B-Spline Polynomials

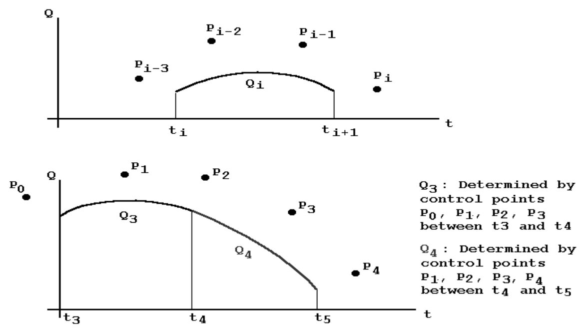
B-Spline Polynomials

- Draftman's spline
  - Flexible metal strip used to lay out object surfaces
  - If not stressed too much, get level-2 (2nd derivative) continuity curve
- Want local control
- Smoother curves
- B-spline curves:
  - Segmented approximating curve
  - 4 control points affect each segment
    - Local control
  - Level-2 continuity everywhere
    - Very smooth

Cubic B-Spline Polynomial Curves

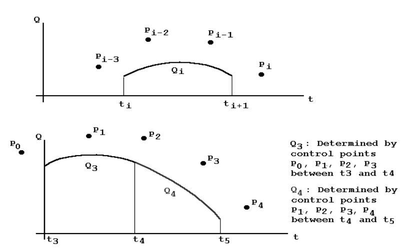
- Approximate m+1 control points Pi (i=0,1,2,…,m)

  with a curve consisting of m-2 cubic polynomial curve <u>segments</u> Q_i (i=3,4,...m), m>=3
- Each Q_i defined in terms of:
  - parameter t: t_i<=t<=t_{i+1} and by four of the m+1 control points

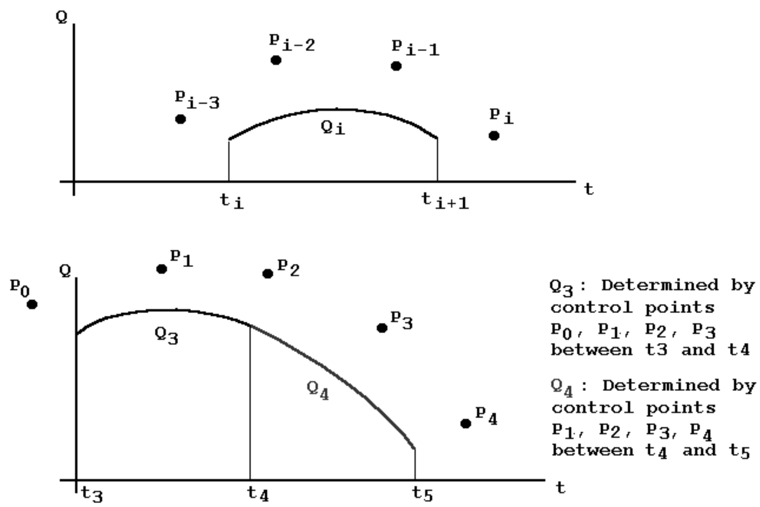


- Segment Q_i determined by control points:

  P_{i-3}, P_{i-2}, P_{i-1}, P_i between t_i and t_{i+1}
- Q_i begins at t = t_i and ends at t = $t_{I+1,}$
- Q_{i+1} joins Q_i at t_{i+1}
  - Join point called a knot.
- For example
  - First segment is Q_3, begins at t_3, ends at t_4
  - Is determined by control points P0,P1,P2,P3
- Each segment is affected by only 4 control points
- Each control point affects at most 4 curve segments



Uniform Cubic B-Spline Curves



Uniform Cubic B-Splines

- A special case where we assume that:
- $t_{i+1} = t_i + 1$
- Polynomial equation for segment Qi:
- Qi(t) = a*(t-ti)^3 + b*(t-ti)^2 + c*(t-ti) + d, ti<=t<=ti+1
- Take independent variable as t-ti
  - Will vary from 0 to 1 for any interval

- Need to get polynomial coefficients (a,b,c,d)
  - from control points
- Find a "B-Spline Basis Matrix"
  - as for Bezier curves
  - but must do computation for each interval

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = M_{BS} * \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_{i} \end{bmatrix}$$

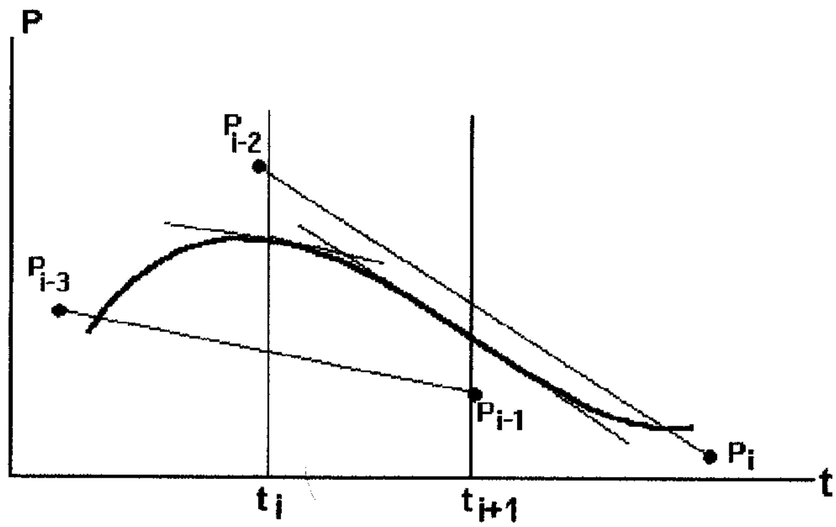
- M_{BS} is the desired matrix

B-Spline Continuity Conditions

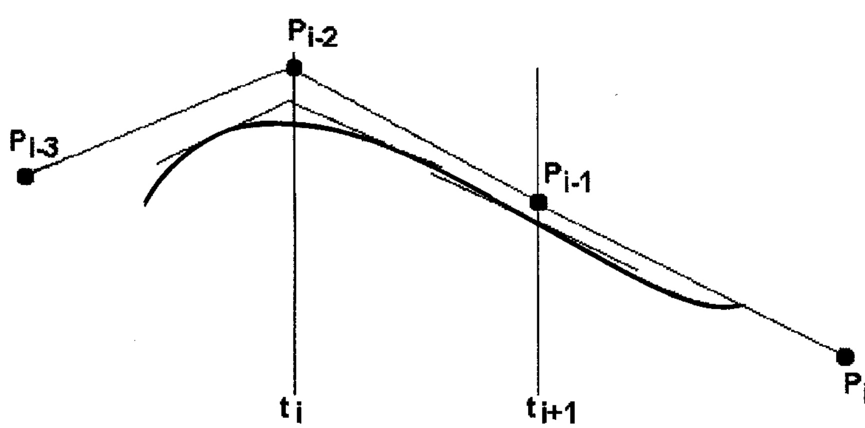
- Conditions on 1st & 2nd derivatives:

1. dQ_i/dt (at $t=t_i$) = slope of line segment joining P_{i-3} and P_{i-1}
2. dQ_i/dt (at $t=t_{i+1}$) = slope of line segment joining P_{i-2} and P_i
3. $(dQ_i/dt)'$ ($t=t_i$) = rate of change in slope at t_i :
   (slope of $[P_{i-3}-P_{i-2}]$ - slope of $[P_{i-2}-P_i]$) / Δt
4. $(dQi/dt)'$ ($t=t_{i+1}$) = rate of change in slope at t_{i+1} :
   (slope of $[P_{i-2}-P_{i-1}]$ - slope of $[P_{i-1}-P_i]$) / Δt

Continuity Conditions 1 and 2



Continuity Conditions 3 and 4



$Qi = a*(t-ti)^3 + b*(t-ti)^2 + c*(t-ti) + d$

$dQi/dt = 3*a*(t-ti)^2 + 2*b*(t-ti) + c$

$(dQi/dt)' = 6a*(t-ti) + 2*b$

- Condition 1: $c = (P_{i-1} - P_{i-3})/2$
- Condition 2: $3*a + 2*b + c = (P_i - P_{i-2})/2$
- Condition 3: $2*b = ((P_{i-1} - P_{i-2}) - (P_{i-2} - P_{i-3})) / 1$
- Condition 4: $6*a + 2*b = ((P_i-P_{i-1})-(P_{i-1}-P_{i-2})) / 1$
- These four equations are not independent
  – Solving gives only a, b, c, but not d

Solution

$a = (1/6) * (-P_{i-3} + 3*P_{i-2} - 3*P_{i-1} + P_i)$

$b = (1/6) * (3*P_{i-3} - 6*P_{i-2} + 3*P_{i-1})$

$c = (1/6) * (-3*P_{i-3} + 3*P_{i-1})$

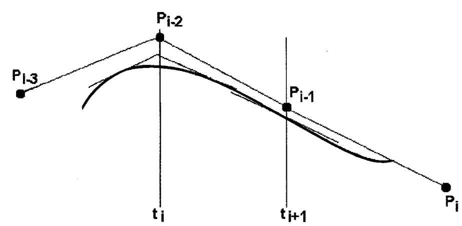
Need another condition to get d

- Choose the following condition:

  Q (at t=ti) = (1/6) * (P_{i-3} + 4*P_{i-2} + P_{i-1})

  – i.e., control point at ti (P_{i-2}) pulls 4 times as hard at t=ti as control points on either side of ti
- Substitute in polynomial equation -->

  d = (1/6) * (P_{i-3} + 4*P_{i-2} + P_{i-1})



Uniform Cubic B-Spline Coefficient Matrix Equation

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = (1/6) * \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} * \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}$$

Could also be written in terms of blending functions

$$Q_i(t) = \sum_{j=0}^{3} B_{i-j,4}(t) * P_{i-j}$$

$B_{i-3,4} = 1/6 * (1-t)^3$

$B_{i-2,4} = 1/6 * (3t^3 - 6t^2 + 4)$

$B_{i-1,4} = 1/6 * (-3t^3 + 3t^2 - 3t +1)$

$B_{i,4} = 1/6 * t^3$

See Foley & Van Dam

Plotting Uniform Cubic B-Splines

- Given m+1 control points P0,P1,P2,...Pm
  - (Recall that each has an x and y coordinate)
    - i.e., P0 --> x0 and y0, etc.
- The following is a "brute force" algorithm to plot the curve
  - delta is a very small increment (e.g., 0.05)

Brute Force Algorithm

```
For (i=3 to m)   // each segment
      Compute ax,bx,cx,dx and ay,by,cy,dy
              from control points i-3, i-2, i-1, i
      For (t=0; t<=1; t+=delta)  // plot ith segment
         x = ax*t^3 + bx*t^2 + cx*t + dx
         y = ay*t^3 + by*t^2 + cy*t + dy
        If (t==0)
          MoveTo(x,y)
        Else
          LineTo(x,y)
```

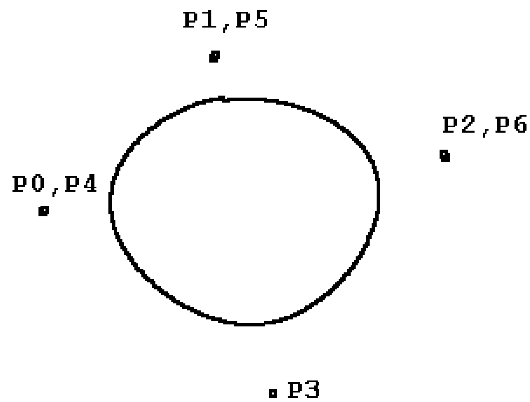
- To increase performance, use forward differences

Closed Cubic B-Splines

Make last 3 control points coincide with 1st 3

0 <--> m-2, 1 <--> m-1, 2 <--> m

Example: m=6



Forcing Interpolation

- Reproduce a control point three times
- Curve will then go through that point

Properties of Uniform B-Splines

1. Local Control
   – Each segment determined by only 4 control points
2. Approximates control points; doesn't interpolate
   (However it will interpolate triplicated control points)
3. Lies inside convex hull of control points
   – Each segment lies inside convex hull of its 4 control points
4. Invariant under affine transformations
5. Very smooth
   – Level-2 continuity everywhere
6. More computations required than for "equivalent" Bezier curve

Bezier vs. B-Spline Curves