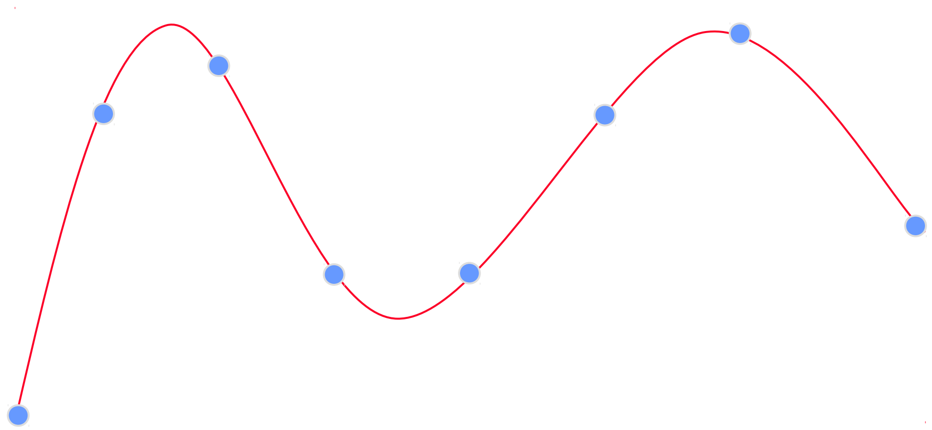
Krivulje

Uvod

- **“Gladke” krivulje so uporabne na mnogih področjih:**
  - **Modeliranje realnih objektov**
  - **Računalniško podprto načrtovanje (CAD)**
  - **Fonti**
  - **Predstavitve podatkov, grafi**
  - **Risbe, skice**

Problem #1

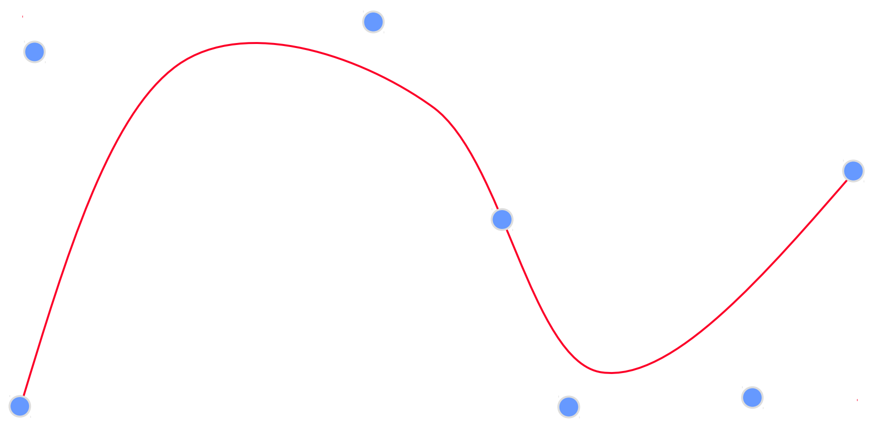
Tvoriti hočemo krivuljo (ploskev), ki poteka skozi množico točk



Problem #2

Radi bi predstavili krivuljo (ploskev) za modeliranje objektov

- **kompaktna oblika**
- **preprosto rokovanje**
- **ni nujno, da poteka skozi točke**



Tipi algebraičnih krivulj

- ***Interpolacije***
  - **krivulja poteka skozi točke**
  - **primerno za znanstveno vizualizacijo in analizo podatkov**
- ***Aproksimacije***
  - **Točke nadzirajo obliko krivulje**
  - **primerno za geometrično modeliranje**

Predstavitve krivulj

- eksplicitne
  eno spremenljivko izrazimo z drugo.
  2D: $y = \cos(x)$
  3D: $y = \cos(x)$ in $z = \sin(x)$

- Implicitne
  funkcija vseh spremenljivk je enaka nič.
  2D: $F(x,y) = x^2+y^3+6 = 0$
  3D: $F(x,y,z) = x^4+y^3 +16 = 0$ and $y^5+z+10 = 0$

- Parametrične
  x,y in z so funkcije parametra, ki se spreminja vzdolž nekega intervala.
  2D: $P(t) = (x(t), y(t))$
  3D: $Q(t) = (x(t),y(t),z(t))$

Eksplicitne predstavitve

Express one variable in terms of the other.

2D: $y = \cos(x)$

3D: $y = \cos(x)$ and $z = \sin(x)$

- Only works for single valued functions
- Easy to determine if the point is on the curve
- Coordinate system dependent

Implicitne predstavitve

Function of all variables equals zero.

2D: $x^2+y^3+6 = 0$

3D: $x^4+y^3 +16 = 0$ and $y^5+z+10 = 0$

- Works for multi-valued functions
- Easy to determine if the point is on the curve
- Coordinate system dependent
- Partial curves require additional constraints
- Easy to determine inside-outside of curve:
  - If $F(x,y,z) < 0$, then (x,y,z) is on the inside of the curve
  - If $F(x,y,z) > 0$, then (x,y,z) is on the outside of the curve

  This feature is useful in solid modeling applications.

Parametrične predstavitve

x,y, and z are all functions of a parameter that varies over an interval.

2D: $P(t) = (x(t), y(t))$

3D: $Q(t) = (x(t),y(t),z(t))$

- Works for multi-valued functions and functions that cross over themselves
- Coordinate system independent - easy to draw
- More points can be placed in areas of high curvature
- Easy to draw partial curves
- Not easy to test if a point lies on the curve

This representation is useful in geometric modeling

Parametrične predstavitve

- Premica

  $x(t) = A_x+(B_x-A_x)t$ in $y(t) = A_y+(B_y-A_y)t$

- Krog

  $x(t) = \cos(t)$ in $y(t) = \sin(t)$

- Elipsa

  $x(t) = W \cos(t)$ in $y(t) = H \sin(t)$

- Hiperbola

  $x(t) = a \sec(t)$ in $y(t) = b \tan(t)$

- Parabola

  $x(t) = at^2$ in $y(t) = 2at$

Parametrične polinomske krivulje

- A parametric polynomial curve is described:

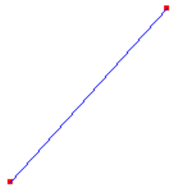
$$x(u) = \sum_{i=0}^{n} a_i u^i$$

$$y(u) = \sum_{i=0}^{n} b_i u^i$$

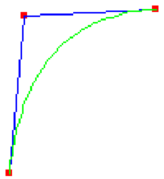
- Advantages of polynomial curves
  - Easy to compute
  - Infinitely differentiable

Stopnja krivulje

- Stopnja določa, kakšna je lahko oblika krivulje
  - Stopnja 1 (linerne) - črta

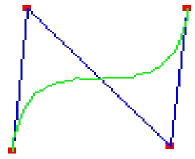


  - Stopnja 2 (kvadrične) – lahko en zavoj



  - Stopnja 3 (kubične) – lahko dva zavoja

- Število kontrolnih točk je vedno stopnja + 1.



Parametrične kubične krivulje

Običajno uporabljamo kubične parametrične polinome

- **enostavna uporaba**
- **dobre lastnosti zveznosti**

$$p(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3$$
$$= \sum_{i=0}^{3} c_i u^i$$

Parametrične kubične krivulje

- **Splošna oblika:**

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

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

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

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

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

Parametrične kubične krivulje

Zakaj kubične krivulje?

- Polinomi nizkih redov omogočajo malo fleksibilnosti pri oblikovanju krivulje
- Polinomi višjih redov lahko povzročijo nezaželene napake v krivulji in so računsko zahtevnejši
- So najnižji red formul, ki omogočajo specifikacijo končnih točk in njihovih odvodov
- Najnižji red, ki ni planaren v 3D

Mešalne funkcije (Blending Functions)

- Računanje statičnih koeficientov je v redu, radi pa bi bolj splošen pristop
- Problem iskanja preprostih polinomov, ki bi jih lahko uporabili kot koeficiente

$$p(u) = \sum_{i=0}^{3} b_i(u)\, p_i$$

Mešalne funkcije (nadaljevanje)

Primer mešalnih funkcij za interpolacijo krivulje

$$p(u) = \sum_{i=0}^{3} b_i(u)\, p_i$$

$$b_0(u) = 1 - 5.5u + 9u^2 - 4.5u^3$$

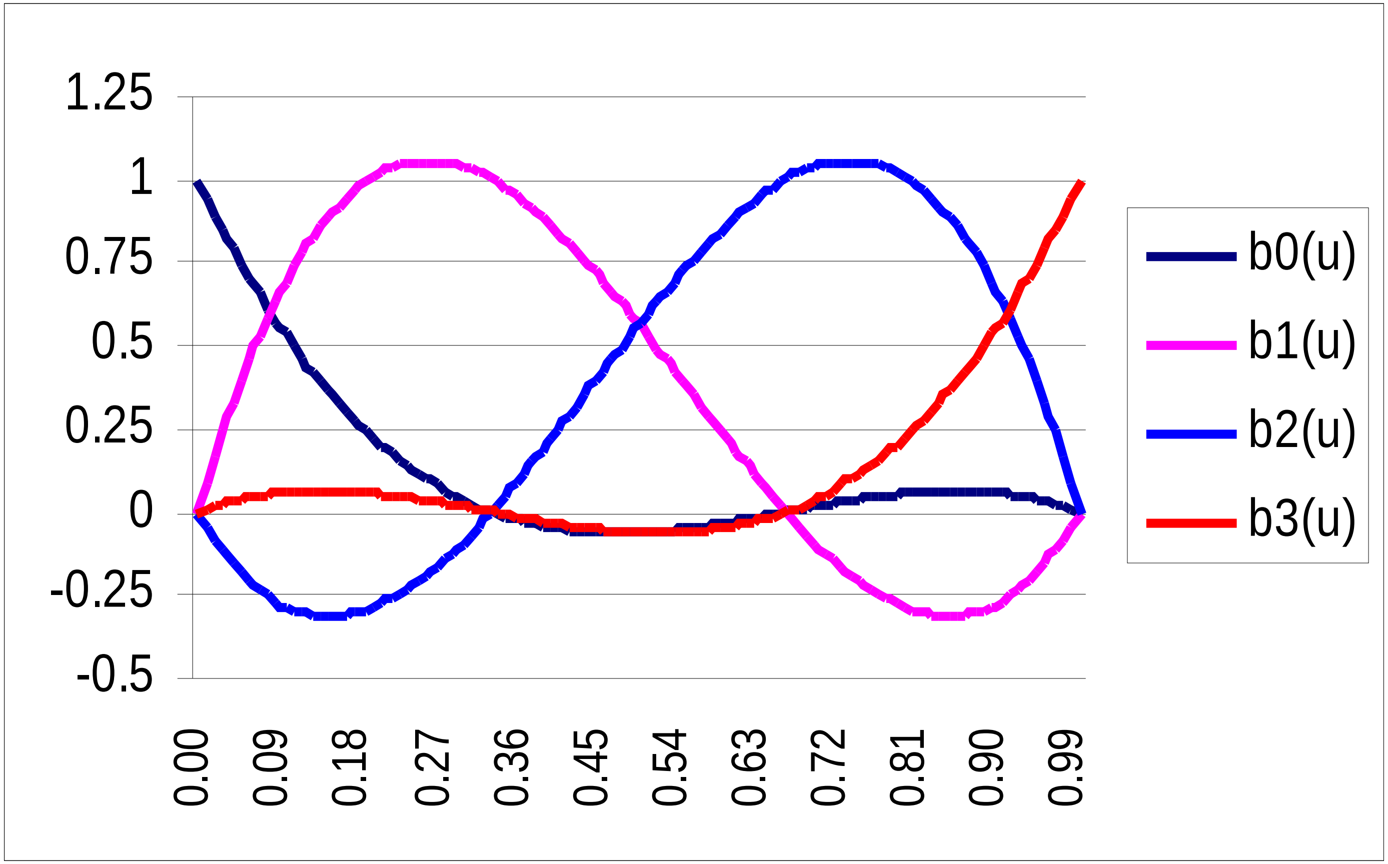
$$b_1(u) = 9u - 22.5u^2 + 13.5u^3$$

$$b_2(u) = -4.5u + 18u^2 - 13.5u^3$$

$$b_3(u) = 1u - 4.5u^2 + 4.5u^3$$

Mešalne funkcije (nadaljevanje)

Mešalne funkcije za primer interpolacije

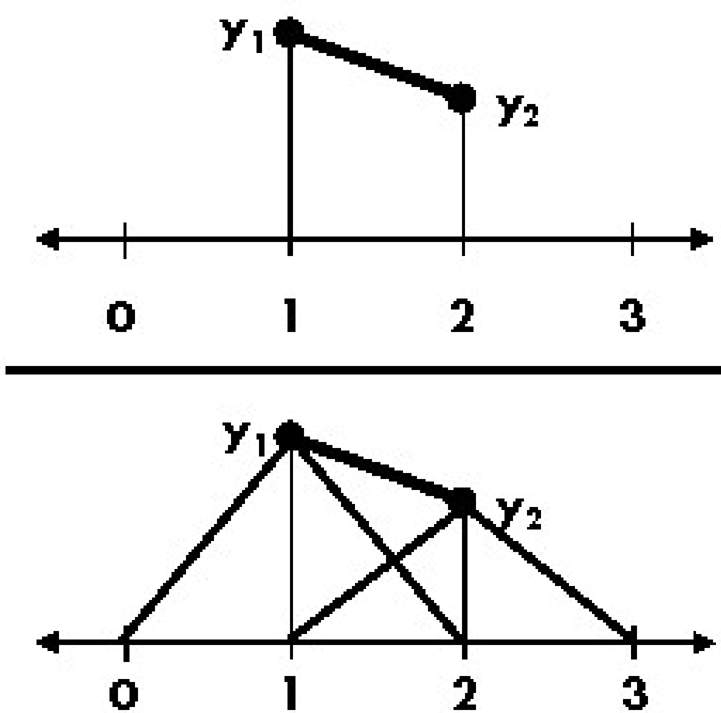


Aproksimacijske krivulje

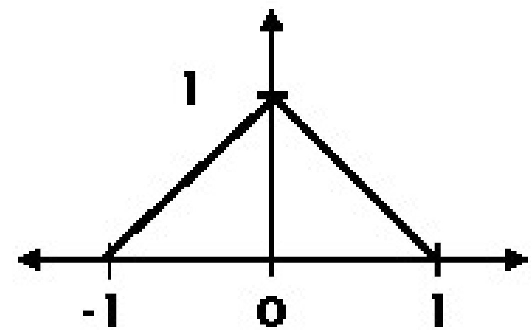
- **Control the shape of the curve by positioning *control points***
- **Multiples types available**
  - **Bezier**
  - **B-Splines**
  - **NURBS (Non-Uniform Rational B-Splines)**
- **Also available for surfaces**

Linearna interpolacija

Linearna interpolacija, trikotniška osnova

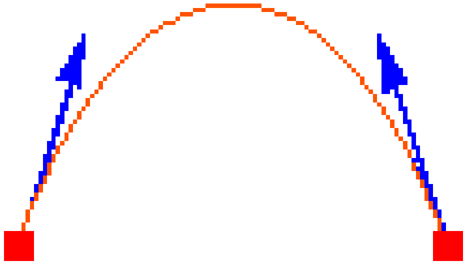


$$P(t) = y_1 T_1(t) + y_2 T_2(t)$$



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

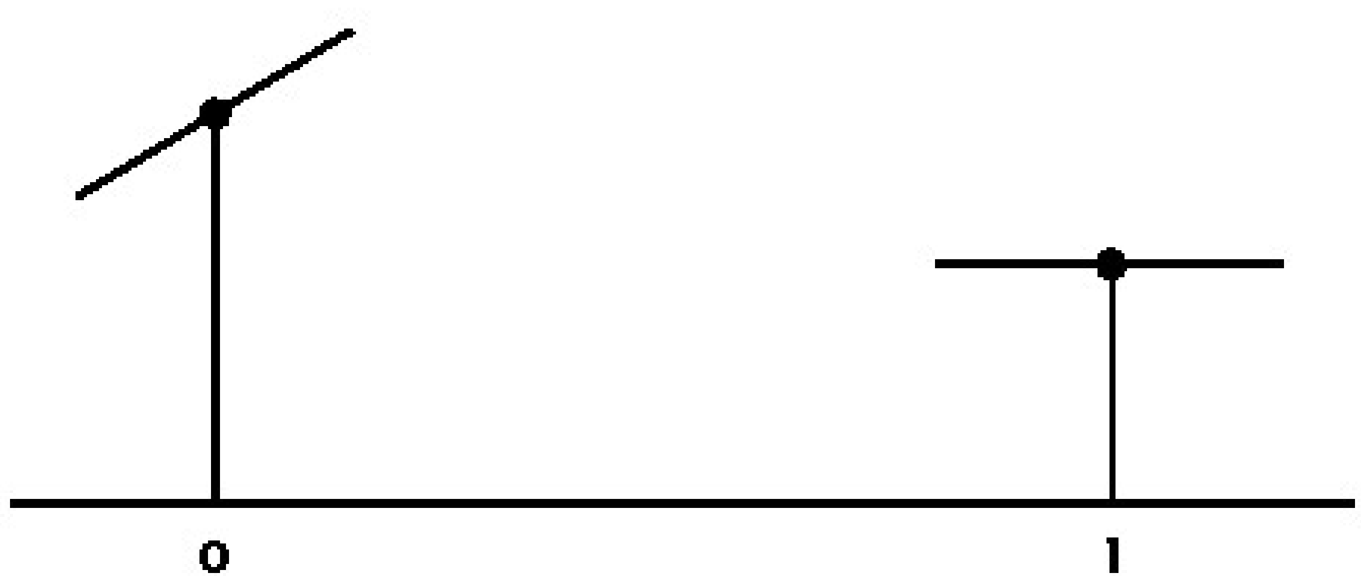
Kubične krivulje



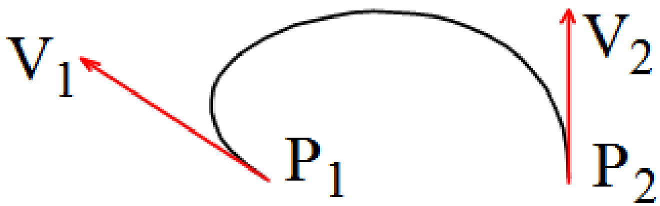
Kubične krivulje imajo štiri koeficiente.

Če rišemo kubično krivuljo med dvema končnima točkama, nam ostaneta še dve točki, ki jih lahko poljubno spreminjamo. Z določanjem položaja in odvoda krivulje (določenega s puščicami) na teh dveh končnih točkah lahko povsem opišemo kubično krivuljo, ki zadošča tem pogojem.

Kubična Hermitova interpolacija



Kubična Hermitova interpolacija



$$\text{Given: } P(0), P(1), P'(0), P'(1)$$

$$\text{Compute: } P(t) = at^3 + bt^2 + ct + d$$ **(Cubic Polynomial)**

$$\text{Derivative: } P'(t) = 3at^2 + 2bt + c$$

$$P(0) = h_0 = d$$
$$P(1) = h_1 = a + b + c + d$$
$$P'(0) = h_2 = c$$
$$P'(1) = h_3 = 3a + 2b + c$$

Solve ⇒

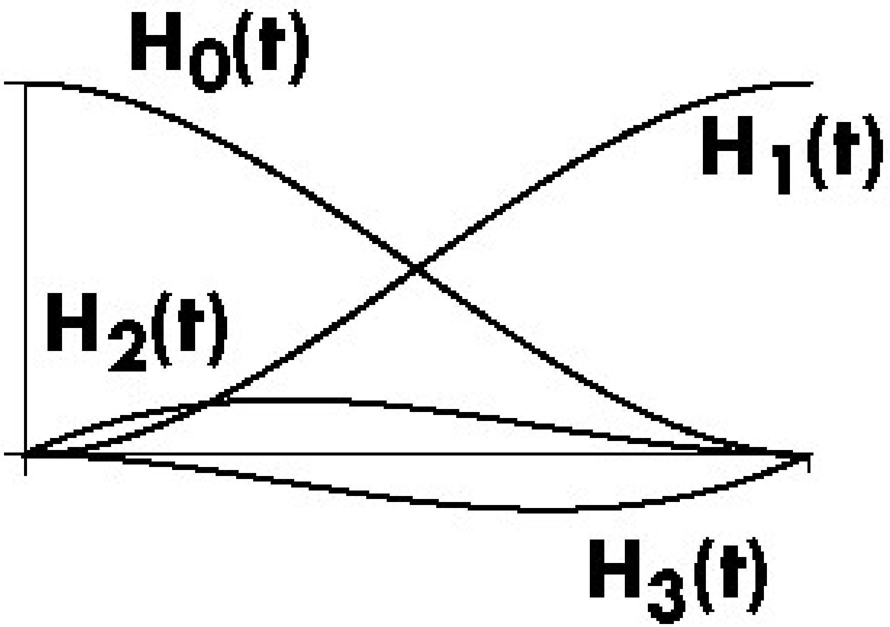
$$a = 2h_0 - 2h_1 + h_2 + h_3$$
$$b = -3h_0 + 3h_1 - 2h_2 - h_3$$
$$c = h_2$$
$$d = h_0$$

Hermitova bazna matrika

$$P(t)=\begin{bmatrix}a & b & c & d\end{bmatrix}\begin{bmatrix}t^3\\ t^2\\ t\\ 1\end{bmatrix}=\underbrace{\begin{bmatrix}a & b & c & d\end{bmatrix}\begin{bmatrix}0 & 1 & 0 & 3\\ 0 & 1 & 0 & 2\\ 0 & 1 & 1 & 1\\ 1 & 1 & 0 & 0\end{bmatrix}}_{\begin{bmatrix}h_0 & h_1 & h_2 & h_3\end{bmatrix}}\underbrace{\begin{bmatrix}2 & -3 & 0 & 1\\ -2 & 3 & 0 & 0\\ 1 & -2 & 1 & 0\\ 1 & -1 & 0 & 0\end{bmatrix}\begin{bmatrix}t^3\\ t^2\\ t\\ 1\end{bmatrix}}_{\begin{bmatrix}H_0^3(t)\\ H_1^3(t)\\ H_2^3(t)\\ H_3^3(t)\end{bmatrix}}$$

$$=\begin{bmatrix}h_0 & h_1 & h_2 & h_3\end{bmatrix}\begin{bmatrix}H_0^3(t)\\ H_1^3(t)\\ H_2^3(t)\\ H_3^3(t)\end{bmatrix}$$

Hermitove bazne funkcije



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

$$H_1(t) = -2t^3 + 3t^2$$

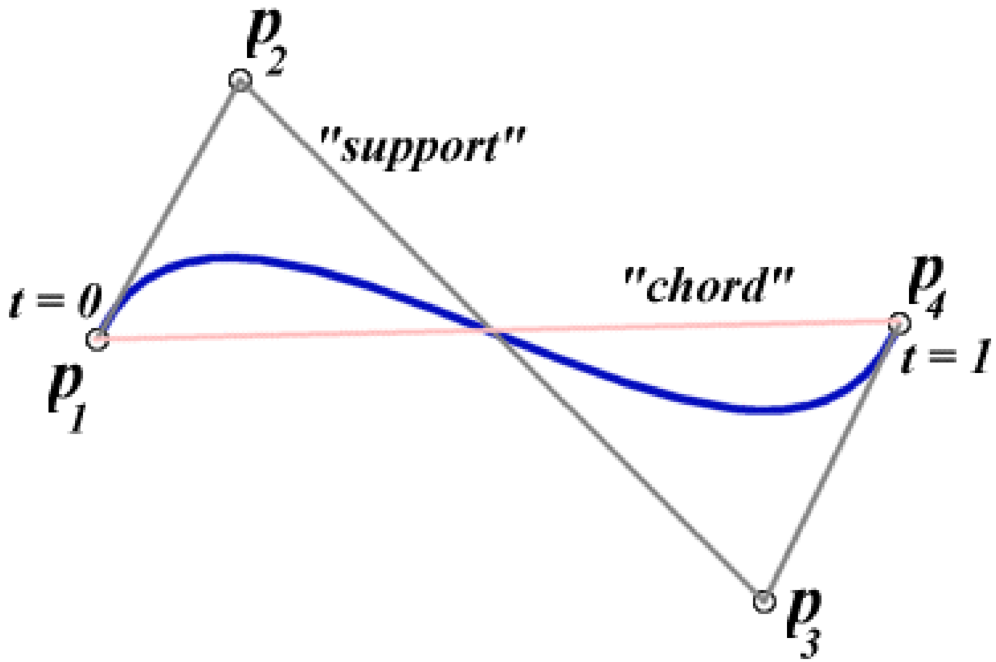
$$H_2(t) = t^3 - 2t^2 + t$$

$$H_3(t) = t^3 - t^2$$

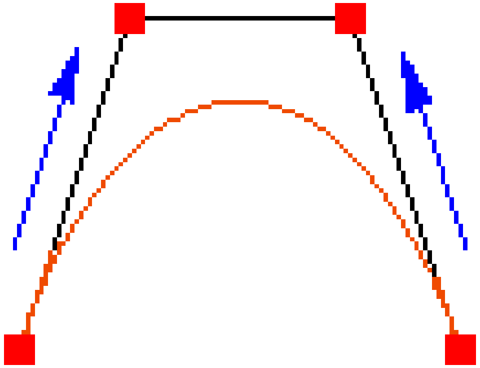
$$\mathbf{M}_H = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Bezierove krivulje

- **Similar to Hermite, but more intuitive definition of endpoint derivatives**
- **Four control points, two of which are knots**



Bezierove krivulje



Po Bezieru je dodatna informacija o krivulji skrita v ostalih dveh točkah. Te štiri točke vsebujejo kontrolni poligon za Bezierovo krivuljo. Seveda pa so odvodi v končnih točkah funkcije teh točk.

Demo

Bezierove krivulje

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

- **The scalar 3 was selected just for this curve**

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

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

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?**

Bezierove mešalne funkcije

- Look at the blending functions
- This family of polynomials is called order-3 Bernstein Polynomials

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

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

Primer kode za 2D prostor

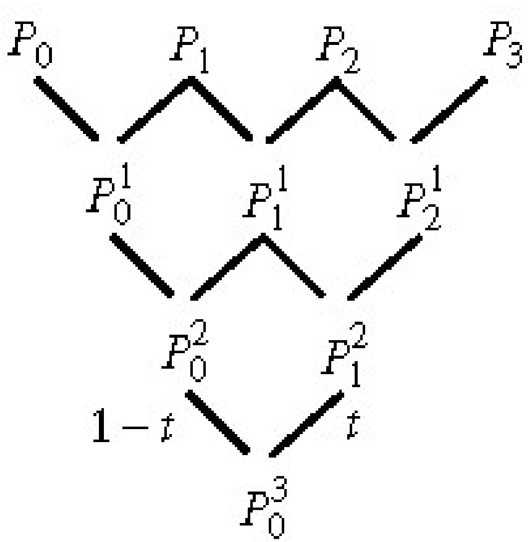
```
t = 0.0;
  delta = 1.0/CURVEMAX;

  for (i=0; i< CURVEMAX+1; i++) {
    vvv = (1.0-t)*(1.0-t)*(1.0-t);
    vvu = 3*(1.0-t)*(1.0-t)*t;
    vuu = 3*(1.0-t)*t*t;
    uuu = t*t*t;

    allCurve[i].x = P[0].x*vvv +
                    P[1].x*vvu +
                    P[2].x*vuu +
                    P[3].x*uuu;

    allCurve[i].y = P[0].y*vvv +
                    P[1].y*vvu +
                    P[2].y*vuu +
                    P[3].y*uuu;
    t += delta;
  }
```

Koeficienti so Bernsteinovi polinomi in se jih da izračunati iz binomske vrste (t + (1-t)) na potenco n.

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

Bernsteinovi polinomi

- **Bezier blending functions are a special set of called the *Bernstein Polynomials***
  - **basis of OpenGL's curve evaluators**



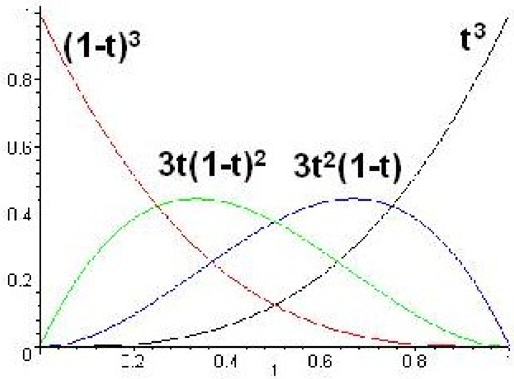
$$P(t) = \sum_{i=0}^{n} P_i B_i^n(t)$$

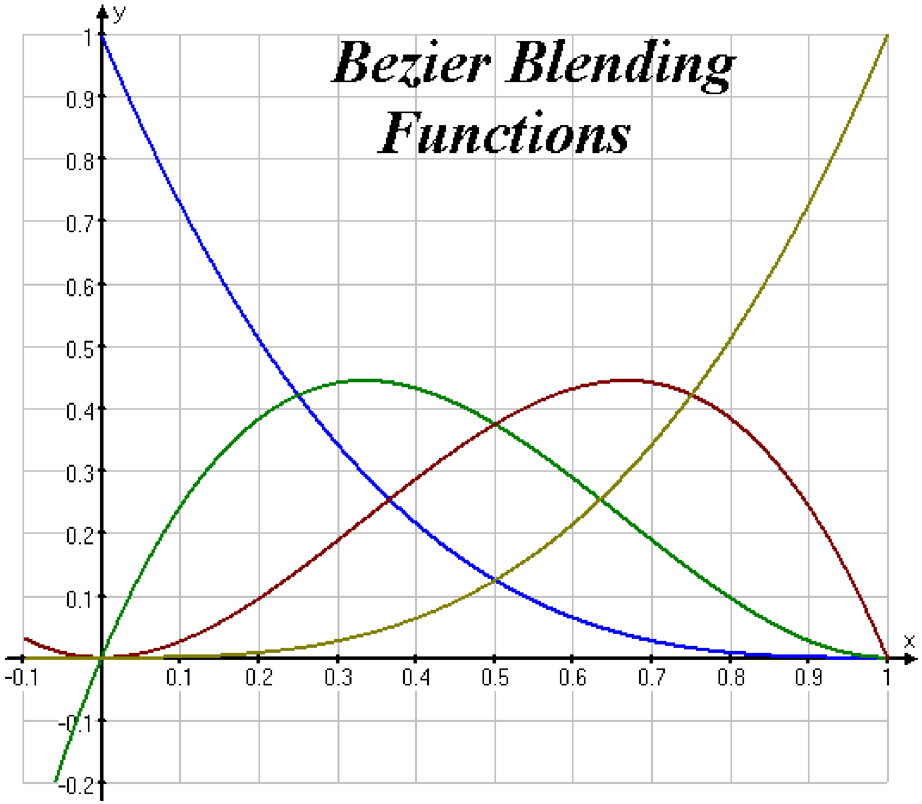
$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Bezierove mešalne funkcije

- 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





Bezierove krivulje

- **Will always remain within bounding region defined by control points**

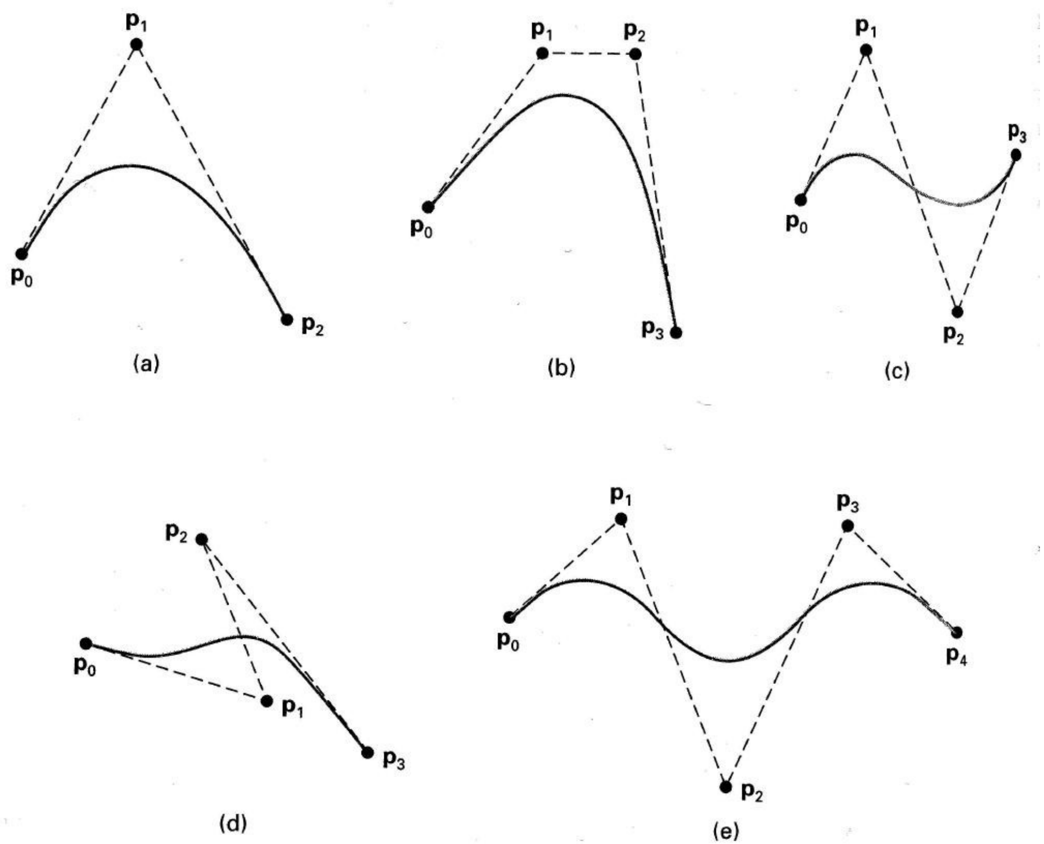
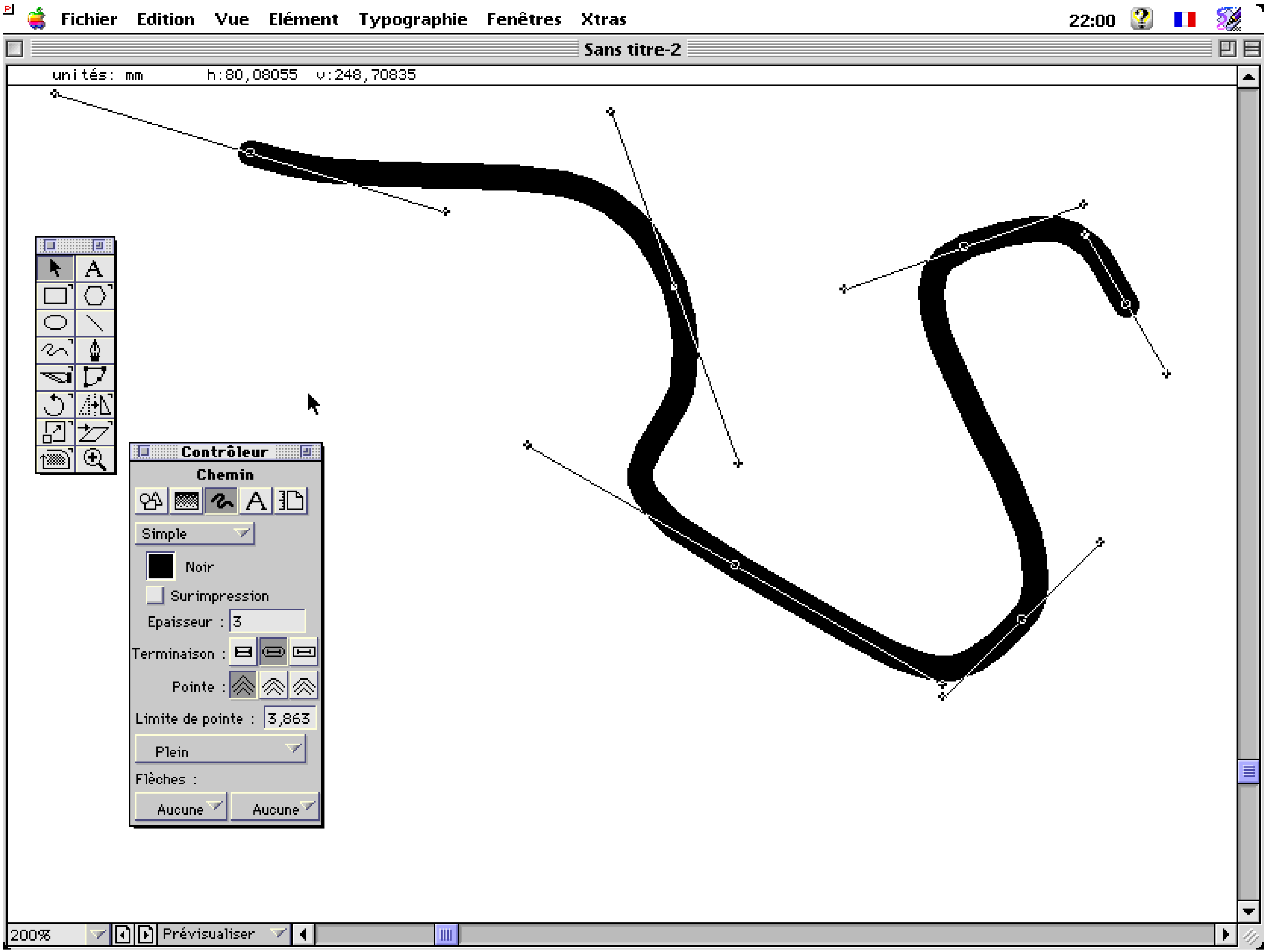


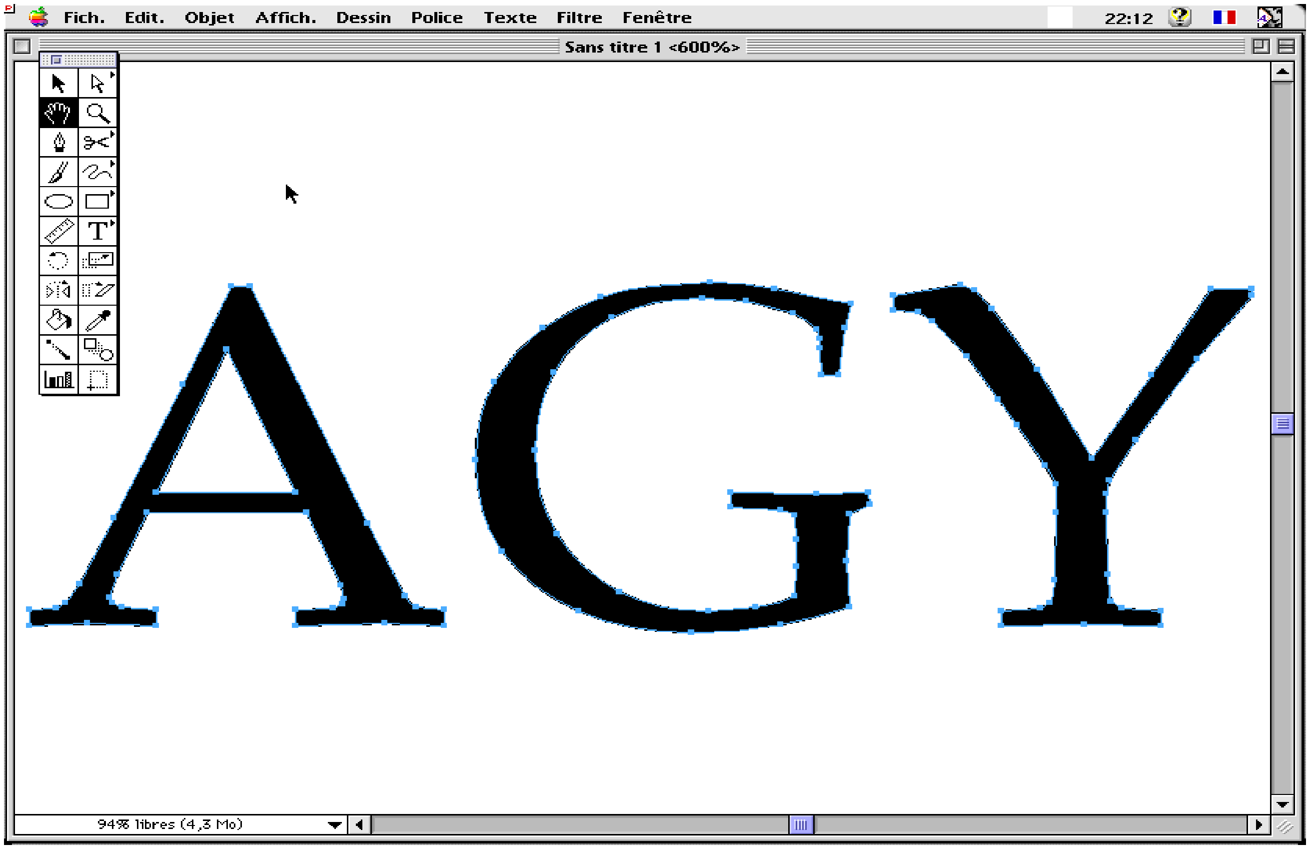
Figure 10-34
Examples of two-dimensional Bézier curves generated from three, four, and five control points. Dashed lines connect the control-point positions.

Bezierjeve krivulje: Freehand



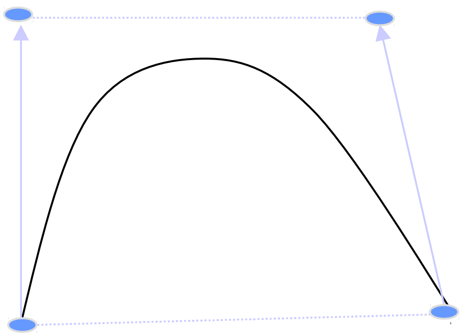
Bezierjeve krivulje : PS

- **Postscript : Bézier stopnje 2**



Bezierove krivulje

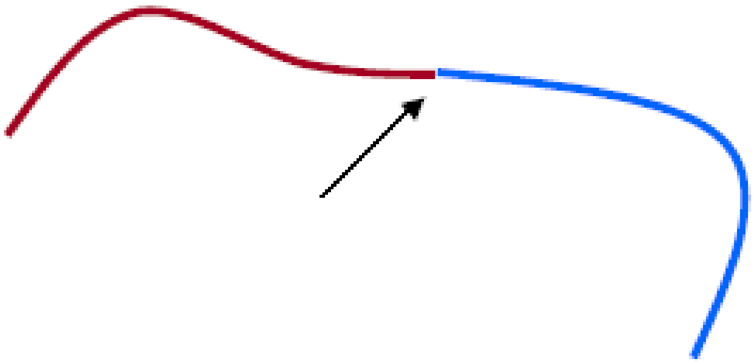
- **Developed by a car designer at Renault**
- **Advantages**
  - **curve contained to *convex hull***
  - **easy to manipulate**
  - **easy to render**
- **Disadvantages**
  - **bad continuity at endpoints**
    - **tough to join multiple Bezier splines**



Segmentne parametrične polinomske krivulje

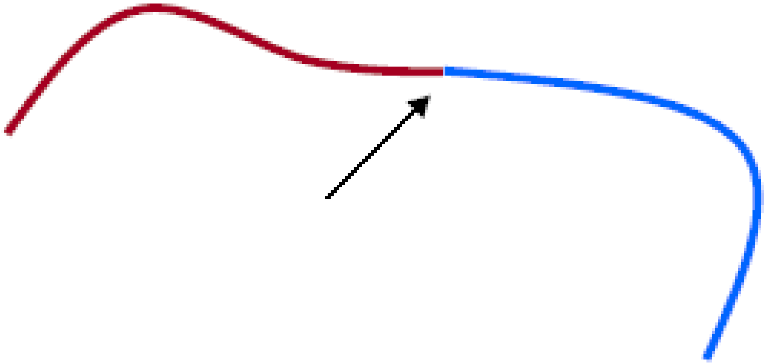
(Piecewise Param Polynomial Curves)

- Idea:
  - Use different polynomial functions on different parts of the curve
- Advantage:
  - Flexibility
  - Control
- Issue:
  - Smoothness at "joints"? (*continuity*)



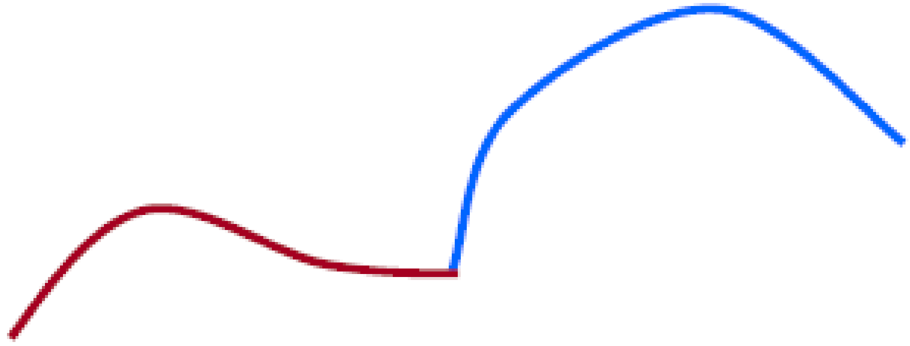
Zveznost

- Continuity C^k indicates adjacent curves have the same kth derivative at their joints



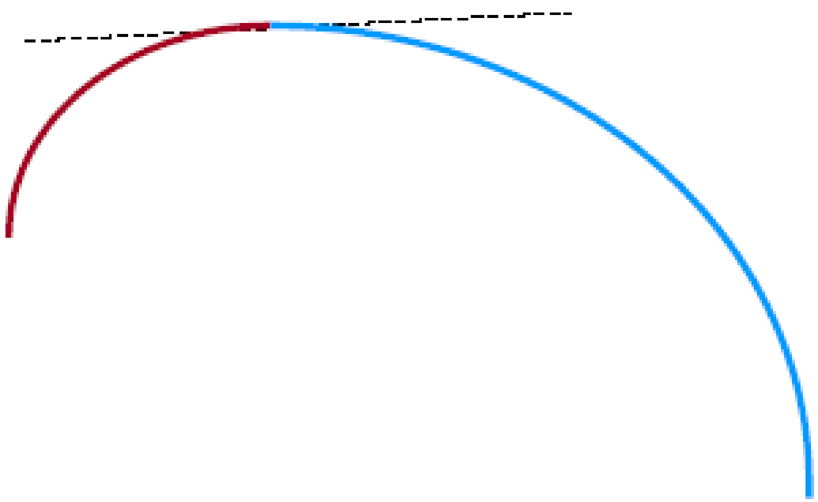
Zveznost C^0

- Adjacent curves share ...
  - Same endpoints: $Q_i(1) = Q_{i+1}(0)$



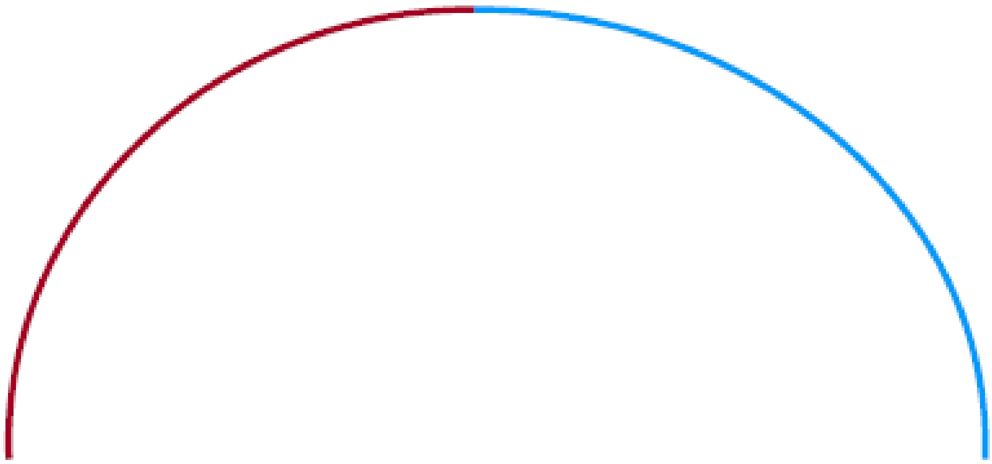
Zveznost C^1

- Adjacent curves share ...
  - Same endpoints: $Q_i(1) = Q_{i+1}(0)$
  - Same derivatives: $Q_i'(1) = Q_{i+1}'(0)$



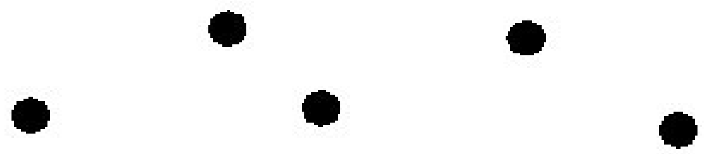
Zveznost C^2

- Adjacent curves share ...
  - Same endpoints: $Q_i(1) = Q_{i+1}(0)$
  - Same derivatives: $Q_i'(1) = Q_{i+1}'(0)$
  - Same second derivatives: $Q_i''(1) = Q_{i+1}''(0)$



Zlepki (splines)

Splines



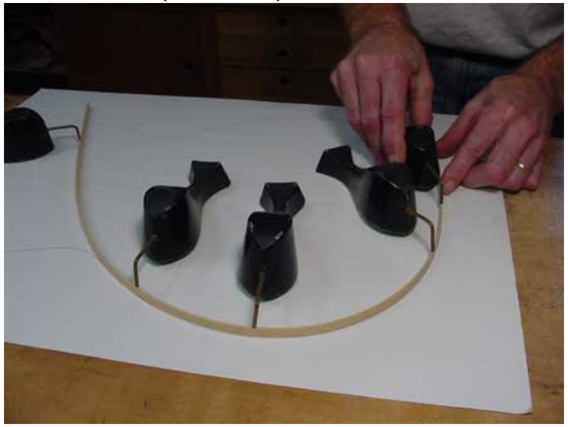
Functions that interpolate/approximate

1. Filtering and reconstruction for images
2. Keyframes and inbetweens for animation
3. Curves and surfaces for modeling

Zlepki - zgodovina

- **Načrtovalci uporabljajo “race” in trakove lesa (plines) za risanje krivulj**
- **Leseni zlepki imajo zveznost drugega reda (second-order continuity)**
- **..in potekajo skozi kontrolne točke**

Raca (utež)



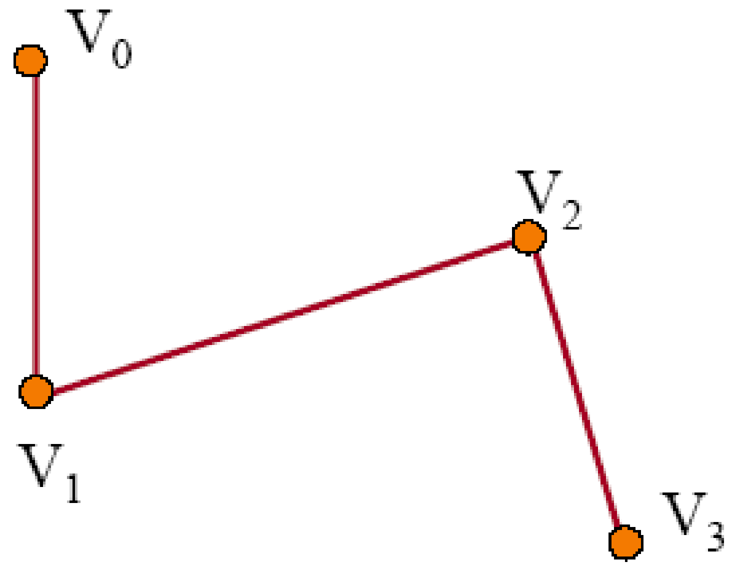
Z racami oblikujejo krivulje

Tvorba zlepkov

- C^2 interpolating splines
- Hermite
- Bezier
- Catmull-Rom
- B-splines

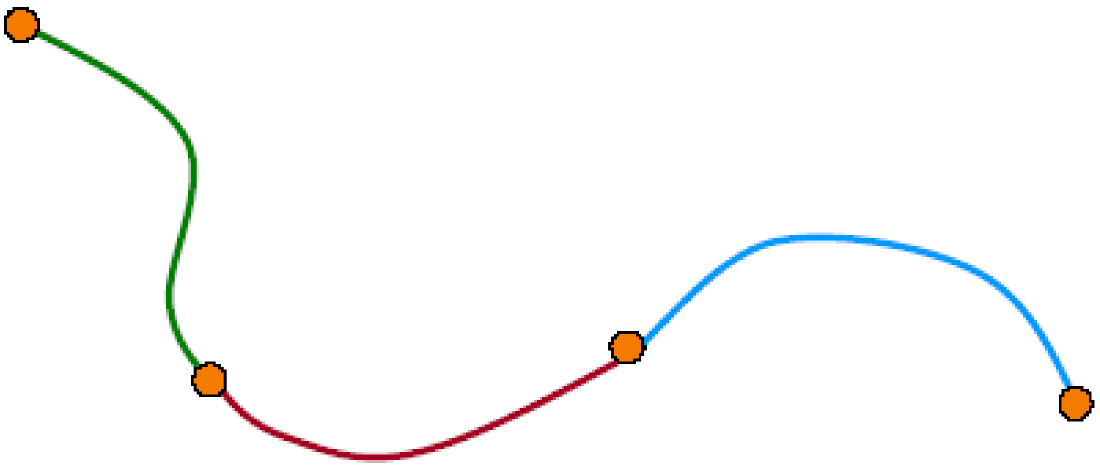
Blending functions

$$Q(u) = \sum_{i=0}^{k} V_i b_i(u)$$



Zlepki z interpolacijo C^2

- Blending functions are chosen so that ...
  - Control points are interpolated
  - Adjacent curves meet with C^2 continuity



B zlepki (b splines)?

- 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

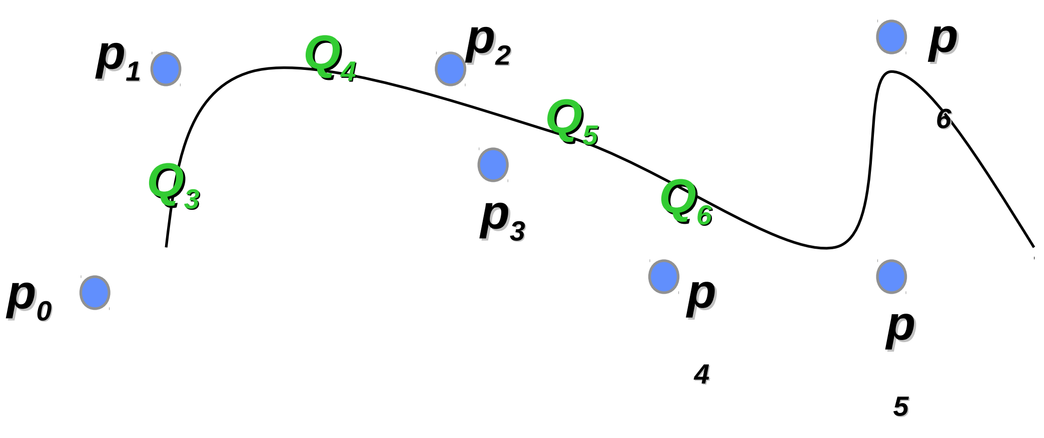
Demo

Krivulje z B-zlepki

- B-zlepki rešujejo dva glavna problema pri Bézierju:
  - Globalna kontrola kontrolnih točk
  - Odvisnost med stopnjo krivulje in številom kontrolnih točk
- Uporabnik poda število kontrolnih točk in sistem avtomatsko konstruira nabor kubičnih krivulj, za katere velja C2
  - To je podobno naboru združenih kubičnih Bézier krivulj
    - Združene kubične Bézier krivulje imajo lahko poljubno število kontrolnih točk in so še vedno kubične, vsaka kontrolna točka pa vpliva le lokalno
  - Vendar so B-zlepki C2 in nimajo omejitev na pozicijo kontrolnih točk, kot pri Bézier krivuljah

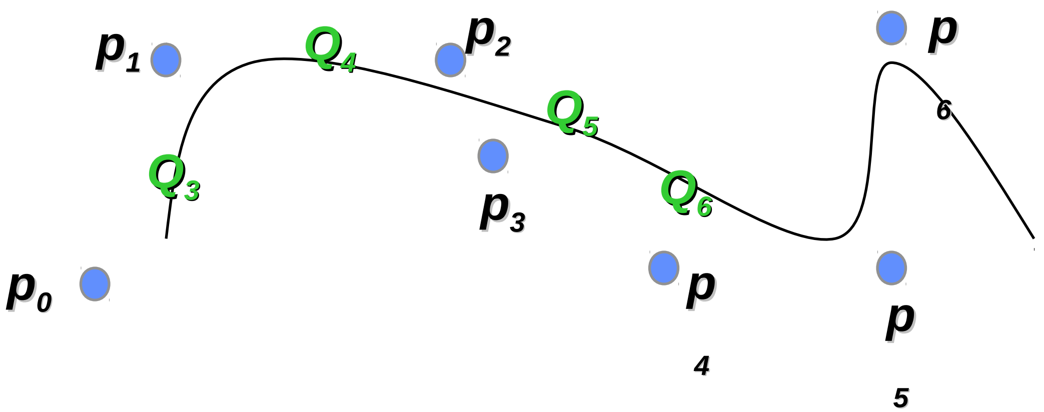
Enakomerni kubični b-zlepki

- 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



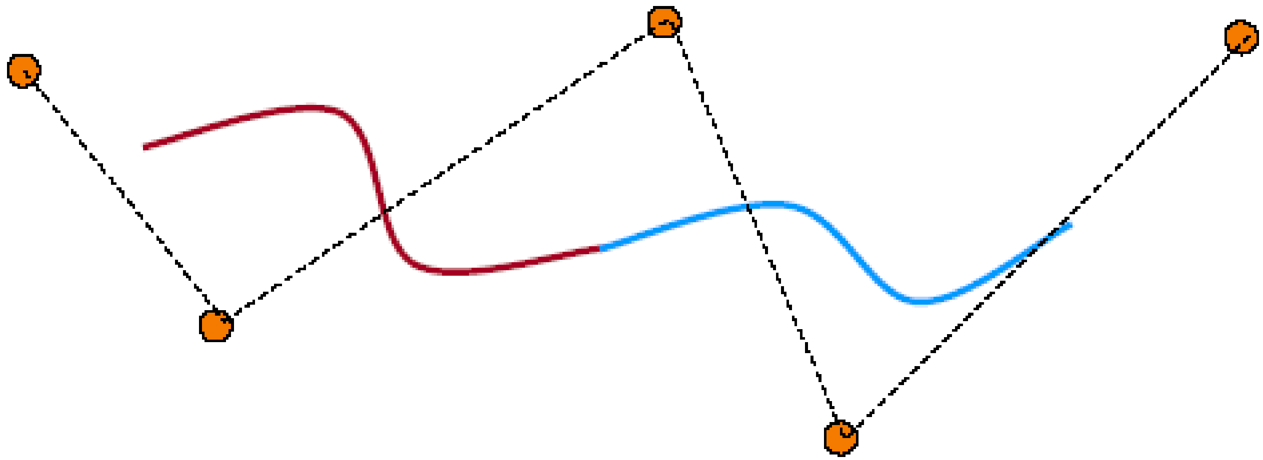
B-Spline Curve

- Start with a sequence of control points
- Select four from middle of sequence (p_{i-2}, p_{i-1}, p_i, p_{i+1})
  - 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



Enakomerni kubični b-zlepki

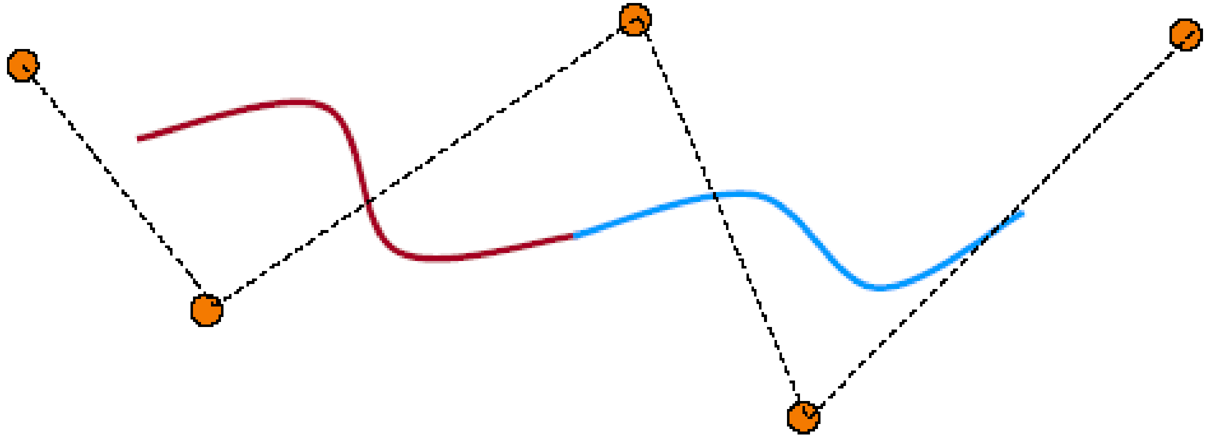
- Choose blending functions so that ...
  - Cubic polynomials
  - C^2 continuity
  - Local control
  - Points not necessarily interpolated



Enakomerni kubični b-zlepki

Derivation:

- Three continuity conditions for each joint J_i ...
  - » Position of two curves are equal at J_i
  - » Derivatives of two curves are equal at J_i
  - » Second derivatives of two curves are equal at J_i
- Also, local control implies ...
  - » Each joint is affected by small set of (4) points



Enakomerni kubični b-zlepki

- Approximating Splines
- Approximates n+1 control points
  - $P_0, P_1, \ldots, P_n, n \geq 3$
- 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

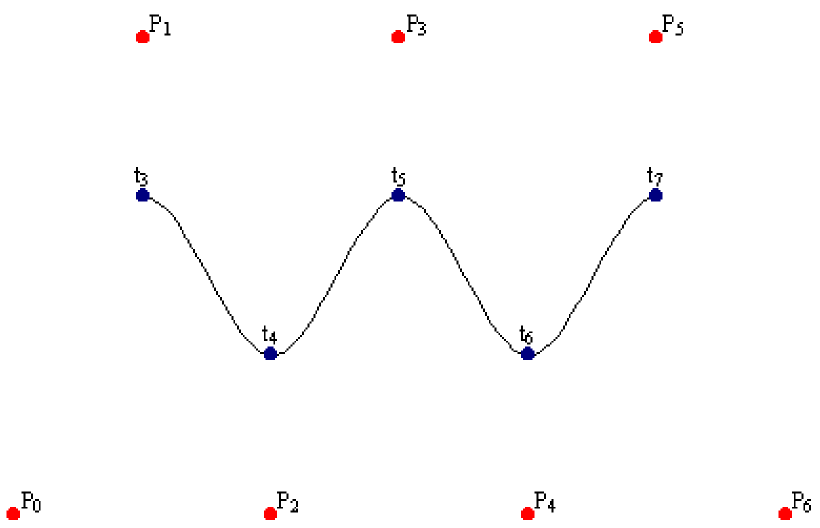
Bazna matrika b-zlepkov

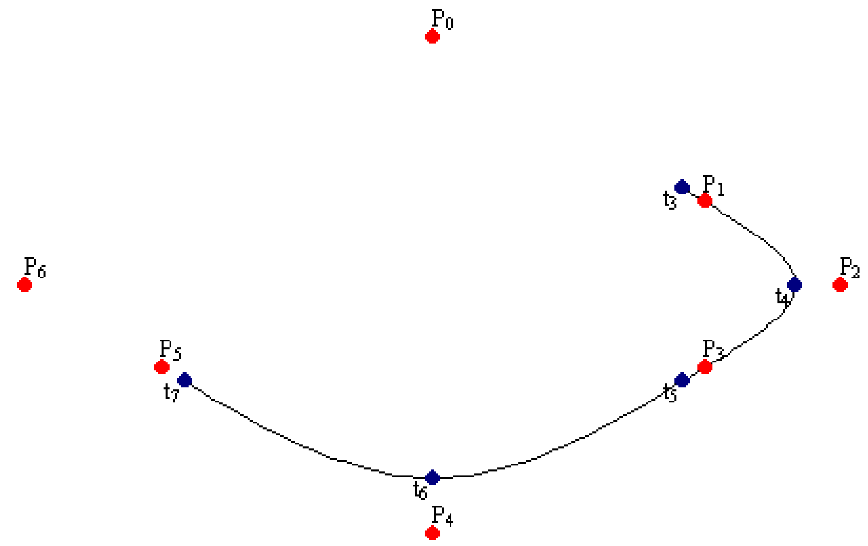
- Formulate 16 equations to solve the 16 unknowns
- The 16 equations enforce the C_0, C_1, and C_2 continuity between adjoining segments, Q

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

b - zlepki

- By far the most popular spline used
- C_0, C_1, and C_2 continuous





b - zlepki

- **Locality of points**

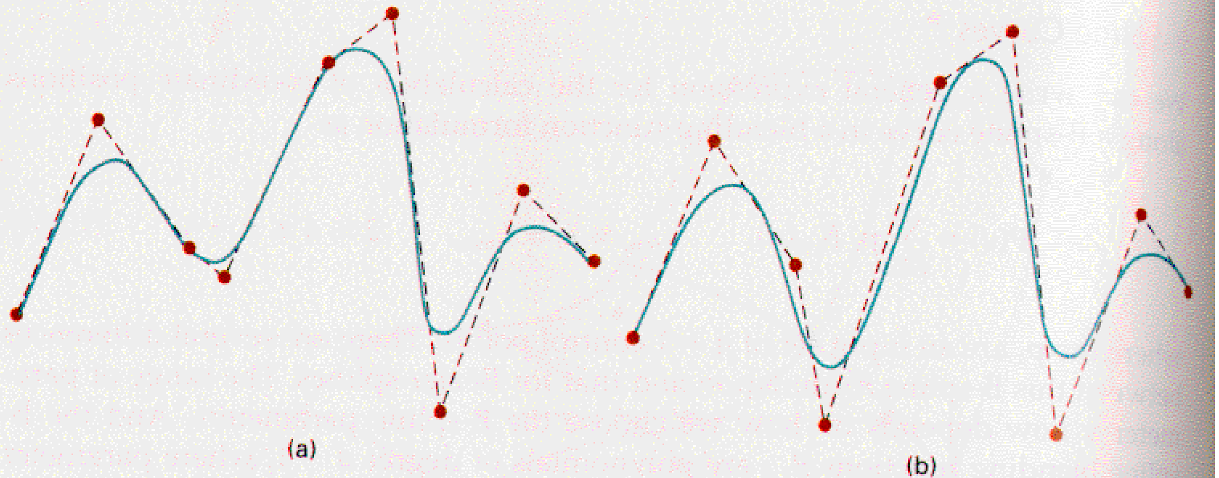


Figure 10-41
Local modification of a B-spline curve. Changing one of the control points in (a) produces curve (b), which is modified only in the neighborhood of the altered control point.

b - zlepki

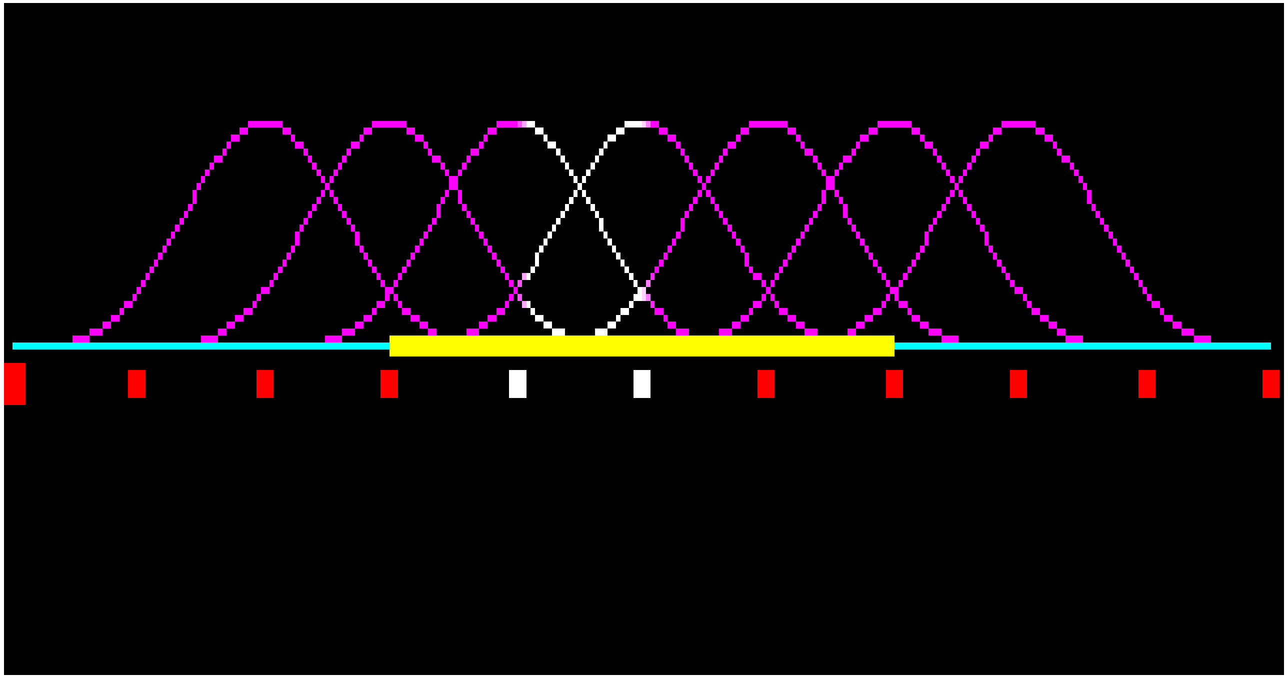
Points along B-Spline are computed just as with Bezier Curves

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

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

Uniform B-Spline Curves

Uniform blending functions get their name from the fact that all the blending functions are uniform

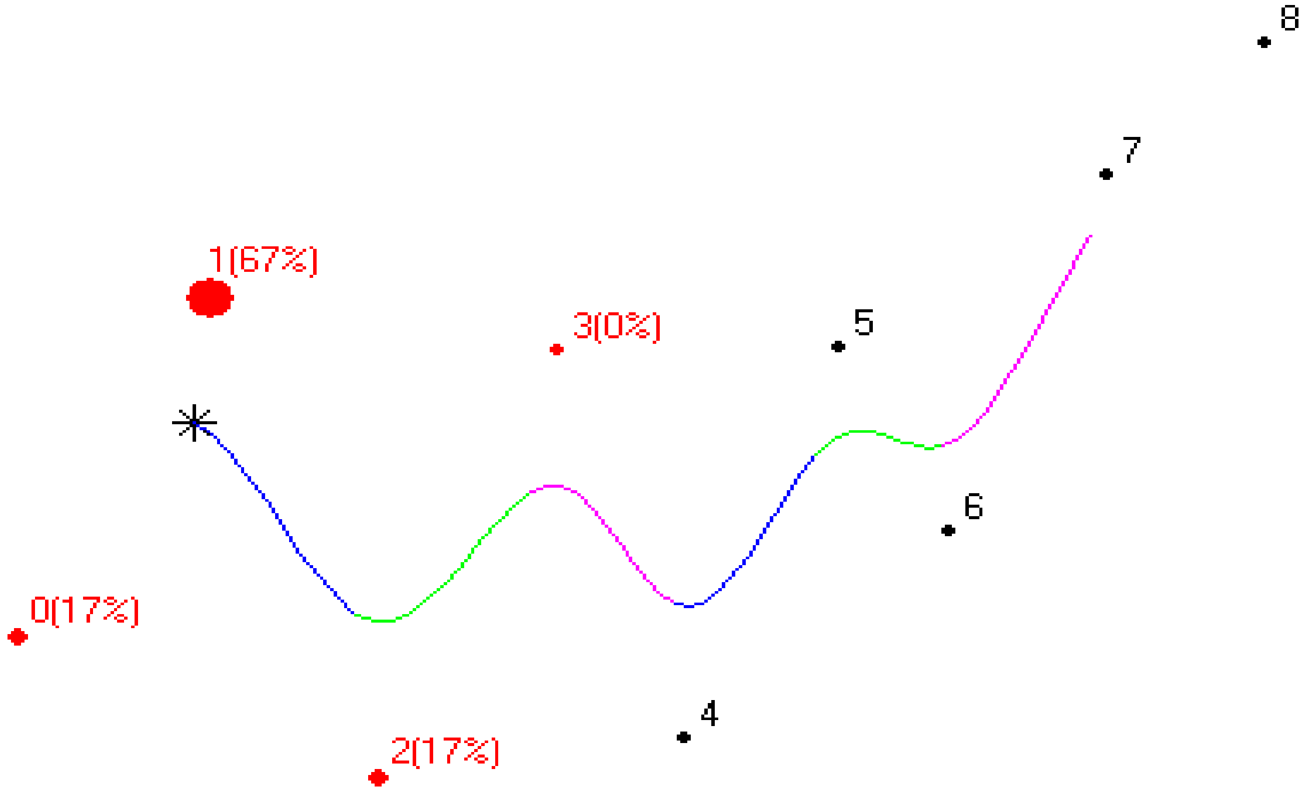


Uniform B-Spline Curves

- Recall that each curve segment composing the full B-Spline needs 4 control points modified by 4 blending functions
- This implies that only the section in yellow on the previous slide can be used to define the curve
  - At either end there are not enough blending functions
- This also implies that the curve will not go through *any* of the control points, including the first and last (as was the case with Bézier)
  - There are multiple non-zero blending functions at u=0 and u=1

Uniform B-Spline Curves

- Here is an example of a Uniform B-spline curve
  - M=8 → 9 control points, 6 curve segments



Non-Uniform B-Spline Curves

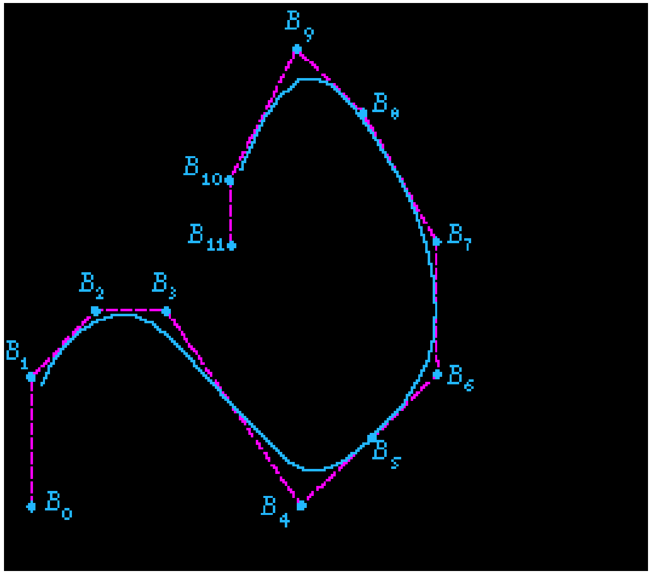
- The main difference between Uniform and Non-Uniform B-Splines is that all the blending functions are not the same
- There are several variations on the blending functions, all controlled by “knot values”
  - We won’t get into the details of knot values
- In particular, the blending function can be specified in such a way as force the curve to go through the endpoints (like Bézier)

Rational Curves

- A rational curve is a curve defined in 4D space that is then projected into 3D space
- The main point of using rational curves is that it allows you to define weights on the control points
  - Giving a control point a higher weight causes the curve to be pulled more towards that control point
- One can have Rational Bézier curves or Rational B-Spline curves
  - Or other types of curves not covered (Hermite, etc.)

NURBS

- NURBS stands for Non-Uniform Rational B-Splines
- It is one of the most popular curve representations used in CAD and graphics work because it allows:
  - Local control of the curve when moving the control points (B-Spline)
  - Ability to adjust the blending functions by moving the “knot values” (Non-Uniform)
  - Ability to weight the control points (Rational)

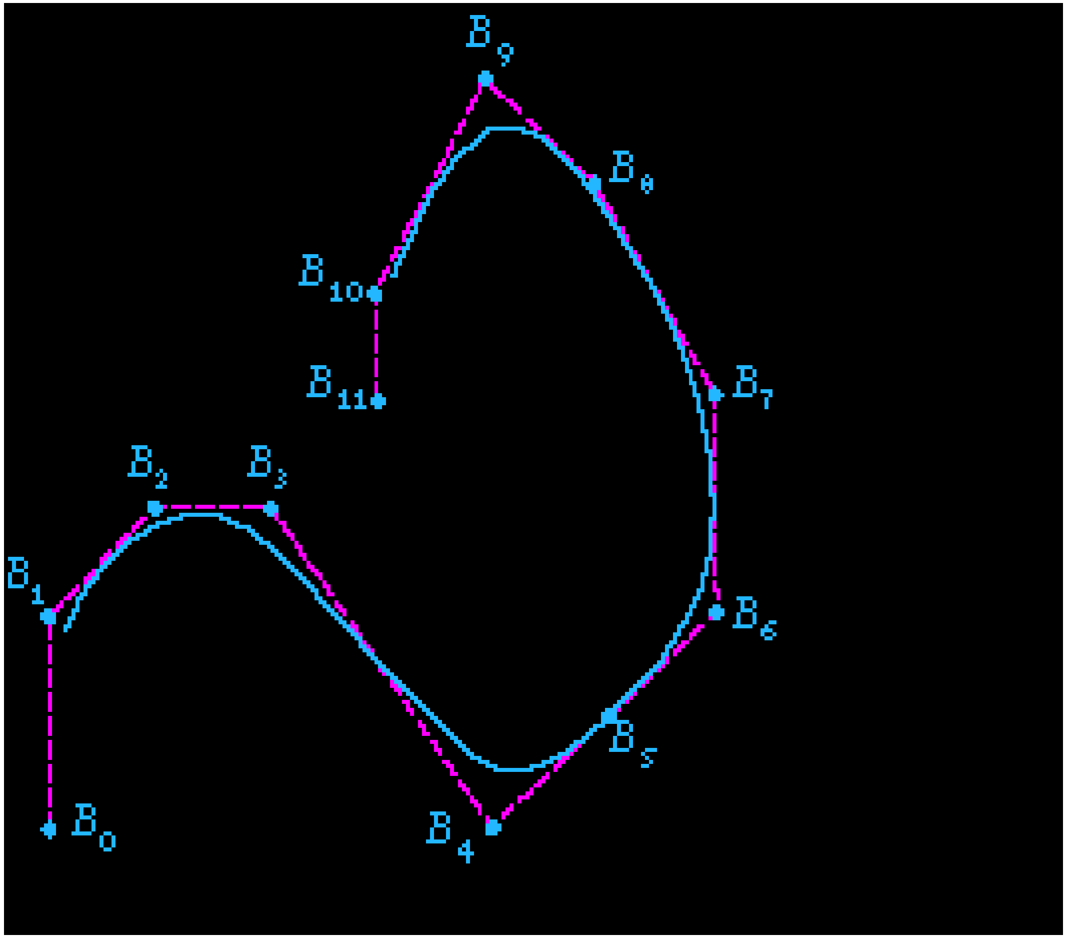


Nonuniform, Rational B-Splines (NURBS)

- 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 – kontrolne točke

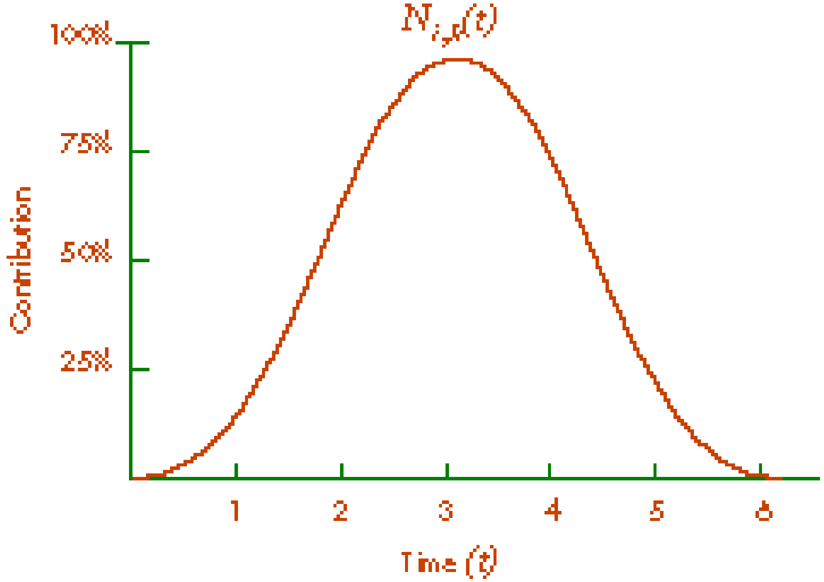
Ena ključnih lastnosti krivulj NURBS je, da njihovo obliko določa položaj kontrolnih točk, kot tiste na sliki, imenovane Bi. Za večjo razpoznavnost smo jih črtkano povezali. Tem povezavam pravimo tudi kontrolni poligon.



Izračun

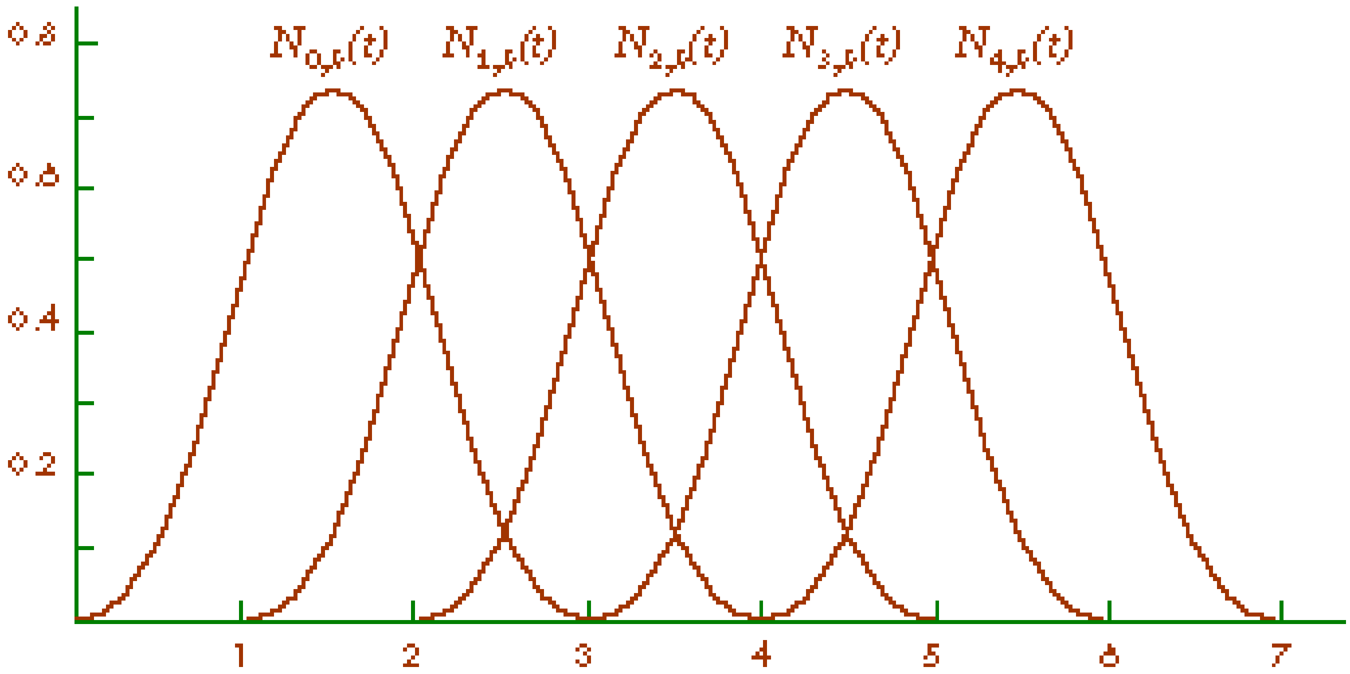
$$C(u) = \sum_{i=0}^{n} N_{i,p} P_i \qquad a \leq u \leq b$$

Funkcijo $N_{i,p}(u)$, ki določa, kako močno vpliva kontrolna točka Bi na krivuljo v času u, imenujemo bazna funkcija kontrolne točke P_i (Zato B v B-zlepkih pomeni bazni).



Slika prikazuje tipičen primer bazne funkcije. Ob določenem času ima maksimum, prej in kasneje pa gladko narašča oz. pada.

Bazne funkcije



Ker ima vsaka kontrolna točka svojo bazno funkcijo, ima krivulja NURBS z recimo petimi kontrolnimi točkami pet takih krivulj, ki vsaka zase pokrivajo svoj del krivulje (torej nek interval časa)

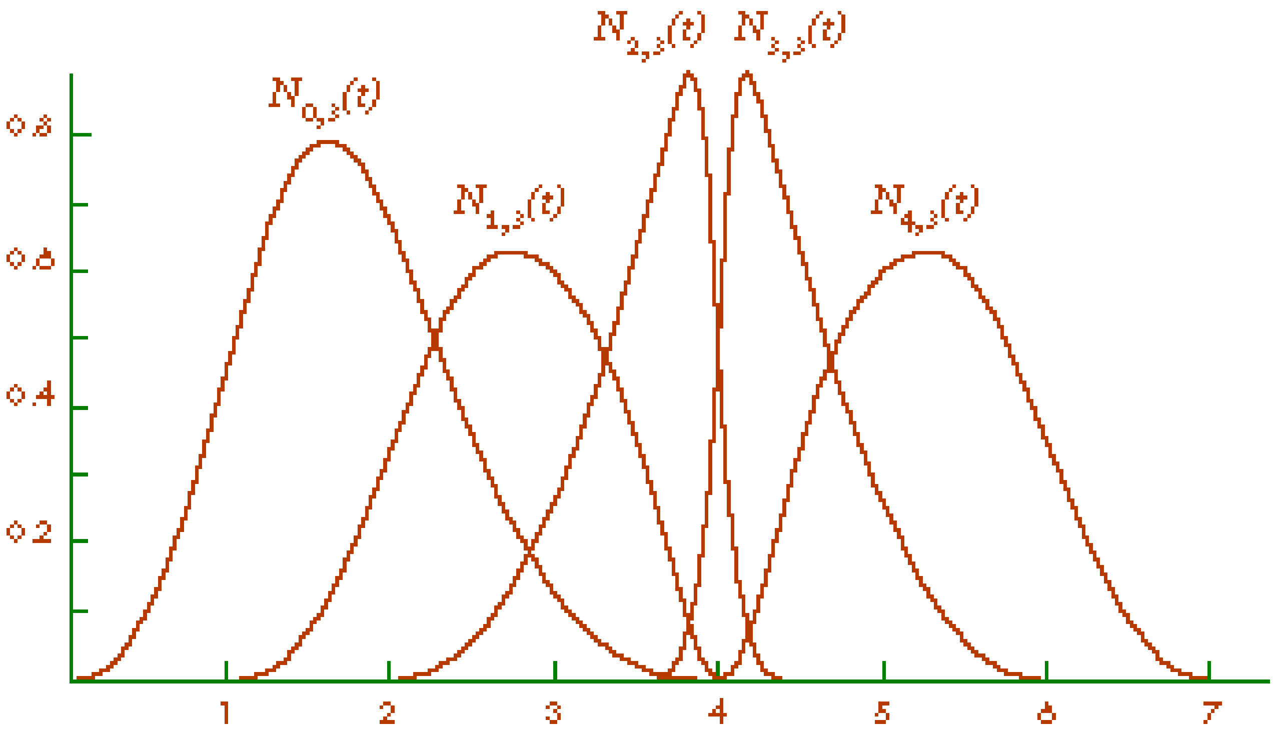
NURBS – vozli (knots)

Opazimo, da imajo vse bazne funkcije enako obliko in da pokrivajo enake intervale časa. Želeli pa bi, da bi kakšna točka vplivala dalj časa kot druga in bolj močno kot kakšna druga. In prav to je pomen črk NU v NURBS, ki pomenijo "non-uniform" (ne-enakomerno).

Problem rešimo tako, da definiramo množico točk, ki razdelijo čas na intervale, ki jih nato uporabimo pri baznih funkcijah, da dosežemo želen učinek.

S spreminjanjem relativne dolžine intervalov lahko spreminjamo čas vpliva kontrolne točke. Točkam, ki označujejo intervale pravimo **vozli** (knots).

NURBS – neenakomerne bazne funkcije



Definicija baznih funkcij

Sedaj smo pripravljeni, da definicijo krivulj NURBS izpopolnimo z bolj podrobno definicijo baznih funkcij:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

kjer je u_i oznaka za i-ti vozel v vektorju vozlov.

Opazimo, da so funkcije pri večjih indeksih *k* (ki jim pravimo tudi *red* bazne funkcije) zgrajene rekurzivno iz tistih nižjega reda. Če je *k* najvisji red bazne funkcije, ki sestavlja krivuljo NURBS, pravimo, da je to krivulja NURBS reda *k* oziroma *stopnje* *p*-1. Na dnu te hieararhije imamo funkcije reda 1, ki so enake 1, če je *u* med *i*-tim in (*i*+1) vozlom, sicer pa enak 0.

Rekurzija

- **Higher degree basis can be constructed from lower degree bases**

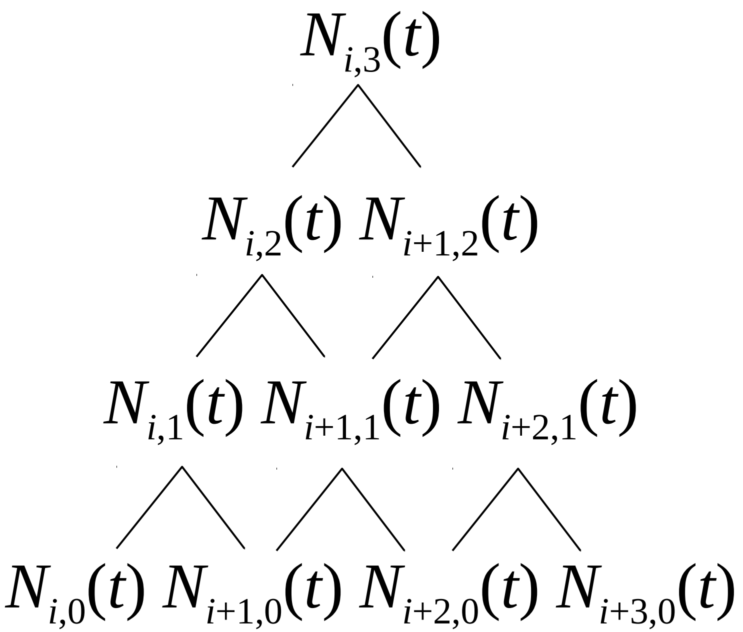
$$N_{i,d}(t) = \frac{t - t_i}{t_{i+d} - t_i} N_{i,d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}} N_{i+1,d-1}(t)$$

- $N_{i,0}(t)$ **=**

  1 if $t_i \le t < t_{i+1}$

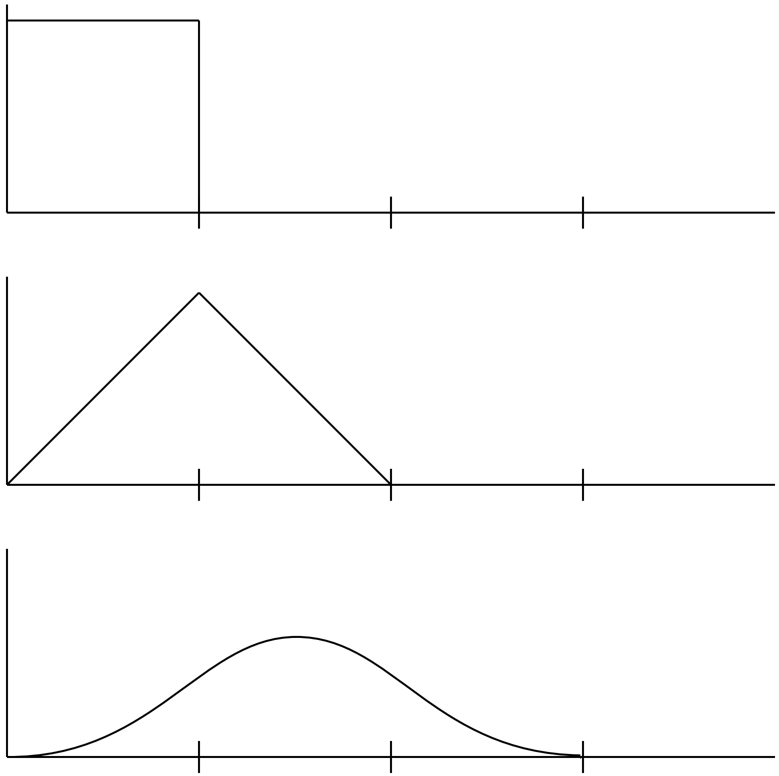
  0 otherwise

- **Non-uniform B-splines constructed using a systolic array**



Konvolucija

- Let $N_{i,k} = N_k(t - i)$
  - Uniform case
  - Translates of the same basis
- Then
  - $N_k(t) = (N_{k-1} * N_1)(t)$
  - $N_1(t) = \{ \quad 1 \text{ if } 0 \leq t < 1, \quad 0 \text{ otherwise}\}$
- $N_2(t)$ is piecewise quadratic Gaussian approximation consisting of three parabola segments
- One more yields cubic B-spline basis



NURBS - Racionalne funkcije

Spoznali smo pomen kontrolnih točk, vozlov in baznih funkcij in razumemo krivulje NUB (**n**on**u**niform **B**-spline). Kaj pa pomeni tisti R v NURB? Čas je, da spregovorimo o racionalnih krivuljah.

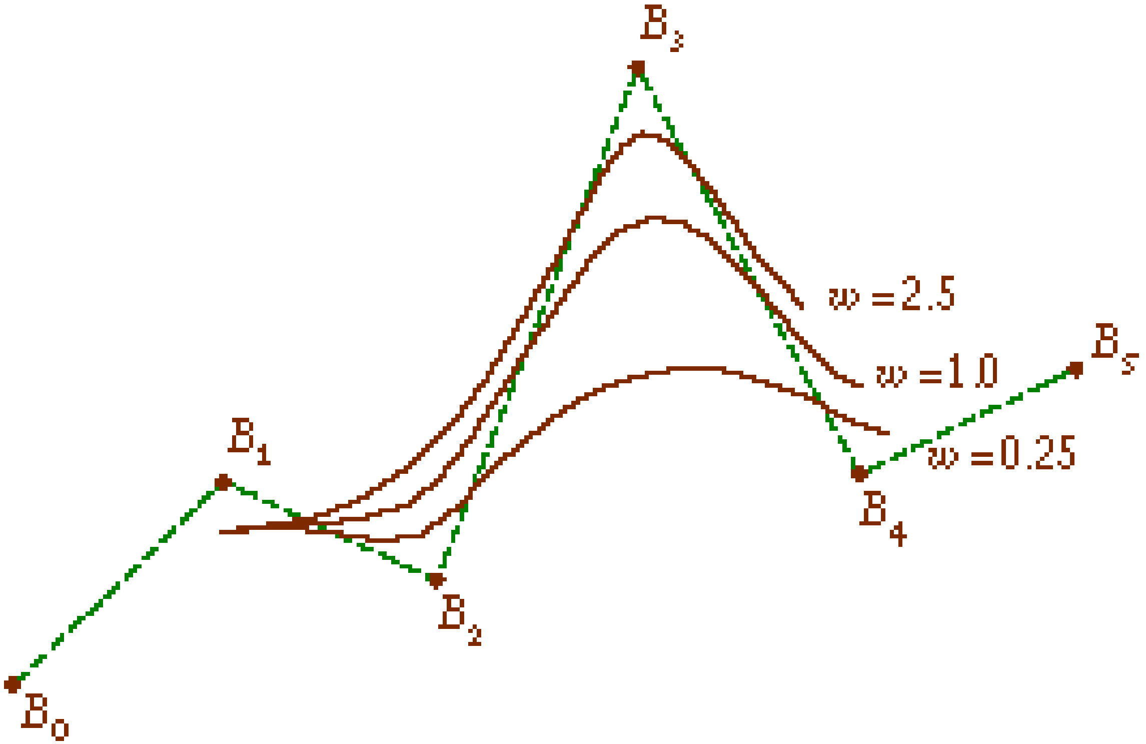
Krivulje, ki so tako definirane z utežjo za vsako kontrolno točko imenujemo racionalne krivulje.

$$\mathcal{X}(t) = \frac{\sum_{i=0}^{n+1} B_i w_i N_{i,k}(t)}{\sum_{i=0}^{n+1} w_i N_{i,k}(t)}$$

NURBS - pomen uteži

Nekateri programi zahtevajo za določanje krivulj NURB štiri-dimenzionalno predstavitev tro-dimenzionalnih kontrolnih točk: {x, y, z, w} namesto {x, y, z}.

Četrta koordinata w se obicajno nanaša na utež kontrolne točke. Navadno imajo vse točke utež 1.0, kar pomeni, da imajo vse enak vpliv na krivuljo. Če povečamo utež ene točke, ji tako damo večji vpliv na "vlečenje" krivulje proti tej točki.

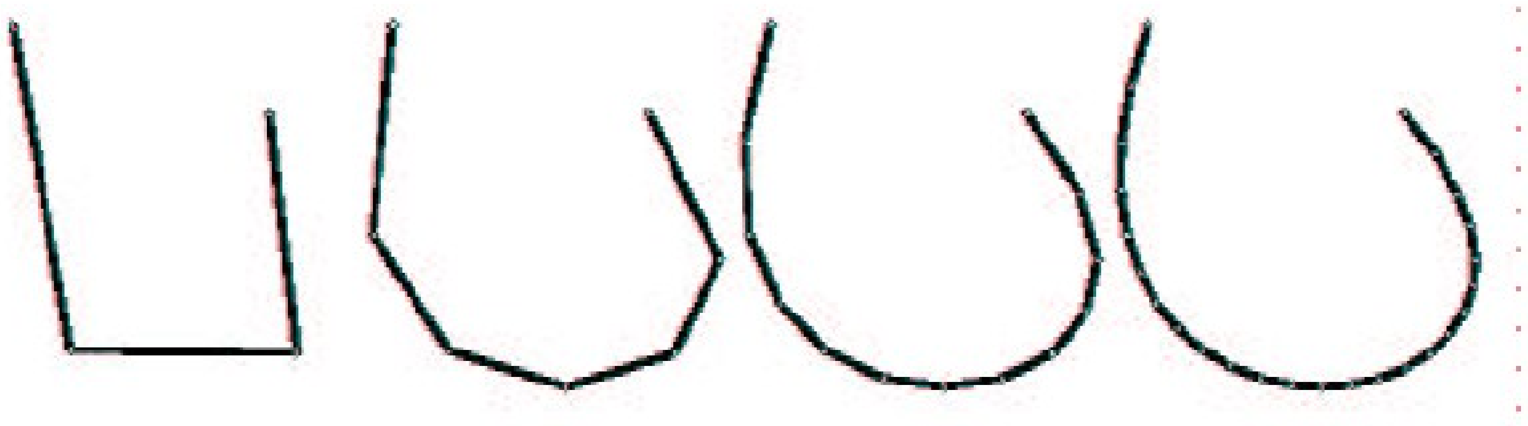


Primerjava kubičnih krivulj

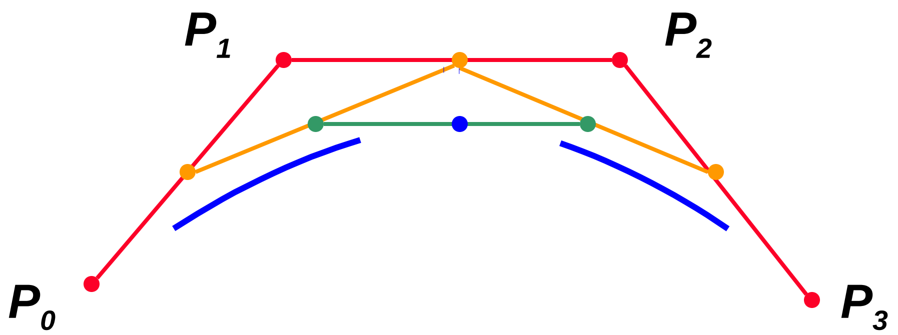
- **Hermitske**
  - Združujejo 4 krivulje; brez CP; polna interpolacija; C^1 in G^1 z omejitvami; hitre
- **Bézierjeve**
  - Konveksni CP, interpolirajo 2 od štirih kontrolnih točk; C^1 in G^1 z omejitvami; najhitrejše
- ***B-zlepki***
  - Uniformni, neracionalni
    - Konveksni CP, 4 točke, brez interpolacije; C^2 in G^2; srednje hitri
  - Neuniformni, neracionalni
    - Konveksni CP, 5 točk, " brez interpolacije"; “zmorejo” C^2 in G^2; počasni
  - Neuniformni, racionalni (NURBS)
    - Konveksni CP, 5 točk, " brez interpolacije"; racionalni; “zmorejo” C^2 in G^2; počasni
- **Beta zlepki (β-Splines)**
  - Konveksni CP; 6 kontrolnih točk (4 lokalne, 2 globalni); C^1 in G^2; srednje hitri
- **Catmull-Rom zlepki**
  - Konveksni CP; interpolirajo ali aproksimirajo 4 točke na CP; C^1 in G^1; srednje
- **Kochanek-Bartels zlepki**
  - Konveksni CP; interpolira 7 točk na CP; C^1 in G^1; srednje

Tvorba krivulj z delitvijo (subdivision)

- How do you make a smooth curve?



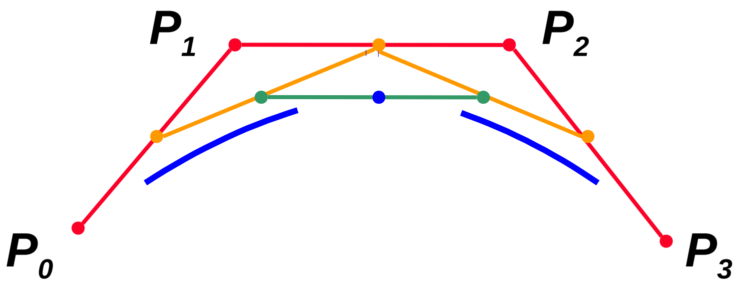
Interpolating Curves [1]: Recursive Subdivision



- Intuitive Idea
  - Given
    - Curve (Bézier or uniform B-spline) defined using control polygons (CPs)
    - 4 control points P_0, P_1, P_2, P_3
  - Problem: can't get quite the right curve shape (not enough control points)
  - Solutions: *increase degree of polynomial segments* OR *add CPs*
  - Technique: recursive subdivision algorithm
    - Add control points by splitting existing CP up recursively
    - Compute CPs for left curve L_0, L_1, L_2, L_3, right curve R_0, R_1, R_2, R_3
    - Stop when variation (curve-to-control point distance) is low enough
    - Purpose: *display curve* OR *allow new control points to be manipulated*

Interpolating Curves [2]:deCasteljau's Algorithm

- Recursive Subdivision Algorithm for Interpolation
  - Purpose: *display curve* OR *allow new control points to be manipulated*
  - Display: fast and cheap (see below)
- Properties
  - Cheap: can implement using subdivision matrices
  - Fast: rapid convergence due to…
  - Variation-diminishing property
    - Monotonic convergence to curve
    - Holds for all splines with convex-hull CPs
- When Does It Work?
  - Uniform splines (uniformly-spaced knots)
    - Q: Can we subdivide NURBS?
    - A: Yes, by adding knots (expensive)
  - Alternative approach: *hierarchical B-splines*



Upodabljanje zlepkov

- **Hornerjeva metoda**
- **Inkrementalna (Forward Difference) metoda**
- **Metode s deljenjem (Subdivision Methods)**

Hornerjeva metoda

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

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

- **Tri množenja, tri vsote**

Forward Difference

$$x_{k+1} = x_k + \Delta x_k$$

$$x_k = a_x u^3 + b_x u^2 + c_x u + d$$

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

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

- **Še vedno veliko računanja**
  - **Reši za spremembo v k (Δ_k) in spremembo v k+1 (Δ_{k+1})**
  - **Opremi z začetnimi vrednostmi za x_0, Δ_0, in Δ_1**
  - **Izračunaj x_3 z seštevanjem $x_0 + \Delta_0 + \Delta_1$**

Delitvene metode

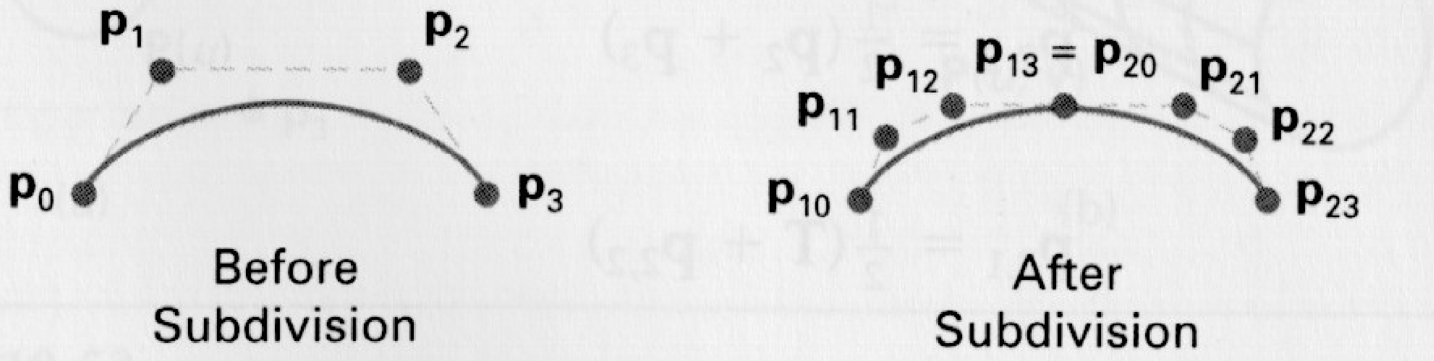
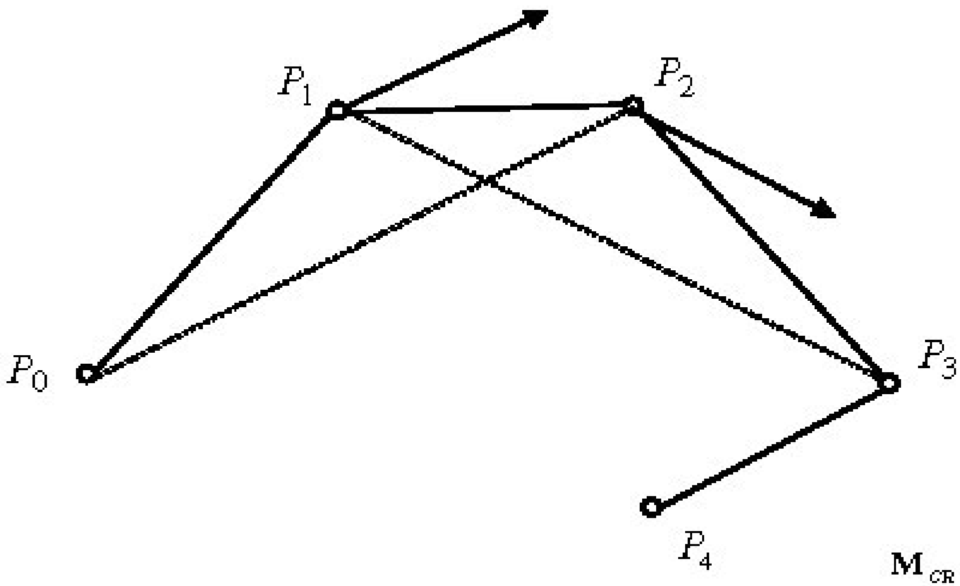


Figure 10-52
Subdividing a cubic Bézier curve section into two sections, each with four control points.

- **Bezier**

Catmull-Rom zlepek

Catmull-Rom Spline



$$T_1 = \tfrac{1}{2}(P_2 - P_1)$$
$$T_2 = \tfrac{1}{2}(P_3 - P_2)$$
$$T_3 = \tfrac{1}{2}(P_4 - P_3)$$
$$\ldots$$

$$\mathbf{M}_{CR} = \frac{1}{2}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

Upodabljanje Bezier zlepkov

```
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);
}
```

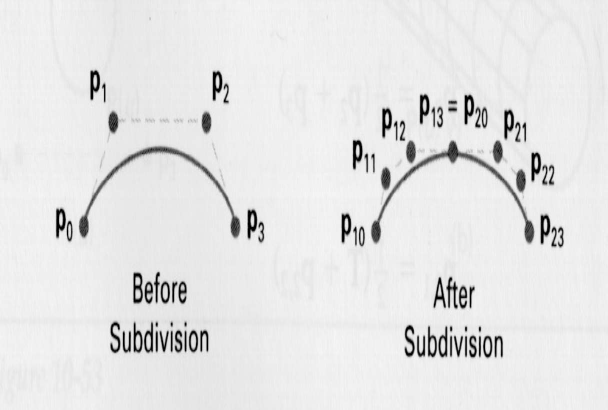


Figure 10-52
Subdividing a cubic Bézier curve section into two sections, each with four control points.