Bézierjeve krivulje

Emil Žagar

Fakulteta za matematiko in fiziko, Univerza v Ljubljani

Matematično modeliranje

Slika: Prepoznate lik na sliki?

Slika: Kaj pa ta dva?

Slika: In te?

Ravninske parametrične krivulje

Ravninske Bézierjeve krivulje

de Casteljauov algoritem

Subdivizija

Zlepki

Drobtinica o ploskvah

Funkcije (ene spremenljivke) dobro poznamo:

$$f : [a, b] \to \mathbb{R}, \quad x \mapsto f(x).$$

Točka na grafu funkcije f je

$$\begin{pmatrix} x \\ f(x) \end{pmatrix}, \quad x \in [a, b].$$

Kaj pa če sta obe komponenti na grafu funkciji iste spremenljivke (parametra) t:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \quad t \in I \subseteq \mathbb{R}.$$

Predpisu

$$\boldsymbol{r} : I \to \mathbb{R}^2, \quad \boldsymbol{r}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

rečemo ravninska parametrična krivulja ali natančneje parametrizacija krivulje $\boldsymbol{r}$.

Podobno lahko definiramo parametrično krivuljo v več dimenzijah:

$$\boldsymbol{r} : I \to \mathbb{R}^d, \quad \boldsymbol{r}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_d(t) \end{pmatrix}, \quad d \in \mathbb{N}.$$

Praktično so najpomembnejše parametrične krivulje, katerih komponente so polinomi:

- preprosta predstavljivost (seznam koeficientov),
- hitro računanje (denimo Hornerjev algoritem).

Polinome pa lahko zapišemo v različnih bazah, na primer:

- standardna:

$$\varphi_j(x) = x^j, \quad j = 0, 1, \ldots, n,$$

- Newtonova:

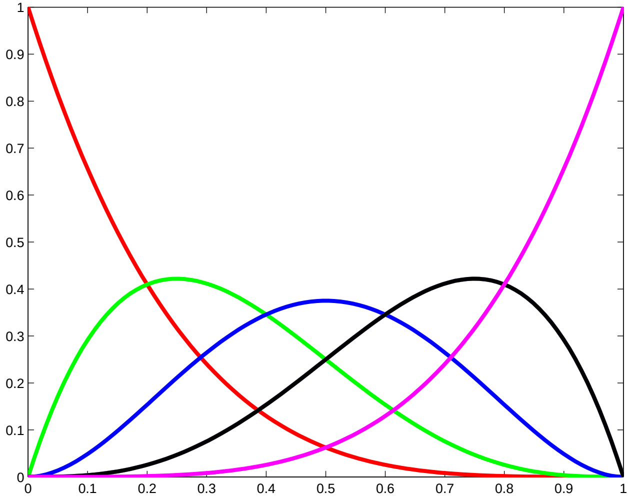
$$\begin{aligned} \varphi_0(x) &= 1, \\ \varphi_j(x) &= (x - x_0)(x - x_1)\cdots(x - x_{j-1}), \quad j = 1, \ldots, n, \end{aligned}$$

- Bernsteinova:

$$\varphi_j(x) = \binom{n}{j} x^j (1 - x)^{n-j}, \quad j = 0, 1, \ldots, n,$$

- ...

- Vsaka baza ima svoje prednosti in pomanjkljivosti.
- Za računalniško podprto geometrijsko oblikovanje (ang. Computer Aided Geometric Design, oz. na kratko CAGD) je pomembna predvsem Bernsteinova baza.
- Podobna področja so še CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), CNC (Computer Numerical Control).



Slika: Bazni Bernsteinovi polinomi stopnje 4 (B_0^4 rdeče, B_1^4 zeleno, B_2^4 modro, B_3^4 črno in B_4^4 rožnato.

- Neodvisno jih v drugi polovici 20. stoletja razvijeta P.E. Bézier[1] in P. de Casteljau[2].
- Temeljijo na Bernsteinovi bazi polinomov.
- Z razvojem računalnikov postanejo nesluteno uporabne.
- Dandanes so nepogrešljive v CAGD (animacije; avtomobilska, ladijska in letalska industrija;...).
- Obstajajo številne posplošitve na ploskve in v več dimenzij.

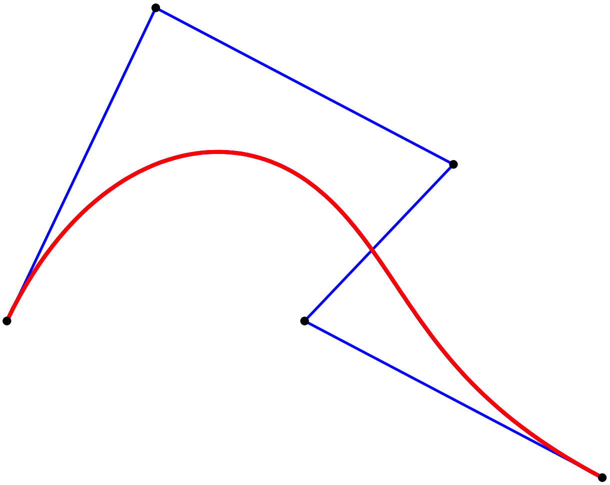
[1] Pierre Étienne Bézier, 1910-1999, francoski razvojni inženir pri Renaultu.

[2] Paul de Faget de Casteljau, 1930-1999, francoski razvojni inženir pri Citroënu.

- Izberimo zaporedje točk v ravnini $\{\boldsymbol{P}_j\}_{j=0}^n$ in definirajmo

$$\boldsymbol{b}(t) = \sum_{j=0}^{n} \boldsymbol{P}_j \, B_j^n(t), \quad t \in [0,1],$$

  kjer so B_j^n Bernsteinovi polinomi.
- Krivulji $\boldsymbol{b}$ rečemo ravninska Bézierova krivulja.
- Točkam $\{\boldsymbol{P}_j\}_{j=0}^n$ pravimo kontrolne točke, poligonu, ki ga določajo, pa kontrolni poligon.



Slika: Bézierova krivulja stopnje 4 (rdeče) s kontrolnimi točkami $\boldsymbol{P}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\boldsymbol{P}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, $\boldsymbol{P}_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$, $\boldsymbol{P}_3 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$, $\boldsymbol{P}_4 = \begin{pmatrix} 4 \\ -1 \end{pmatrix}$.

Bézierjeve krivulje imajo za oblikovanje nekaj pomembnih lastnosti:

- Prva in zadnja kontrolna točka sta interpolacijski

$$\boldsymbol{b}(0) = \boldsymbol{P}_0, \quad \boldsymbol{b}(1) = \boldsymbol{P}_n.$$

- Tangentna vektorja na krivuljo v začetni in končni točki sta

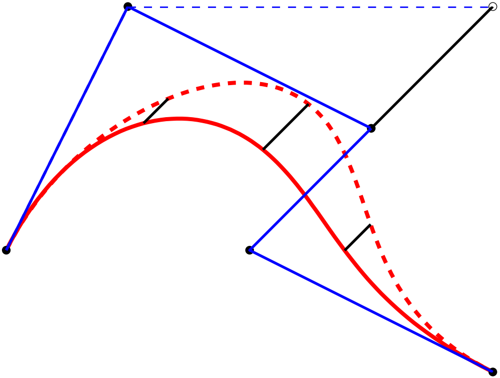
$$\boldsymbol{b}'(0) = n(\boldsymbol{P}_1 - \boldsymbol{P}_0), \quad \boldsymbol{b}'(1) = n(\boldsymbol{P}_n - \boldsymbol{P}_{n-1}).$$

- Krivulja leži v konveksni ovojnici kontrolnih točk.

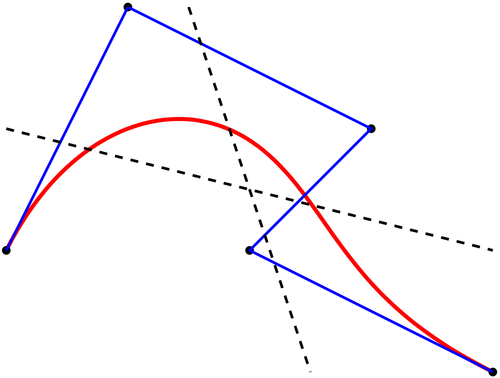
- Če j-to kontrolno točko premaknemo za vektor $\boldsymbol{v}$, se točke na krivulji premaknejo za

$$B_j^n(t)\,\boldsymbol{v}.$$

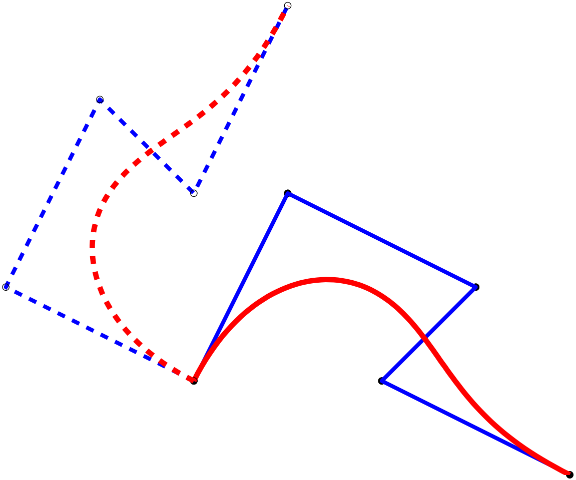
- Vsaka premica seka krivuljo kvečjemu tolikokrat kot kontrolni poligon.
- Afine transformacije lahko izvajamo le na kotrolnih točkah (uporabno recimo v PostScriptu).
- ...



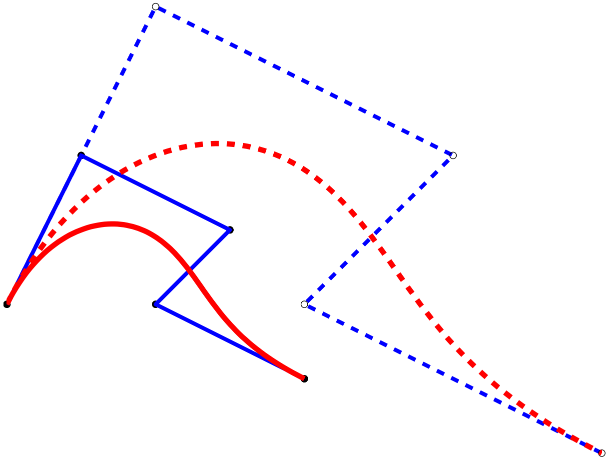
Slika: Premik tretje kontrolne točke Bézierjeve krivulje za vektor $\boldsymbol{v} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.



Slika: Premica lahko seka krivuljo kvečjemu tolikokrat kot kontrolni poligon.



Slika: Rotacija Bézierjeve krivulje za 90° okrog prve kontrolne točke v pozitivni smeri.



Slika: Skaliranje Bézierjeve krivulje za faktor 2.

- Osnovna naloga pri Bézierovih krivuljah je računanje točk na krivulji.
- Neposredni račun je na dlani: izberemo parameter $t \in [0, 1]$ in izračunamo

$$\boldsymbol{b}(t) = \sum_{j=0}^{n} \boldsymbol{P}_j \, B_j^n(t).$$

- Žal je tak način časovno prezahteven in nestabilen.
- Obstaja alternativa, de Casteljauov algoritem

Oglejmo si primer: dane so kontrolne točke

$$\boldsymbol{P}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \boldsymbol{P}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \ \boldsymbol{P}_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \ \boldsymbol{P}_3 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \ \boldsymbol{P}_4 = \begin{pmatrix} 4 \\ -1 \end{pmatrix}$$

in parameter $t = 3/4$.

- 1. korak: definirajmo $\boldsymbol{b}_j^0 := \boldsymbol{P}_j$, $j = 0, 1, 2, 3, 4$.
- 2. korak: izračunajmo

$$\boldsymbol{b}_j^1 = (1-t)\, \boldsymbol{b}_j^0 + t\, \boldsymbol{b}_{j+1}^0, \quad j = 0, 1, 2, 3,$$

torej

$$\boldsymbol{b}_0^1 = \begin{pmatrix} 3/4 \\ 3/2 \end{pmatrix}, \ \boldsymbol{b}_1^1 = \begin{pmatrix} 5/2 \\ 5/4 \end{pmatrix} \ \boldsymbol{b}_2^1 = \begin{pmatrix} 9/4 \\ 1/4 \end{pmatrix}, \ \boldsymbol{b}_3^1 = \begin{pmatrix} 7/2 \\ -3/4 \end{pmatrix}.$$

- 3. korak: izračunajmo

$$\boldsymbol{b}_j^2 = (1-t)\,\boldsymbol{b}_j^1 + t\,\boldsymbol{b}_{j+1}^1, \quad j = 0, 1, 2,$$

  torej

$$\boldsymbol{b}_0^2 = \begin{pmatrix} 33/16 \\ 21/16 \end{pmatrix}, \; \boldsymbol{b}_1^2 = \begin{pmatrix} 37/16 \\ 1/2 \end{pmatrix} \; \boldsymbol{b}_2^2 = \begin{pmatrix} 51/16 \\ -1/2 \end{pmatrix}.$$

- 4. korak: izračunajmo

$$\boldsymbol{b}_j^3 = (1-t)\,\boldsymbol{b}_j^2 + t\,\boldsymbol{b}_{j+1}^2, \quad j = 0, 1,$$

  torej

$$\boldsymbol{b}_0^3 = \begin{pmatrix} 9/4 \\ 45/64 \end{pmatrix}, \; \boldsymbol{b}_1^3 = \begin{pmatrix} 95/32 \\ -1/4 \end{pmatrix}.$$

- 5. korak: izračunajmo

$$\boldsymbol{b}_j^4 = (1-t)\,\boldsymbol{b}_j^3 + t\,\boldsymbol{b}_{j+1}^3, \quad j = 0,$$

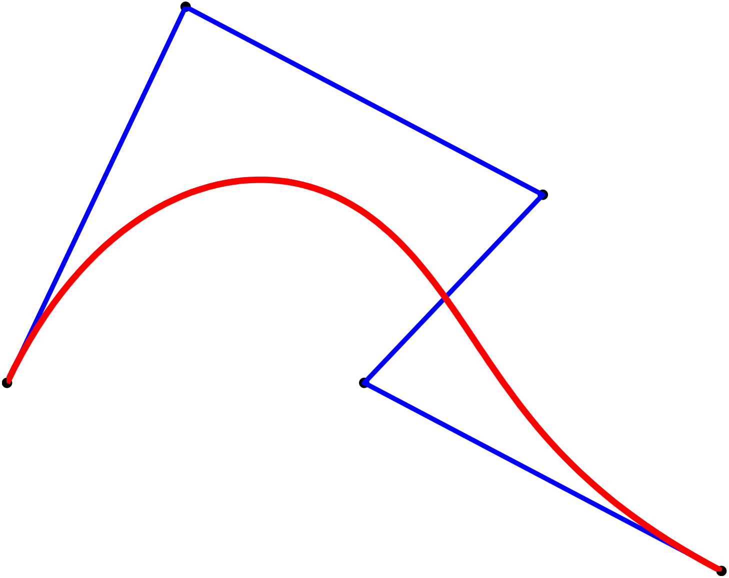
torej

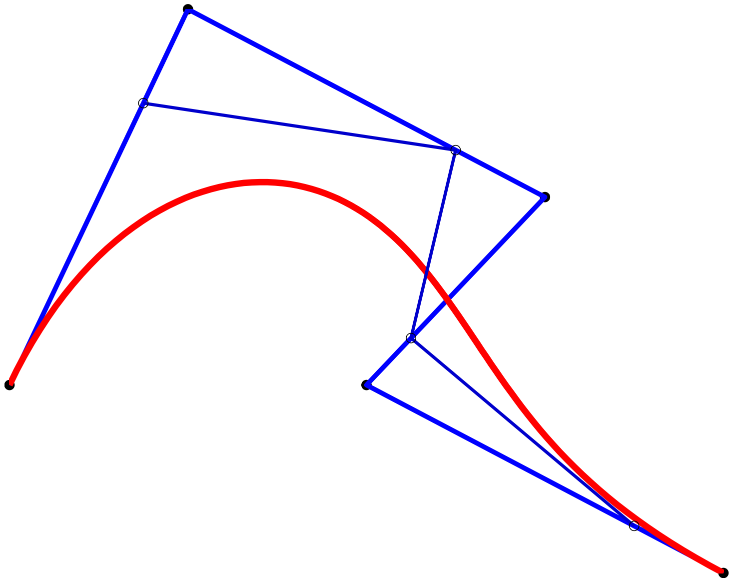
$$\boldsymbol{b}_0^4 = \begin{pmatrix} 357/128 \\ -3/256 \end{pmatrix}.$$

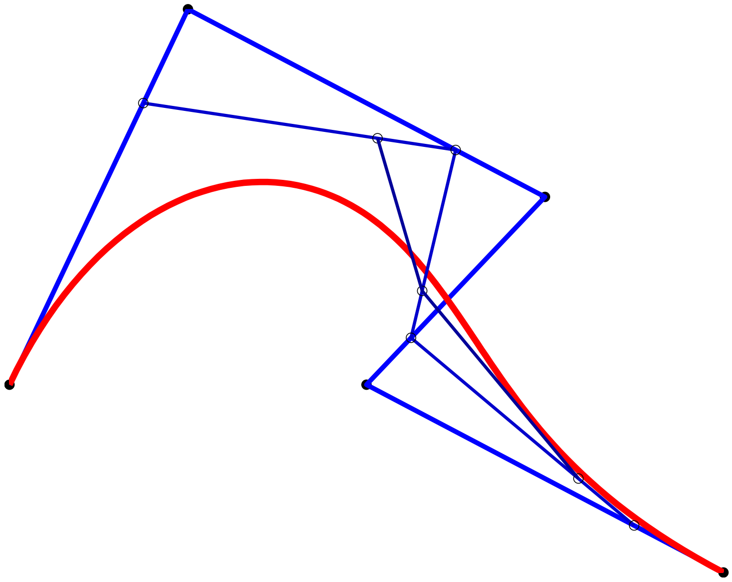
Izkaže se, da je

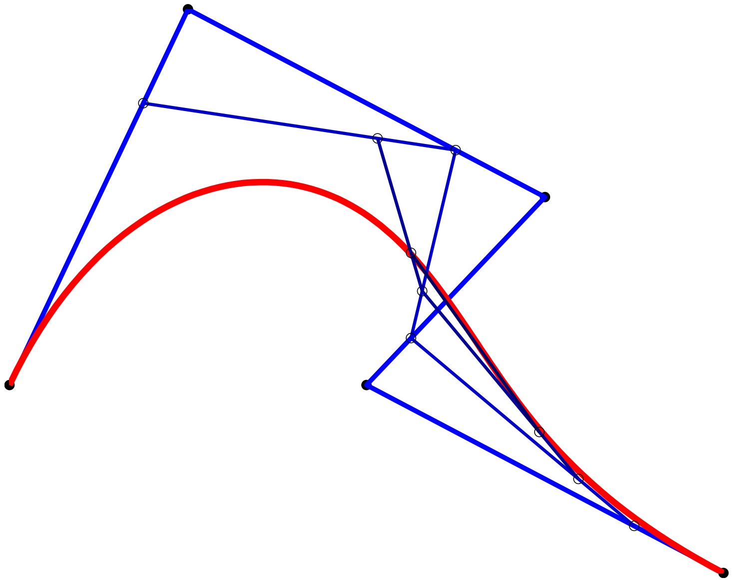
$$\boldsymbol{b}(t) = \boldsymbol{b}_0^4.$$

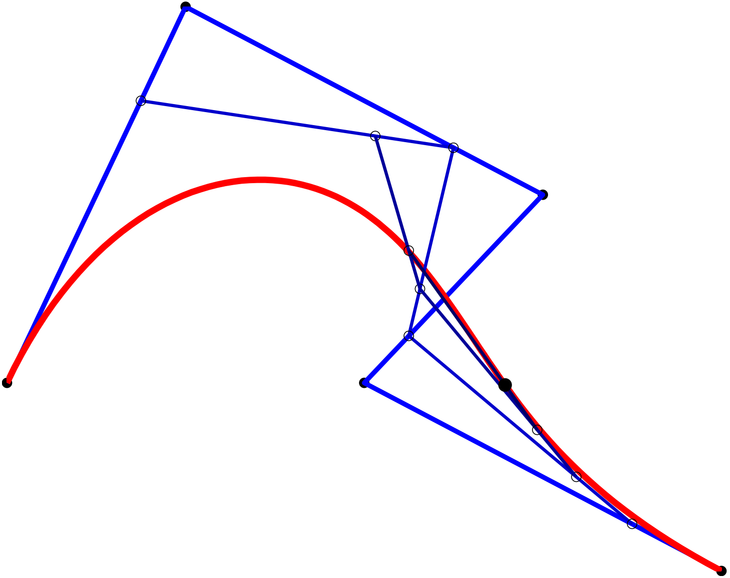
Oglejmo si še grafični prikaz računanja:











Splošni de Casteljauov algoritem se torej glasi:

Algoritem

Podatki: $\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n \in \mathbb{R}^2$ *in* $t \in [0, 1]$.

Definirajmo: $\boldsymbol{b}_j^0(t) = \boldsymbol{P}_j$, $j = 0, 1, \ldots, n$.

Ponavljajmo:

$$\boldsymbol{b}_j^k(t) = (1 - t)\, \boldsymbol{b}_j^{k-1} + t\, \boldsymbol{b}_{j+1}^{k-1}(t), \qquad \begin{aligned} k &= 1, 2, \ldots, n, \\ j &= 0, 1, \ldots, n - k. \end{aligned}$$

Izhod: $\boldsymbol{b}_0^n(t)$ *je točka na Bézierovi krivulji pri parametru* t.

Ponavadi vmesne točke de Casteljauovega algoritma zapišemo v tabelo:

$$\begin{array}{ccccc}
\boldsymbol{b}_0^0 & & & & \\
\boldsymbol{b}_1^0 & \boldsymbol{b}_0^1 & & & \\
\boldsymbol{b}_2^0 & \boldsymbol{b}_1^1 & \ddots & & \\
\vdots & \vdots & \ddots & \boldsymbol{b}_0^{n-1} & \\
\boldsymbol{b}_n^0 & \boldsymbol{b}_{n-1}^1 & \cdots & \boldsymbol{b}_1^{n-1} & \boldsymbol{b}_0^n
\end{array}$$

Če shranimo vse točke, potrebujemo tabelo dimenzije $(n+1) \times (n+1) \times 2$.
Če nas zanima samo končna točka, je dovolj tabela dimenzije $(n+1) \times 2$. Zakaj?

- de Casteljauov algoritem pa je veliko več kot samo način izračuna točke na krivulji.
- Vmesne točke pri de Casteljauovem algoritmu podajo dodatno informacijo o krivulji.
- Vemo, da je $\boldsymbol{b}_0^n$ točka na krivulji, obenem pa dobimo še kontrolne točke dveh delov krivulje.

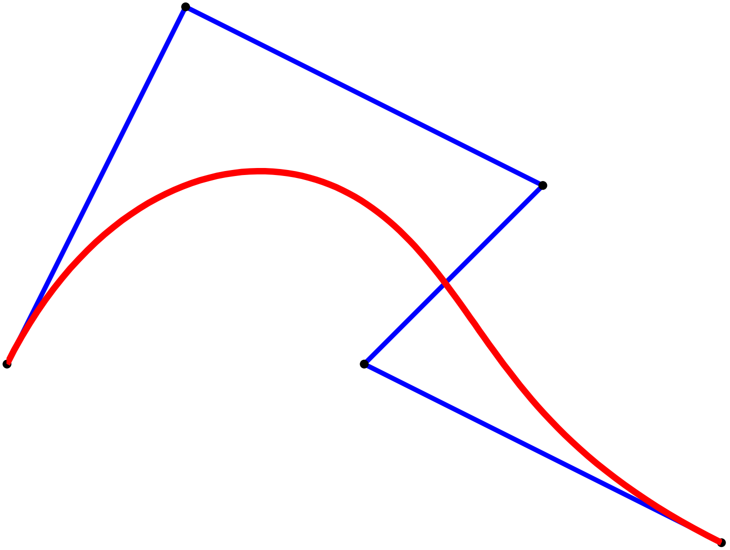
$$
\begin{array}{ccccc}
\mathbf{b}_0^0 & & & & \\
\mathbf{b}_1^0 & \mathbf{b}_0^1 & & & \\
\mathbf{b}_2^0 & \mathbf{b}_1^1 & \ddots & & \\
\vdots & \vdots & \ddots & \mathbf{b}_0^{n-1} & \\
\mathbf{b}_n^0 & \mathbf{b}_{n-1}^1 & \cdots & \mathbf{b}_1^{n-1} & \mathbf{b}_0^n
\end{array}
$$

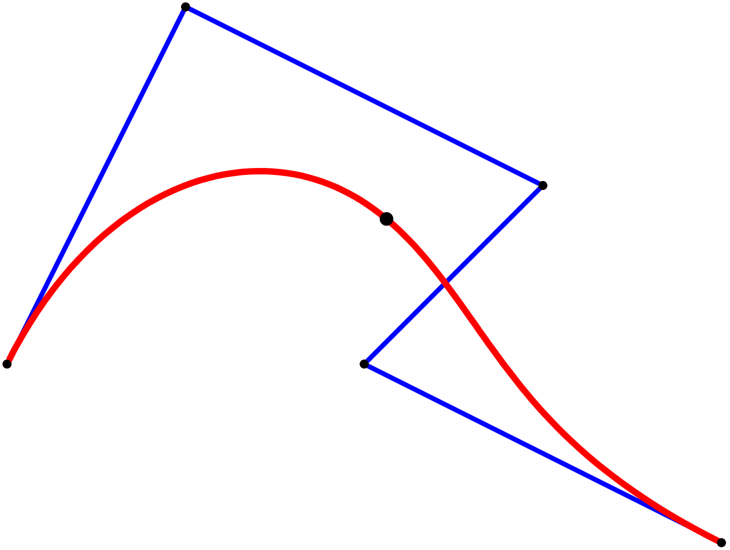
Rdeče točke predstavljajo kontrolne točke dela krivulje na eni strani točke $\mathbf{b}_0^n$, rožnate pa kontrolne točke na drugi strani te točke.

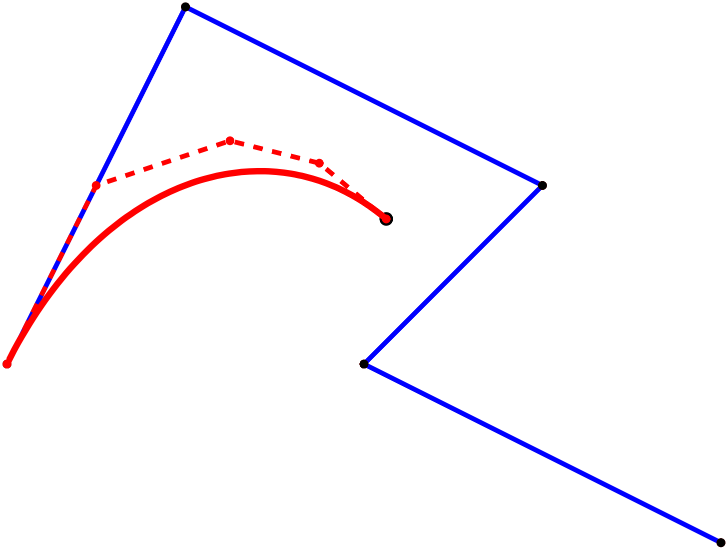
Oglejmo si že znani primer krivulje s kontrolnimi točkami

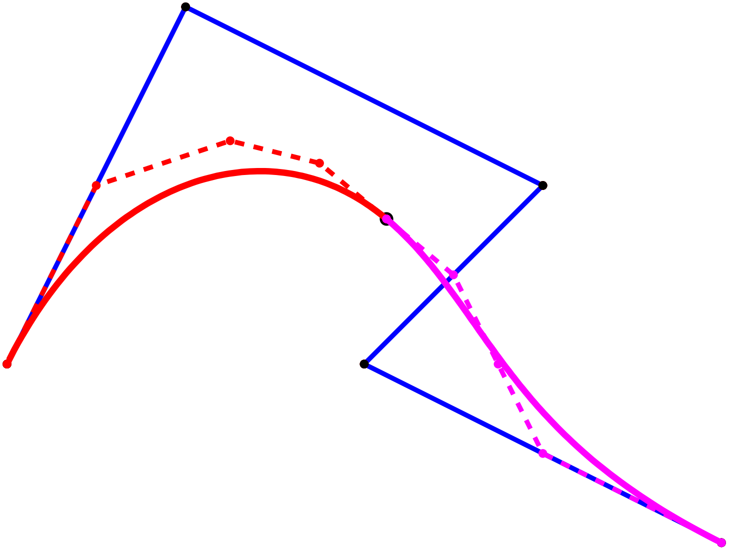
$$
\mathbf{P}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ \mathbf{P}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \ \mathbf{P}_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \ \mathbf{P}_3 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \ \mathbf{P}_4 = \begin{pmatrix} 4 \\ -1 \end{pmatrix}
$$

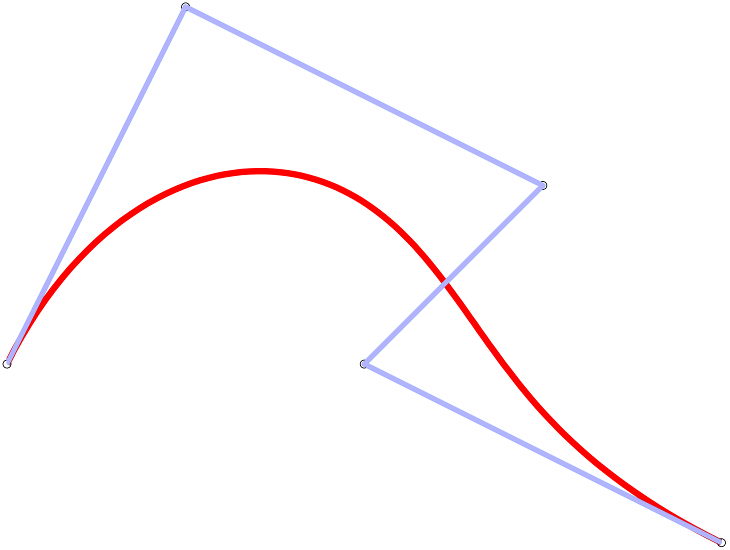
in izberimo $t = 1/2$.

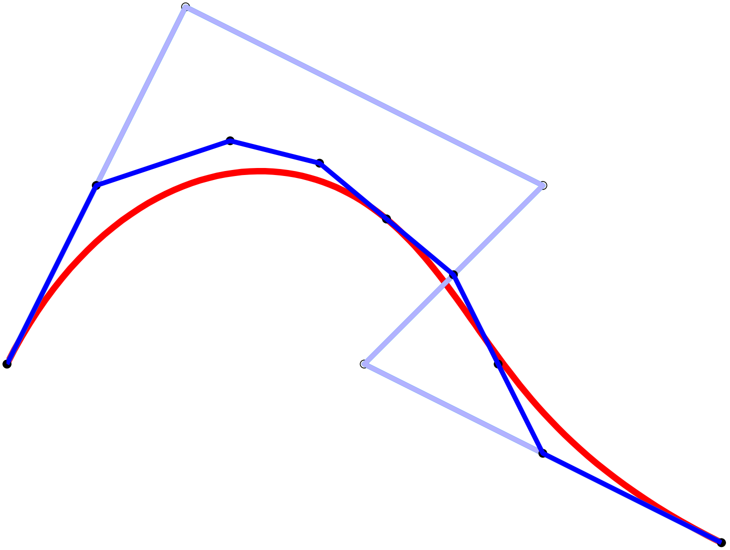


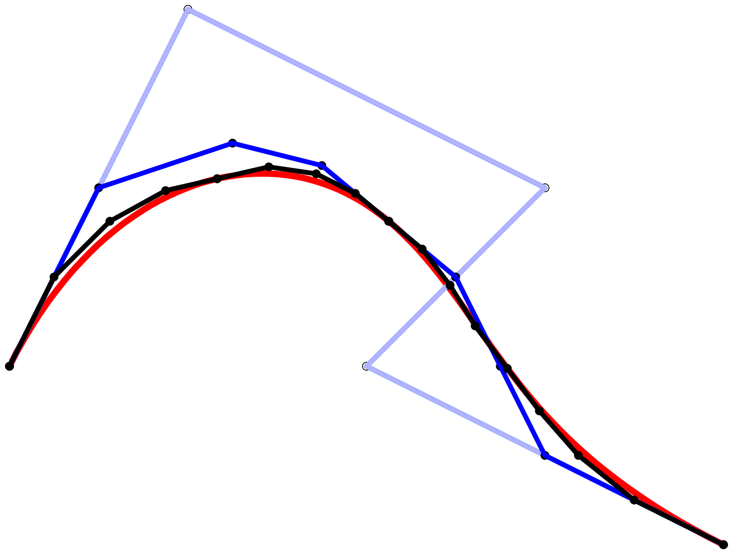












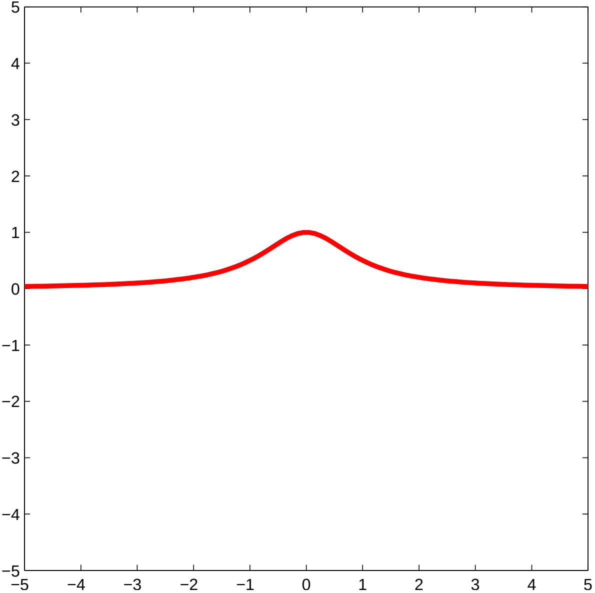
Povzemimo prednosti de Casteljauovega algoritma pred računanjem neposredno po formuli

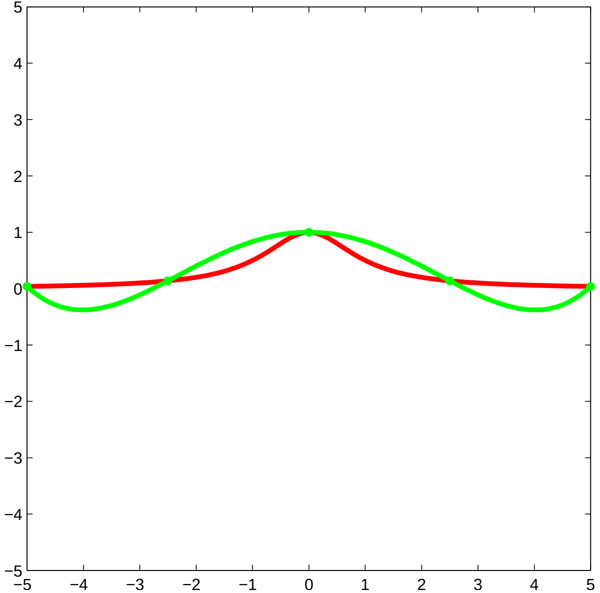
$$\boldsymbol{b}(t) = \sum_{j=0}^{n} \boldsymbol{P}_j \, B_j^n(t), \quad t \in [0,1],$$

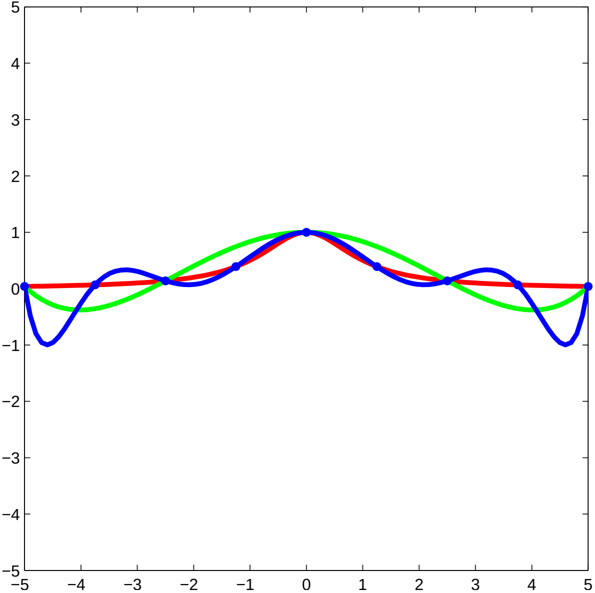
- Hitrejši izračun.
- Enostavna implementacija.
- Stabilnost in manjše numerične napake.
- Možnost izvajanja subdivizije.

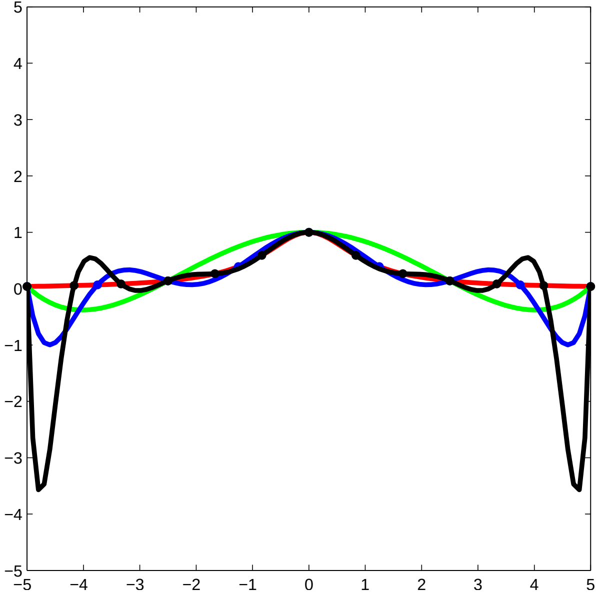
- Včasih želimo s krivuljo “zadeti” (interpolirati) nekaj predpisanih točk.
- To posredno lahko pomeni, da mora biti stopnja krivulje velika.
- Izkaže se, da je to lahko težava.
- Oglejmo si primer interpolacije točk na krivulji

$$\boldsymbol{r}(t) = \begin{pmatrix} t \\ (1+t^2)^{-1} \end{pmatrix}, \quad t \in [-5, 5].$$

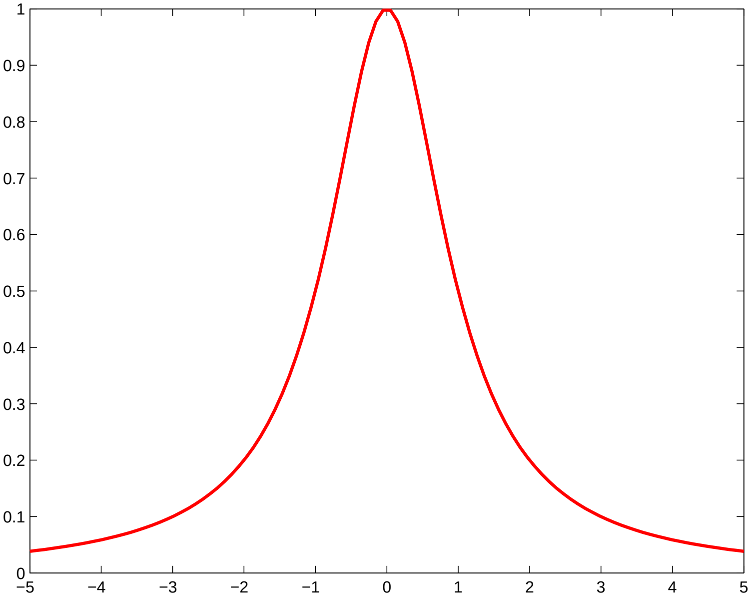

5
4
3
2
1
0
−1
−2
−3
−4
−5
−5
−4
−3
−2
−1
0
1
2
3
4
5

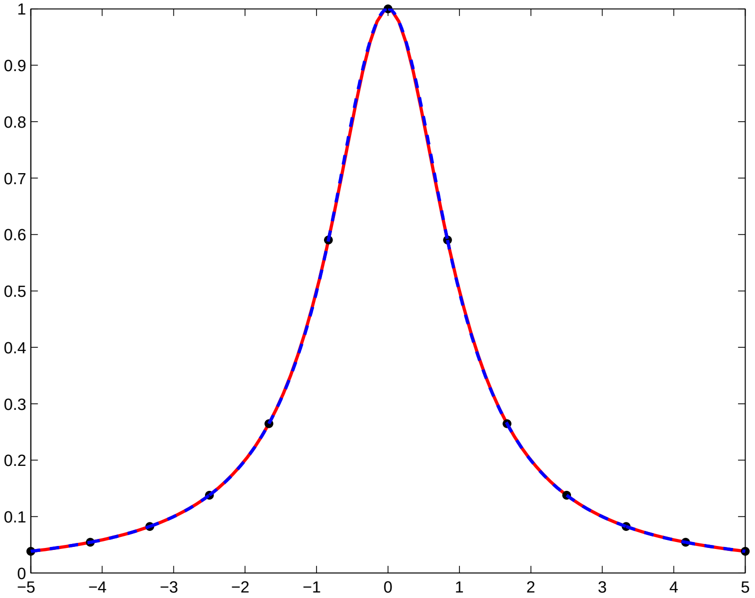

5
4
3
2
1
0
−1
−2
−3
−4
−5
−5
−4
−3
−2
−1
0
1
2
3
4
5


5
4
3
2
1
0
−1
−2
−3
−4
−5
−5
−4
−3
−2
−1
0
1
2
3
4
5


5
4
3
2
1
0
−1
−2
−3
−4
−5
−5
−4
−3
−2
−1
0
1
2
3
4
5

- Višanje stopnje polinomske krivulje ni rešitev.
- Interpolacijske točke radelimo na manjše skupine in jih interpoliramo s krivuljami nižje stopnje.
- Posamezne interpolacijske krivulje zlepimo$\longrightarrow$zlepki.
- Poznamo veliko različnih zlepkov.
- Oglejmo si enega za prejšnji primer podatkov.


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
−5
−4
−3
−2
−1
0
1
2
3
4
5



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
−5
−4
−3
−2
−1
0
1
2
3
4
5

- Obstaja več posplošitev parametričnih krivulj na ploskve.
- Med najbolj preprostimi so tenzorski produkti Bézierovih krivulj.
- Ponovno jih definiramo s kontrolnimi točkami (kontrolno mrežo) in Bernsteinovimi polinomi.

- Formalno jih definiramo takole:

$$\boldsymbol{s}(u,v)=\sum_{j=0}^{m}\sum_{k=0}^{n}\boldsymbol{P}_{j,k}\,B_j^m(u)\,B_k^n(v),\quad u,v\in[0,1],$$

  kjer so $\boldsymbol{P}_{j,k}\in\mathbb{R}^3$ kontrolne točke, ki določajo kontrolno mrežo.
- Zanje veljajo podobne lastnosti kot za Bézierjeve krivulje.
- Še bolj uporabne pa so trikotne krpe.