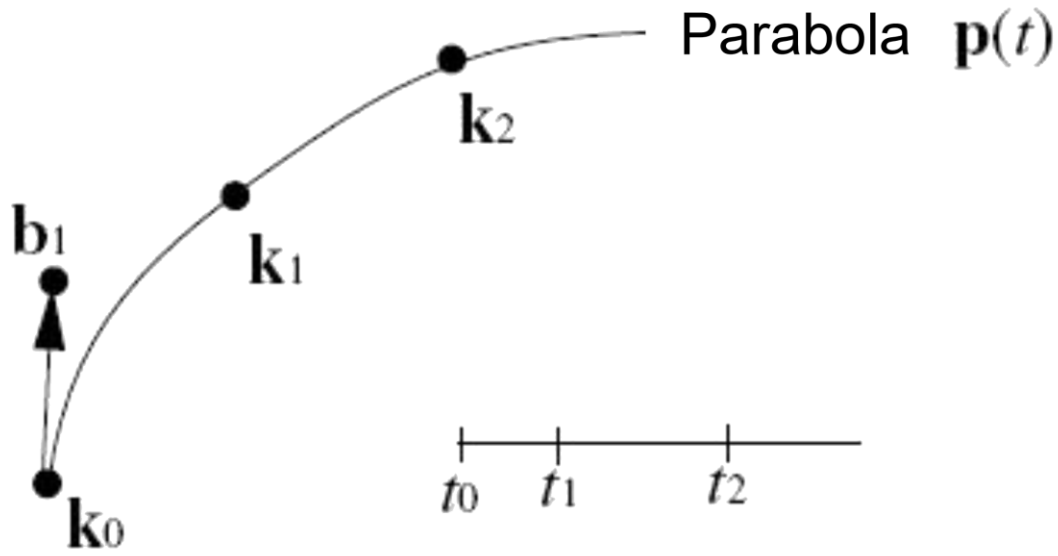


Bezier Splines

C^2 Cubic Bezier Splines: End conditions

Bezier spline curves: End conditions

- **Bessel's end condition**
  - The tangential vector in $\boldsymbol{k_0}$ is equivalent to the tangential vector of the parabola interpolating $\{\boldsymbol{k_0}, \boldsymbol{k_1}, \boldsymbol{k_2}\}$ at $\boldsymbol{k_0}$:



$$\dot{\boldsymbol{x}}(t_i) = \frac{n \cdot (\boldsymbol{b}_1 - \boldsymbol{b}_0)}{t_{i+1} - t_i}$$

Bezier spline curves: End conditions

Parabola Interpolating $\{\boldsymbol{k}_0, \boldsymbol{k}_1, \boldsymbol{k}_2\}$

$$\boldsymbol{p}(t) = \frac{(t_2 - t)(t_1 - t)}{(t_2 - t_0)(t_1 - t_0)}\boldsymbol{k}_0 + \frac{(t_2 - t)(t - t_0)}{(t_2 - t_1)(t_1 - t_0)}\boldsymbol{k}_1 + \frac{(t_0 - t)(t_1 - t)}{(t_2 - t_1)(t_2 - t_0)}\boldsymbol{k}_2$$

Its derivative

$$\dot{\boldsymbol{p}}(t_0) = -\frac{(t_2 - t_0) + (t_1 - t_0)}{(t_2 - t_0)(t_1 - t_0)}\boldsymbol{k}_0 + \frac{(t_2 - t_0)}{(t_2 - t_1)(t_1 - t_0)}\boldsymbol{k}_1 - \frac{(t_1 - t_0)}{(t_2 - t_1)(t_2 - t_0)}\boldsymbol{k}_2$$

Location of $\boldsymbol{b}_1$

$$\boldsymbol{b}_1 = \boldsymbol{b}_0 + \frac{t_1 - t_0}{3}\dot{\boldsymbol{p}}(t_0)$$

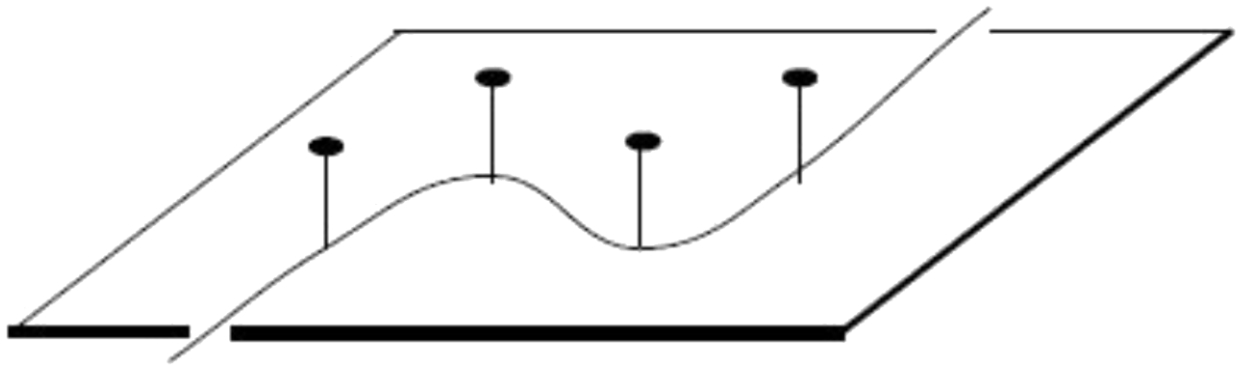
$$\ddot{\boldsymbol{x}}(t_i) = \frac{n \cdot (n-1) \cdot (\boldsymbol{b}_2 - 2\boldsymbol{b}_1 + \boldsymbol{b}_0)}{(t_{i+1} - t_i)^2}$$

Bezier spline curves: End conditions

- Natural end condition:

$$\ddot{\boldsymbol{x}}(t_0) = 0 \Leftrightarrow \boldsymbol{b}_1 = \frac{\boldsymbol{b}_2 + \boldsymbol{b}_0}{2}$$

$$\ddot{\boldsymbol{x}}(t_n) = 0 \Leftrightarrow \boldsymbol{b}_{3n-1} = \frac{\boldsymbol{b}_{3n-2} + \boldsymbol{b}_{3n}}{2}$$



End conditions: Examples

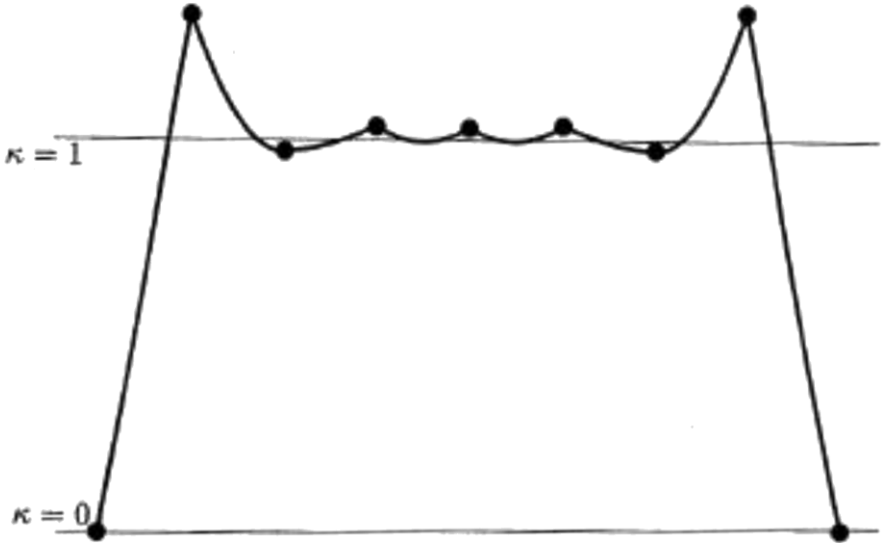
- ***Bessel* end condition**

Curve: circle of radius 1

Curvature plot

End conditions: Examples

- ***Natural*** **end condition**



Curve: circle of radius 1

Curvature plot

Bezier Splines

C^2 Cubic Bezier Splines: parameterization

Bezier spline curves: Parameterization

- **Approach so far:**
  - Given: control points $\boldsymbol{k}_0, \ldots, \boldsymbol{k}_n$ and knot sequence $t_0 < \cdots < t_n$
  - Wanted: interpolating curve
  - Problem: Normally, the knot sequence is not given, but it influences the curve

Bezier spline curves: Parameterization

- **Equidistant (uniform) parameterization**
  - $t_{i+1} - t_i = \text{const}$
  - e.g. $t_i = i$
  - Geometry of the data points is not considered
- **Chordal parameterization**
  - $t_{i+1} - t_i = \|\boldsymbol{k}_{i+1} - \boldsymbol{k}_i\|$
  - Parameter intervals proportional to the distances of neighbored control points

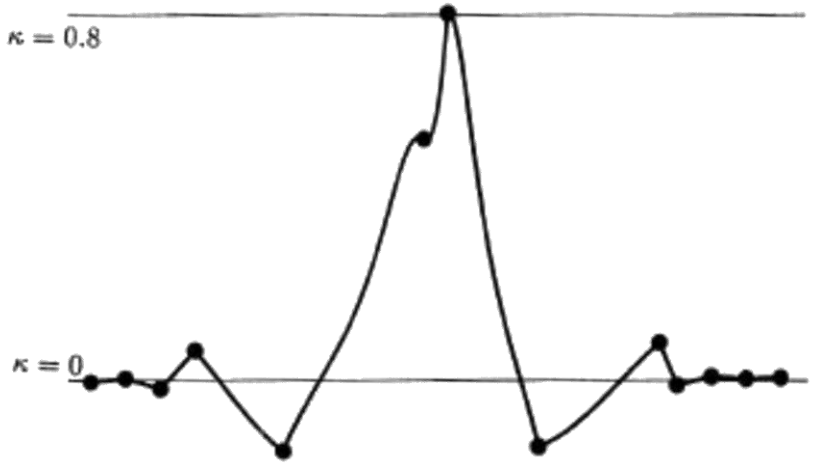
Bezier spline curves: Parameterization

- Centripetal parameterization
  - $t_{i+1} - t_i = \sqrt{\|\boldsymbol{k}_{i+1} - \boldsymbol{k}_i\|}$
- Foley parameterization
  - Involvement of angles in the control polygon
  - $t_{i+1} - t_i = \|\boldsymbol{k}_{i+1} - \boldsymbol{k}_i\| \cdot \left(1 + \frac{3}{2}\frac{\hat{\alpha}_i \|\boldsymbol{k}_i - \boldsymbol{k}_{i-1}\|}{\|\boldsymbol{k}_i - \boldsymbol{k}_{i-1}\| + \|\boldsymbol{k}_{i+1} - \boldsymbol{k}_i\|} + \frac{3}{2}\frac{\hat{\alpha}_{i+1} \|\boldsymbol{k}_{i+1} - \boldsymbol{k}_i\|}{\|\boldsymbol{k}_{i+1} - \boldsymbol{k}_i\| + \|\boldsymbol{k}_{i+2} - \boldsymbol{k}_{i+1}\|}\right)$
  - with $\hat{\alpha}_i = \min\left(\pi - \alpha_i, \frac{\pi}{2}\right)$
  - and $\alpha_i = \mathrm{angle}(\boldsymbol{k}_{i-1}, \boldsymbol{k}_i, \boldsymbol{k}_{i+1})$
- Affine invariant parameterization
  - Parameterization on the basis of an affine invariant distance measure (e.g. G. Nielson)

Bezier spline curves: Parameterization

- Examples: Chordal parameterization

Curve

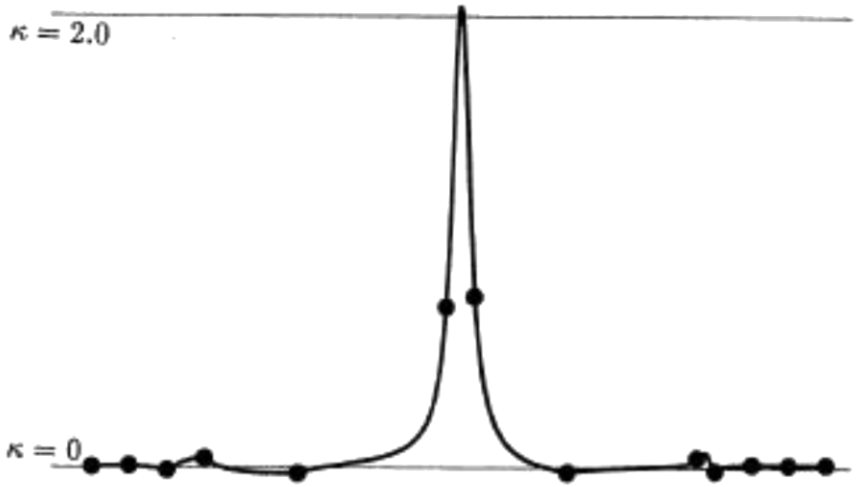


Curvature plot

Bezier spline curves: Parameterization

- Examples: Centripetal parameterization

Curve

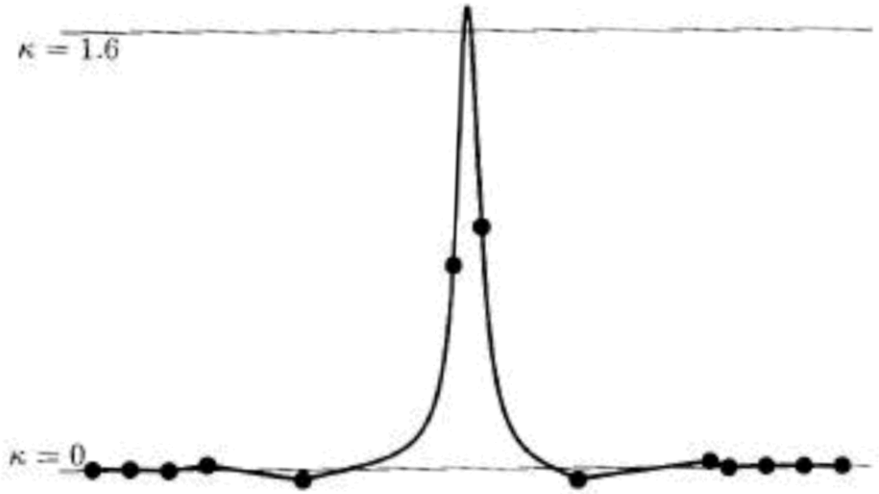


Curvature plot

Bezier spline curves: Parameterization

- Examples: Foley parameterization

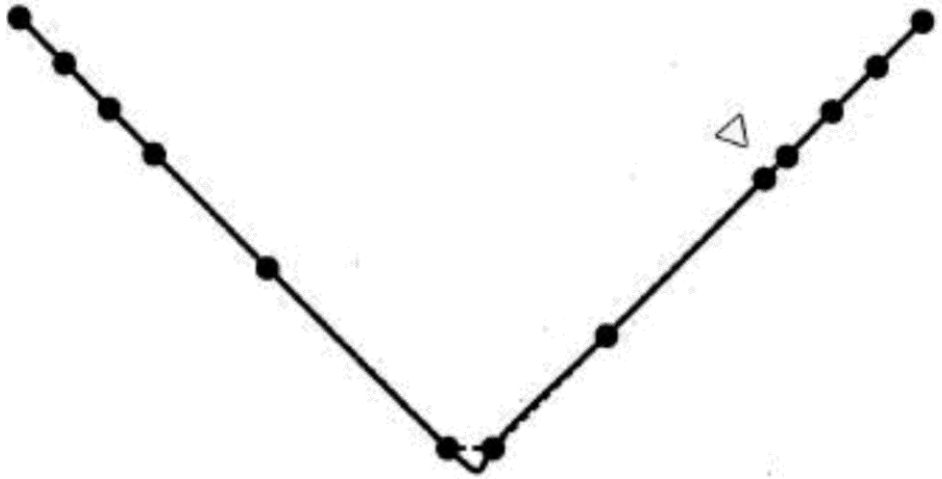
Curve



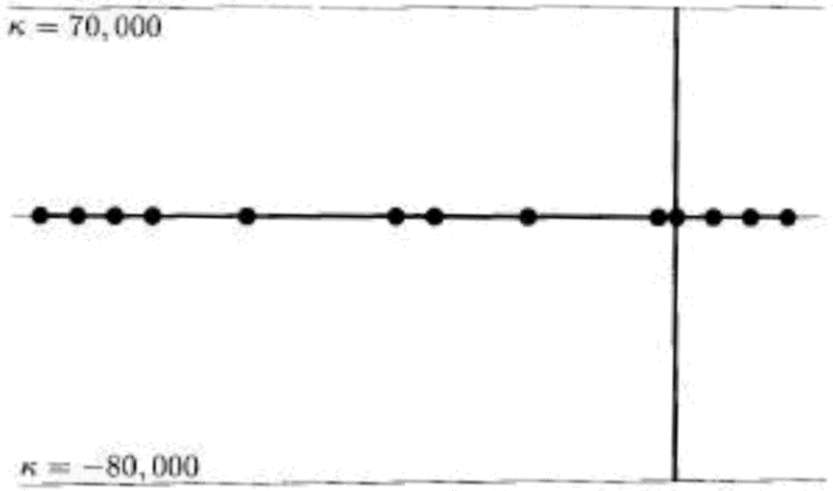
Curvature plot

Bezier spline curves: Parameterization

- Examples: Uniform parameterization



Curve



Curvature plot

Bezier Splines

C^2 Cubic Bezier Splines: closed curves

Closed cubic Bezier spline curves

- **Closed cubic Bezier spline curves**
  - Given:

    $\boldsymbol{k}_0, \ldots, \boldsymbol{k}_{n-1}, \boldsymbol{k}_n = \boldsymbol{k}_0$: control points

    $t_0 < \cdots < t_n$: knot sequence
  - As an “end condition” for the piecewise cubic curve we place:

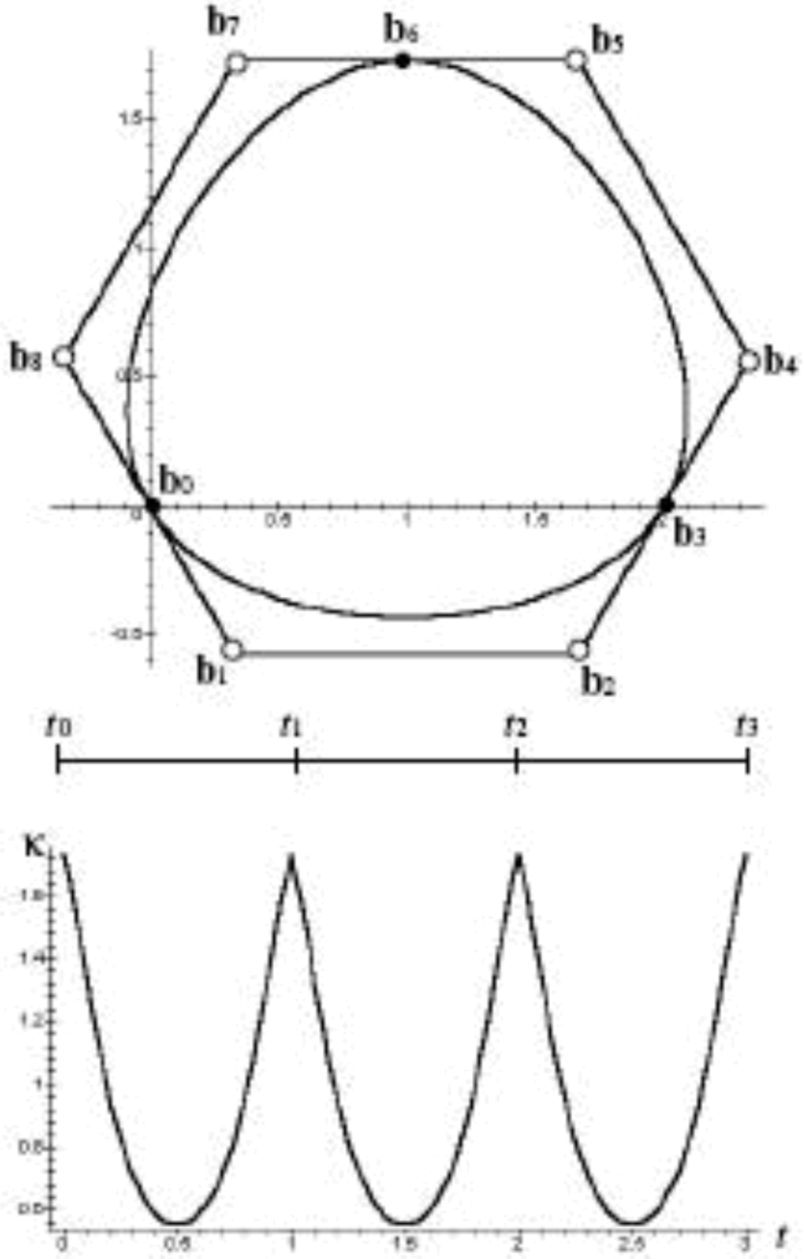
$$\dot{\boldsymbol{x}}(t_0) = \dot{\boldsymbol{x}}(t_n)$$

$$\ddot{\boldsymbol{x}}(t_0) = \ddot{\boldsymbol{x}}(t_n)$$

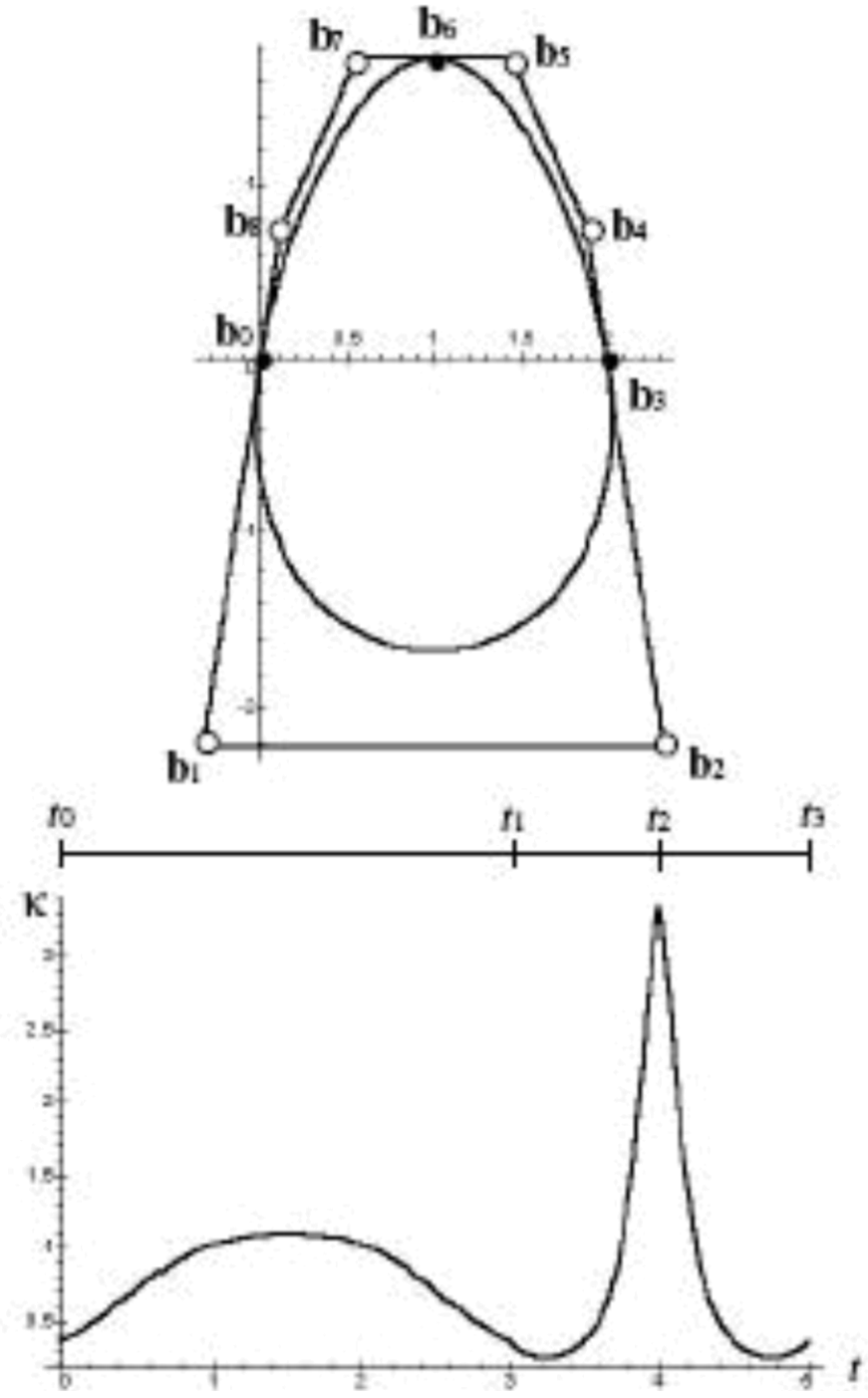
Closed cubic Bezier spline curves

- **Closed cubic Bezier spline curves**
  - $\rightarrow C^2$-continuous and closed curve
  - Advantage of closed curves: selection of the end condition is not necessary!
  - Examples (on the next 3 slides): $n = 3$

Examples



Examples



Examples