Representing Curves and Surfaces

Introduction

- We need smooth curves and surfaces in many applications:
  - model real world objects
  - computer-aided design (CAD)
  - high quality fonts
  - data plots
  - artists sketches

Introduction

- The need to present curves and surfaces arises in two cases:
  - In modeling **existing objects** (a face, a mountain).
  - In modeling "from scratch" where **no preexisting physical object** is being presented.

Introduction

- Most common representation for surfaces:
  - polygon mesh
  - parametric surfaces
  - quadric surfaces
- Solid modeling

Introduction

- Polygon mesh:
  - Is a collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons.
  - An edge connects two vertices, and a polygon is a closed sequence of edges.
  - good for boxes, cabinets, building exteriors.
  - bad for curved surfaces.
  - errors can be made arbitrarily small at the cost of space and execution time

Representing Polygon Meshes

- explicit representation
- by a list of vertex coordinates

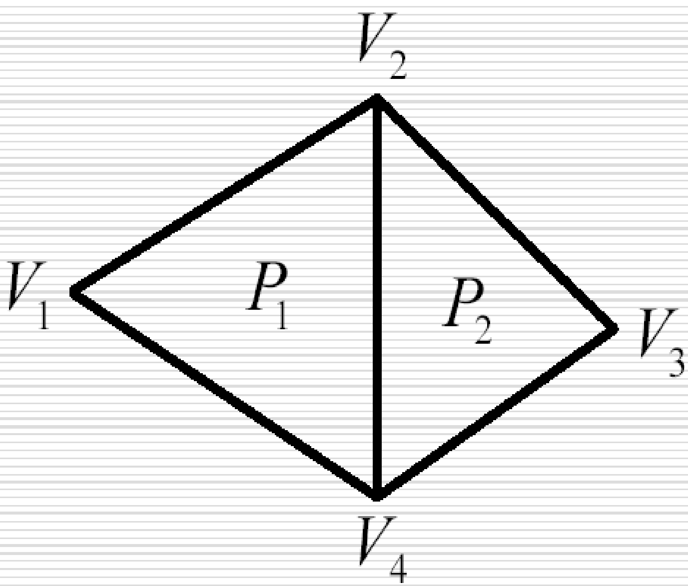
$$P = ((x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n))$$

- pointers to a vertex list
- pointers to an edge list

Explicit representation

- Advantages:
  - It is space efficient for a single polygon.
- Disadvantages:
  - In polygon mesh representation much space is lost because the coordinates of shared vertices are duplicated
  - There is no explicit representation of shared edges and vertices.
  - If edges are being drawn, each shared edge is drawn twice.

Pointers to a Vertex List

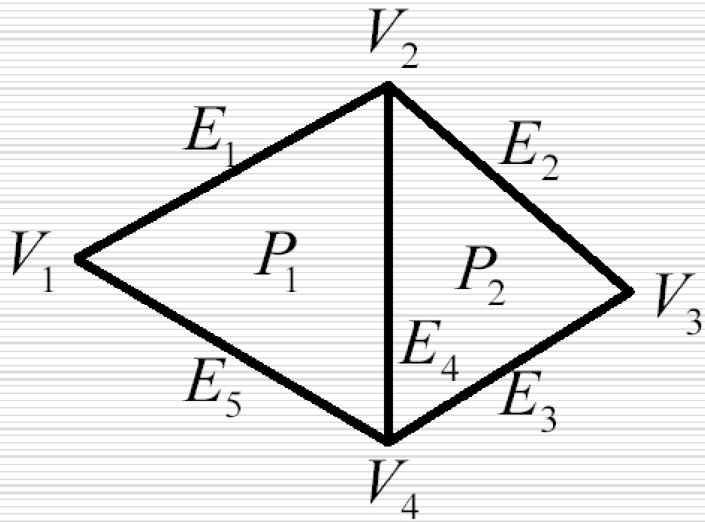


$$
\begin{aligned}
V &= (V_1, V_2, V_3, V_4) \\
&= ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4)) \\
P_1 &= (1,2,4) \\
P_2 &= (4,2,3)
\end{aligned}
$$

Pointers to a Vertex List

- Advantages:
  - Since each vertex is stored once, considerable space is saved.
  - Coordinates of a vertex can be changed easily
- Disadvantages:
  - It is difficult to find polygons that share an edge.
  - Shared polygon edges are drawn twice when polygons outlines are displayed.

Pointers to an Edge List



$$
\begin{aligned}
V &= (V_1, V_2, V_3, V_4) \\
&= ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4)) \\
E_1 &= (V_1, V_2, P_1, \lambda) \\
E_2 &= (V_2, V_3, P_2, \lambda) \\
E_3 &= (V_3, V_4, P_2, \lambda) \\
E_4 &= (V_4, V_2, P_1, P_2) \\
E_5 &= (V_4, V_1, P_1, \lambda) \\
P_1 &= (E_1, E_4, E_5) \\
P_2 &= (E_2, E_3, E_4)
\end{aligned}
$$

Pointers to a Edge List

- Advantages:
  - Redundant clipping, transformation and scan conversion are avoided
  - Filled polygons are displayed easily
- Disadvantages:
  - It is not easy to determine which edges are incident to a vertex.

Introduction

- Parametric polynomial curves:
  - point on 3D curve = $(x(t), y(t), z(t))$
  - $x(t)$, $y(t)$, and $z(t)$ are polynomials
  - usually cubic: cubic curves

Parametric cubic curves

- Polylines and polygons:
  - large amounts of data to achieve good accuracy
  - interactive manipulation of the data is tedious
- Higher-order curves:
  - more compact (use less storage)
  - easier to manipulate interactively
- Possible representations of curves:
  - explicit, implicit, and parametric

Parametric cubic curves

- General form:

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

$$y(t) = a_y t^3 + b_y t^2 + c_y t \quad + d_y$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t \quad + d_z$$

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \qquad T = [t^3 \quad t^2 \quad t \quad 1]$$

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot C = T \cdot M \cdot G$$

Tangent Vector

$$\frac{d}{dt}Q(t) = Q'(t) = \left[\frac{d}{dt}x(t) \quad \frac{d}{dt}y(t) \quad \frac{d}{dt}z(t) \right]^{\mathrm{T}}$$

$$= \frac{d}{dt}C \bullet T = C \bullet \left[3t^2 \quad 2t \quad 1 \quad 0 \right]^{\mathrm{T}}$$

$$= \left[3a_x t^2 + 2b_x t + c_x \quad 3a_y t^2 + 2b_y t + c_y \quad 3a_z t^2 + 2b_z t + c_z \right]^{\mathrm{T}}$$

Parametric Cubic Curves

- $Q(t) = C \bullet T$
- Rewrite the coefficient matrix as $C = G \bullet M$
  - where M is a 4×4 **basis matrix**, G is called the **geometry matrix**
  - so

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

4 endpoints or tangent vectors

Parametric Cubic Curves

- $Q(t) = G \bullet M \bullet T = G \bullet B$

  where $B = M \bullet T$ is called the **blending function**

Why Parametric Cubic Curves?

- Why cubic?
  - lower-degree polynomials give too little flexibility in controlling the shape of the curve
  - higher-degree polynomials can introduce unwanted wiggles and require more computation
  - lowest degree that allows specification of endpoints and their derivatives
  - lowest degree that is not planar in 3D

Continuity between curve segments

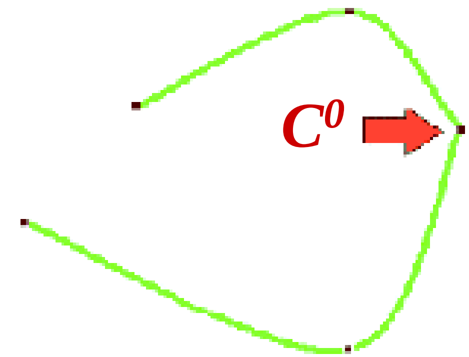
- G^0 geometric continuity
  – two curve segments join together
- G^1 geometric continuity
  – the directions *(but not necessarily the magnitudes)* of the two segments' tangent vectors are equal at a join point

Continuity between curve segments

- C^1 continuous
  - the tangent vectors of the two cubic curve segments are equal *(both directions and magnitudes)* at the segments' join point
- C^n continuous
  - the direction and magnitude of through the nth derivative are equal at the join point

$$d^n / dt^n[Q(t)]$$

Measure of Smoothness

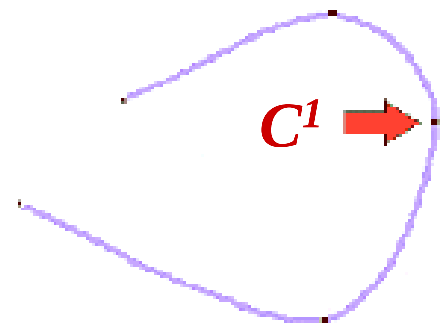


G^0 Geometric Continuity ⇔ C^0 Parametric Continuity

If two curve segments join together.

G^1 Geometric Continuity

If the **directions** (but not necessarily the magnitudes) of the two segments' tangent vectors are equal at a join point.

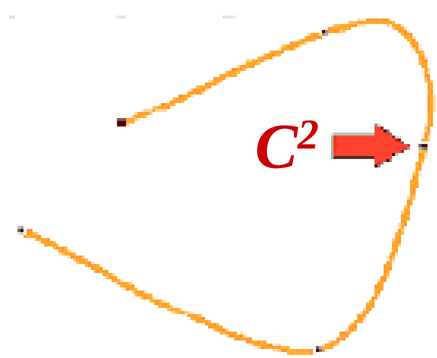


C^1 Parametric Continuity

If the **directions and magnitudes** of the two segments' tangent vectors are equal at a join point.

C^2 Parametric Continuity

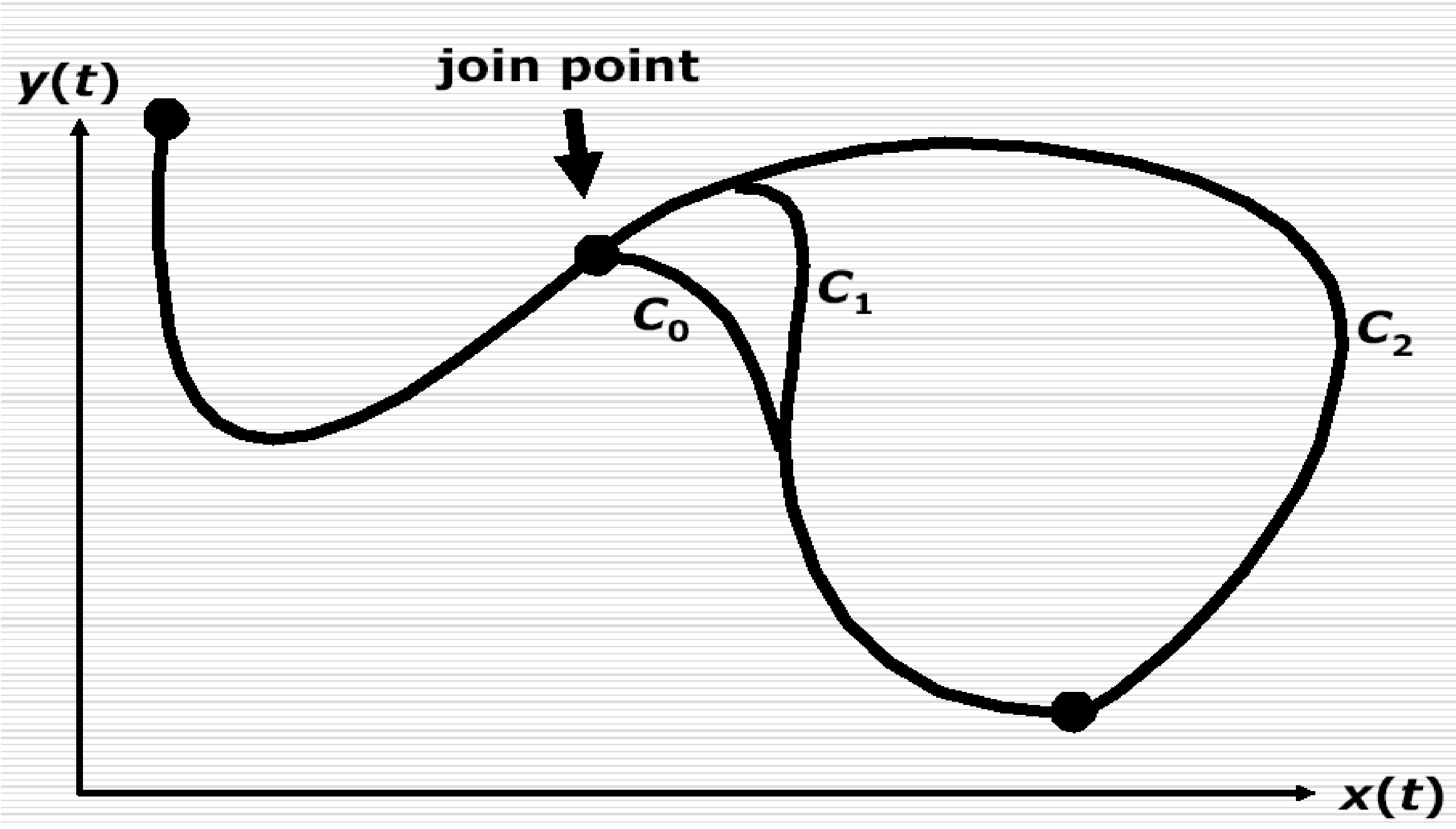
If the direction and magnitude of $Q^2(t)$ (curvature or **acceleration**) are equal at the join point.



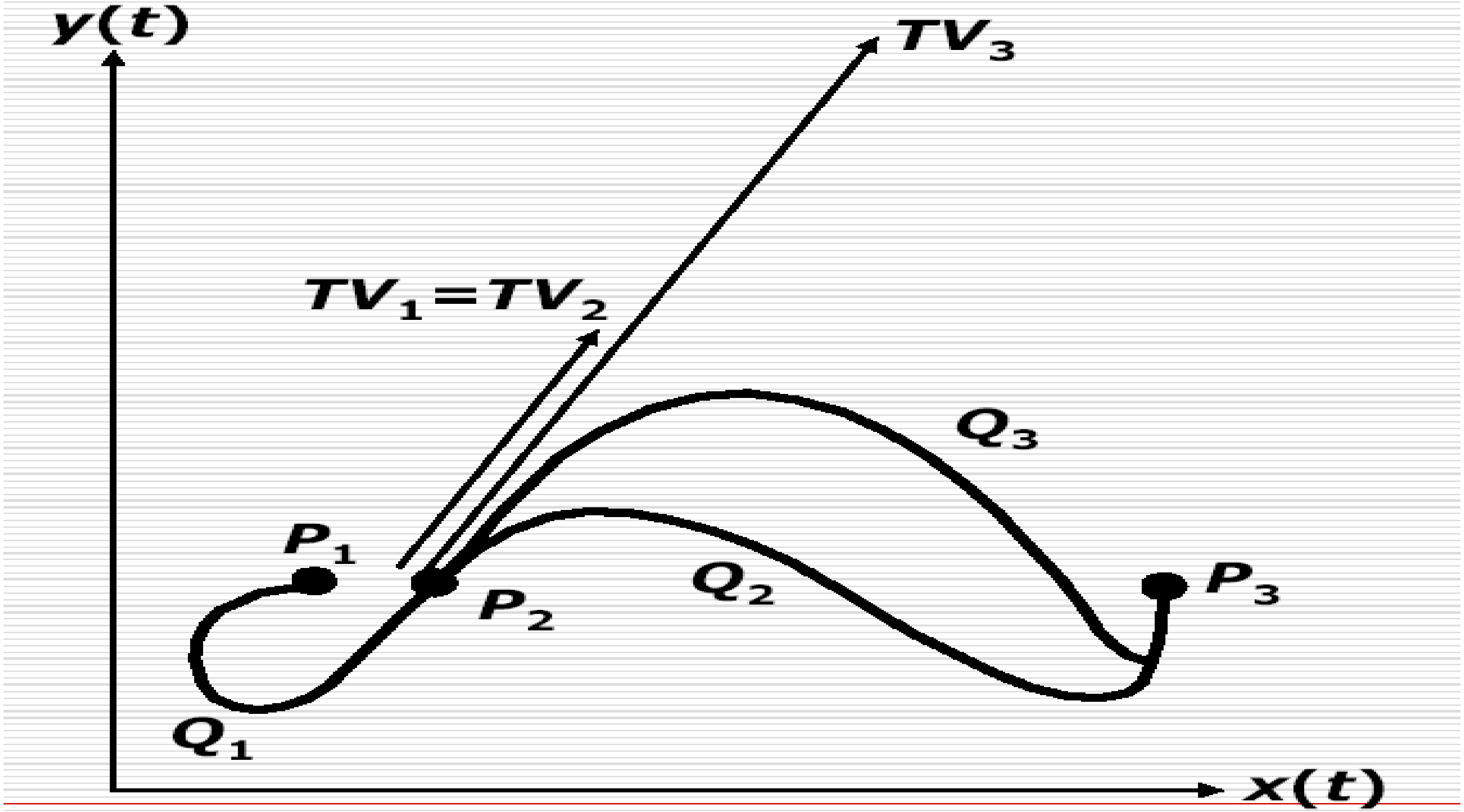
C^n Parametric Continuity

If the direction and magnitude of $Q^n(t)$ through the nth derivative are equal at the join point.

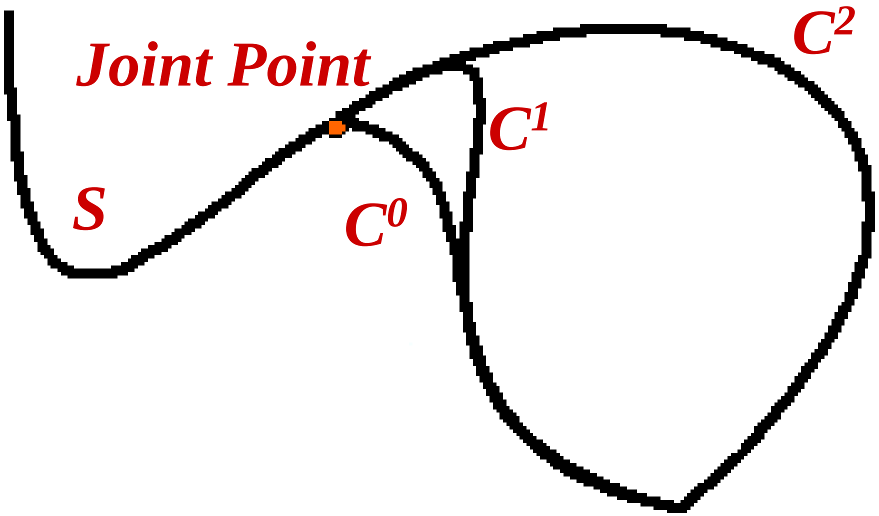
Continuity between curve segments



Continuity between curve segments

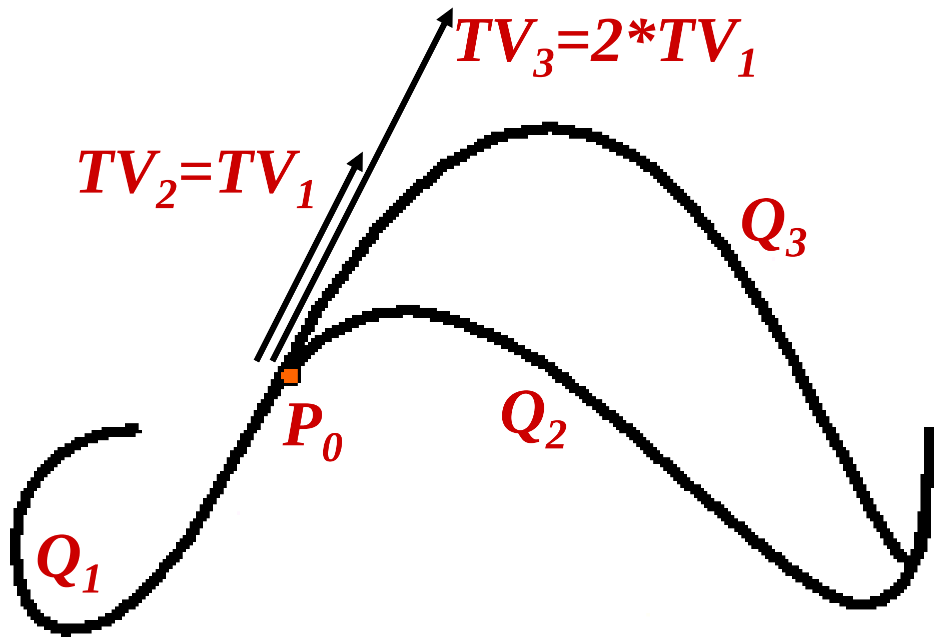


Measure of Smoothness



- By increasing parametric continuity we can increase smoothness of the curve.

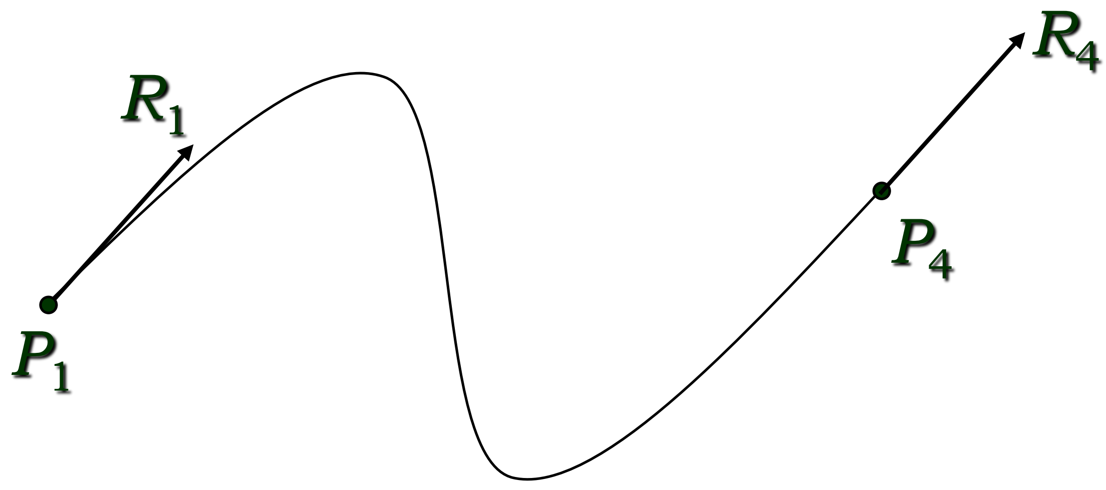
- Q_1& Q_2 are C^1 and G^1 continuous
- Q_1& Q_3 are G^1 continuous only as Tangent vectors have different magnitude.
- Observe the effect of increasing in magnitude of TV



Three Types of Parametric Cubic Curves

- Hermite Curves
  - defined by two **endpoints** and two endpoint **tangent vectors**
- Bézier Curves
  - defined by two **endpoints** and two **control points** which control the endpoint’ **tangent vectors**
- Splines
  - defined by four **control points**

Hermit Curves



A cubic Hermite curve segment interpolating the endpoints P_1 and P_4 is determined by constraints on
the endpoints P_1 and P_4 and

tangent vectors at the endpoints R_1 and R_4

Hermite Curves

- Given the endpoints P_1 and P_4 and tangent vectors at R_1 and R_4
- What are
  - Hermite basis matrix M_H
  - Hermite geometry vector G_H
  - Hermite blending functions B_H
- By definition

$$G_{\mathrm{H}} = \begin{bmatrix} P_1 & P_4 & R_1 & R_4 \end{bmatrix}$$

Hermit Curves (Continue)

The Hermite Geometry Vector: $G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x = T \cdot C_x = T \cdot M_H \cdot G_{H_x}$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} M_H \cdot G_{H_x}$$

The constraints on $x(0)$ and $x(1)$:

$$x(0) = P_{1x} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} M_H \cdot G_{H_x}$$

$$x(1) = P_{4x} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} M_H \cdot G_{H_x}$$

Hermit Curves (Continue)

$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

Hence the tangent-vector constraints:

$$x'(0) = R_{1x} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

$$x'(1) = R_{4x} = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

The 4 constraints can be written as:

$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x = G_{H_x} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} M_H \cdot G_{H_x}$$

Hermit Curves (Continue)

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

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T \cdot M_H \cdot G_H = B_H \cdot G_H$$

$$= \left(2t^3 - 3t^2 + 1\right)P_1 + \left(-2t^3 + 3t^2\right)P_4$$

$$+ \left(t^3 - 2t^2 + t\right)R_1 + \left(t^3 - t^2\right)R_4$$

Bezier Curves

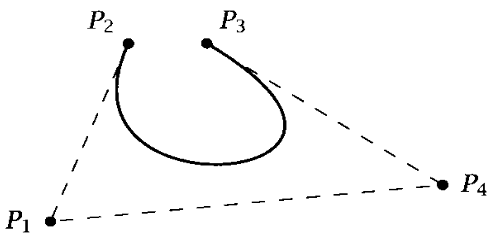


Figure 4.14

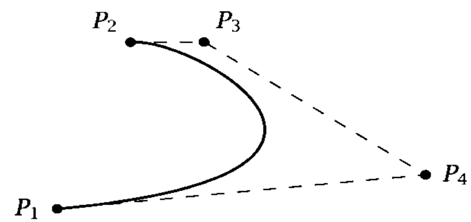


Figure 4.15

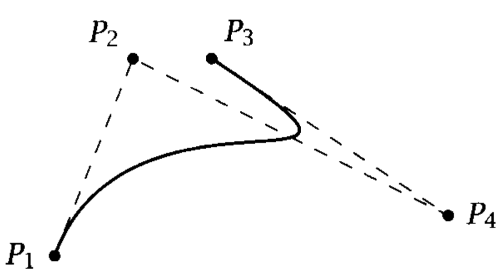


Figure 4.16

P_1, P_2, P_4, P_3

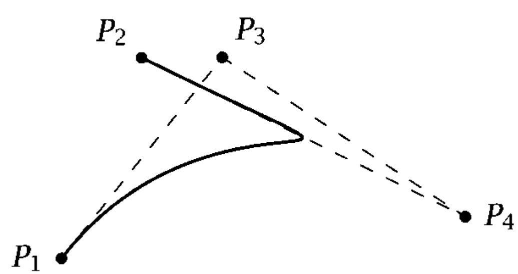


Figure 4.17

P_1, P_3, P_4, P_2

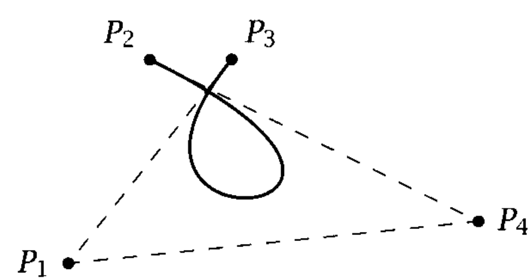
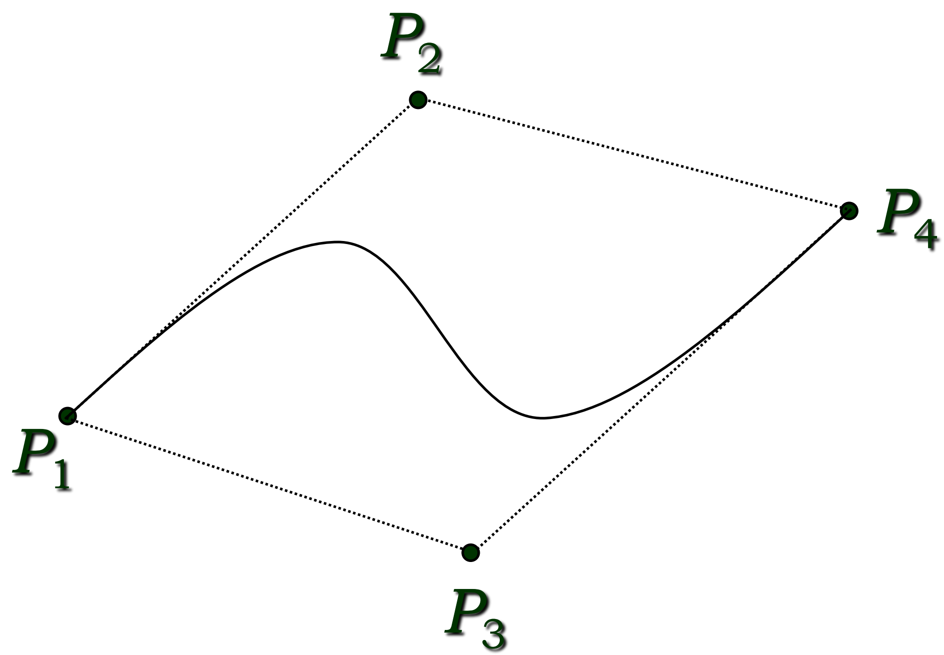


Figure 4.18

P_3, P_1, P_4, P_2

Bézier Curves



Indirectly specifies the endpoint tangent vectors by specifying two intermediate points that are not on the curve.

$$R_1 = Q'(0) = P_1P_2 = 3(P_2 - P_1)$$
$$R_4 = Q'(1) = P_3P_4 = 3(P_4 - P_3)$$

Bézier Curves (Continue)

The Bézier **Geometry Vector:** $G_B = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = M_{HB} \cdot G_B$$

$$\begin{aligned} Q(t) &= T \cdot M_H \cdot G_H = T \cdot M_H \cdot (M_{HB} \cdot G_B) \\ &= T \cdot (M_H \cdot M_{HB}) \cdot G_B = T \cdot M_B \cdot G_B \end{aligned}$$

Bézier Curves (Continue)

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

$$Q(t) = T \cdot M_B \cdot G_B$$

$$= (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$$

The 4 polynomials in $B_B = T \,.\, M_B$ are called the Bernstein polynomials.

B-Spline Curves

- Most shapes are simply too complicated to define using a single Be´zier curve.
- A spline curve is a sequence of curve segments that are connected together to form a single continuous curve.
- For example, a piecewise collection of B´ezier curves, connected end to end, can be called a spline curve.
- The word “spline” can also be used as a verb, as in “Spline together some cubic B´ezier curves.”

B-Spline Curves…

- While C^1 continuity is straightforward to attain using Be´zier curves, C^2 and higher continuity is cumbersome. This is where B-spline curves come in.
- In practical terms, B-spline curves can be thought of as a method for defining a sequence of degree n Be´zier curves that join automatically with C^{n-1} continuity, regardless of where the control points are placed.
- An open string of m Be´zier curves of degree n involve ($nm + 1$) distinct control points (shared control points counted only once), that same string of Be´zier curves can be expressed using only ($m + n$) B-spline control points (assuming all neighboring curves are C^{n-1}).

Polar Form

- All of the important algorithms for Be´zier and B-spline curves can be derived from the following rules for polar values.
- For degree *n* Be´zier curves over the parameter interval [a, b], the control points are relabeled

$$P_i = P(u_i, u_2, \ldots, u_n) \text{ where } u_j = a, (j \le n - i) otherwise$$

$$u_j = b$$

- For a degree two curve over the interval [a, b],

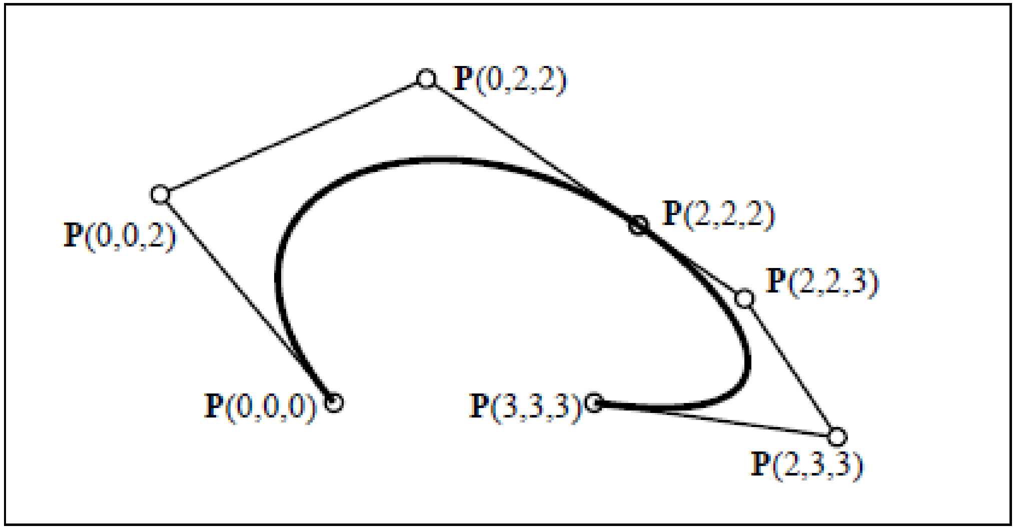
P0 = P(a, a); P1 = P(a, b); P2 = P(b, b).

- For a degree three Be´zier curve,

P0 = P(a, a, a); P1 = P(a, a, b);

P2 = P(a, b, b); P3 = P(b, b, b);

Be´zier curves in Polar Form



• Figure shows two cubic Be´zier curves labeled using polar values. The first curve is defined over the parameter interval [0, 2] and the second curve is defined over the parameter interval [2, 3].

B-Spline curves in Polar Form

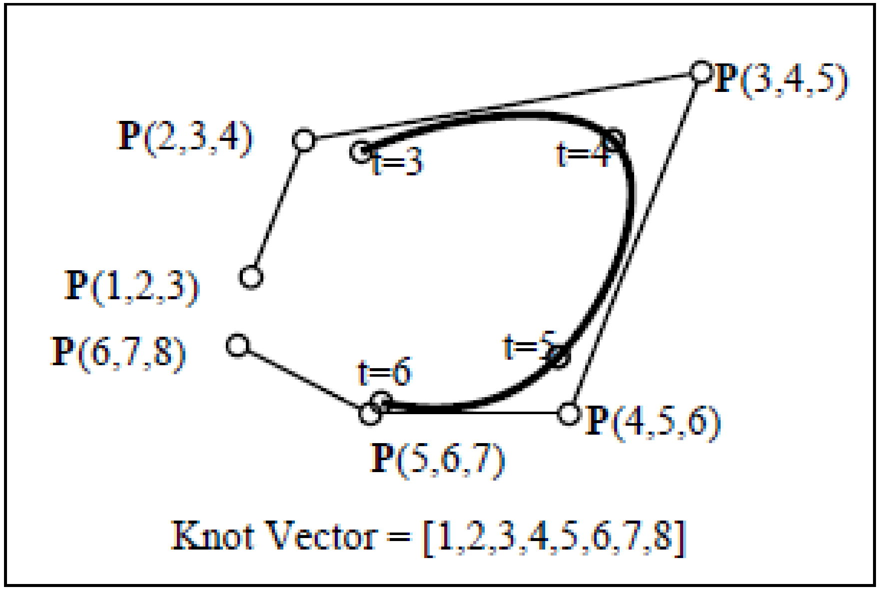
- For a degree *n* B-spline with a knot vector (explained later) of $[t_1, t_2, t_3, t_4, \ldots]$, the arguments of the polar values consist of groups of *n* adjacent knots from the knot vector, with the i^{th} polar value being $P(t_i, \ldots., t_{i+n-1})$, as shown in the next slide.
- A polar value is symmetric in its arguments. This means that the order of the arguments can be changed without changing the polar value. For example,

P(1, 0, 0, 2) = P(0, 1, 0, 2) = P(0, 0, 1, 2) = P(2, 1, 0, 0), etc.

knot vector

- A knot vector is a list of parameter values, or knots, that specify the parameter intervals for the individual Be´zier curves that make up a B-spline.
- For example, if a cubic B-spline is comprised of four Be´zier curves with parameter intervals [1, 2], [2, 4], [4, 5], and [5, 8], the knot vector would be [t0, t1, 1, 2, 4, 5, 8, t7, t8].
- Notice that there are two (one less than the degree) extra knots prepended and appended to the knot vector. These knots control the end conditions of the B-spline curve.

B-spline curve labeled using polar form



References

- J. D. Foley, A. V. Dan, S. K. Feiner and J. F. Hughes, Computer Graphics, Principles & Practice, Second Edition.
- T. W. Sederberg, An Introduction to B-Spline Curves, March 14, 2005.