ALGORITEM ZA DOLOČITEV POVEZOVALNIH FUNKCIJ KRIVULJ B-ZLEPKOV IN NURB KRIVULJ V POLINOMSKEM ČASU

INFORMATICA 3/91

Keywords: B-spline curves, NURB curves, blending functions, de Boor algorithm, time complexity, computer graphics

Borut Žalik, Nikola Guid,
Andrej Tibaut in Aleksander Vesel
Tehniška fakulteta Maribor

Povzetek: Povezovalne funkcije krivulj B-zlepkov in NURB krivulj so definirane z rekurzivno formulo, čigar direktna vgradnja v algoritem pripelje do eksponentne časovne zahtevnosti izračunavanja povezovalnih funkcij glede na njihov red. S pravilno pripravo podatkovnih struktur in upoštevanjem lastnosti povezovalnih funkcij lahko sestavimo algoritem, ki izračuna povezovalne funkcije in nariše krivuljo v polinomskem času. Izdelan algoritem smo primerjali glede na porabljen čas CPU z iterativnim de Boorovim algoritmom, s katerim določimo točke na krivulji brez izračunavanja povezovalnih funkcij. Ugotovimo, da je naš algoritem učinkovitejši celo od de Boorovega algoritma v primeru periodičnih in neperiodičnih krivulj B-zlepkov, v primeru NURB krivulj pa je slabši.

An algorithm for B-spline and NURB blending functions determination in a polynomial time. B-spline and NURB blending functions are defined by recursive formula. Its direct implementation into a program lead us to the exponential time complexity regarding to the degree of blending functions. With paying attention to a correct data structure we have been able to construct an algorithm which evaluates the blending functions and plots the curve in a polynomial time. This algorithm has been compared with iterative de Boor algorithm which determines the points on a curve without blending function calculation. The values of the CPU time spend on curve generation show, that our algorithm is better in the case of periodical and nonperiodical B-spline curves, and in the case of NURB curves it is worse than de Boor algorithm.

1. UVOD

Pri realizaciji algoritmov za izračun in izris krivulj B-zlepkov se srečamo s problemom izračunavanja povezovalnih ali baznih funkcij. Za hiter izris krivulj B-zlepkov in NURB krivulj sicer obstaja iterativni de Boorov algoritem, kjer povezovalnih funkcij sploh ni potrebno izračunavati, vendar je mnogokrat zelo ugodno, če lahko prikažemo tudi povezovalne funkcije, predvsem zaradi učenja in raziskovanja. Seveda je najugodneje, če so povezovalne funkcije izračunane analitično in zakodirane v algoritmu (bazne funkcije so polinomi ustreznega reda). Periodične bazne funkcije nižjih redov (2 ali 3) lahko hitro izpeljemo, najdemo pa jih tudi v literaturi. V primeru uporabe neperiodičnih baznih funkcij moramo vložiti že več napora, da izpeljemo vse potrebne povezovalne funkcije. Le-te že redkeje najdemo (npr. v [TURK80] in [YAMA88]). Določitev koeficientov povezovalnih funkcij višjih redov pa zahteva precej truda.

2. DEFINICIJA KRIVULJ B-ZLEPKOV

Definicijo in lastnosti krivulj B-zlepkov najdemo na mnogih mestih v literaturi ([BÖHM84], [MORT85], [BART87], [GUID88], [PIEG87], [YAMA88]). V nadaljevanju bomo uporabljali oznake in zaključke iz vira [GUID90]. Krivulja B-zlepkov je definirana kot:

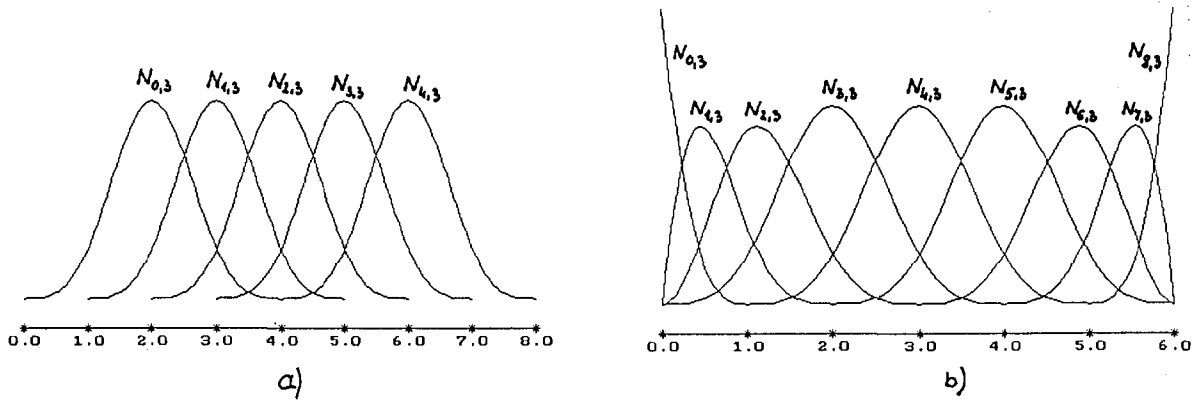
$$p(u) = \sum_{i=0}^{n} r_i \, N_{i,k}(u) \qquad (1)$$

kjer so r_i kontrolne točke in $N_{i,k}$ bazne ali povezovalne funkcije reda k, ki so definirane z rekurzivno formulo:

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

in

$$N_{i,k}(u) = \frac{(u - u_i)\, N_{i,k-1}(u)}{u_{i+k} - u_i} + \frac{(u_{i+k+1} - u)\, N_{i+1,k-1}(u)}{u_{i+k+1} - u_{i+1}} \qquad (2)$$



Slika 1 Kubične bazne funkcije, ki jih določa vozliščni vektor
a) $U_{knot} = [0, 1, 2, 3, 4, 5, 6, 7, 8]$
b) $U_{knot} = [0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6]$

Množico u_i shranimo v vozliščni vektor $U_{knot} = [u_0, u_1, u_2, \ldots, u_m]$. Obravnavali bomo dve obliki vozliščnega vektorja:

a) **Vozliščni vektor, ki določa nesklenjene periodične krivulje B-zlepkov.** Za nas so zanimivi le tisti odseki, kjer sodeluje k+1 povezovalnih funkcij, saj je krivulja B-zlepkov definirana v intervalu $u \in [u_0 + k, u_m - k]$. Definiran je kot [GUID90]:

$$U_{knot} = [u_0, u_1, u_2, \ldots, u_m] = [0, 1, 2, \ldots, m]$$

m je določen s formulo:

$$m = n + k + 1 \qquad (3)$$

kjer je:

n: število kontrolnih točk minus 1

k: red povezovalnih funkcij.

Bazne funkcije za n = 4 in k = 3 nam kaže slika 1a.

b) **Vozliščni vektor, ki določa neperiodične krivulje B-zlepkov.** Definiran je kot [GUID90]:

$$U_{knot} = [u_0, u_1, u_2, \ldots, u_m]$$

$u_i = 0$; če $i \le k$

$u_i = i - k$; če $k+1 \le i \le m - k - 1$

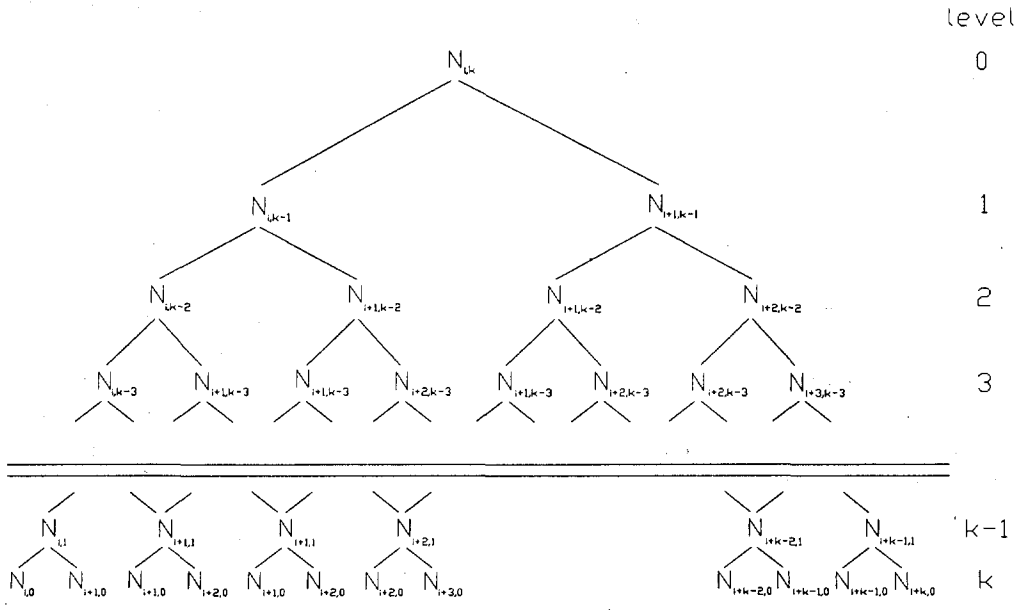
$u_i = m - 2k$; če $m - k \le i \le m$

kjer je m definiran z enačbo 3. Bazne funkcije za n = 8 in k = 3 nam kaže slika 1b.

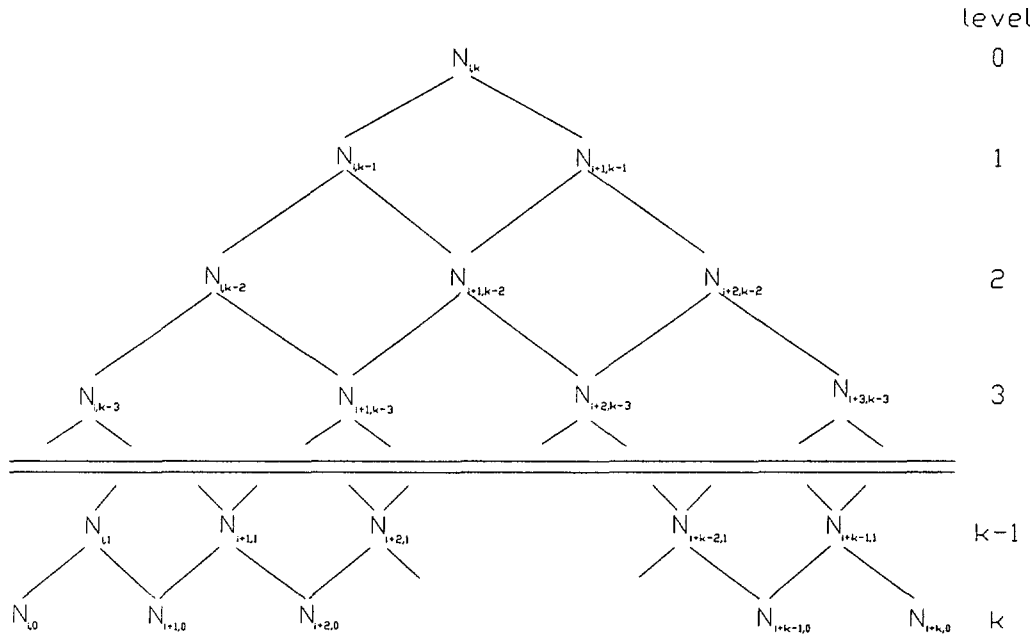
3. PRIPRAVA PODATKOVNE STRUKTURE ZA DOLOČITEV POVEZOVALNIH FUNKCIJ

Potek izračunavanja povezovalnih funkcij z algoritmom, ki neposredno rešuje enačbo 2, nam kaže slika 2. Rekurzivni klici nam ustvarijo polno dvojiško drevo. Število vozlišč v takem drevesu je določeno z enačbo

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1. \qquad (4)$$



Slika 2 Drevo rekurzivnih klicev funkcije $N_{i,k}$



Slika 3 Modificirana struktura za izračun funkcije $N_{l,k}$

2^k vrednosti povezovalnih funkcij na globini k določimo brez računanja (enačba 2), vse preostale $2^k - 1$ vrednosti povezovalnih funkcij pa moramo izračunati, kar vodi k eksponencialni časovni zahtevnosti

$$T(k) = 2^k - 1 = O(2^k). \qquad (5)$$

Če pa boljše pogledamo sliko 2, vidimo, da se nekatere povezovalne funkcije, shranjene v vozliščih drevesa, ponavljajo. Z združevanjem vozlišč, ki hranijo iste povezovalne funkcije, dobimo novo strukturo, ki jo vidimo na sliki 3. Število vozlišč v taki strukturi je določeno z enačbo

$$\sum_{i=0}^{k} (1 + i) = \frac{1}{2} (k^2 + 3k + 2). \qquad (6)$$

Tudi tokrat k + 1 začetnih vrednosti povezovalnih funkcij na globini k določimo brez računanja, tako da nam ostane le $\frac{1}{2}$ k(k + 1) izračunov, ki pa jih lahko opravimo v polinomskem času:

$$T(k) = \frac{1}{2} k(k+1) = O(k^2) \qquad (7)$$

Da bi lahko določili vrednost povezovalne funkcije glede na izbiro parametra u v polinomskem času, moramo pripraviti ustrezno podatkovno strukturo. Funkcijo na globini j ($0 \le j < k$) izračunamo s pomočjo dveh funkcij iz globine j+1. Ugodno bi bilo našo shemo s slike 3 obrniti tako, da bodo prej dostopna vozlišča z večjo globino. Podatkovno strukturo tvorimo kot seznam enosmerno povezanih seznamov, kjer imamo na vrhu k+1 vozlišč s povezovalnimi funkcijami reda 0 in na dnu samo eno vozlišče reda k. Strukturo deklariramo na naslednji način:

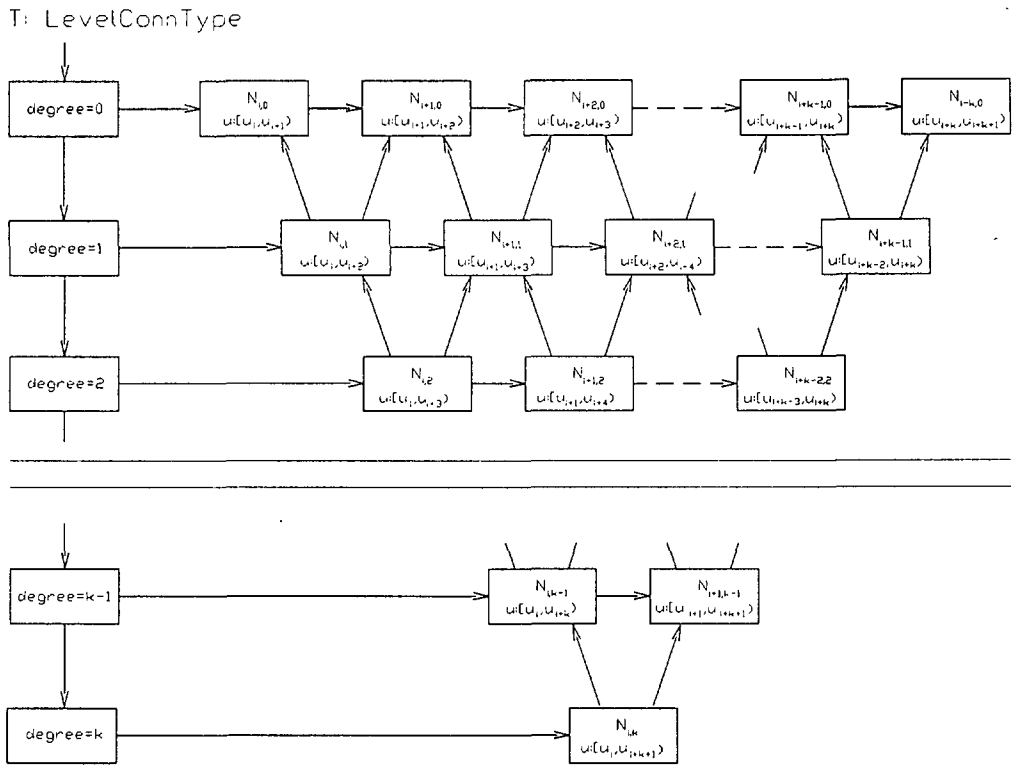
```
TreePointer = ^TreeType;
TreeType = record
   Ni: integer;                  {indeks povezovalne funkcije}
   LeftValue,                    {spodnja meja določena z Uknot}
   RightValue,                   {zgornja meja določena z Uknot}
   LDenominator,                 {imenovalec levega izraza}
   RDenominator,                 {imenovalec desnega izraza}
   Nu: real;  {vrednost povezovalne funkcije pri izbranem u}
   PermanentLL, {zastavica na globini 0; če TRUE,potem Nu=0}
   Permanent: Boolean;           {če TRUE, izračun Nu ni potreben}
   Lson, Rson,                   {kazalca na zapise višje v strukturi}
   list: TreePointer;            {kazalec na naslednji seznam}
end; {record TreeType}

LevelConnType = ^ListOfPointers;
ListOfPointers = record
   ToLevel: TreePointer;         {kazalec na seznam TreePointer}
   Nk: Integer;                  {stopnja povezovalne funkcije}
   NextLevel: LevelConnType;     {kazalec na naslednji zapis}
end; {record ListOfPointers}
```

Slika 4 nam kaže uporabljeno podatkovno strukturo.

Slika 4 Podatkovna struktura za izračun povezovalnih funkcij v polinomskem času

Inicializacija podatkovne strukture

S tako pripravljeno podatkovno strukturo, ki jo vidimo na sliki 4, lahko izračunamo poljubno povezovalno funkcijo glede na postavljeni vozliščni vektor, le prave podatke moramo vpisati v posamezne zapise. Podatkovno strukturo začnemo polniti pri zapisih, ki hranijo povezovalne funkcije stopnje 0 in so na vrhu seznama seznamov. Najprej za vsako povezovalno funkcijo na globini 0 preverimo mejni vrednosti iz vozliščnega vektorja in s tem določimo interval, v katerem je posamezna povezovalna funkcija definirana. Če sta obe vrednosti enaki, postavimo zastavico PermanentLL na TRUE, vrednost povezovalne funkcije shranjene v komponenti Nu pa je 0. V primeru, ko se vrednosti LeftValue in



Slika 4 Podatkovna struktura za izračun povezovalnih funkcij v polinomskem času

RightValue razlikujeta, postavimo zastavico PermanentLL v FALSE, Nu dobi vrednost 1, vrednosti iz vozliščnega vektorja vpišemo v LeftValue ter RightValue in določimo konstanti v imenovalcih, ki ju shranimo v komponenti zapisa LDenominator in RDenominator. Nato napolnimo še preostale zapise v podatkovni strukturi s slike 4 tako, da se spuščamo po seznamu tipa LevelConnType. Če imata oba zapisa na višjem nivoju v podatkovni strukturi zastavico PermanentLL TRUE, dobi tudi zastavica pravkar obravnavanega zapisa vrednost TRUE, Nu pa vrednost 0, sicer pa postavimo zastavico PermanentLL v FALSE. V tem primeru določimo interval, v katerem ta povezovalna funkcija deluje, kot unijo dveh sosednjih intervalov z nižjega nivoja dostopnih preko kazalcev Lson in Rson, nato pa z njihovo pomočjo določimo še imenovalca.

Določitev vrednosti povezovalne funkcije pri izbranem parametru

Pri izbrani vrednosti parametra u v večini primerov sodeluje pri izračunu funkcije $N_{1,k}$ manjše število povezovalnih funkcij, kot smo jih označili z zastavico PermanentLL. Tako npr. na globini 0 sodeluje kvečjemu le ena povezovalna funkcija, ki ima po enačbi 2 vrednost 1. Zato v vsakem zapisu tipa TreeType uvedemo še zastavico Permanent, ki ima vrednost TRUE pri vseh funkcijah $N_{1,k}$, kjer velja $u \notin$ (LeftValue, RightValue). Seveda dobi zastavica Permanent takoj vrednost TRUE, če ima takšno vrednost tudi PermanentLL. Zastavice na globinah večjih kot 0 določimo v primeru, ko ima zastavica PermanentLL vrednost FALSE, z naslednjim logičnim izrazom

Permanet := Lson^.Permanent and Rson^.Permanent.

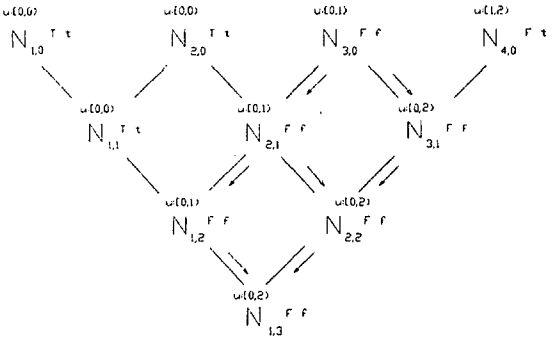
Postavitev zastavic spremenimo, ko tekoči parameter u spremeni interval v vozliščnem vektorju U_{knot}. Izračun vrednosti povezovalnih funkcij pri parametru u sedaj opravimo tako, da od globine 1 do k izračunamo le povezovalne funkcije v tistih zapisih, ki imajo postavljeno zastavico Permanent na vrednost FALSE.

Na sliki 5 vidimo primer, ko želimo izračunati neperiodično povezovalno funkcijo $N_{1,3}$ s slike 1b. Če je $u \in [0,1)$, moramo opraviti 5 izračunov (vrednost funkcije reda 0 je določena že vnaprej in je ne upoštevamo). Potek izračuna vidimo na sliki 5, če sledimo puščicam. Vrednost zastavice PermanentLL smo označili z velikima črkama T in F, vrednost zastavice Permanent pa z malima črkama t in f. Če je $u \in [1,2)$, bi morali opraviti le tri izračune. Na sliki 6 vidimo algoritem, ki uporablja opisano podatkovno strukturo, za izračun vrednosti povezovalne funkcije.

4. PRIPRAVA PODATKOVNE STRUKTURE IN ALGORITEM ZA IZRIS KRIVULJ B-ZLEPKOV

Priprava podatkovne strukture

Vsako povezovalno funkcijo bomo izračunali v diskretnih točkah. Naj bo na intervalu $[u_l, u_{l+1})$ dovolj NoOfSteps izračunov. Ker se vsaka povezovalna funkcija razteza največ na k+1 intervalih (slika 1a in 1b), bo za posamezno povezovalno funkcijo potrebnih največ (k+1) * NoOfSteps izračunov [ŽALI90]. Za vsako bazno funkcijo si shranimo še dejansko število izračunanih vrednosti. Vse podatke shranimo v zapis BfType.



Slika 5 Potek izračuna povezovalne funkcije $N_{1,3}$

```
BfType = record
           Bf: array [0..UValuesMax] of real;
           NoOfUvalues: integer;
end

UValuesMax ≥ (k+1) * NoOfSteps
```

Da bi lahko direktno uporabili enačbo 1 za določitev krivulj B-zlepkov, bomo vse funkcije $N_{i,k}$ predstavili kot kazalce na zapise tipa BfType in jih shranili v polje tipa AllBFunctionType:

```
AllBfunctionType = array [0..NoOfBf] of ^BfType.
```

Izračun in izris periodičnih krivulj B-zlepkov

Potrebujemo n+1 enakih periodičnih povezovalnih funkcij, ki so med seboj le premaknjene (slika 1a). Zato je dovolj, če izračunamo le eno povezovalno funkcijo in zanjo rezerviramo prostor, za vse ostale n bazne funkcije pa postavimo le kazalce na zapis tipa BfType. Potrebno podatkovno strukturo vidimo na sliki 7a.

Časovna in prostorska zahtevnost določitve povezovalnih funkcij sta neodvisni od števila kontrolnih točk, tako da lahko zapišemo:

$$T(n) = 1$$
$$S(n) = 1 \qquad (8)$$

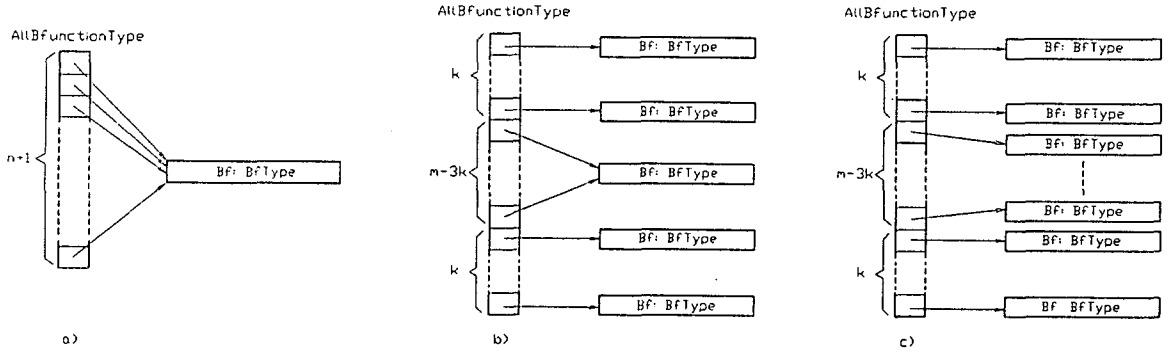
Funkcijo za izračun baznih funkcij pokličemo le ((k+1)NoOfSteps) - krat.

```
function ModifiedNf(t: LevelConnType; u: real): real;
{ Funkcija izračuna vrednost povezovalne funkcije krivulje
  B-zlepkov.
  - t: kazalec na podatkovno strukturo,
  - u: vrednost parametra, pri kateri bomo izračunali
                                        povezovalno funkcijo }
  var
    bufferNu: real;
    q: TreePointer;

begin (* ModifiedNF *)
  t := t^. NextLevel;
  while t <> nil do
  begin
    q := t^.ToLevel;
    while q <> nil do
      with q^ do
      begin
        if not permanet then
          Nu := LeftDenominator * (u - LeftValue) * Lson^.Nu +
                RightDenominator * (RightValue - u)* Rson^.Nu;
        end;
        BufferNu := Nu;
        q := q^.list;
    end;
    t := t^.NextLevel;
  end;
  ModifiedNf := BufferNu;
end; { ModifiedNF }
```

Slika 6 Algoritem za izračun vrednosti povezovalne funkcije

Odsek krivulje B-zlepkov narišemo tako, da upoštevamo le tiste povezovalne funkcije, ki na tem odseku nastopajo in jih zmnožimo s pripadajočimi kontrolnimi točkami. V začetku algoritma za izris krivulj B-zlepkov določimo, katere povezovalne funkcije sodelujejo pri odseku, ki ga trenutno rišemo (slika 8). Prvo in zadnjo povezovalno funkcijo, ki hkrati določata tudi indeks ustrezne kontrolne točke, shranimo v spremenjivki firstBf in lastBf. V drugem delu algoritma določimo še indeks diskretne vrednosti vsake povezovalne funkcije, shranjene v zapisu tipa BfType, ki jo pri izračunu točk na krivulji po enačbi 1 potrebujemo.

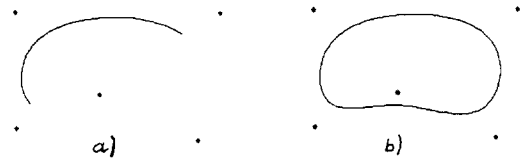
Nesklenjeno periodično krivuljo B-zlepkov, čigar povezovalne funkcije so na sliki 1a, vidimo na sliki 9a. Z majhno spremembo pri pripravi podatkov lahko s tem algoritmom narišemo tudi sklenjeno krivuljo B-zlepkov, ki jo vidimo na sliki 9b.



Slika 7 Podatkovna struktura baznih funkcij za:
a) periodične krivulje B-zlepkov
b) neperiodične krivulje B-zlepkov
c) NURB krivulje

```
Procedure PlotBSpl(k,m: integer; periodic: boolean;
                   Pts: AllBfunctionType; rx,ry: PointsType);
{  - k: red povezovalnih funkcij,
   - m: število vozliščnih vrednosti,
   - periodic: TRUE, če rišemo periodične krivulje B-zlepkov
   - Pts: polje kazalcev na povezovalne funkcije,
   - rx,ry: polje kontrolnih točk.
var
  x, y: PolyLineType;
   j, ui, ni, ind, firstBf, lastBf: integer;
  a: real;
begin
  firstBf := -1;                    lastBf := k - 1;
  for ui := 0 to m-2*k-1 do
  begin
    firstBf := firstBf + 1;         lastBf := lastBf + 1;
    for j := 0 to NoOfSteps do
    begin
      x[j] := 0;     y[j] := 0;
      for ni := firstBf to lastBf do
      begin
        if periodic then
          ind := (lastBf - ni) * NoOfSteps + j
        else
          if ni <= k then
            ind := (lastBf - k) * NoOfSteps + j
          else
            ind := (lastBf - ni) * NoOfSteps + j;
        a  :=  Pts[ni]^.Bf[ind];
        x[j] := x[j] + rx[ni] * a;
        y[j] := y[j] + ry[ni] * a;
      end;
    end;
    gPolyLine(NoOfSteps+1, x, y);
  end;
end; { PlotBspl }
```

Slika 8 Algoritem za izris krivulj B-zlepkov



Slika 9 a) Nesklenjena periodična krivulja B-zlepkov
b) Sklenjena periodična krivulja B-zlepkov

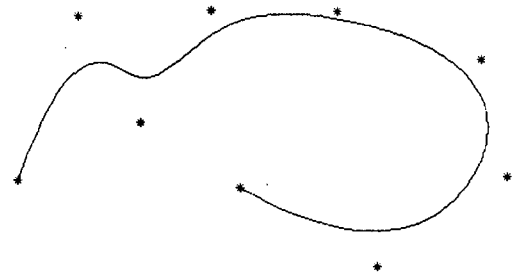
Izračun in izris neperiodičnih krivulj B-zlepkov

V primeru neperiodičnih krivulj B-zlepkov je število povezovalnih funkcij, ki jih moramo izračunati, večje. V najslabšem primeru moramo določiti k+1 povezovalnih funkcij, od katerih jih je k neperiodičnih in ena periodična. Na primeru s slike 1b, ko je k = 3 in n = 8 vidimo, da je prvih k baznih funkcij ($N_{0,3}$, $N_{1,3}$, $N_{2,3}$) zrcalnih zadnjim k baznim funkcijam ($N_{7,3}$, $N_{6,3}$, $N_{5,3}$). Zato je dovolj, če izračunamo le prve, druge pa dobimo z enostavno zamenjavo mest koeficientov v polju. Med neperiodičnimi baznimi funkcijami se pojavijo periodične povezovalne funkcije, če je n ≥ 2k [GUID90]. Kot smo že videli, pa te zahtevajo samo en izračun.

Tudi tokrat je prostorska in časovna zahtevnost določitve povezovalnih funkcij neodvisna od števila kontrolnih točk, tako da velja enačba 8. Funkcijo za izračun baznih funkcij v najslabšem primeru pokličemo

$$1*NoOfSteps + 2*NoOfSteps + ... + (k+1)*NoOfSteps =$$
$$= 0.5 * (k^2 + 3k + 2) * NoOfSteps.$$

Potrebno podatkovno strukturo vidimo na sliki 7b, ki jo uporablja algoritem za izris krivulje B-zlepkov (slika 8). Na sliki 10 vidimo primer neperiodične krivulje B-zlepkov, katere povezovalne funkcije so na sliki 1b.



Slika 10 Neperiodična krivulja B-zlepkov

5. DEFINICIJA NURB KRIVULJ

NURB krivulje ("nonuniform rational B-spline curves") so s svojo univerzalnostjo pritegnile mnogo raziskovalcev ([WAYN83], [BÖHM84], [PIEG87], [PIEG89], [ROGE90]). Z njimi lahko eksaktno predstavimo tako daljice, stožnice, kot poljubno oblikovane krivulje. NURB krivulje B-zlepkov uporabljajo tudi nekateri komercialni modelirni sistemi. Definirane so kot:

$$p(u) = \frac{\sum_{i=0}^{n} N_{i,k}(u)\, w_i\, r_i}{\sum_{j=0}^{n} N_{j,k}(u)\, w_j} = \sum_{i=0}^{n} R_{i,k}(u)\, r_i \qquad (9)$$

kjer so r_i kontrolne točke in $R_{i,k}$ racionalne povezovalne funkcije, ki so definirane s kvocientom:

$$R_{i,k}(u) = \frac{N_{i,k}(u)\, w_i}{\sum_{j=0}^{n} N_{j,k}(u)\, w_j} \qquad (10)$$

V enačbi 10 nastopajo bazne funkcije $N_{i,k}$, ki so rekurzivno definirane z enačbo 2. Z vsoto v imenovalcu izraza 10 funkcijo $R_{i,k}$ pri vsaki vrednosti parametra u normaliziramo. Množico vrednosti w_i, $w_i \geq 0$, $0 \leq i \leq n$ imenujemo uteži. Shranimo jih v vektor uteži $W = [w_0, w_1, ..., w_n]$. Število uteži je enako številu točk, tako da točko r_i utežimo z utežjo w_i. Vozliščni vektor je definiran podobno kot tisti, ki določa neperiodične krivulje B-zlepkov (poglavje 2):

$$U_{knot} = [u_0, u_1, ..., u_{m-1}, u_m],$$

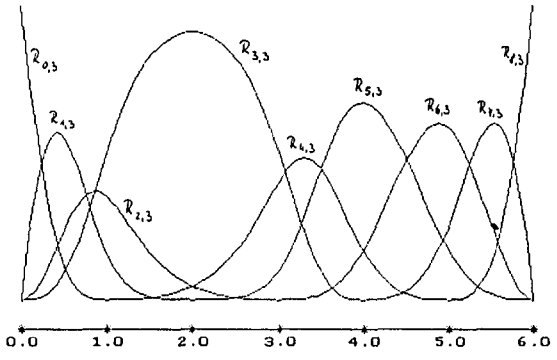
kjer velja: $u_i = 0$; če $i \leq k$ (a)

$u_i = m-2k$; če $m-k \leq i \leq m$ (b)

$u_i \geq u_k$; če $i > k$ in $k+1 \leq i,k \leq m-k-1$ (c)

Za vozle na intervalu (b) velja, da so lahko neuniformni. To pomeni, da razmaki med vozli niso nujno konstantni. Vrednosti sosednjih vozlov so lahko enake, vendar pri tem

velja, da ne sme biti enakih več kot k+1 sosednjih vozlov. Če vsem utežem w_i, $0 \le i \le n$ dodelimo vrednost 1, preide NURB krivulja v krivuljo B-zlepkov. Torej so $R_{i,k}$ posplošitev povezovalnih funkcij $N_{i,k}$. Primer povezovalnih funkcij za NURB krivulje vidimo na sliki 11.



Slika 11 Kubične racionalne povezovalne funkcije določene z:

U_{knot} = [0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6] in
W = [1, 1, 1, 5, 1, 1, 1, 1, 1]

6. ALGORITEM ZA IZRAČUN NURB KRIVULJ

Za izračun povezovalnih funkcij $N_{i,k}$ uporabljamo že znano dinamično drevesno strukturo (poglavje 3). Tudi za shranjevanje vrednosti $R_{i,k}$ vzamemo isti tip podatkovne strukture kot za $N_{i,k}$:

NURB: AllBFunctionType,

s to razliko, da ima tokrat vsaka funkcija $R_{i,k}$ kazalec na svoj zapis tipa BfType (slika 7c).

Za izračun racionalnih povezovalnih funkcij upoštevamo naslednje:

- potrebujemo n+1 povezovalnih funkcij $N_{i,k}$, ki so lahko vse, zaradi morebitnega neuniformnega vozliščnega vektorja, različne;
- ker v splošnem niti dve racionalni povezovalni funkciji nista enaki, moramo določiti tudi vseh n+1 $R_{i,k}$.

Pseudo kod algoritma za izračun povezovalnih funkcij NURB krivulj vidimo na sliki 12.

Pri analizi časovne zahtevnosti za izračun neperiodične krivulje B-zlepkov smo ugotovili, da je le-ta neodvisna od števila točk. Tokrat pa so zaradi neuniformnega vozliščnega vektorja vse funkcije $N_{i,k}$ med seboj različne. Vsako izmed njih izračunamo v polinomskem času (enačba 7), za izračun vseh funkcij $N_{i,k}$ pa moramo poklicati rutino za izračun baznih funkcij

$$2*(1*NoOfSteps + 2*NoOfSteps + \ldots + k*NoOfSteps) +$$
$$+ (k+1)(n+1-2k)*NoOfSteps = (k+1)(n-k+1)*NoOfSteps.$$

Število klicev je tokrat v linearni odvisnosti od števila kontrolnih točk krivulje. Ko smo izračunali vse povezovalne funkcije $N_{i,k}$, je potrebno izračunati še racionalne povezovalne funkcije $R_{i,k}$. Vsaka $R_{i,k}$ se razteza največ čez k+1 segmentov vozliščnega vektorja. Za vsako vrednost u

```
Type
  Ptr = ^TypePtr;
  TypePtr = array [0..knotmax] of record
                                    knot  : real;
                                    index : integer;
                                  end;

procedure MakeNURBf(ABF: AllBFunctionType;

      Var NURB: AllBFunctionType; Wei: PointerToWeights;
   Knt: PointerToKnots; NoAllBf, k: integer; var indexs: Ptr);

{ - ABF: kazalec na povezovalne funkcije Ni,k,
  - NURB: kazalec na NURB povezovalne funkcije,
  - Knt: kazalec na vozliščni vektor,
  - Wei: kazalec na polje, kjer so shranjene uteži,
  - NoAllBf: število neperiodičnih pov. funkcij v ABF,
  - k: stopnja racionalnih povezovalnih funkcij
  - indexs: kazalec na polje vozlov.

var
  i, num, c, m, NoUv : integer;

  function Summa(indexs: Ptr; i,j,k,m: integer;
           N: AllBFunctionType; W: PointerToWeights;
           Knt: PointerToKnots): real;
    { - i: indeks funkcije, ki se trenutno izračunava,
      - j: indeks vrednosti povezovalne funkcije,
      - k: stopnja povezovalnih funkcij,
      - N: kazalec na polje vrednosti povezovalnih funkcij,
      - W: kazalec na polje uteži.
  var
    Sum, u          : real;
    Bfu, ind, kplus1 : integer;
  begin
    Kplus1 := 1; Bfu := 0;
    u := GetuValue(Knt,indexs,i,j);
    while (kplus1  ≤ k + 1) and (Bfu ≤ NoAllBf) do
    begin
      if (u ≥ Knt^[Bfu]) and (u ≤ Knt^[Bfu + k +1]) then
      begin
        ind := GetIndex(Knt,indexs,i,j,Bfu);
        Sum:= Sum + N[Bfu]^.Bf[ind] * W^[Bfu];
        kplus1 := kplus1 + 1;
      end;
      Bfu := Bfu +1;
    end;
    Summa:= Sum;
  end;  { Summa }

begin { MakeNURBf }
  new(indexs);
  indexs^[0].index := 0;    indexs^[0].knot := Knt^.[0];
  m := NoAllBf + k + 1;     c := 0;
  for i := 0 to k do
  begin
    num := i;
    while (num ≤ m - k) and (c ≤ m) do
    begin
      if num ≤ k then
      begin
        indexs^[num + 1].index := ABF^[num]^.NoOfUValues;
        indexs^[num + 1].knot := Knt^[i + k + 1];
      end
      else
      begin
        indexs^[num + 1].index := indexs^[num - k].index +
                                  ABF^[num].NoOfUValues;
        indexs^[num + 1].index := Knt^[num + k + 1];
      end;
      num := num + k + 1; c := c + 1;
    end;
  end;
  GetDiffKnots(indexs,NoAllBf + k + 1,NoOffDiff);
  for i := 0 to NoAllBf do
  begin
    m := NoAllBf + k + 1; NoUv := ABF[i]^.NoOfUValues;
    NURB[i]^.NoOfUValues := NoUv;
    for j := 0 to NoUv do
      NURB[i]^.Bf[j] := ABF[i]^.Bf[j] * Wei^.[i] /
                        Summa(indexs,i,j,k,m,ABF,Wei,Knt);
  end;
end; { MakeNURBf }
```

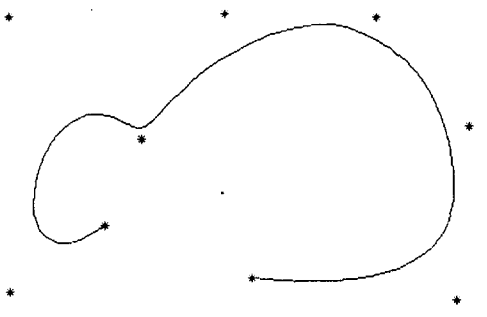
Slika 12 Algoritem za izračun NURB krivulj

moramo določiti vsoto imenovalca v enačbi 10, ki je seštevek k+1 vrednosti funkcij $N_{i,k}$ pomnoženih z utežmi w_i, kar opravimo v polinomskem času.

Algoritem za izris NURB krivulj, ki uporablja podatkovno strukturo s slike 7c, vidimo na sliki 13. Primer NURB krivulje določene s povezovalnimi funkcijami na sliki 11 pa vidimo na sliki 14.

```
procedure PlotNURBSpline (m, n, k : integer;
                          Knt : PointerToKnots; rx, ry : PtoPoints;
                          NURB : AllbFunctionType; indexs : Ptr);
{ - m: število vozlišč vozliščnega vektorja,
  - n: število kontrolnih točk in število povez. funkcij,
  - k: stopnja racionalnih povezovalnih funkcij,
  - Knt: kazalec na vozliščni vektor,
  - rx,ry: kazalca na x- in y-koordinate kontrolnih točk,
  - NURB:  kazalec na vrednosti povezovalnih funkcij
  - indexs: kazalec na polje vozlov.                          }
var
  x, y     : PolyLine_Type;
  i,j,ind,NoOfKnt,firstRBf,lastRBf,func,ni,Steps: integer;
  first    : boolean;
begin
  NoOfKnt := NoOfDiffKnots(indexs);
  for i:=1 to NoofKnt-1 do
  begin
    first:=true;
    for j:=0 to n do
      if  (Knt^[j]<=rKnt[i]) and (rKnt[i+1]<=Knt^[j+k+1])
      then if first then
        begin
          firstRbf:=j;
          first:=false;
        end
        else lastRbf:=j;
    Steps:=round((rKnt[i+1]-rKnt[i])*NoofSteps)+1;
    for ni:=1 to Steps do
    begin
      x[ni]:=0;
      y[ni]:=0;
      for func:=firstRbf to lastRbf do
      begin
        ind:=round((rKnt[i]-Knt^[func])*NoofSteps)+ni-1;
        x[ni]:=x[ni]+ NURB[func]^.Bf[ind]*rx^[func];
        y[ni]:=y[ni]+ NURB[func]^.Bf[ind]*ry^[func];
      end;
    end;
    gPolyLine(Steps,x,y);
  end;
end; { PlotNURBSpline }
```

Slika 13 Algoritem za izris NURB krivulj



Slika 14 NURB krivulja

7. ZAKLJUČEK

Opisana algoritma za izris krivulj B-zlepkov in NURB krivulj smo uporabili v programskem orodju za študij krivulj v CAGD, ki smo ga izdelali v našem laboratoriju [GUID91]. Časovno zahtevnost obeh izdelanih algoritmov smo primerjali z de Boorovim iterativnim algoritmom [YAMA88] in iterativnim racionalnim de Boorovim algoritmom [FARI89], ki izračunata ustrezno krivuljo brez določevanja povezovalnih funkcij. Rezultati so prikazani v tabeli 1 za primer, ko imamo 35 kontrolnih točk. Vidimo, da je naš algoritem za izris krivulj B-zlepkov hitrejši od de Boorovega algoritma, saj s sestavo primerne podatkovne strukture izkorišča lastnosti krivulj B-zlepkov oziroma njihovih povezovalnih funkcij (t.j. da je potrebno določiti samo k+1 različnih povezovalnih funkcij). De Boorov racionalni iterativni algoritem za izris NURB krivulj pa je hitrejši od našega algoritma, saj je tokrat potrebno izračunati vseh n povezovalnih funkcij. Seveda pa de Boorov algoritem ne omogoča prikaz povezovalnih funkcij, ki pa smo jih želeli vključiti v omenjen paket za študij krivulj.

Tabela 1 Čas (msek) porabljen za izris krivulj B-zlepkov in NURB krivulj z različnimi algoritmi

		algoritem								
		A	B	C	D	E	F	G	H	I
red	2	1.59	1.71	1.74	1.83	3.46	3.47	22.95	5.54	22.96
	3	1.97	2.18	2.14	2.59	5.27	5.33	34.56	9.50	34.88
	5	2.74	4.07	4.04	7.84	10.49	10.60	63.25	20.65	66.96
	8	4.40	22.84	11.53	87.49	20.87	21.26	119.91	43.11	197.65

A: naš algoritem - periodične krivulje
B: Cox-de Boorova formula - periodične krivulje
C: naš algoritem - neperiodične krivulje
D: Cox-de Boorova formula - neperiodične krivulje
E: de Boorov algoritem - periodične krivulje
F: de Boorov algoritem - neperiodične krivulje
G: naš algoritem za NURB krivulje
H: de Boorov racionalni algoritem za NURB krivulje
I: rekurzivni algoritem za NURB krivulje

Opisan algoritem lahko brez težav priredimo tudi za izračun in izris drugih tipov krivulj in ploskev, kar smo storili tudi v našem programskem paketu za študij krivulj v CAGD. Algoritem smo napisali v pascalu in za izris uporabili funkcije standarda GKS. Izdelali smo ga na skromni aparaturni opremi: IBM PC/AT-286/16Mh z uporabo matematičnega koprocesorja.

8. LITERATURA

[BART87] Bartels, R. H., J. C. Beatty, and B. A. Barsky, "An Introduction to Splines for Use in Computer Graphics and Geometric Modeling", Morgan Kaufman Publishers, Los Altos, California, 1987.

[BÖHM84] Böhm, W., G. Farin, and J. Kahmann, "A survey of curve and surface methods in CAGD", Comput. Aided Geometric Design, Vol. 1, No. 1, 1984, pp. 1-60.

[FARI89] Farin, G., "Rational Curves and Surfaces", in Mathematical Methods in Computer Aided Geometric Design, (eds. T. Lyche and L. L. Schumaker), Academic Press, 1989, pp. 215-238.

[GUID88] Guid, N., "Računalniška grafika", Tehniška fakulteta Maribor, Maribor, 1988.

[GUID90] Guid, N. and B. Žalik, "Contributions to practical considerations of B-splines", Automatica, Zagreb, Vol. 31, No. 1-2, 1990, pp. 83-88.

[GUID91] Guid, N., B. Žalik, and A. Vesel, "A software teaching tool for curve methods in CAGD", Proc. of Computer Graphics and Education '91, Barcelona, 1991, pp. 62 - 70.

[MORT85] Mortenson, M. E., "Geometric Modeling", John Wiley & Sons, New York, 1985.

[PIEG87] Piegel, L. and R. F. Sproull, "Curve and Surface Construction Using Rational B-splines", Comput. Aided Design, Vol. 19, No. 9, 1987, pp. 485 - 498.

[PIEG89] Piegl, L., "Modifying the shape of rational B-splines. Part 1: curves", Comput. Aided Design, Vol. 21, No. 8, 1989, pp. 509-518.

[ROGE90] Rogers, D. F. and L. A. Adlum, "Dynamic rational B-spline surfaces", Comput. Aided Design, Vol. 22, No. 9, 1990, pp. 609 - 616.

[TURK80] Turk, S., "Računarska grafika. Osnovi teorije i primjene", Školska knjiga Zagreb, Zagreb, 1980.

[WAYN83] Wayne, T., "Rational B-splines for Curve and Surface Representation", IEEE Comput. Graphics and Application, Vol. 3, No. 6, pp. 61-69.

[YAMA88] Yamaguchi, F., Curves and Surfaces in Computer Aided Geometric Design", Springer-Verlag, Berlin, Heidelberg, 1988.

[ŽALI90] Žalik, B. and N. Guid, "Časovna in prostorska zahtevnost pri izrisu krivulj B-zlepkov", Proc. 34. konferenca ETAN, Zagreb, Vol. 8, 1990, pp. 65-72.