Aproksimacija spodnjega dela krožnice s Hermiteovo interpolacijo

Poročilo o projektni nalogi pri predmetu Matematično modeliranje

Urša Kumelj

25. februar 2024

Kazalo

1 Naloga **2**

2 Matematično ozadje **2**

2.1 Bézierjeve krivulje 2

2.1.1 De Casteljaujev algoritem 3

2.1.2 Enostranski Hermiteov interpolant 3

3 Reševanje **3**

3.1 Implementacija naloge 3

3.2 Rešitev točke a 4

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 $\boldsymbol{T_1} = \frac{1}{\|(-1,b)\|}(-1, b)$ v točko $\boldsymbol{T_2} = \frac{1}{\|(1,-b)\|}(1, -b)$, $b \in \mathbb{R}$, $b > 0$, po kubični Bézierjevi krivulji p, ki aproksimira spodnji del kronžice, katera ima središče v izhodišču in gre skozi $\boldsymbol{T_1}$ ter $\boldsymbol{T_2}$. Določite iskani enostranski Hermiteov interpolant $\boldsymbol{p}$, 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 $\boldsymbol{p}$?

(c) Kakšna je absolutna vrednost hitrosti kroglice, ki potuje po $\boldsymbol{p}$, ko pride v točko $\boldsymbol{T_2}$?

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 Casteljaujevega algoritma. Osredotočili se bomo le na kubične Bézierjeve krivulje. Ta je sestavljena iz štirih kontrolnih točk P_0, P_1, P_2, P_3 in ima 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 lastonosti 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

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

(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 točki. Postopek torej vsaki vrednosti parametra $t \in [0,1]$ učinkovito priredi točko na Bézierjevi krivulji. Implementacija v Matlabu izgleda tako:

```
function tocka = deCasteljau(b,t)
  n = size(b,2);
  for i = 2:n
  b = (1-t).*b(:,1:end-1) + t.*b(:,2:end);
  end
  tocka = b(:,end);
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 tangentne vektorje v teh točkah, ki so usmerjeni vzdolž tangent na krožnico v teh točkah. Vrednot parametra L, ki ustreza enostranskemu Hermiteovemu interpolantu je $L = \frac{4}{3}\tan(\frac{1}{4}\alpha)$. Kako dosežemo to vrednost?

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. Torej 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 potrebno zarotirati 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:

```
function c = aproksimacija_kroznice(b)
 fi = pi/2;

 T1 = [-cos(fi); -sin(fi)];
 T2 = [cos(fi); sin(fi)];
 dT1 = [sin(fi); -cos(fi)];
 dT2 = [-sin(fi); cos(fi)];

 L = 4/3*tan(pi/4);

 c = [T1,T1+L*dT1,T2-L*dT2,T2];

 kot_rotacije = atan(-b)-pi/2;
 M = [cos(kot_rotacije),-sin(kot_rotacije);sin(
    kot_rotacije),cos(kot_rotacije)];
 c = M*c;
 plotBezier(c);
 axis equal;
end
```

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

3.2 Rešitev točke a

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

```
 b = 1;
 t = 1/2;
```

```
kontrolne_tocke = aproksimacija_kroznice(b);
tocka = deCasteljau(kontrolne_tocke,t);
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 $\boldsymbol{p}$ 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`.

```
function tocke = deCast(b,t)
 [~,n] = size(t);
 [~,m] = size(b);
 tocke = zeros(1,n);
 b_nov = b;
 for i=1:n
   b_nov = b;
   for j=1:m-1
    [~,l] = size(b_nov);
    b_nov = b_nov(1:l-1).*(1-t(i)) + b_nov(2:l).*t(i);
   end
 tocke(i) = b_nov;
end
```

```
kontrolne_tocke_y = kontrolne_tocke(2,:);
funkcija = @(t) deCast(kontrolne_tocke_y,t);
iskani_t = fminsearch(funkcija,0.7);
minimum = deCasteljau(kontrolne_tocke, iskani_t)
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 pride 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`. Sedaj enostavno vstavimo v enačbo za hitrost vse y kontrolne točke ter 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.

```
g = 9.81;
t = linspace(0,1);
y_tocke_krivulje = deCast(kontrolne_tocke,t);
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.