

Aproksimacija spodnjega dela krožnice s Hermiteovo interpolacijo

*Poročilo o projektni nalogi pri predmetu
Matematično modeliranje*

Urša Kumelj

5. maj 2024

Kazalo

1	Naloga	2
2	Matematično ozadje	2
2.1	Bézierjeve krivulje	2
2.1.1	De Casteljauev algoritem	3
2.1.2	Enostranski Hermiteov interpolant	3
3	Reševanje	4
3.1	Implementacija naloge	4
3.2	Rešitev točke a	5
3.3	Rešitev točke b	5
3.4	Rešitev točke c	6
4	Viri in literatura	6

1 Naloga

Denimo, da potuje kroglica pod vplivom gravitacije (brez trenja) iz točke $\mathbf{T}_1 = \frac{1}{\|(-1, b)\|}(-1, b)$ v točko $\mathbf{T}_2 = \frac{1}{\|(1, -b)\|}(1, -b)$, $b \in \mathbb{R}$, $b > 0$, po kubični Bézierjevi krivulji p , ki aproksimira spodnji del krožnice, katera ima središče v izhodišču in gre skozi \mathbf{T}_1 ter \mathbf{T}_2 . Določite iskani enostranski Hermiteov interpolant \mathbf{p} , tj. kubično Bézierovo krivuljo, ki zadošča Hermiteovim interpolacijskim pogojem in ima parameter $L = \frac{4}{3} \tan(\frac{1}{4}\alpha)$ ter odgovorite na naslednja vprašanja.

- (a) Koliko je kroglica oddaljena od koordinatnega izhodišča pri parametru $t = \frac{1}{2}$?
- (b) Katera je najnižja točka, ki jo doseže kroglica med potovanjem po \mathbf{p} ?
- (c) Kakšna je absolutna vrednost hitrosti kroglice, ki potuje po \mathbf{p} , ko pride v točko \mathbf{T}_2 ?

2 Matematično ozadje

2.1 Bézierjeve krivulje

Bézierjeve krivulje so parametrične krivulje, pomembne v računalniški grafiki. Določene so z zaporedjem kontrolnih točk, ki jih izračunamo s pomočjo De Casteljauevega algoritma. Osredotočili se bomo le na kubične Bézierjeve krivulje. Te so sestavljene iz štirih kontrolnih točk P_0, P_1, P_2, P_3 in imajo obliko

$$b(t) = P_0(1-t)^3 + 3P_1(1-t)^2t + 3P_2(1-t)t^2 + P_3t^3, \quad t \in [0, 1]$$

Pomembne lastnosti Bézierjevih krivulj so:

- (i) Prva in zadnja kontrolna točka sta interpolacijski

$$b(0) = P_0, \quad b(1) = P_3$$

- (ii) Tangentna vektorja v prvi in zadnji točki sta

$$b'(0) = 3(P_1 - P_0), \quad b'(1) = 3(P_3 - P_2)$$

- (iii) Krivulja leži v konvekcijski ovojnici kontrolnih točk.

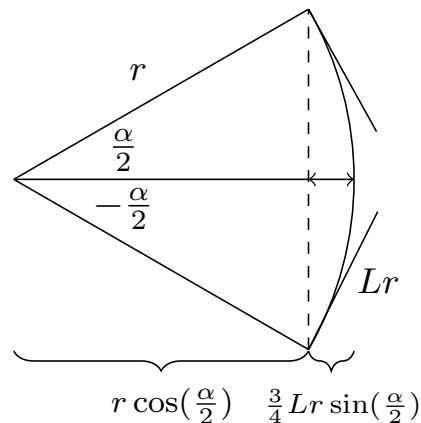
2.1.1 De Casteljaujev algoritem

Osnovna ideja algoritma je rekurzivno deljenje kontrolnih točk, dokler nam ne ostane ena sama točka. Postopek torej vsaki vrednosti parametra $t \in [0, 1]$ učinkovito priredi točko na Bézierjevi krivulji. Implementacija v Matlabu izgleda tako:

```
1 function tocka = deCasteljau(b,t)
2     n = size(b,2);
3     for i = 2:n
4         b = (1-t).*b(:,1:end-1) + t.*b(:,2:end);
5     end
6     tocka = b(:,end);
7 end
```

2.1.2 Enostranski Hermiteov interpolant

Za aproksimacijo krožnih lokov z enostranskim Hermiteovim interpolantom tipično določimo točke na krožnici, ki jih želimo interpolirati, in tangente vektorje v teh točkah, ki so usmerjeni vzdolž tangent na krožnico v teh točkah. Ideja pri tej metodi je potisniti ekstrem v 0. Vrednot parametra L , ki ustreza enostranskemu Hermiteovemu interpolantu, je tako $L = \frac{4}{3} \tan(\frac{1}{4}\alpha)$, $\alpha \in [0, 2\pi]$. Kako dosežemo to vrednost?



Iz slike vidimo, da je razdalja od izhodišča do sredine loka enaka

$$r \cos(\frac{\alpha}{2}) + \frac{3}{4} Lr \sin(\frac{\alpha}{2}).$$

Mi bi želeli dobiti tak parameter L , da bo zgornja razdalja enaka radiju r .

$$\begin{aligned} r &= r \cos\left(\frac{\alpha}{2}\right) + \frac{3}{4} L r \sin\left(\frac{\alpha}{2}\right) \\ L &= \frac{3}{4} \cdot \frac{1 - \cos\left(\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} = \frac{3}{4} \tan\left(\frac{\alpha}{4}\right) \end{aligned}$$

3 Reševanje

3.1 Implementacija naloge

Začnimo z implementacijo same naloge in nato rešimo preostale podnaloge. Potrebujemo aproksimacijo spodnjega dela krožnice iz točke T_1 do T_2 , ki sta normirani, kar pomeni, da ležita na enotski krožnici. Problem bomo rešili tako, da bomo aproksimirali lok krožnice iz točke, ki leži v četrtem kvadrantu do točke, ki leži v prvem kvadrantu. Dobljeno implementacijo bomo potem zarotirali za določen kot, da bomo dobili ravno željeno. Vmesni kot med točkama pa je vedno π .

Najprej potrebujemo določiti kontrolne točke kubične Bézierove krivulje. To-rej za prvo kontrolno točko vzemimo $P_0 = (-\cos(\varphi), -\sin(\varphi))$, ki je točka v četrtem kvadrantu na krožnici in $P_3 = (\cos(\varphi), \sin(\varphi))$ kot zadnjo kontrolno točko, ki je v prvem kvadrantu. Kot φ je polovični kot med njima, zaradi simetrije. Da bo ta krivulja zadoščala enostranskemu Hermiteovemu pogoju, mora biti odvod usmerjena tangenta same krožnice. Tako bosta preostali dve kontrolni točki oblike $P_1 = P_0 + LP'_0$ in $P_2 = P_3 - LP'_3$, kjer je $P'_0 = (\sin(\varphi), -\cos(\varphi))$, $P'_3 = (-\sin(\varphi), \cos(\varphi))$ in $L = \frac{4}{3} \tan(\frac{1}{4}\pi)$, ki določa dolžino tangentnega vektorja v krajiščih.

Sedaj potrebujemo dobljeno le zarotirati. Da dobimo spodnji del krožnice, je potrebna rotacija za kot $-\frac{\pi}{2}$, za pravilno postavitev točk T_1 in T_2 pa lahko kot dobimo na naslednji način. Ker poznamo koordinati točke lahko preprosto s trigonometrijo dobimo

$$\tan(\phi) = \frac{y}{x} = \frac{b}{-1} \Leftrightarrow \phi = \arctan(-b).$$

Implementacija v Matlabu:

```
1 function c = aproksimacija_kroznice(b)
2   fi = pi/2;
3
4   T1 = [-cos(fi); -sin(fi)];
5   T2 = [cos(fi); sin(fi)];
6   dT1 = [sin(fi); -cos(fi)];
```

```

7   dT2 = [-sin(fi); cos(fi)];
8
9   L = 4/3*tan(pi/4);
10
11  c = [T1, T1+L*dT1, T2-L*dT2, T2];
12
13  kot_rotacije = atan(-b)-pi/2;
14  M = [cos(kot_rotacije), -sin(kot_rotacije); sin(
      kot_rotacije), cos(kot_rotacije)];
15  c = M*c;
16  plotBezier(c);
17  axis equal;
18  end

```

V implementaciji je prav tako uporabljena funkcija `plotBezier`, ki nam vse skupaj izriše.

3.2 Rešitev točke a

Glede na to, kako je izračunan parameter L , bo razdalja pri $t = \frac{1}{2}$ vedno enaka radiju krožnice, torej 1, saj smo na enotski krožnici. Rezultat pa lahko seveda preverimo v Matlabu. S pomočjo `deCasteljau` izračunamo točke na krivulji pri parametru t .

```

1   b = 1;
2   t = 1/2;
3   kontrolne_tocke = aproksimacija_kroznice(b);
4   tocka = deCasteljau(kontrolne_tocke, t);
5   razdalja = sqrt(tocka(1)^2 + tocka(2)^2)

```

Seveda parameter b poljubno spreminjamo.

3.3 Rešitev točke b

Najnižja točka, ki jo doseže kroglica med potovanjem po \mathbf{p} izračunamo s pomočjo vgrajene funkcije `fminsearch`. Glede na to, da točke na Bézierjevi krivulji pridobivamo z De Casteljaujevim algoritmom, kjer parameter t točno določa posamezno točko, je potrebno poiskati, pri katerem t je dosežen minimum. Za iskanje minimuma pa so za nas važne le y komponente kontrolnih točk, zato sem uporabila funkcijo `deCast`, ki deluje le na vektorju kontrolnih točk b velikosti $(n + 1)$. Na koncu je potrebno dobljeni t vstaviti v De Casteljajev algoritem, da dobimo konkretno točko minimuma in jo na sliko tudi narišemo z ukazom `scatter`.

```

1 function tocke = deCast(b,t)
2     [~,n] = size(t);
3     [~,m] = size(b);
4     tocke = zeros(1,n);
5     b_nov = b;
6     for i=1:n
7         b_nov = b;
8         for j=1:m-1
9             [~,l] = size(b_nov);
10            b_nov = b_nov(1:l-1).*(1-t(i)) + b_nov(2:l).*t(i);
11        end
12    tocke(i) = b_nov;
13 end

1 kontrolne_tocke_y = kontrolne_tocke(2,:);
2 funkcija = @(t) deCast(kontrolne_tocke_y,t);
3 iskani_t = fminsearch(funkcija,0.7);
4 minimum = deCasteljau(kontrolne_tocke, iskani_t)
5 scatter(minimum(1), minimum(2), 10, 'filled', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g')

```

3.4 Rešitev točke c

Za izračun hitrosti velja enačba $v = \sqrt{2g(-y)}$, ki velja zaradi zakona o ohranitvi energije. Kot pri prejšnji točki tudi tukaj že iz enačbe vidimo, da potrebujemo zgolj y komponente točk na krivulji, zato bomo uporabili `deCast`. Vse y kontrolne točke sedaj enostavno vstavimo v enačbo za hitrost vse in jo izračunamo za vsak $t \in [0,1]$. Na koncu vrnemo le absolutno vrednost zadnjega elementa vektorja, saj je pri $t = 1$ dosežena točka T_2 .

```

1 g = 9.81;
2 t = linspace(0,1);
3 y_tocke_krivulje = deCast(kontrolne_tocke,t);
4 v = abs(sqrt(2*g*(-y_tocke_krivulje(end))))

```

4 Viri in literatura

- Zapiski s predvanj in vaj pri predmetu Matematično modeliranje, vse je dostopno na letošnji spletni učilnici kot tudi na učilnicah preteklih let.

- Wikipedia o Bézierjevih krivuljah, dostopno na Wikipedia.
- Calculate control points of cubic bezier curve approximating a part of a circle, dostopno na StackExchange.
- Git repozitorij Nika Erzetica, dostopno na GitHub.