Išvestinių reikšmės kraštiniuose taškuose: $(x_1; y_1)$ ***ir*** $(x_N; y_N)$**.** Jos apskaičiuojamos panaudojant necentruotas trijų taškų formules, t.y. per tris kraštinius taškus $((x_1, y_1), (x_2, y_2), (x_3, y_3)$ kairiajame intervalo gale ir $(x_{N-2}, y_{N-2}), (x_{N-1}, y_{N-1}), (x_N, y_N)$ dešiniajame intervalo gale) pravedama kvadratinė parabolė ir šios parabolės išvestinės reikšmė kraštiniame taške sutapatinama su ieškoma išvestine.

Tuo būdu:

$$yi_1 = \frac{(2h_1 + h_2)d_1 - h_1 d_2}{h_1 + h_2}.$$

Jei yi_1 ir d_1 yra priešingų ženklų, tai $yi_1 = 0$.

Jei d_1 ir d_2 yra priešingų ženklų, be to dar, $|yi_1| > |3d_1|$, tai $yi_1 = 3d_1$.

Analogiškai:

$$yi_N = \frac{(2h_{N-1} + h_{N-2})d_{N-1} - h_{N-1} d_{N-2}}{h_{N-1} + h_{N-2}}.$$

Jei yi_N ir d_{N-1} yra priešingų ženklų, tai $yi_N = 0$.

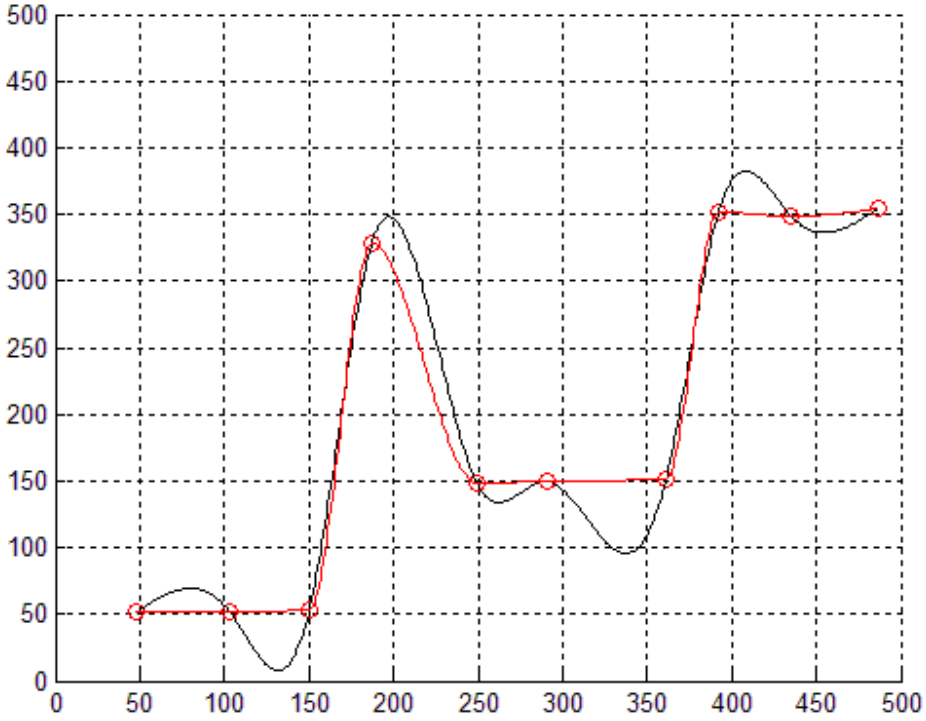
Jei d_{N-1} ir d_{N-2} yra priešingų ženklų, be to dar, $|yi_N| > |3d_{N-1}|$, tai $yi_N = 3d_{N-1}$.

Žemiau pateikta funkcija ***akimam***, kuri apskaičiuoja modifikuotą Akima splainą, o 5.13 paveiksle pavaizduoti kubinis interpoliacinis splainas su natūraliomis kraštinėmis sąlygomis ir modifikuotas Akima splainas.

```
function fv=akimam(x,y,xx)
% AKIMAM  apskaičiuoja Ermito splainą, kai duoti
% interpoliavimo taškai (x,y).Išvestinių reikšmės taškuose (x,y)
% apskaičiuojamos pagal modifikuotas Akima formules.
% Įė jimo parametrai
% x  - interpoliavimo taškų abscisės,
% y  - interpoliavimo taškų ordinates,
% xx - reikšmės abscisių, kuriose apskaičiuojamos
%      Ermito splaino reikšmės.
% Išėjimo parametrai
% v  - Ermito splaino reikšmės taškuose xx.

% Ar teisingi įėjimo duomenys?
n = length(x); m = length(y);
if n==2
    error('per mažas x elementų skaičius')
end
if n~=m
    error('x-o ir y-ko elementų skaičiai yra skirtingi')
end
% Išvestinių yi reikšmių apskaičiavimas
h = diff(x);
d = diff(y)./h;
%  Išvestinės vidiniuose taškuose (x(k);y(k))
   yi = zeros(size(y));
   k = find(sign(d(1:n-2)).*sign(d(2:n-1)) > 0);
   hs = h(k)+h(k+1);
   w1 = (h(k)+hs)./(3*hs); w2 = (hs+h(k+1))./(3*hs);
   dmax = max(abs(d(k)), abs(d(k+1)));
   dmin = min(abs(d(k)), abs(d(k+1)));
```

```
   yi(k+1) = dmin./(w1.*(d(k)./dmax) + w2.*(d(k+1)./dmax));
% Išvestinės yi(1) ir yi(n) kraštiniuose taškuose.
   yi(1) = ((2*h(1)+h(2))*d(1) - h(1)*d(2))/(h(1)+h(2));
   if  (sign(yi(1)) ~= sign(d(1)))
      yi(1) = 0;
   elseif (sign(d(1)) ~= sign(d(2))) && (abs(yi(1)) > abs(3*d(1)))
      yi(1) = 3*d(1);
   end
   yi(n) = ((2*h(n-1)+h(n-2))*d(n-1) - h(n-1)*d(n-2))/...
           (h(n-1)+h(n-2));
   if  (sign(yi(n)) ~= sign(d(n-1)))
      yi(n) = 0;
   elseif (sign(d(n-1)) ~= sign(d(n-2))) &&...
                                        (abs(yi(n)) > abs(3*d(n-1)))
      yi(n) = 3*d(n-1);
   end
   fv=ermitspln(x,y,yi,xx); % žr. Ermito kubiniai splainai
```



5.13 pav. Kubinis interpoliacinis splainas su natūraliomis kraštinėmis sąlygomis (juoda spalva) ir modifikuotas Akima spalinas (raudona spalva).

5.2. Kreivių konstravimas

Atliekant kompiuterinės grafikos darbus arba projektuojant gaminius, pavyzdžiui, automobilių kėbulus, tenka konstruoti norimos formos glodžias kreives. Tuomet labai svarbu turėti galimybę keisti norimą kreivės dalį, nekeičiant kitų dalių. Sprendžiant šį uždavinį, reikia naudoti lokaliuosius parametrinius splainus.

Apie 1960-uosius metus kreivių konstravimo uždavinį pirmieji pradėjo spręsti prancūzų inžinieriai. Polinomines struktūras kreivėms konstruoti *Citroen* firmoje pirma-

sis pradėjo P. de Kastelžo (P. de Casteljau) apie 1959 metus. Tačiau jo darbai buvo įslaptinti. Vėliau, maždaug 1962 metais, *Renault* automobilių kompanijos inžinieriaus P. Bezjė (P. Bezier) sukurta kreivių ir paviršių projektavimo sistema UNISURF buvo plačiai publikuota. Dėl to šioms kreivėms, kaip ir panašios konstrukcijos paviršiams, prigijo Bezjė vardas.

Norint gauti glodžias ir pakankamai sudėtingas kreives, kurias po to būtų galima lokaliai keisti, Bezje konstrukcijų neužtenka. Kreivėms konstruoti buvo pradėti naudoti B splainai. Pirmasis B splainus kreivėms konstruoti panaudojo K. de Būras (C. de Boor), dirbdamas *General Motors* kompanijoje.

Anksčiau, iki 1960 metų, kreivėms ir paviršiams modeliuoti (pavyzdžiui, *Boeing* firmoje lėktuvų fiuzeliažams modeliuoti) buvo naudojamos konikės — antrosios eilės kreivės. (Senovės graikų matematikas Apolonijus antrosios eilės kreives – apskritimą, elipsę, hiperbolę ir parabolę — nagrinėjo kaip plokščiuosius kūgio pjūvius, todėl jos dažnai vadinamos konikėmis.) Atsiradus splaininei kreivių konstravimo technologijai, šiuos skirtingus metodus bandyta derinti. Tam tikslui S. Kunsas (S. A. Coons) sukūrė racionaliuosius Bezjė polinomus ir B splainus.

5.2.1. Bezjė kreivės

Tarkime, kad turime aibę taškų $p_i = \left(x_i; y_i\right)$, $i = \overline{0, n}$. Jie, dar vadinami Bezjė taškais arba kontroliniais taškais, gali būti išdėstyti plokštumoje iš kairės į dešinę, nebūtinai jų indeksų didėjimo tvarka. Toliau taškus p_i traktuosime kaip dviejų komponenčių vektorius

$$p_i = \begin{pmatrix} x_i & y_i \end{pmatrix}^t,\ i = \overline{0, n}.$$

Tada *n*-tosios eilės Bezjė polinomo, kurį nusako taškai $p_i \left(i = \overline{0, n}\right)$, parametrinė išraiška bus nusakoma taip:

$$P(u) = \sum_{i=0}^{n} C_n^i (1-u)^{n-i} u^i p_i \text{ ; čia } C_n^i = \frac{n!}{i!(n-i)!},\ 0 \le u \le 1.$$

Polinomai $B_i^n(u) = C_n^i (1-u)^{n-i} u^i$ vadinami *n*-jo laipsnio Bernšteino polinomais. Kitaip tariant, Bezjė kreivė (polinomas) — tai parametriškai užrašytas Bernšteino polinomų $B_i^n(u)$ tiesinis darinys, kai darinio koeficientai sutapatinami su kontrolinių taškų koordinatėmis:

$$P(u) = p_0 B_0^n(u) + p_1 B_1^n(u) + \ldots + p_n B_n^n(u).$$

Kai $n = 1$, turime tiesės atkarpą $p_0 p_1$, kai $n = 2$ — kvadratinės parabolės lanką, einantį per taškus p_0 ir p_2, o atkarpos $p_0 p_1$ ir $p_1 p_2$ yra to lanko liestinės taškuose p_0 ir p_2.

Nesunku patikrinti, kad:

— p_0 ir p_n yra galiniai kreivės taškai;

— liestinės galiniuose taškuose p_0 ir p_n turi atkarpų $p_0 p_1$ ir $p_{n-1} p_n$ kryptis;

— visa kreivė yra kontrolinių taškų $p_0, p_1, \ldots, p_n$ iškilame apvalkale.

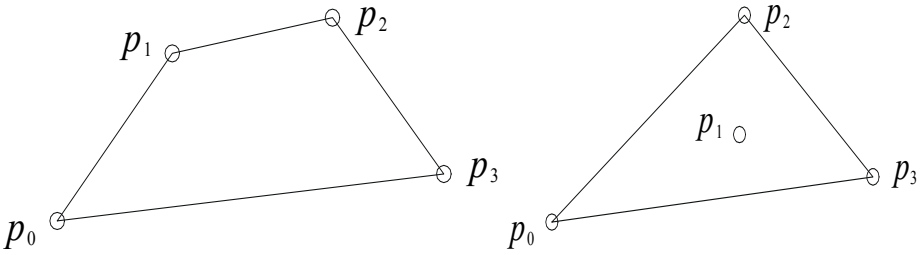
Praktikoje dažniausiai naudojami trečiosios eilės Bezjė polinomai:

$$x(u) = (1-u)^3 x_0 + 3(1-u)^2 u x_1 + 3(1-u) u^2 x_2 + u^3 x_3,$$
$$y(u) = (1-u)^3 y_0 + 3(1-u)^2 u y_1 + 3(1-u) u^2 y_2 + u^3 y_3.$$

Juos nusako keturi taškai: p_0, p_1, p_2, p_3.

Trečiosios eilės Bezjė polinomai turi aiškią geometrinę prasmę. Kai $u = 0$, tai $x(0) = x_0$, $y(0) = y_0$; kai $u = 1$, tai $x(1) = x_3$, $y(1) = y_3$;

$$y'_x(0) = \frac{y'_u(0)}{x'_u(0)} = \frac{3(y_1 - y_0)}{3(x_1 - x_0)} = \frac{y_1 - y_0}{x_1 - x_0},$$
$$y'_x(1) = \frac{y'_u(1)}{x'_u(1)} = \frac{3(y_3 - y_2)}{3(x_3 - x_2)} = \frac{y_3 - y_2}{x_3 - x_2}.$$



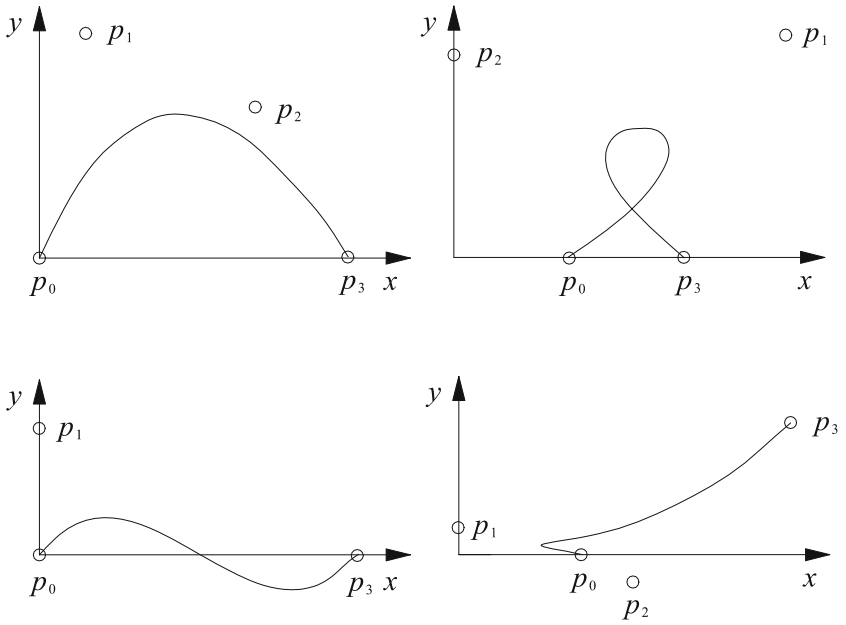
5.14 Sritis, apribota iškilojo apvalkalo, apibrėžto taškais p_0, p_1, p_2, p_3

Vadinasi, trečiosios eilės Bezjė polinomai eina per taškus $p_0 = (x_0; y_0)$ ir $p_3 = (x_3; y_3)$, o tiesės, einančios per taškus p_0, p_1 ir p_2, p_3, yra tų polinomų liestinės atitinkamai taškuose p_0 ir p_3.

5.15 paveiksle parodyti trečiosios eilės Bezjė polinomai, esant įvairiam taškų p_i išsidėstymui. Taškai p_1 ir p_2 vadinami gidais. Keičiant jų padėtį, labai keičiasi kreivių forma, tačiau tos kreivės visada yra srityje, apribotoje iškilojo apvalkalo, kurį apibrėžia taškai p_0, p_1, p_2, p_3 (žr. 5.14 pav.).

Kreivės prigludimą prie liestinių p_0p_1 bei p_2p_3 (o kartu ir kreivės formą) galima valdyti keičiant atkarpų p_0p_1 ir p_2p_3 ilgį. Pavyzdžiui, kuo ilgesnė atkarpa p_0p_1, tuo labiau Bezjė polinomas prigludęs prie liestinės p_0p_1.

Kreivės iš trečiosios eilės Bezjė polinomų konstruojamos taip. Displėjaus ekrane pele atidedami keturi taškai p_0, p_1, p_2, p_3. Pagal toliau pateiktą algoritmą brėžiama kreivės dalis, kurią žymi šie taškai. Po to atidedami trys papildomi taškai. Jie pažymimi p_1^*, p_2^* ir p_3^*; $p_0^* = p_3$. Šie taškai nusako naują kreivės dalį tarp taškų p_0^* ir p_3^*. Šis procesas tęsiamas toliau: atidedamas naujas papildomų taškų trejetas, prie jo prijungiamas nulinis taškas, kuris yra prieš tai buvusio ketvertuko paskutinis taškas, ir brėžiama nauja kreivės dalis.



5.15 pav. Trečiosios eilės Bezjė polinomai

Kastelžo algoritmas. Bezjė kreivės taškai $P(t)$ (čia $t \in [0;1]$) efektyviai gali būti apskaičiuoti pagal Kastelžo algoritmą.

Suformuluokime uždavinį. Duoti *n*-tosios eilės Bezjė kreivės kontroliniai taškai $p_0, p_1, \ldots, p_n$ ir parametro t reikšmės; čia $t \in [0;1]$. Reikia rasti tašką $P(t)$.

Taškas $P(t)$ apskaičiuojamas pagal toliau pateiktą Kastelžo funkciją ***kastelzo***.

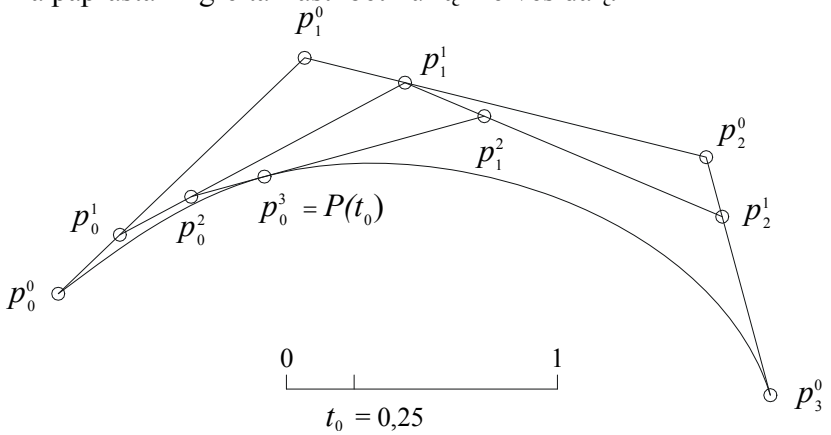
```
function [xt,yt]=kastelzo(n,x,y)
% KASTELZO - tai funkcija, kuri apskaičiuoja n-os eilės
% Bezje kreivę.
% Įėjimo parametrai
% n     - kreivės eilė,
% (x,y) - kontroliniai taškai.
% Išėjimo parametrai
% (xt yt) - Bezje kreivės taškai, apskaičiuoti kai t=0:0.001:1.

u=0:0.001:1; m=numel(u); t=u';
ax=repmat(x,m,1); ay=repmat(y,m,1);
for j=1:n
 for i=1:n+1-j
   ax(:,i)=(1-t).*ax(:,i)+t.*ax(:,i+1);
   ay(:,i)=(1-t).*ay(:,i)+t.*ay(:,i+1);
end
end
xt=ax(:,1); yt=ay(:,1);
```

Aptarkime geometrinę Kastelžo algoritmo interpretaciją.

5.16 paveiksle pavaizduota, kaip brėžiama kubinė Bezjė kreivė. Viršutinis *p* indeksas žymi iteracijos (parametro *j*) numerį. Čia kiekvieną atkarpą $p_i^j p_{i+1}^j$ taškas p_i^{j+1} dalija santykiu $t_0 : (1-t_0)$. Tarpiniuose skaičiavimuose gauti taškai teikia svarbią informaciją. Pavyzdžiui, taškai p_0^0, p_0^1, p_0^2 ir p_0^3 yra Bezjė kreivės lanko, atitinkančio parametrą *t*

$(t \in [0; t_0])$, kontroliniai taškai. Analogiškai, taškai p_0^3, p_1^2, p_2^1 ir p_3^0 yra Bezjė kreivės lanko, atitinkančio parametrą t $(t \in [t_0; 1])$, kontroliniai taškai. Taigi Kastelžo algoritmas įgalina paprastai ir greitai rasti bet kurią kreivės dalį.



5.15 pav. Kastelžo algoritmo kubinei Bezjė kreivei brėžti iliustracija

Racionaliosios Bezjė kreivės. Iš antrosios eilės kreivių: apskritimo, elipsės, parabolės ir hiperbolės — tik parabolę galima priskirti prie anksčiau aptartų Bezjė kreivių. Tuo tarpu apskritimas, elipsė ir hiperbolė nėra polinominės kreivės. Norint ir jas įtraukti į Bezjė kreivių šeimą, reikia naudoti racionaliąsias Bezjė kreives, kurios gaunamos centrinės projekcijos metodu.

Erdvės R^3 koordinačių sistemoje *Oxyz*, nagrinėkime plokštumą $z = 1$ ir centrinės projekcijos centrą — koordinačių pradžią *O*. Tada kiekvieno erdvės R^3 taško $p = (x; y; z)$ centrinė projekcija plokštumoje $z = 1$ yra taškas, kuriame spindulys *Op* kerta plokštumą $z = 1$. Taigi *racionalioji plokščioji n-tojo laipsnio Bezjė kreivė apibrėžiama kaip erdvinės n-tojo laipsnio Bezjė kreivės centrinė projekcija plokštumoje* $z = 1$.

Nesunku įsitikinti, kad erdvės R^3 taško $(x; y; z)$ centrinė projekcija plokštumoje $z = 1$ yra taškas $(x/z; y/z)$. Paskutinę koordinatę *z* praleidžiame, nes visų plokštumos taškų ji lygi 1.

Aišku, kad erdvinės Bezjė kreivės $\widetilde{P}(t)$ kontrolinio taško $\widetilde{p}_i = (w_i x_i; w_i y_i; w_i)$ $(i = \overline{0, n})$ centrinė projekcija yra taškas $p_i(x_i; y_i)$. Tada erdvinės Bezjė kreivės

$$\widetilde{P}(t) = \sum_{i=0}^{n} \widetilde{p}_i B_i^n(t) = \left(\sum_{i=0}^{n} w_i x_i B_i^n(t); \sum_{i=0}^{n} w_i y_i B_i^n(t); \sum_{i=0}^{n} w_i B_i^n(t) \right)$$

centrinė projekcija plokštumoje $z = 1$ yra plokščioji kreivė

$$P(t) = \left(\frac{\sum_{i=0}^{n} w_i x_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)}; \frac{\sum_{i=0}^{n} w_i y_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)} \right),$$

kuri vadinama racionaliąja *n*-tojo laipsnio Bezjė kreive. Vadinasi, vektorinė racionaliosios *n*-tojo laipsnio Bezjė kreivės išraiška yra tokia:

$$P(t)=\frac{\sum_{i=0}^{n} w_i p_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)}.$$

Lengva patikrinti, kad racionaliosios kreivės $P(t)$ galiniai taškai yra p_0 ir p_n, kuriuose ji liečia tieses p_0p_1 ir $p_{n-1}p_n$. Tuo racionaliosios kreivės yra panašios į paprastąsias Bezjė kreives. Tačiau kontrolinių taškų svoriai racionaliosioms kreivėms suteikia papildomų savybių. Kuo didesnis kontrolinio taško p_i svoris w_i, tuo labiau kreivė artėja prie šio taško.

Jei visi w_i $\left(i=\overline{0,n}\right)$ yra lygūs, tai racionalioji Bezjė kreivė sutampa su paprastąja, nes $\sum_{i=0}^{n} B_i^n(t)=1$.

Vis dėlto svarbiausia racionaliųjų kreivių savybė yra ta, kad *racionalioji kvadratinė Bezjė kreivė yra konikė, t. y. antrosios eilės kreivė*.

Teorema. R*acionalioji kvadratinė Bezjė kreivė yra antrosios eilės kreivė. Ji yra:*

1) *parabolė, kai* $w_1^2-w_0w_2=0$;

2) *hiperbolė, kai* $w_1^2-w_0w_2>0$;

3) *elipsė, kai* $w_1^2-w_0w_2<0$.

Timerio (H. Timmer) kreivė. H. Timmer iš McDonell Douglas modifikavo kubinę Bezjė kreivę, imdamas ne Bernšteino polinomus, o kitas bazines funkcijas. Jo kreivė, kaip ir Bezjė kreivė eina per taškus p_0 ir p_3, atkarpos p_0p_1 ir p_2p_3 yra kreivės liestinės atitinkamai taškuose p_0 ir p_3. Be to, Timerio kreivė eina per atkarpos p_1p_2 vidurio tašką, tačiau atkarpa p_1p_2 nėra kreivės liestinė. Timerio kreivė, lyginant su Bezjė kreive, yra labiau įtempta: eina arčiau poligono, kurį nusako kontroliniai taškai, kontūro.

Timerio bazinės funkcijos yra tokie kubiniai polinomai:

$f_0(t)=-2t^3+5t^2-4t+1; f_1(t)=4t^3-8t^2+4t;$

$f_2(t)=-4t^3+4t^2; f_3(t)=2t^3-t^2,$

o pati kreivė, kurią apibrėžia kontroliniai taškai p_0, p_1, p_2, p_3, užrašoma taip:

$P(t)=p_0f_0(t)+p_1f_1(t)+p_2f_2(t)+p_3f_3(t), 0\le t\le 1.$

Bolo kreivės. Šias kreives 1974m. sukonstravo Alanas Bolas (Alan Ball), kuris dirbo British Aircraft Corporation. Panašiai, kaip Timerio kreivės, Bolo kubinės kreivės gali būti atskiri Bezjė kreivės atvejai. Tačiau jų skiriamasis bruožas yra tas, kad Bolo kreivių sekcijos yra konikės.

Bazinės funkcijos, apibrėžiančios Bolo kreives, yra:

$f_0(t)=(1-t)^2=1-2t+t^2, f_1(t)=2t(1-t)^2=2t-4t^2+2t^3,$

$f_2(t)=2t^2(1-t)=2t^2-2t^3, f_3(t)=t^2,$

o pati kreivė, kurią apibrėžia kontroliniai taškai p_0, p_1, p_2, p_3, užrašoma taip:

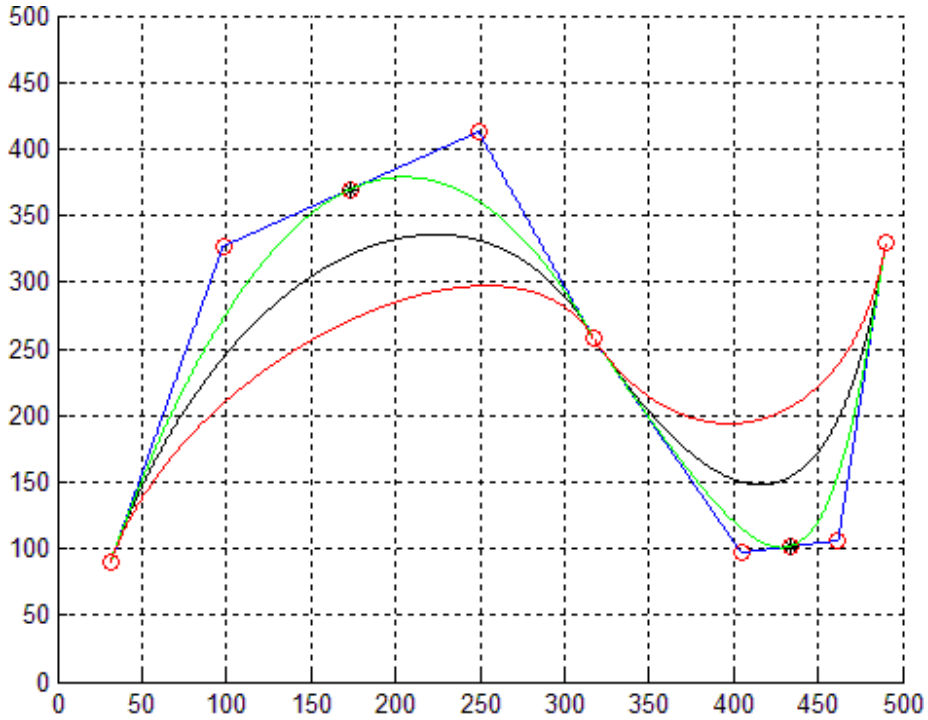
$P(t) = p_0 f_0(t) + p_1 f_1(t) + p_2 f_2(t) + p_3 f_3(t), 0 \le t \le 1.$

Reikia pastebėti, jei $p_1 = p_2$, tai Bolo kreivė sutampa su kvadratine Bezjė kreive:

$p_0(1-t)^2 + p_1 2t(1-t)^2 + p_1 2t^2(1-t) + p_3 t^2 == p_0(1-t)^2 + p_1(2t(1-t)^2 + 2t^2(1-t)) + p_3 t^2 =$

$= p_0(1-t)^2 + p_1 2t(1-t) + p_3 t^2.$

5.16 paveiksle pavaizduotos Bezjė, Bolo ir Timerio kreivės, kurias nusako tie patys kontroliniai taškai.



5.16 pav. Bezjė (juoda), Bolo (raudona) ir Timerio(žalia) kreivės ir kontrolinių taškų poligonas (mėlynas).

5.2.2. Splaininės kreivės

Normalizuotieji B splainai (toliau B splainai) yra patogi splainų erdvės bazė. Bet kokia splaininė kreivė arba paviršius yra tiesinis B splainų darinys. Konstruojant splainines kreives ir paviršius, jie užrašomi parametrine forma, sutapatinant tiesinio darinio koeficientus su kontrolinių (de Būro) taškų koordinatėmis. Todėl, keičiant kurio nors kontrolinio taško padėtį, lokaliai keičiasi splaininės kreivės (paviršiaus) forma. Dėl šios savybės, o taip pat dėl kreivės (paviršiaus) glodumo splaininės kreivės ir paviršiai plačiai naudojami geometriniame dizaine, pvz., projektuojant automobilių kėbulus, lėktuvų fuzeliažus ir kt.

Panagrinėkime teorinius ir praktinius kreivių ir paviršių konstravimo aspektus naudojant MATLAB‘o aplinką. Ši aplinka dėka paelemenčių operacijų įgalina dviem eilėm sutrumpinti splaininių kreivių ir paviršių apskaičiavimo laiką, lyginant su laiku, reikalingu jiems apskaičiuoti, naudojant kitas aukšto lygio programavimo kalbas, o taip pat lengvai pavaizduoti juos grafiškai.

B splainų parametrinė forma. Splaininėms kreivėms ir paviršiams konstruoti naudojami B splainai, kuriuos apibrėžia vienetinio žingsnio sveikaskaitinis tinklelis. Tokių B splainų parametrinė forma yra invariantiška tinklelio mazgo atžvilgiu. Todėl nagrinėsime *n*-osios eilės B splaino, apibrėžto mazgo $x = 0$ atžvilgiu, parametrinę formą $B_n^0(u), 0 \le u \le 1$. Splaino $B_n^0(u)$ analitinę išraišką intervale $[k, k+1]$, $k = \overline{0,n}$, pažymėkime $B_n^{0[k,k+1]}(u)$ ir aptarkime, kaip ją apskaičiuoti.

Kaip buvo minėta anksčiau, B-splainai tenkina rekurenčiąją formulę:

$$B_n^i(x) = \frac{x - x_i}{x_{i+n} - x_i} \cdot B_{n-1}^i(x) + \frac{x_{i+n+1} - x}{x_{i+n+1} - x_{i+1}} \cdot B_{n-1}^{i+1}(x), \quad n \ge 1,$$
$$B_0^i(x) = \begin{cases} 1, & x \in [x_i, x_{i+1}), \\ 0, & x \notin [x_i, x_{i+1}). \end{cases} \tag{5.17}$$

Remdamiesi šia išraiška, išvesime rekurenčiąją formulę apskaičiuoti $B_n^{0[k,k+1]}(u)$.

Kadangi tinklelis sveikaskaitinis ir jo žingsnis lygus 1, tai, įvedus pakeitimą, $u = x - k$ ir įvertinus, kad 1) $B_n^{1[k,k+1]}(u) = B_n^{0[k-1,k]}(u)$, 2) $B_0^{[0,1]}(u) = 1$, iš (5.17) formulės gausime

$$B_n^{[k,k+1]}(u) = \frac{1}{n}\left((u+k)B_{n-1}^{[k,k+1]}(u) + (n+1-k-u)B_{n-1}^{[k-1,k]}(u)\right), \ k = \overline{0,n},$$
$$B_0(u) = \begin{cases} 1, & x \in [0,1], \\ 0, & x \notin [0,1], \end{cases} \tag{5.18}$$

čia viršutinis indeksas “0” praleistas.

(5.18) formulė sudaro *n*-osios eilės splaininių kreivių ir paviršių konstravimo pagrindą. Ji įgalina sudaryti universalią procedūrą, kuri leidžia duotoms *n* ir *u* reikšmėms apskaičiuoti $B_n^{[k,k+1]}(u)$, $k = \overline{0,n}$, reikšmes, iš kurių apskaičiuojami splaininių kreivių ir paviršių taškai.

Kreivių konstravimas. Tarkime, $p_i = (x_i, y_i)$, $i = \overline{1,N}$ – kontrolinių taškų seka. Tada konstruojamos *n*-osios eilės splaininės kreivės dalies, kurią apibrėžia taškai $p_i, p_{i+1}, \ldots, p_{i+n}$, parametrinė forma yra:

$$p(u) = \sum_{l=i}^{n+i} p_l \cdot B_n^{[n+i-l,n+i+1-l]}(u), \quad u \in [0,1]. \tag{5.19}$$

Norint, kad kreivės pradžios taškas būtų p_1, o galo – p_N, kontrolinių taškų seką iš kairės papildykime (*n*-1) taškais p_1, o iš dešinės – (*n*-1) taškais p_N. Kadangi $\sum_{k=0}^{n} B_n^{[k,k+1]}(u) = 1$, tai lengva įsitikinti, kad taip papildyta kontrolinių taškų seka apibrėš kreivę nuo p_1 iki p_N.

Greičiausiai kontrolinių taškų seką papildysime atlikę veiksmą:

```
p_naujas = [repmat(p(1), 1, n-1)   p   repmat(p(N), 1, n-1)]
```

Šioje formulėje MATLAB funkcija *repmat* (*n*-1) kartą pakartoja pirmojo parametro reikšmę. Todėl *p_naujas* apibrėžia anksčiau nurodytu būdu papildytą kontrolinių taškų seką. Tolesniame dėstyme šią seką žymėsime ankstesniu simboliu $p_i, i = \overline{1, N+2n-2}$.

Taigi, splaininės kreivės taškams apskaičiuoti reikia:

1) vieną kartą pagal (5.18) formulę apskaičiuoti $B_n^{[k,k+1]}(u)$, $k = \overline{0,n}$, visoms *u*=0:*h*:1 reikšmėms ; (ši formulė reiškia, kad *u* įgauna reikšmes nuo 0 iki 1, kurios nutolę viena nuo kitos žingsniu *h*; čia ir toliau naudosime MATLAB sintaksę),
2) šias reikšmes naudoti kiekvienos kreivės dalies taškams apskaičiuoti pagal (5.19) formulę.

MATLAB'as įgalina efektyviai apskaičiuoti B splainų reikšmes, kai *u*=0:*h*:1.

B splaino reikšmių apskaičiavimas

```
h=0.001; u=0:h:1; m=numel(u);
t=u'; b=zeros(1,n+1); b(1)=1;a=repmat(b,m,1);
for l=1:n
 for k=l+1:-1:1
  if k~=1
   a(:,k)=((t+k-1).*a(:,k)+(l+2-k-t).*a(:,k-1))/l;
  else
   a(:,k)=(t+k-1).*a(:,k)/l;
  end
 end
end
```

Šiame algoritme standartinės MATLAB funkcijos apskaičiuoja:

- *numel(u)* – masyvo *u* elementų skaičių,
- *zeros(1,n+1)* – nulinį vektorių, turintį (*n*+1) elementą,
- *repmat(b,m,1)* – *m*×(*n*+1) formato matricą, kurios eilutės yra vektorius *b*,

Be to, simboliai " ' " ir " .* " atitinkamai žymi transponavimo ir paelementę daugybos operacijas, o – *a*(:, *k*) žymi matricos *a* *k*-ąjį stulpelį.

Nesunku pastebėti, kad $a(i,k) = B_n^{[k,k+1]}(i \cdot h), k = \overline{0,n}, i = \overline{0,(1/h)}$.

Dabar pagal (5.19) formulę brėšime splaininę kreivę.

n-osios eilės splaininės kreivės brėžimas

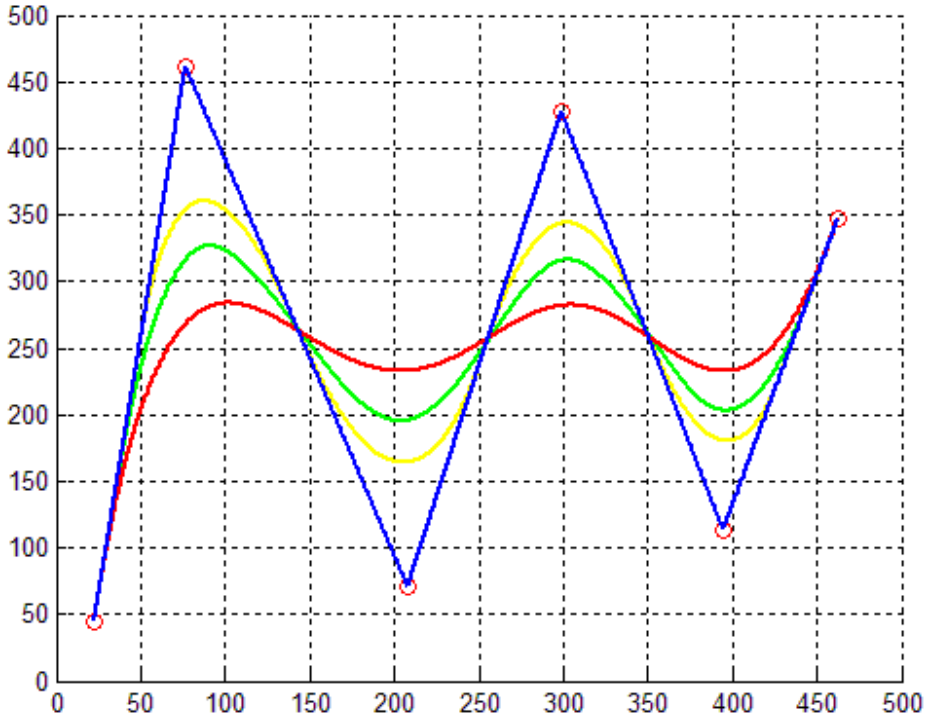
```
for i=1:N+n-2
  c=x(i:i+n); d=c(end:-1:1); p=repmat(d,m,1); xx=sum((a.*p)');
  c=y(i:i+n); d=c(end:-1:1); p=repmat(d,m,1); yy=sum((a.*p)');
  plot(xx,yy); hold on;
end
```

Čia simbolis *x*(*i*:*i*+*n*) žymi masyvo *x* elementus, pradedant indeksu *i* ir baigiant (*i*+*n*); simbolis *c*(*end*:-*1*:*1*) žymi, kad masyvo *c* elementai surašomi atvirkščia tvarka, pradedant paskutiniuoju ir baigiant pirmuoju; funkcija *sum*(*g*) apskaičiuoja matricos *g* stulpelių elementų sumas; funkcija *plot*(*xx*,*yy*) brėžia funkcijos, nusakytos reikšmių lentele (*xx*, *yy*), grafiką, o funkcija *hold on* reiškia, kad grafiko dalys bus brėžiamos tame pačiame lange.

Kaip matyti iš teksto, kreivė brėžiama dalimis, kurias nusako (*n*+1) gretimų kontrolinių taškų aibės: taškai $p_i, p_{i+1}, \ldots, p_{i+n}$ sudaro *i*-ąją, $i = \overline{1, N+n-2}$, aibę. Be to,

MATLAB‘o funkcijos įgalina visus *i*-osios dalies taškus apskaičiuoti vienu metu. Ši galimybė dviem eilėm sutrumpina kreivės konstravimo laiką.

5.17 paveiksle pavaizduotos pirmos, antros, trečios ir penktos eilės splaininės kreivės, kurias nusako raudonai pažymėti de Būro taškai.



5.17 pav. Pirmos (mėlyna), antros (geltona), trečios (žalia) ir penktos (raudona) eilės splaininės kreives, kurias apibrežia de Būro taškai (raudoni apskritimėliai).

Overhauserio kreivės. Overhauserio kreivės buvo sukonstruotos ir naudojamos Ford Motor Company. Jos dar yra žinomos, kaip Catmull- Rom splainai. Splaininės kreivės turi trukumą: jos neinterpoliuoja kontrolinių taškų, t.y. *n*-os eilės splaininė kreivė eina tik per pirmąjį ir paskutinįjį kontrolinius taškus. Tuo tarpu Overhauserio kreivės interpoliuoja visus kontrolinius taškus.

Kubinę Overhauserio kreivę apibrėžia tokios bazinės funkcijos:

$$f_0(t) = -\frac{1}{2}t + t^2 - \frac{1}{2}t^3, f_1(t) = 1 - \frac{5}{2}t^2 + \frac{3}{2}t^3,$$

$$f_2(t) = \frac{1}{2}t + 2t^2 - \frac{3}{2}t^3, f_3(t) = -\frac{1}{2}t^2 + \frac{1}{2}t^3,$$

o pati kreivė, kurią apibrėžia kontroliniai taškai p_0, p_1, p_2, p_3, užrašoma taip:

$$P(t) = p_0 f_0(t) + p_1 f_1(t) + p_2 f_2(t) + p_3 f_3(t), 0 \le t \le 1.$$

Kaip matome iš kreivės formulės, $P(0) = p_1, P(1) = p_2$. Vadinasi, kai *t* kinta nuo nulio iki vieneto, yra suformuojamas Overhauserio kreivės segmentas, prasidedantis kontroliniame taške p_1 ir pasibaigiantis taške p_2. Be to, kreivės $P(t)$ išvestinė taške p_1 yra tiktai taškų p_0 ir p_2 funkcija; analogiškai, kreivės $P(t)$ išvestinė taške p_2 yra tiktai

taškų p_1 ir p_3 funkcija: $P'(0)=\frac{1}{2}(p_2-p_0), P'(1)=\frac{1}{2}(p_3-p_1)$. Tai reiškia, kad kreivės segmentų p_1p_2 ir p_2p_3, nusakytų atitinkamai kontrolinių taškų ketvertais: p_0, p_1, p_2, p_3 ir p_1, p_2, p_3, p_4, taške p_2 reikšmės - iki pirmosios išvestinės imtinai - sutampa. Vadinasi, Overhauserio kreivė yra kubinis defekto du splainas, interpoliuojantis kontrolinius taškus, ir yra panaši į Akima splainus.

Kaip brėžti Overhauserio kreivę, kurią nusako kontroliniai taškai $p_0, p_1, p_2, \ldots, p_n$?

Tam, kad kreivė prasidėtų taške p_0 ir baigtųsi taške p_n, kontrolinių taškų masyvą papildysime dviem fiktyviais taškais: masyvo pradžioje pakartosime tašką p_0, - gale pakartosime tašką p_n. Toliau, imdami gretimų kontrolinių taškų ketvertus:

$p_i, p_{i+1}, p_{i+2}, p_{i+3}, i=\overline{0,n-1}$, nuosekliai brėšime kreivės segmentus: p_{i+1}, p_{i+2}.

Catmull-Rom kreivės. Overhauserio kreivės yra atskirasis Catmull-Rom kreivių atvejis. Šios kreivės yra interpoliaciniai defekto du splainai, kuriuos apibrėžia kontroliniai taškai $p_0, p_1, p_2, \ldots, p_n$. Kiekvienas gretimų kontrolinių taškų ketvertukas:

$p_i, p_{i+1}, p_{i+2}, p_{i+3}, i=\overline{0,n-3}$ apibrėžia kreivės segmentą: p_{i+1}, p_{i+2}.

Catmull-Rom kreivę nusako tokios bazinės funkcijos:

$f_0(t)=-\alpha t+2\alpha t^2-\alpha t^3, f_1(t)=1+(\alpha-3)t^2+(2-\alpha)t^3,$

$f_2(t)=\alpha t+(3-2\alpha)t^2+(\alpha-2)t^3, f_3(t)=-\alpha t^2+\alpha t^3, 0\le\alpha\le 1,$

$0\le t\le 1.$

Kreivė, kurią apibrėžia kontroliniai taškai $p_i, p_{i+1}, p_{i+2}, p_{i+3}$, užrašoma taip:

$P(t)=p_i f_0(t)+p_{i+1}f_1(t)+p_{i+2}f_2(t)+p_{i+3}f_3(t), 0\le t\le 1.$

Kai $\alpha=\frac{1}{2}$, tai Catmull-Rom kreivė sutampa su Overhauserio kreive.

Kiekviename vidiniame taške $p_i, i=\overline{1,n-1}$ kreivės pirmoji išvestinė lygi $\alpha(p_{i+1}-p_{i-1})$. Reikia pažymėti, kad Catmull-Rom kreivė neguli kontrolinių taškų iškiliajame apvalkale.

Kaip brėžti Catmull-Rom kreivę, kurią nusako kontroliniai taškai $p_0, p_1, p_2, \ldots, p_n$?

Jos brėžimas pilnai sutampa su Overhauserio kreivės brėžimu.

5.18 paveiksle pavaizduotos Overhauserio ir Catmull-Rom ($\alpha=0{,}25, \alpha=0{,}75$) kreivės.

Žemiau pavaizduota funkcija ***plokstkreives***, kuri įgalina apskaičiuoti Timerio, Bolo, Overhauserio ir Catmull-Rom kreives.

```
function [xt,yt]= plokstkreives(x,y,p,alfa);
% PLOKSTKREIVES  - tai funkcija, kuri apskaičiuoja 3-os eiles
% plokštumines Timerio, Bolo, Overhauserio ir Catmull-Rom kreives.
% Įėjimo parametrai
% (x,y) - kontrolinių taškų ketvertas,
% p     - kreivės požymis:
% p=1   - Timerio kreivė;
% p=2   - Bolo kreivė;
```

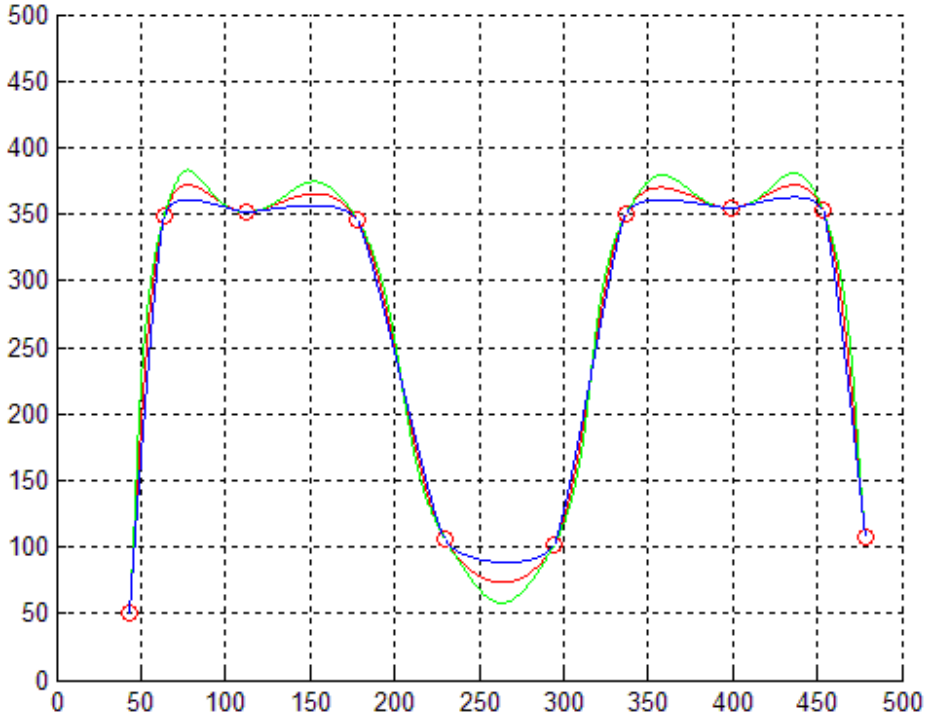
```
% p=3   - Catmull-Rom kreivė;
% alfa  - Catmull-Rom kreivės parametras;
%         alfa yra tarp 0 ir 1.
%         Jei alfa=0.5, tai Overhauserio kreivė;
%         Jei p=1 arba p=2, tai alfa reikšmė lygi nuliui.
% Išėjimo parametrai
% (xt,yt) -  kreivės taškai, apskaičiuoti kai t=0:0.001:1.

switch p
case 1
    n=[3 3 3 3];
    a1=[-2 5 -4 1]; a2=[4 -8 4 0]; a3=[-4 4 0 0]; a4=[2 -1 0 0];
    case 2
        n=[2 3 3 2];
        a1=[1 -2 1]; a2=[2 -4 2 0];  a3=[-2 2 0 0]; a4=[1 0 0];
        case 3
            n=[3 3 3 3];
            a1=[-alfa 2*alfa -alfa 0]; a2=[2-alfa alfa-3 0 1];
            a3=[alfa-2 3-2*alfa alfa 0]; a4=[alfa -alfa 0 0];
            otherwise
                disp('klaidinga p reikšmė');
end
t=0:0.001:1; m=numel(t);
%Kreivės bazinių funkcijų reikšmių apskaičiavimas
b1=horn(n(1),a1,t); b2=horn(n(2),a2,t);
b3=horn(n(3),a3,t); b4=horn(n(4),a4,t);
%Kreivės bazinių funkcijų reikšmių matrica
b=[b1' b2' b3' b4'];
ax=repmat(x,m,1); ay=repmat(y,m,1);
% Kreivės taškų apskaičiavimas
xt=sum((ax.*b)'); yt=sum((ay.*b)');
```

5.3. Dvimatė interpoliacija

5.3.1. Polinominė interpoliacija.

Ankstesniuose paragrafuose aptarėme vienmatę polinominę ir splainininę interpoliacijas. Dabar panagrinėsime, kaip elgtis dvimatės interpoliacijos atveju, kai funkcijos $z = f(x, y)$ reikšmės apibrėžtos reguliaraus arba nereguliaraus stačiakampio tinklelio mazguose (x_i, y_j). Jei interpoliavimo mazgai nesudaro stačiakampio tinklelio, tai paprastai interpoliavimo sritis skaidoma į trikampius, kurių visuma dengia interpoliavimo sritį. Toks srities išskaidymas vadinamas trianguliacija. Interpoliacijos, naudojant trianguliaciją, čia nenagrinėsime.



5.18 pav. Overhauserio (raudona), Catmul-Rom ($\alpha = 0.75$, žalia) ir Catmul-Rom ($\alpha = 0.25$, mėlyna) kreivės

Tarkime, kad žinome funkcijos $z = f(x, y)$ reikšmes stačiakampio tinklelio mazguose $(x_i, y_j) \in [x_1, x_2, \ldots, x_n] \times [y_1, y_2, \ldots, y_n]$ ir reikia apskaičiuoti funkcijos z reikšmę taške $(u, v) \in [x_i, x_{i+1}] \times [y_j, y_{j+1}], i \in \overline{1, n}, j \in \overline{1, m}$.

Šiuo atveju dvimatės interpoliacijos uždavinys keičiamas nuosekliu vienmačiu interpoliavimu, kuris polinominės interpoliacijos atveju atliekamas taip.

1. Pasirenkame interpoliacinio polinomo eilę *k*.
2. Remdamiesi interpoliavimo mazgais (jei galima) $y_{j+t}, t = \overline{-\left\lfloor \frac{k}{2} \right\rfloor, \ldots, \left\lceil \frac{k}{2} \right\rceil}$ ir funkcijos z reikšmėmis $z = f(x_l, y_{j+t})$ interpoliuojame funkciją z visiems $l = \overline{i - \left\lfloor \frac{k}{2} \right\rfloor, \ldots, i + \left\lceil \frac{k}{2} \right\rceil}$, ir apskaičiuojame interpolacinių polinomų reikšmes $fv_l(v)$.

   Pastaba. Jei taškas (u, v) yra tinklelio pakraštyje, tai interpoliavimo taškai nebus simetriškai išsidėstę to taško atžvilgiu.
3. Remdamiesi interpoliavimo mazgais $x_l, l = \overline{i - \left\lfloor \frac{k}{2} \right\rfloor, \ldots, i + \left\lceil \frac{k}{2} \right\rceil}$ ir reikšmėmis $fv_l(v)$, apskaičiuojame interpoliacinio polinomo reikšmę, kai $x = u$. Ši reikšmė ir yra apytikslė ieškomoji funkcijos *z* reikšmė.

Pateikta skaičiavimo schema vaizdžiai matosi 5.19 paveiksle.

Šiame paveiksle pateikta trečio laipsnio dvimatės interpoliacijos schema, norint apskaičiuoti funkcijos $z = f(x, y)$ reikšmę taške $(1.75;2.25)$.

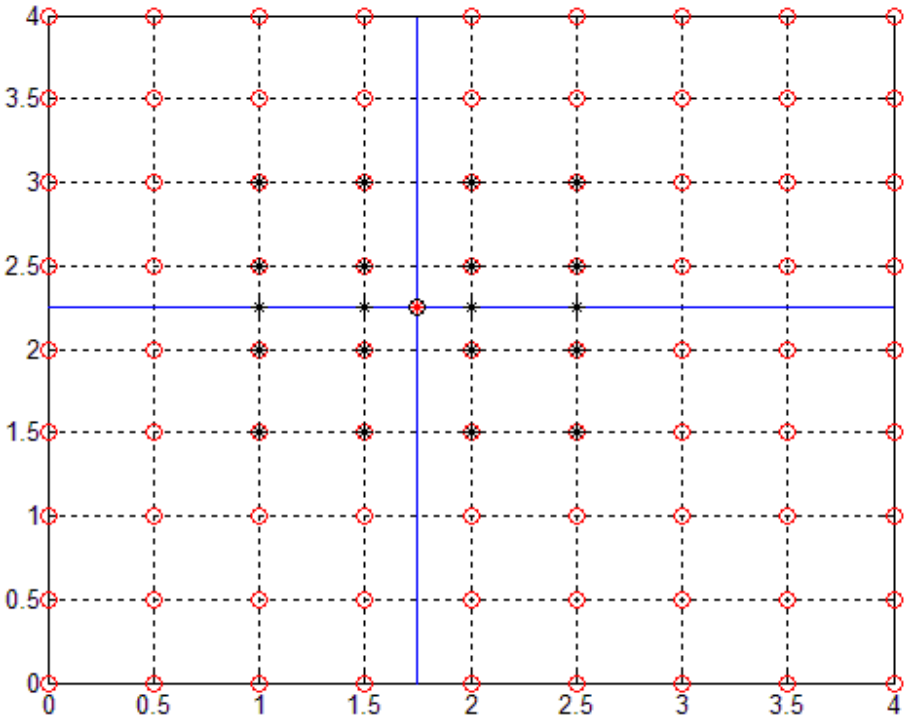
Juodos žvaigždutės raudoname apskritimėlyje žymi interpoliavimo mazgus pagal kintamąjį y. Juodos žvaigždutės horizontalėje $y = 2.25$ žymi interpoliavimo mazgus pagal kintamąjį x, o raudona žvaigždute juodame apskritimėlyje žymi tašką, kuriame reikia apskaičiuoti funkcijos $z = f(x, y)$ reikšmę.

Pavyzdys. Apskaičiuokime funkcijos $z = \frac{x^2}{2} - \frac{y^2}{2}$ reikšmę taške $(1.75;2.25)$, kai žinomos funkcijos reikšmes stačiakampio tinklelio $0 \le x \le 4, 0 \le y \le 4$, kurio žingsnis tiek x, tiek ir y ašies kryptimis lygus 0.5 (žr. 5.19 pav.).

Funkcijos reikšmės tinklelio mazguose yra:

```
z =
    0.0000   0.1250   0.5000   1.1250   2.0000   3.1250   4.5000   6.1250   8.0000
   -0.1250   0.0000   0.3750   1.0000   1.8750   3.0000   4.3750   6.0000   7.8750
   -0.5000  -0.3750   0.0000   0.6250   1.5000   2.6250   4.0000   5.6250   7.5000
   -1.1250  -1.0000  -0.6250   0.0000   0.8750   2.0000   3.3750   5.0000   6.8750
   -2.0000  -1.8750  -1.5000  -0.8750   0.0000   1.1250   2.5000   4.1250   6.0000
   -3.1250  -3.0000  -2.6250  -2.0000  -1.1250   0.0000   1.3750   3.0000   4.8750
   -4.5000  -4.3750  -4.0000  -3.3750  -2.5000  -1.3750   0.0000   1.6250   3.5000
   -6.1250  -6.0000  -5.6250  -5.0000  -4.1250  -3.0000  -1.6250   0.0000   1.8750
   -8.0000  -7.8750  -7.5000  -6.8750  -6.0000  -4.8750  -3.5000  -1.8750   0.0000
```

Apskaičiuotos $fv_l(2.25)$ reikšmės yra: `-2.0313  -1.4063  -0.5313  0.5938.` Tada, interpoliuodami šias reikšmes ir imdami x=1.75, apskaičiuosime ieškomą funkcijos z reikšmę taške $(1.75;2.25)$. Ši reikšmė lygi -1 ir tai yra tiksli funkcijos z reikšmė. Kadangi hiperbolinis paraboloidas $z = \frac{x^2}{2} - \frac{y^2}{2}$ yra antros eilės paviršius, todėl taikant net antros eilės interpoliaciją būtume gavę tikslų atsakymą.



5.19 pav. Dvimatės polinominės interpoliacijos schema

5.3.2. Dvimatė splaininė interpoliacija.

Dvimatės interpoliacijos atveju, kai funkcijos $z = f(x, y)$ reikšmės apibrėžtos stačiakampio tinklelio mazguose (x_i, y_j), vietoje polinomų galime taikyti splainus. Tačiau interpoliuojant splainais, skirtingai nei naudojant polinomus, reikia papildomai nurodyti kraštines sąlygas. Todėl šiuo atveju patogiau visą paviršių aproksimuoti, pavyzdžiui, bikubiniu splainu.

Bikubinio interpoliacinio splaino konstravimas.

Uždavinio formulavimas. Tarkime, kad srityje D turime tinklelį

$$D_h = \{x_j, y_i : a = x_0 < x_1 < ... < x_n = b;\ c = y_0 < y_1 < ... < y_m = d\}$$

ir funkcijos $z = f(x, y)$ reikšmes jo mazguose. Reikia apskaičiuoti bikubinį splainą $g = g(x, y)$, tenkinantį šias sąlygas:

1) $g(x, y) \in C^2(D)$;

2) kiekviename langelyje (j, i): $x_j \le x \le x_{j+1}$, $y_i \le y \le y_{i+1}$,

$$g(x, y) = \sum_{k,l=0}^{3} a_{kl}^{ji} (x_j - x)^k (y_i - y)^l \ ;$$

3) $g(x_j, y_i) = f(x_j, y_i) = z_{ji},\ j = \overline{0, n},\ i = \overline{0, m}$.

Kad šis splainas būtų apibrėžtas vienareikšmiškai, kaip ir vienmačiu atveju, turime reikalauti, kad $g(x, y)$ tenkintų kraštines sąlygas, kurios yra analogiškos vienmačio kubinio splaino kraštinėms sąlygoms. 5.20 paveiksle parodyta šio uždavinio iliustracija, kai sritis D yra stačiakampė ir jos kontūro taškuose galioja sąlyga $\frac{\partial g}{\partial x} = \frac{\partial f}{\partial x}$, $\frac{\partial g}{\partial y} = \frac{\partial f}{\partial y}$. Aptarsime, kaip apskaičiuojami bikubiniai interpoliaciniai splainai, užrašyti tiesiniu B splainų dariniu.

Bikubinis splainas užrašomas formule

$$g(x, y) = \sum_{i=-3}^{m-1} \sum_{j=-3}^{n-1} b_{ij} B_j(x) \widetilde{B}_i(y); \qquad (5.20)$$

čia $B_j(x)$ — kubinis B splainas, užrašytas taško x_j atžvilgiu ir apibrėžiamas tinkleliu $x_0 < x_1 < ... < x_n$, $\widetilde{B}_i(y)$ — kubinis B splainas, užrašytas taško y_i atžvilgiu ir apibrėžiamas tinkleliu $y_0 < y_1 < ... < y_m$, b_{ij} — koeficientai.

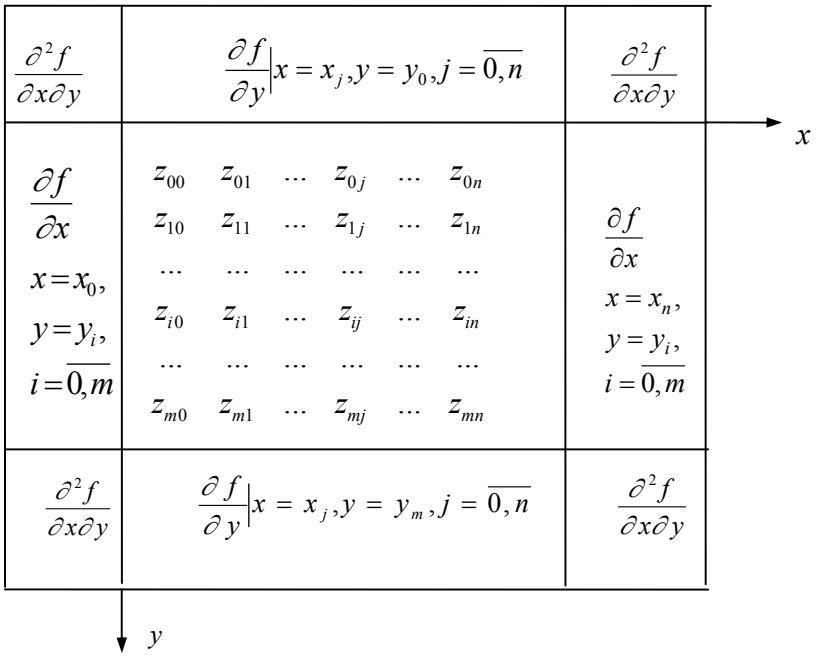
Imkime splaino funkcijų sistemą

$$v_j(y) = \sum_{i=-3}^{m-1} b_{ij} \widetilde{B}_i(y),\ j = \overline{-3, n-1}\ . \qquad (5.21)$$

Tada (5.20) splainą galėsime užrašyti formule

$$g(x, y) = \sum_{j=-3}^{n-1} v_j(y) B_j(x) \qquad (5.22)$$

ir apskaičiuoti taip.

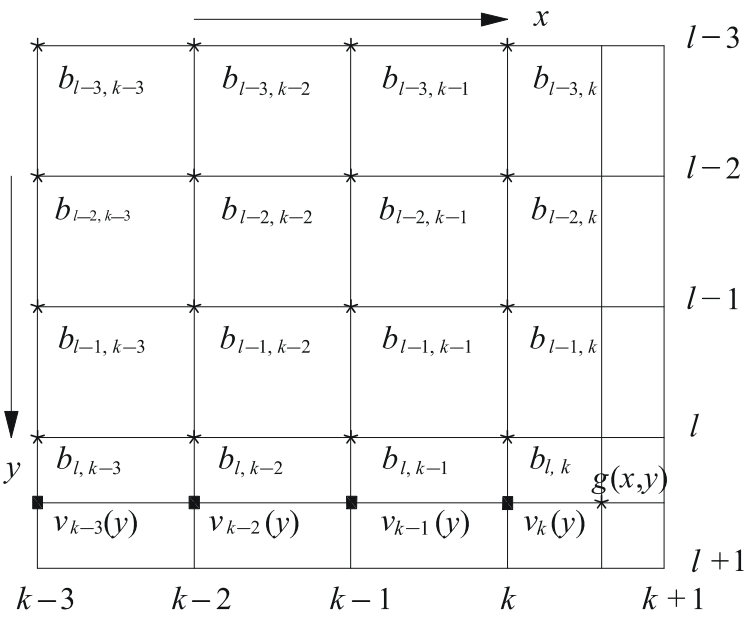


5.20 pav. Sąlygos, nusakančios bikubinį interpoliacinį splainą, kai naudojamos III tipo kraštinės sąlygos

Pirmasis žingsnis. Taikydami vienmačio interpoliacinio splaino apskaičiavimo funkciją ***bnspln*** (žr. 5.1.2.3 paragrafą), randame kubinius interpoliacinius splainus pagal visas lentelės (žr. 5.20 pav.) eilutes $i = \overline{0, m}$, taip pat pagal eilutes $\left.\frac{\partial f}{\partial y}\right|_{\substack{x = x_j,\\ y = y_0}}$ $(i = -1)$ ir $\left.\frac{\partial f}{\partial y}\right|_{\substack{x = x_j,\\ y = y_m}}$ $(i = m+1)$. Šio žingsnio rezultatas — apskaičiuoti koeficientai v_{ij} $\left(i = \overline{-1, m+1},\ \ j = \overline{-3, n-1}\right)$, nusakantys kubinius splainus pagal visas lentelės eilutes.

Antrasis žingsnis. Naudodami koeficientus v_{ij} ir vienmačio interpoliacinio splaino apskaičiavimo funkciją ***bnspln***, randame kubinius interpoliacinius splainus pagal visus stulpelius $j = \overline{-3, n-1}$, kai v_{ij} $\left(i = \overline{0, m}\right)$ yra funkcijos reikšmės, o $v_{-1\,j}$ ir $v_{m+1,\,j}$ — kraštinės sąlygos. Šio žingsnio rezultatas — matrica $\left[b_{ij}\right]$ $\left(i = \overline{-3, m-1},\ j = \overline{-3, n-1}\right)$, nusakanti interpoliacinį bikubinį splainą.

Kaip apskaičiuoti $g(x, y)$ reikšmę, kai taškas $(x; y)$ priklauso sričiai D ir $x_k \le x \le x_{k+1},\ \ y_l \le y \le y_{l+1}$? Šis uždavinys lengvai sprendžiamas turint vienmačio splaino apskaičiavimo funkciją ***bg*** (žr. 5.1.2.2 paragrafą). Tarkime, kad $x \in \left[x_k; x_{k+1}\right]$. Tada, atsižvelgdami į tai, kad kubinis B splainas nelygus nuliui tik keturiuose intervaluose, iš pradžių pagal (5.21) formulę apskaičiuojame $v_j(y)$ $\left(\text{čia } j = \overline{k-3, k}\right)$, paskui pagal (5.22) formulę — $g(x, y)$. Kaip randamas $g(x, y)$, parodyta 5.21 paveiksle.



5.21 pav. $g(x,y)$ *apskaičiavimas*

Žemiau pateikta funkcijos ***dbnspln***, kuri apskaičiuoja bikubinio interpoliacinio splaino koeficientus, o taip pat funkcija ***dbg***, kuri apskaičiuoja šio splaino reikšmę taške (u,v).

```
function b=dbnspln(n,m,rx,ry,xk0,xkn,yk0,ykm,m00,m10,m01,m11,x,y,z)
% DBSPLN apskaičiuoja 3-os eilės dvimatį interpoliacinį splainą,
% kai interpoliavimo mazgai sudaro stačiakampį tinklelį.
% Įėjimo parametrai
% n   - tinklelio stulpelių skaičius,
% m   - tinklelio eilučių skaičius,
% rx  - kraštinių sąlygų išvestinės  eilė pagal x,
% ry  - kraštinių sąlygų išvestinės  eilė pagal y,
% xk0 - dalinių išvestinių pagal x reikšmės taškuose
%        (x(0),y(i))i=1,...,m,
% xkn - dalinių išvestinių pagal x reikšmės taškuose
%        (x(n),y(i))i=1,...,m,
% yk0 - dalinių išvestinių pagal y reikšmės taškuose
%        (x(j),y(0))j=1,...,n,
% ykm - dalinių išvestinių pagal y reikšmės taškuose
%        (x(j),y(m))j=1,...,n,
% m00,m10,m01,m11 - daliniu išvestinių pagal x ir y reikšmės
%     mazguose: x(0),y(0)),(x(n),y(0)),(x(0),y(m)),(x(n),y(m)),
% x   - stačiakampio tinklelio mazgų abscisių masyvas,
% y   - stačiakampio tinklelio mazgų ordinačių masyvas,
% z   - dviejų kintamųjų funkcijos reikšmių matrica tinklelio
%        mazguose.
% Išėjimo parametrai
% b - bikubinio splaino koeficientai.

if (rx<=0)|(rx>=3)
    disp('netinkama rx reiksme'); return;
end
if (ry<=0)|(ry>=3)
    disp('netinkama ry reiksme'); return;
```

```
end
%  Matricos v formavimas
for i=1:m
    [temp,poz]=bnspln(3,[rx rx],[xk0(i) xkn(i)],x,z(i,:));
    if poz == 1
        v(i,:)=temp';
    else
        displ('splainas neegzistuoja'); return;
    end
end
[v0,poz]=bnspln(3,[rx rx],[m00 m10],x,yk0);
[vm,poz]=bnspln(3,[rx rx],[m01 m11],x,ykm);
% Matricos b formavimas
for j=1:n+2
    [temp,poz]=bnspln(3,[ry ry],[v0(j) vm(j)],y,v(:,j));
    if poz == 1
        b(:,j)=temp;
    else
        displ('splainas neegzistuoja'); return;
    end
end

function zuv=dbg(x,y,rx,ry,b,u,v)
% GBG funkcija apskaičiuoja bikubinio splaino
% dalinės (rx+ry)išvestinės reikšmė taške (u,v).
% Įėjimo parametrai
% x    - tinklelio x mazgai,
% y    - tinklelio y mazgai,
% rx   - dalinės išvestinės pagal x eilė,
% ry   - dalinės išvestines pagal y eilė,
% b    - bikubinį splainą nusakanti koeficientų matrica,
%(u,v) - taškas, kuriame apskaičiuojama ieškomoji reikšmė.
% Išėjimo parametrai
% zuv  - apskaičiuota ieškomoji reikšmė.

n=numel(x); m=numel(y);
if (rx<0)
    disp('neigiama rx reikšmė'); return;
end
if (ry<0)
    disp('neigiama ry reikšmė'); return;
end
if (u<x(1))|(u>x(n))
    disp('netinkama u reikšmė'); return;
end
if (v<y(1))|(v>y(m))
    disp('netinkama v reikšmė'); return;
end
% Kuriam intervalui priklauso u?
t=repmat(u,1,n);
ind=n-sum((x>=t));
if ind==0
    ind=1;
end;
fr=size(b); fv=zeros(1,fr(1));
for k=ind:ind+3
    [grt,temp]=bg(3,ry,y,b(:,k),v);
```

```
    fv(k)=grt;
end
[grt,temp]=bg(3,rx,x,fv,u);
zuv=grt;
```

Pavyzdys. Apskaičiuokime funkcijos $z=\frac{x^2}{2}-\frac{y^2}{2}$ reikšmę taške $(1.75;2.25)$, kai žinomos funkcijos reikšmes tinklelio $x\times y$, čia *x*=[0, 1, 3, 4, 6, 7], *y*=[0, 1, 2, 3, 5, 8], mazguose.

Spręsdami šį uždavinį, pirmiausia, funkciją $z=\frac{x^2}{2}-\frac{y^2}{2}$ aproksimuosime bikubiniu interpolaciniu splainu , o po to, apskaičiuosime to splaino reikšmę taške $(1.75;2.25)$. Funkcijos reikšmės tinklelio mazguose yra:

```
z =
    0.0000    0.5000    4.5000    8.0000   18.0000   24.5000
   -0.5000    0.0000    4.0000    7.5000   17.5000   24.0000
   -2.0000   -1.5000    2.5000    6.0000   16.0000   22.5000
   -4.5000   -4.0000    0.0000    3.5000   13.5000   20.0000
  -12.5000  -12.0000   -8.0000   -4.5000    5.5000   12.0000
  -32.0000  -31.5000  -27.5000  -24.0000  -14.0000   -7.5000.
```

Pasinaudodami funkcija ***dbnspln*** ir kraštines sąlygas sutapatindami su tiksliomis funkcijos $z=\frac{x^2}{2}-\frac{y^2}{2}$ dalinių išvestinių reikšmėmis, turėsime tokius bikubinio splaino koeficientus:

```
b =
   -0.0000   -0.5000    0.1667    2.8333    8.6667   15.3333   24.0000   31.5000
    0.5000   -0.0000    0.6667    3.3333    9.1667   15.8333   24.5000   32.0000
   -0.0000   -0.5000    0.1667    2.8333    8.6667   15.3333   24.0000   31.5000
   -1.5000   -2.0000   -1.3333    1.3333    7.1667   13.8333   22.5000   30.0000
   -4.8333   -5.3333   -4.6667   -2.0000    3.8333   10.5000   19.1667   26.6667
  -12.8333  -13.3333  -12.6667  -10.0000   -4.1667    2.5000   11.1667   18.6667
  -30.1667  -30.6667  -30.0000  -27.3333  -21.5000  -14.8333   -6.1667    1.3333
  -58.6667  -59.1667  -58.5000  -55.8333  -50.0000  -43.3333  -34.6667  -27.1667
```

Remdamiesi funkcija ***dbg,*** apskaičiuosime bikubinio splaino reikšmę taške $(1.75,2.25)$. Splaino reikšmė, kaip ir turėjo būti, yra lygi -1, ir tai yra tiksli funkcijos *z* reikšmė tame taške.

5.4. Paviršių konstravimas

5.4.1. Splaininių paviršių konstravimas.

Tarkime, kad *n*-osios eilės splaininio paviršiaus kontroliniai taškai nusakyti matrica $P=\left[p_{ij}\right], i=\overline{1,M}, j=\overline{1,N}$, čia $p_{ij}=(x_{ij},y_{ij},z_{ij})$. Tada šio paviršiaus dalį, kurią nusako kontroliniai taškai, patenkantys į $(n+1)\times(n+1)$ formato submatricą $L=\left[p_{s,t}\right],\quad s=\overline{i,i+n},\quad t=\overline{j,j+n}$, sudarys taškai

$$t(u,v)=(x(u,v),y(u,v),z(u,v)),\ 0\le u\le 1,\ 0\le v\le 1,$$

apskaičiuoti pagal formulę

$$t(u,v)=\sum_{k=1}^{n+1}\left(B_n^{[n+1-k,n+2-k]}(v)\cdot\left(\sum_{l=1}^{n+1}\left(p_{i+l,j+k}\cdot B_n^{[n+1-l,n+2-l]}(u)\right)\right)\right). \qquad (5.23)$$

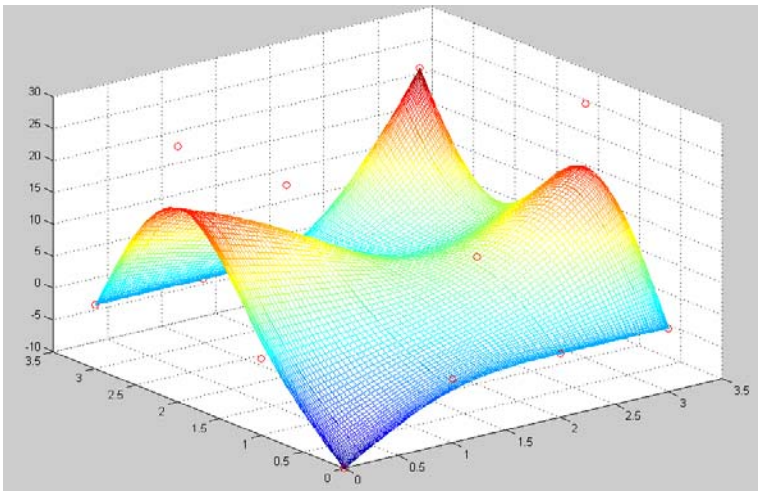
Iš šios formulės matyti, kad pirmiausia apskaičiuojami taškai splaininių kreivių, kurias apibrėžia submatricos $L=\left[p_{s,t}\right],\quad s=\overline{i,i+n},\quad t=\overline{j,j+n}$ stulpeliai, o po to, apskaičiuojamos reikšmės splaininių kreivių, kurias apibrėžia tie apskaičiuoti taškai.

Aišku, kad konstruojant paviršių, submatrica L turi nuosekliai (po vieną stulpelį iš kairės į dešinę ir po vieną eilutę iš viršaus į apačią) slinkti per matricą P.

Norint, kad kontroliniai taškai $p_{11}, p_{1N}, p_{M1}, p_{MN}$ priklausytų paviršiui, matricą P reikia papildyti, įvedant viršuje ir apačioje po (n-1) eilutę, o kairėje ir dešinėje – po (n-1) stulpelį. Papildomos eilutės viršuje pakartoja matricos P pirmąją, o apačioje – paskutiniąją eilutę. Papildomi stulpeliai kairėje pakartoja matricos P pirmąjį, o dešinėje – paskutinįjį stulpelį.

Žemiau pateikta funkcija ***splpavirsius***, kuri konstruoja splaininį paviršių, nusakomą Debūro taškais, o 5.22 paveiksle pavaizduotas trečios eilės splaininis paviršius, kurį apibrėžia Debūro taškų matricos:

$$xp = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{pmatrix}, \quad yp = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{pmatrix}, \quad zp = \begin{pmatrix} -10 & 0 & 0 & 0 \\ 2 & 3 & 10 & 30 \\ 30 & 20 & 0 & 0 \\ 0 & 0 & 0 & 25 \end{pmatrix}.$$



5.22 pav. Splaininis paviršius ir Debūro taškai (pažymėti raudonai).

```
function splpavirsius(n,xp,yp,zp)
%  SPLPAVIRŠIUS- tai funkcija, kuri apskaičiuoja splaininį paviršių,
%  nusakoma Debūro taškais.
% Įėjimo parametrai
%  n   - paviršiaus  eilė,
%  xp  - Debūro taškų x-u matrica,
%  yp  - Debūro taškų y-u matrica,
%  zp  - Debūro taškų z-u matrica.

% Debūro taškų papildymas fiktyviais taškais,
% kad paviršius eitų per kampinius Debūro taškus.
[mm,nn]=size(xp);
xt=[repmat(xp(1,:),n-1,1); xp; repmat(xp(mm,:),n-1,1)];
xx=[repmat(xt(:,1),1,n-1) xt repmat(xt(:,nn),1,n-1)];
yt=[repmat(yp(1,:),n-1,1); yp; repmat(yp(mm,:),n-1,1)];
yy=[repmat(yt(:,1),1,n-1) yt repmat(yt(:,nn),1,n-1)];
zt=[repmat(zp(1,:),n-1,1); zp; repmat(zp(mm,:),n-1,1)];
zz=[repmat(zt(:,1),1,n-1) zt repmat(zt(:,nn),1,n-1)];
```

```
h=0.05; u=0:h:1; m=numel(u); a=zeros(m,n+1); p=zeros(m,n+1);
% B splaino reikšmių apskaičiavimas, kai t=0:0.05:1.
t=u'; b=zeros(1,n+1); b(1)=1; a=repmat(b,m,1);
for l=1:n
 for k=l+1:-1:1
  if k~=1
   a(:,k)=((t+k-1).*a(:,k)+(l+2-k-t).*a(:,k-1))/l;
  else
   a(:,k)=(t+k-1).*a(:,k)/l;
  end
 end
end
%Paviršiaus brėžimas
for i=1:mm+n-2
 for j=1:nn+n-2
 % (n+1)*(n+1) formato matricos išskirimas
  x=xx(i:i+n,j:j+n);  y=yy(i:i+n,j:j+n);  z=zz(i:i+n,j:j+n);
 % Paviršiaus dalies, kurią nusako išskirta matrica,
 % taškų apskaičiavimas
for k=1:n+1
  c=x(:,k)'; d=c(end:-1:1); p=repmat(d,m,1); xu(k,:)=sum((a.*p)');
  c=y(:,k)'; d=c(end:-1:1); p=repmat(d,m,1); yu(k,:)=sum((a.*p)');
  c=z(:,k)'; d=c(end:-1:1); p=repmat(d,m,1); zu(k,:)=sum((a.*p)');
end
for k=1:m
  c=xu(:,k)'; d=c(end:-1:1); p=repmat(d,m,1); xuv(k,:)=sum((a.*p)');
  c=yu(:,k)'; d=c(end:-1:1); p=repmat(d,m,1); yuv(k,:)=sum((a.*p)');
  c=zu(:,k)'; d=c(end:-1:1); p=repmat(d,m,1); zuv(k,:)=sum((a.*p)');
end
% Apskaičiuotos paviršiaus dalies brėžimas
mesh(xuv,yuv,zuv)
hold on
 end
end
% Debūro taškų vaizdavimas
hold on
plot3(xp,yp,zp,'or')
```

5.4.2. Bezjė paviršiaus konstravimas.

Bezjė paviršius konstruojamas analogiškai, kaip ir splaininis paviršius. Šį paviršių nusako tos pačios kontrolinių taškų matricos: xp, yp, zp. Tačiau šiuo atveju, skirtingai nei splaininio paviršiaus atveju, jų nereikia papildyti fiktyviais taškais. Be to, konstruojant *n*-os eilės Bezjė paviršių, $(n+1)\times(n+1)$ formato submatrica $L=\left[p_{s,t}\right]$, $s=\overline{i,i+n}$, $t=\overline{j,j+n}$ turi nuosekliai slinkti per matricą *P*, šokdama per *n* stulpelių, judant iš kairės į dešinę, ir per *n* eilučių, slenkant iš viršaus žemyn.

Aišku, kad *n*-os eilės Bezjė paviršiaus konstravimo pagrindas yra Kastelžo algoritmas, kuris įgalina efektyviai apskaičiuoti *n*-os eilės Bezjė kreivę, kurią apibrėžia duoti kontroliniai taškai.

Žemiau pateikta funkcija ***bezjepavirsius***, kuri konstruoja Bezjė paviršių, nusakomą kontrolinių taškų matricomis xp, yp, zp.

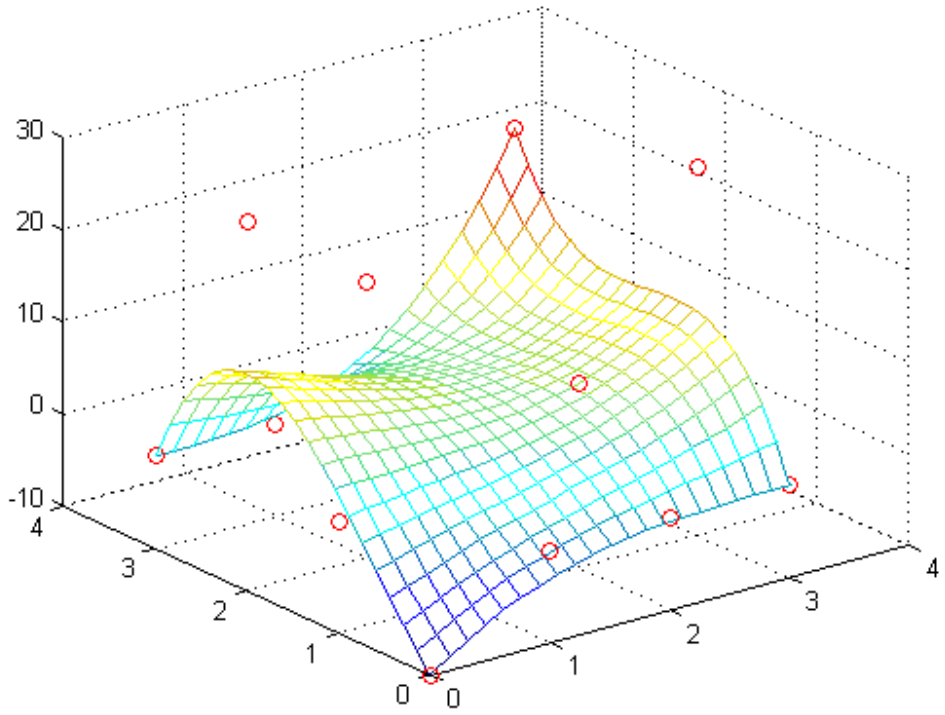
```
function bezjepavirsius(n,xp,yp,zp)
% BEJEPAVIRSIUS - tai funkcija, kuri apskaičiuoja Bezjė paviršių,
```

```
% nusakomą kontroliniais taškais (xp,yp,zp).
% Įėjimo parametrai
% n   - paviršiaus  eilė,
% xp  - kontrolinių taškų x-u matrica,
% yp  - kontrolinių taškų y-u matrica,
% zp  - kontrolinių taškų z-u matrica.

[mm,nn]=size(xp); tn=nn-n-1; tm=mm-n-1;
if (mm < n+1)|(nn < n+1)
    disp('per mažas eilučių arba stulpelių skaičius')
    return
end
if (mod(tm,n)~=0)|(mod(tn,n)~=0)
    disp('neteisingas matricos formatas')
    return
end
h=0.05; u=0:h:1;t=u'; m=numel(u);
%Paviršiaus brėžimas
for i=1:n:mm-n
 for j=1:n:nn-n
% (n+1)*(n+1) formato matricos išskirimas
  x=xp(i:i+n,j:j+n); y=yp(i:i+n,j:j+n); z=zp(i:i+n,j:j+n);
% Paviršiaus dalies, kurią nusako išskirta matrica,
% taškų apskaičiavimas
for k=1:n+1
    c=x(:,k)'; xu(:,k)=kastelzo(n,c,t);
    c=y(:,k)'; yu(:,k)=kastelzo(n,c,t);
    c=z(:,k)'; zu(:,k)=kastelzo(n,c,t);
end
for k=1:m
    c=xu(k,:); xt=kastelzo(n,c,t); xuv(k,:)=xt';
    c=yu(k,:); yt=kastelzo(n,c,t); yuv(k,:)=yt';
    c=zu(k,:); zt=kastelzo(n,c,t); zuv(k,:)=zt';
end
% Apskaičiuotos paviršiaus dalies brėžimas
mesh(xuv,yuv,zuv)
hold on
 end
end
% Kontrolinių taškų vaizdavimas
hold on
plot3(xp,yp,zp,'or')
% Antrinė funkcija "kastelzo"
function xt=kastelzo(n,x,t)
% kastelzo  - tai funkcija, kuri apskaičiuoja n-os eiles
% Bezjė kreives taškų abscises.
% Įėjimo parametrai
% n     - kreivės eilė,
% x     - kontrolinių taškų abscisės,
% t     - parametras, apibrėžiantis Bezjė kreivės taškus.
% Išėjimo parametrai
% xt    - Bezjė kreivės taškų abscisės nurodytoms t reikšmėms.

m=numel(t); ax=repmat(x,m,1);
for j=1:n
 for i=1:n+1-j
   ax(:,i)=(1-t).*ax(:,i)+t.*ax(:,i+1);
```

```
end
end
xt=ax(:,1);
```



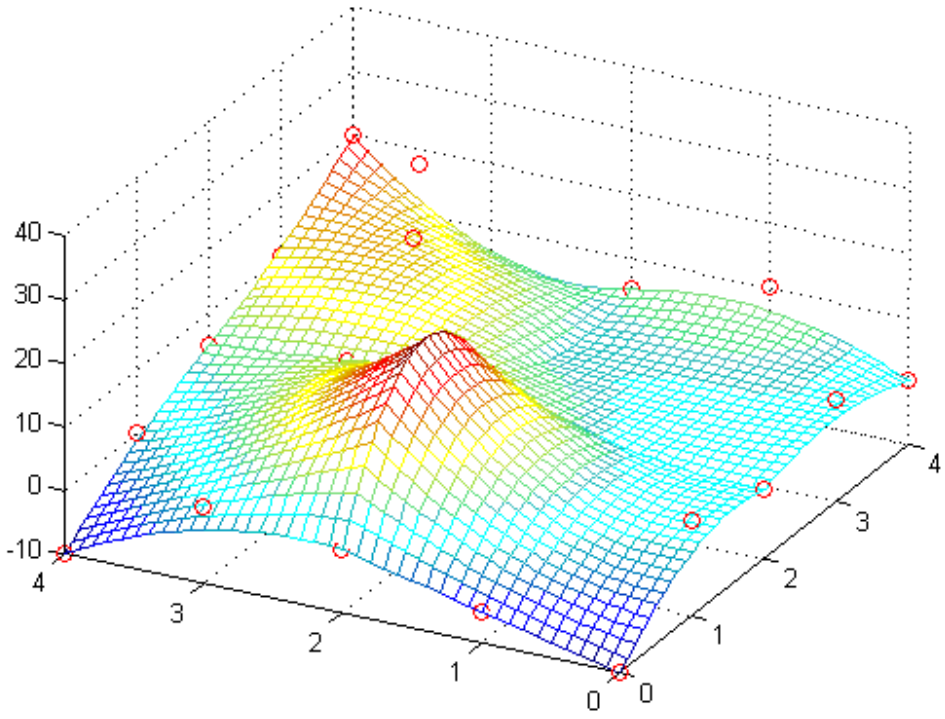
5.23 pav. Trečios eilės Bezjė paviršius ir kontroliniai taškai (pažymėti raudonai).

5.23 paveiksle pavaizduotas trečios eilės Bezjė paviršius, kurį apibrėžia tos pačios matricos, kaip ir 5.22 paveiksle pavaizduotą splaininį paviršių.

5.24 paveiksle pavaizduotas antros eilės Bezjė paviršius, kurį apibrėžia matricos:

$$xp = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}, \quad yp = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{pmatrix}, \quad zp = \begin{pmatrix} -10, 5, 1, 6, 0 \\ -5, -2, 0, 1, 10 \\ 0, 40, 10, 0, 5 \\ 2, 5, 7, 29, 0 \\ -10, 0, 5, 10, 20 \end{pmatrix}.$$

Kaip matyti iš 5.24 paveikslo, Bezjė paviršius, kaip ir kreivė, nėra glodus.



5.24 pav. Antros eilės Bezjė paviršius ir kontroliniai taškai (pažymėti raudonai).

Pastaba. Bendru atveju Bezjė paviršius gali būti konstruojamas iš bilaipsnio $m \times n$ Bezjė skiaučių. Tokio paviršiaus konstravimas iš principo nesiskiria nuo aptarto *n*-os eilės Bezjė paviršiaus konstravimo.

5.5. *Suglodinimo metodas*

Nagrinėdami interpoliavimo uždavinį, funkciją $f(x)$, apibrėžtą reikšmių lentele, stengėmės pakeisti tokia aproksimuojančiąja funkcija $F(x)$ (*n*-tojo laipsnio polinomu, splainu ir pan.), kad taškuose x_i $\left(i = \overline{0,n}\right)$ $F(x_i)$ reikšmės būtų lygios $f(x_i)$ reikšmėms.

Labai dažnai $f(x_i)$ reikšmės yra eksperimento rezultatai ir turi matavimo bei metodo paklaidų. Todėl reikalauti, kad aproksimuojančioji funkcija $F(x)$ tenkintų sąlygą $F(x_i) = f(x_i)$ $\left(i = \overline{0,n}\right)$, būtų neprotinga. Geriau rasti tokią funkciją $F(x)$, kuri pagal pasirinktą kriterijų geriausiai aproksimuotų $f(x)$. Toks funkcijos $F(x)$ apskaičiavimo - metodas vadinamas suglodinimu. Atsižvelgiant į suglodinimo kriterijų, galima gauti įvairias $F(x)$ išraiškas.

Suglodinimo uždavinio formulavimas. *Sakykime, funkcija* $y = f(x)$ *nusakyta reikšmių lentele* $\left(x_i, \tilde{y}_i\right)$*; čia* $i = \overline{1,m}$*. Simbolis* $\tilde{y}_i$ *rodo, kad* $f(x)$ *reikšmės taškuose* x_i *yra apytikslės. Taip pat žinoma aproksimuojančiosios funkcijos* $F(x)$ *analizinė išraiška:* $F\left(x, a_0, \ldots, a_n\right)$*; čia* a_i $\left(i = \overline{0,n}\right)$ *— nežinomi parametrai ir* $n << m$*. Reikia rasti tokias parametrų* a_i *reikšmes, su kuriomis* $F(x)$ *„geriausiai“ pagal pasirinktą kriterijų*

aproksimuotų funkciją $f(x)$.

Sprendžiant šį uždavinį, taikomi įvairūs suglodinimo kriterijai, o kartu ir įvairūs suglodinimo metodai.

Pasirinktų taškų metodas. Taikant šį metodą, iš lentelės $\left(x_i, \widetilde{y}_i\right)$ (čia $i = \overline{1, m}$) pasirenkamas $n+1$ taškas $\left(x_{i_k}; \widetilde{y}_{i_k}\right)$ (čia $k = \overline{1, n}$), kurio y_{i_k} reikšmės yra tiksliausios, ir parametrai a_k $\left(k = \overline{0, n}\right)$ apskaičiuojami atsižvelgiant į sąlygą $F\left(x_{i_k}\right) = \widetilde{y}_{i_k}$, $k = \overline{0, n}$.

Aišku, kad taip apskaičiuota funkcija $F(x)$ sutampa su interpoliuojančiąja funkcija, einančia per taškus $\left(x_{i_k}; \widetilde{y}_{i_k}\right)$, $k = \overline{0, n}$.

Vidurkių metodas. Šiuo metodu parametrai a_k $\left(k = \overline{0, n}\right)$ apskaičiuojami taip, kad $f\left(x_i\right)$ ir $F\left(x_i\right)$ skirtumų suma būtų lygi nuliui, t. y.

$$z = \sum_{i=1}^{m}\left(f\left(x_i\right) - F\left(x_i\right)\right) = 0.$$

Minimakso metodas. Šiuo metodu parametrai a_k $\left(k = \overline{0, n}\right)$ apskaičiuojami taip, kad $f\left(x_i\right)$ ir $F\left(x_i\right)$ didžiausias skirtumo modulis būtų galimai mažesnis, t.y.

$z = \min\max\left\{\left|f(x_i) - F(x_i)\right|\right\}$ visiems $i = \overline{1, m}$. Minimakso kriterijus retai naudojamas, nes:

- per didelį svorį skiria didžiausią paklaidą turinčiam taškui,
- uždavinys neišsprendžiamas elementariomis priemonėmis.

Absoliučiojo nuokrypio metodas. Šiuo metodu parametrai a_k $\left(k = \overline{0, n}\right)$ apskaičiuojami taip, kad nuokrypių modulių suma būtų mažiausia, t.y.

$$z = \min\sum_{i=1}^{m}\left|f\left(x_i\right) - F\left(x_i\right)\right|.$$

Absoliučiojo nuokrypio metodas, nesuteikdamas nei vienam taškui per daug svorio, suvidurkina nuokrypas. Norint šiuo metodu apskaičiuoti parametrų a_k $\left(k = \overline{0, n}\right)$ reikšmes, reikia apskaičiuoti funkcijos *z* dalines išvestines pagal šiuos parametrus ir jas prilyginti nuliui. Pagrindinis metodo trūkumas yra tas, kad neegzistuoja absoliučios reikšmės funkcijos išvestinė taške nulis. Tai reiškia, kad šio uždavinio sprendinys nebūtinai egzistuoja.

Mažiausių kvadratų metodas. Tai dažniausiai taikomas suglodinimo metodas. Jis formuluojamas taip: koeficientus a_k $\left(k = \overline{0, n}\right)$ reikia apskaičiuoti taip, kad $f\left(x_i\right)$ ir $F\left(x_i\right)$ skirtumų kvadratų suma būtų pati mažiausia, t. y. reikia minimizuoti

$$z = \sum_{i=1}^{m}\left(F\left(x_i, a_0, \ldots, a_n\right) - \widetilde{y}_i\right)^2. \qquad (5.24)$$

Nors mažiausių kvadratų metodas suteikia daugiau svorio blogai išmatuotam taškui, tačiau neleidžia jam dominuoti kitų taškų atžvilgiu. Be to, šio uždavinio sprendinys visada egzistuoja. Todėl praktikoje šis metodas daugiausia naudojamas.

Toliau nagrinėsime mažiausių kvadratų metodą.

(5.24) formule nusakyta tikslo funkcija turi vienintelį ekstremumą, kuris apskaičiuojamas iš lygčių sistemos

$$\frac{\partial z}{\partial a_k} = 0,\ k = \overline{0, n}. \qquad (5.25)$$

Bendruoju atveju ši sistema yra netiesinė, taigi ją galima spręsti taikant netiesinių lygčių sistemų sprendimo metodus, išnagrinėtus trečiajame skyriuje.

(5.25) sistemos formavimą ir sprendimą galima palengvinti dvejopai:

1) jei $F\left(x, a_0, \ldots, a_n\right)$ yra *n*-tojo laipsnio polinomas, tai (5.25) lygčių sistema yra tiesinė ir jos formavimas bei sprendimas nesudaro sunkumų;

2) galima taikyti vadinamąjį ištiesinimo metodą, kurio esmė tokia: atitinkamai parinktoje koordinačių sistemoje $\left(X, Y\right)$ taškai $\left(x_i; \widetilde{y}_i\right)$ $\left(\text{čia } i = \overline{1, m}\right)$ apytiksliai tenkina tiesės $Y = kX + b$ lygtį, tada mažiausių kvadratų metodas toje sistemoje realizuojamas labai paprastai.

Pavyzdžiui, tarkime, kad taškai $\left(x_i; \widetilde{y}_i\right)$ tenkina dėsnį

$$y = ae^{cx}. \qquad (5.26)$$

Logaritmuodami abi šios lygybės puses, gauname: $\ln y = \ln a + cx$.

Pažymėkime: $X = x,\ Y = \ln y,\ b = \ln a$. Tada (5.26) lygtį koordinačių sistemoje $\left(X, Y\right)$ galėsime užrašyti taip: $Y = cX + b$.

Ištiesinimo metodas gali būti taikomas, norint nustatyti funkcijos $F\left(x, a_0, \ldots, a_n\right)$ analizinę išraišką, t. y. sužinoti, kokį dėsnį tenkina taškai $\left(x_i; \widetilde{y}_i\right)$. Šio metodo esmė yra ta, kad, turėdami testinį koordinačių sistemų $\left(X, Y\right)$ rinkinį, ieškome, kurioje koordinačių sistemoje taškai $\left(x_i; \widetilde{y}_i\right)$ apytiksliai tenkina tiesės $Y = kX + b$ lygtį. Tačiau šio metodo taikymas yra daugiau menas negu mokslas, todėl detaliau jo nenagrinėsime.

5.5.1. *Mažiausių kvadratų metodas, taikomas tada, kai $F\left(x, a_0, \ldots, a_n\right)$ yra n-tosios eilės polinomas*

Uždavinio formulavimas. Duoti taškai $\left(x_i; \widetilde{y}_i\right)$, $i = \overline{1, m}$. Reikia rasti tokį *n*-tosios eilės $(n << m)$ polinomą

$$F\left(x, a_0, \ldots, a_n\right) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0,$$

su kuriuo

$$z = \sum_{i=1}^{m} \left(a_n x_i^n + a_{n-1} x_i^{n-1} + \ldots + a_k x_i^k + \ldots + a_1 x_i + a_0 - \widetilde{y}_i\right)^2$$

turėtų mažiausią reikšmę.

Nesunku pastebėti, kad šiuo atveju (5.25) lygčių sistema įgis išraišką

$$a_0 \sum_{i=1}^{m} x_i^k + a_1 \sum_{i=1}^{m} x_i^{k+1} + \ldots + a_n \sum_{i=1}^{m} x_i^{k+n} = \sum_{i=1}^{m} \widetilde{y}_i x_i^k,\ k = \overline{0, n}. \quad (5.27)$$

Pavyzdžiui, kai $n = 2$, tai (5.27) lygčių sistema bus tokia:

$$ma_0 + a_1\sum_{i=1}^{m} x_i + a_2\sum_{i=1}^{m} x_i^2 = \sum_{i=1}^{m} \widetilde{y}_i ,$$
$$a_0\sum_{i=1}^{m} x_i + a_1\sum_{i=1}^{m} x_i^2 + a_2\sum_{i=1}^{m} x_i^3 = \sum_{i=1}^{m} \widetilde{y}_i x_i ,$$
$$a_0\sum_{i=1}^{m} x_i^2 + a_1\sum_{i=1}^{m} x_i^3 + a_2\sum_{i=1}^{m} x_i^4 = \sum_{i=1}^{m} \widetilde{y}_i x_i^2 .$$

(5.27) lygčių sistema yra tiesinė, tačiau labai dažnai ji būna blogai sąlygota, t. y. mažos paklaidos apskaičiuojant sumas sukelia dideles parametrų a_k paklaidas.

Žemiau pateikta funkcija ***mkm***, įgalinanti mažiausiu kvadratų metodu apskaičiuoti suglodinantį polinomą.

```
 function a=mkm(n,x,y)
% MKM funkcija maziausių kvadratų metodu apskaičiuoja n-os eiles
% suglodinantį polinomą.
% Įėjimo parametrai
% n - polinomo eilė,
% x,y - aproksimuojamos funkcijos taškai.
% Išėjimo parametrai
% a - polinomo koeficientų masyvas, pradedant laisvuoju nariu

%Sumų skaičiavimas
m=numel(x); pa=zeros(m,2*n+1); pxy=zeros(m,n+1); xk=ones(m,1);
for j=1:2*n+1
    pa(:,j)=xk;
    if j<=n+1
        pxy(:,j)=xk.*y';
    end;
    xk=xk.*x';
end
 s=sum(pa); b=sum(pxy);
 % Matricos c formavimas
 for i=1:n+1
  c(i,1:n+1)=s(i:i+n);
 end
 % Sistemos ca=b sprendimas
 a=c\b';
```

Pavyzdys. Mažiausių kvadratų metodu kubiniu polinomu aproksimuokime funkciją ***y=sin(x)+randn(1,m)*0.25,*** kai ***x=linspace(0,2*pi,10)***, o ***m=numel(x),***.

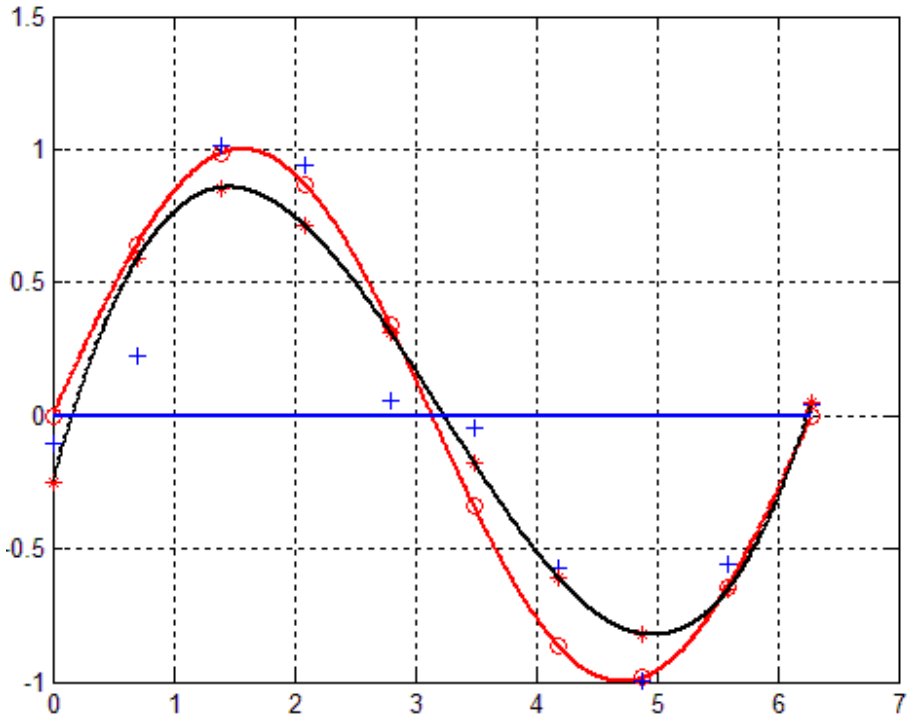
Aproksimavimo geometrinė iliustracija parodyta 5.25 paveiksle.

Raudona spalva pavaizduotas funkcijos yra $y=sin(x)$ grafikas.

Raudoni apskritimėliai žymi tikslias diskretizuotas $y=sin(x)$ reikšmes.

Mėlyni pliusiukai žymi $y=sin(x)$ apytiksles reikšmes.

Raudonos žvaigždutės žymi mažiausių kvadratų metodu suglodintas $y=sin(x)$ reikšmes, o juodas grafikas vaizduoja funkcijos $y=sin(x)$ aproksimaciją kubiniu polinomu.



Pav. 5.25. Funkcijos $y = \sin(x)$ *aproksimacija mažiausių kvadratų metodu*

Be išnagrinėtos diskrečiosios aproksimacijos, dažnai yra naudojama tolydžioji aproksimacija. Tam, kad mažiausių kvadratų metodu apskaičiuotume tolydžiąją aproksimaciją, kai aproksimuojančioji funkcija yra polinomas, reikės išspręsti tiesinių lygčių sistemą, analogišką (5.27) sistemai. Skirtumas yra tik toks, kad tolydžiosios aproksimacijos atveju, (5.27) sistemoje kiekviena suma keičiama integralu. Tokios tiesinės lygčių sistemos matrica yra Hilberto matrica, kuri yra blogai sąlygota, t.y. sistemos sprendinys labai jautrus apvalinimo paklaidoms. Norint šios blogybės išvengti, aproksimuojantysis polinomas užrašomas ortogonaliųjų (dažniausiai Čiobyševo arba Ležandro) polinomų bazėje. Tada tokios tiesinės lygčių sistemos matrica yra įstrižainė, ir tai labai palengvina aproksimuojančio polinomo apskaičiavimą. Šio klausimo plačiau čia nenagrinėsime, nurodydami, kad tolydžioji aproksimacija išnagrinėta B. Kvedaro ir M. Sapagovo skaičiavimo metodų vadovėlyje.

MATLAB'o funkcija polyfit. Ši funkcija mažiausių kvadratų metodu apskaičiuoja suglodinantįjį polinomą. Kreipinys į funkcija yra ***p = polyfit(x,y,n)***, čia (x, y) aproksimuojamos funkcijos taškai, n – suglodinančio polinomo eilė, o p – apskaičiuoti polinomo koeficientai, išdėstyti pradedant vyriausiuoju koeficientu.

5.5.2. *Suglodinančiųjų kubinių splainų konstravimas*

Plėtojant suglodinančiųjų splainų teoriją, svarbią reikšmę turėjo 1957 m. Dž. K. Holidėjaus (J. C. Holliday) įrodyta teorema, kad iš visų funkcijų, priklausančių klasei $C^2[a,b]$ ir interpoliuojančių tuos pačius taškus, kubiniai interpoliaciniai splainai su

natūraliomis kraštinėmis sąlygomis minimizuoja funkcionalą $\int_a^b \left(\varphi''(x)\right)^2 dx$.

Dž. H. Albergas (J. H. Ahlberg), E. N. Nilsonas (E. N. Nilson) ir Dž. L. Volšas (J. L. Walsh) 1962 m. apibendrino šią savybę, pritaikydami jį kubiniams interpoliaciniams splainams, tenkinantiems periodines ir III tipo (žr. 5.10 paragrafą) kraštines sąlygas, taip pat nelyginio laipsnio splainams. Taigi iš visų funkcijų $\varphi(x)$, priklausančių klasei $K^n(a,b)$ ir interpoliuojančių tuos pačius taškus, $(2n-1)$-osios eilės interpoliacinis splainas, kurio p-tosios $\left(p = n, n+1, \ldots, 2n-2\right)$ eilės išvestinės, kai $x = a$ ir $x = b$, lygios nuliui (natūralios kraštinės sąlygos), minimizuoja funkcionalą $\int_a^b \left(\varphi^{(n)}(x)\right)^2 dx$. Simbolis $K^n(a,b)$ žymi klasę funkcijų, kurios intervale $[a;b]$ yra tolydžios ir turi tolydžiąsias išvestines iki $(n-1)$-osios eilės imtinai, o jų n-tosios eilės išvestinės kvadratas intervale $[a;b]$ yra integruojamas.

Remdamasis šiais rezultatais, I. J. Šionbergas (I. J. Shoenberg) 1964 m. suformulavo funkcionalo

$$\alpha\int_a^b \left(\varphi^{(n)}(x)\right)^2 dx + \sum_{i=0}^{N}\left(\varphi(x_i) - z_i\right)^2$$

minimizavimo uždavinį ir drauge įrodė, kad šį funkcionalą minimizuoja $(2n-1)$-osios eilės splainas su natūraliomis kraštinėmis sąlygomis, kurio n-tosios eilės išvestinės kvadratas intervale $[a;b]$ yra integruojamas.

Darbo formules, apibūdinančias kubinį splainą, 1967 m. išvedė K. H. Rainšas (C. H. Reinsh).

Toliau nagrinėsime tokį uždavinį: reikia rasti suglodinantįjį kubinį splainą, kuris minimizuotų funkcionalą

$$\Phi(\varphi) = \int_a^b \left(\varphi''(x)\right)^2 dx + \sum_{i=0}^{N} p_i\left(\varphi(x_i) - \tilde{y}_i\right)^2;$$

čia $p_i > 0$ — svoriniai koeficientai, o $(x_i, \tilde{y}_i)$ — funkcijos, kurią reikia suglodinti, reikšmių lentelė $\left(i = \overline{0, N}\right)$.

Iš anksčiau pateikto nagrinėjimo darome išvadą, kad tai bus kubinis splainas $g(x)$, tenkinantis natūralias kraštines sąlygas. Jo ieškosime naudodami dalimis polinominę splaino išraišką, taigi splainas bus visiškai apibrėžtas, jei žinosime $x[0..N]$, $y[0..N]$ ir $m[0..N]$ (žr. 5.11 paragrafą).

Kadangi kubinis splainas vienareikšmiškai nusakomas taškais $(x_i; y_i)$, $\left(\text{čia } i = \overline{0, N}\right)$, tai $\Phi(g)$ minimizavimas pakeičiamas funkcijos nuo y_i minimizavimu.

Žinome, kad

$$g''(x) = m_{i-1}\frac{x_i - x}{h_i} + m_i\frac{x - x_{i-1}}{h_i},$$

kai $x \in [x_{i-1}; x_i]$; čia $m_{i-1} = g''(x_{i-1})$, $m_i = g''(x_i)$, $h_i = x_i - x_{i-1}$, $i = \overline{1, N}$; be to, $m_0 = m_N = 0$. Tada

$$\Phi(g) = \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} \left(m_{i-1} \frac{x_i - x}{h_i} + m_i \frac{x - x_{i-1}}{h_i} \right)^2 dx + \sum_{i=0}^{N} p_i (y_i - \tilde{y}_i)^2 . \quad (5.28)$$

(5.28) formulėje atlikę integravimo veiksmą, gauname:

$$\Phi(g) = (Am, m) + \sum_{i=0}^{N} p_i (y_i - \tilde{y}_i)^2 ,$$

čia A — kvadratinė $(N-1)$-osios eilės matrica, apibrėžta 5.11 paragrafe.

$Am = Hy$ (žr. 5.11 paragrafą), todėl m tiesiškai išreiškiamas dydžiais $y_0, y_1, \ldots, y_n$ ir $\Phi(g)$ ekstremumas yra minimumas, kurio būtinoji ir pakankamoji sąlyga tokia:

$$\frac{\partial \Phi}{\partial y_i} = \frac{\partial}{\partial y_i} (Am, m) + 2 p_i (y_i - \tilde{y}_i) = 0, \; i = \overline{0, N} . \quad (5.29)$$

Kadangi $Am = Hy$, tai

$$\frac{\partial}{\partial y_i} (Am, m) = \frac{\partial}{\partial y_i} (Hy, m) = 2 \left(\frac{\partial (Hy)}{\partial y_i}, m \right) = 2 (H^t m)_i , \; i = \overline{0, N} ;$$

čia H^t — transponuotoji matrica H, apibrėžta 5.11 paragrafe.

Taigi (5.29) minimumo sąlyga, užrašyta matricine išraiška, yra

$$H^t m + Py = P\tilde{y} ; \quad (5.30)$$

čia P — $(n+1)$-osios eilės įstrižainė matrica, kurios i-tasis įstrižaininis elementas lygus p_i.

(5.30) lygtį iš kairės daugindami iš HP^{-1}, gauname:

$$Hy + HP^{-1} H^t m = H\tilde{y} . \quad (5.31)$$

Kadangi $Hy = Am$, tai (5.31) lygtį galime užrašyti taip:

$$(A + HP^{-1} H^t) m = H\tilde{y} . \quad (5.32)$$

(5.32) lygčių sistema yra tiesinė su penkiaįstrižaine simetrine matrica, ir šios sistemos sprendinys lygus kubinio splaino, minimizuojančio (5.28) funkcionalą, antrųjų išvestinių reikšmėms. Turėdami tas reikšmes, iš (5.30) lygčių sistemos apskaičiuosime šio splaino y reikšmes pagal formulę

$$y = \tilde{y} - P^{-1} H^t m . \quad (5.33)$$

Iš (5.33) formulės matyti, kad kuo didesnė svorinio koeficiento p_i reikšmė, tuo mažesnis skirtumas tarp y_i ir $\tilde{y}_i$, t. y. tarp suglodintos ir pradinės funkcijos reikšmės.

Labai dažnai yra žinomos matavimo paklaidų ribos, pavyzdžiui, $\tilde{y}_i$ reikšmės, išmatuotos 5% tikslumu. Tokiu atveju protinga reikalauti, kad suglodinantysis splainas būtų koridoriuje $(1 \pm 0{,}05)\tilde{y}(x)$. Tai galima pasiekti atitinkamai parinkus svorinius koeficientus p_i.

Toliau pateikiama suglodinančiųjų kubinių splainų apskaičiavimo funkcija ***ssplin***.

```
function [fm,ya]=ssplin(x,y,p)
% SSPLIN apskaičiuoja suglodinantį kubinį interpoliacinį splainą
% su natūraliomis kraštinėmis sąlygomis.
% Įėjimo parametrai
% (x,y) - duotieji taškai,
% p     - taškų svoriai.
% Išėjimo parametrai
% fm    - splaino antrųjų išvestinių reikšmės,
% ya    - suglodintos y reikšmės.
N=numel(x); % taškų skaičius n=N-1; % intervalų skaičius
a1=zeros(1,n-1); a2=zeros(1,n-1); a3=zeros(1,n-1); a4=zeros(1,n-1);
a5=zeros(1,n-1); h=x(2:N)-x(1:N-1);
% Sistemos (A+HP^-1H^t)m=Hỹ formavimas
p1=ones(1,n-3);
a1=p1./(p(3:n-1).*h(3:n-1).*h(2:n-2));
a5=[a1 0 0]; a1=[0 0 a1];
p2=ones(1,n-2);
a2=-(p2./h(1:n-2)+p2./h(2:n-1))./(p(2:n-1).*h(2:n-1))-...
    (p2./h(2:n-1)+p2./h(3:n))./(p(3:n).*h(2:n-1))+h(2:n-1)/6;
a4=[a2 0]; a2=[0 a2];
p3=ones(1,n-1);
a3=-(p3./(p(1:n-1).*h(1:n-1).^2)+...
    (p3./h(1:n-1)+p3./h(2:n)).^2./p(2:n)+...
     p3./(p(3:n+1).*h(2:n).^2))-...
     (h(1:n-1)+h(2:n))/3;
pd=ones(1,n-1);
d(1:n-1)=y(1:n-1)./h(1:n-1)-y(2:n).*(pd./h(1:n-1)+pd./h(2:n))+...
         y(3:n+1)./h(2:n);

fm=perkelt5(a1,a2,a3,a4,a5,d);
fm=[0 fm(1:n-1) 0];
% Suglodintų y apskaiciavimas
ya(1)=y(1)-fm(2)/(p(1)*h(1));
ya(2)=y(2)-(fm(3)/h(2)-fm(2)*(1/h(1)+1/h(2)))/p(2);
pya=ones(1,n-3);
ya(3:n-1)=y(3:n-1)-(fm(2:n-2)./h(2:n-2)-...
          fm(3:n-1).*(pya./h(2:n-2)+pya./h(3:n-1))+...
          fm(4:n)./h(4:n))./p(3:n-1);
ya(n)=y(n)-(fm(n-1)/h(n-1)-fm(n)*(1/h(n-1)+1/h(n)))/p(n);
ya(n+1)=y(n+1)-fm(n)/(p(n+1)*h(n));
% Antrinė funkcija
function x=perkelt5(a,b,c,d,g,h)
% PERKELT5  sprendžia penkiaįstrižaininę tiesinių lygčių sistemą
%perkelties metodu (žr. K. Plukas, Skaitiniai metodai ir algoritmai).
% a,b,c,d,g - sistemos matricos įstrižaininiai elementai,
% h         - sistemos laisvieji nariai,
% x         - sistemos sprendinys.

n=numel(c);
s=zeros(1,n+3); e=zeros(1,n+3); f=zeros(1,n+3);
% Tiesioginis etapas
s(1)=0; s(2)=0; e(1)=0; e(2)=0; f(1)=0; f(2)=0;
for i=1:n
v=c(i)-a(i)*s(i)*s(i+1)-a(i).*e(i)-b(i).*s(i+1);
s(i+2)=(a(i)*s(i)*e(i+1)+b(i)*e(i+1)+d(i))/v;
e(i+2)=g(i)/v;
```

```
f(i+2)=(a(i)*s(i)*f(i+1)+a(i)*f(i)+b(i)*f(i+1)-h(i))/v;
end
% Atvirkstinis etapas
x(n+2)=0; x(n+1)=0;
for i=n:-1:1
    x(i)=s(i+2)*x(i+1)+e(i+2)*x(i+2)+f(i+2);
end
```

5.26 paveiksle pavaizduotas funkcijos

$$y = 0.1a_1x^5 + 0.5a_2x^4 + a_3x^3 + a_4x^2 + a_5x + a_6 + 5\sin x,$$

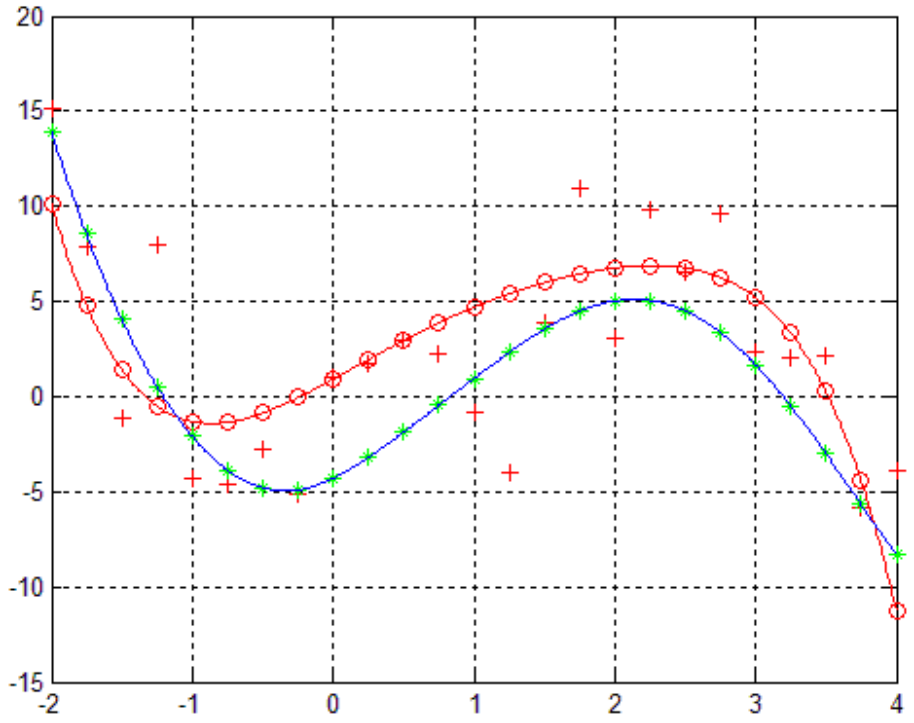
grafikas (raudona spalva), čia koeficientai $a_1, a_2, a_3, a_4, a_5, a_6$ yra iš intervalo [0;1] atsitiktinai sugeneruoti skaičiai; be to, koeficientai a_2, a_4, a_6 yra neigiami.

Raudoni pliusiukai žymi suglodinamos funkcijos diskretizuotas apytiksles reikšmes mazguose : *x=-2:0.25:4* .

Visų taškų svoriai $p_i, i = \overline{1,25}$ yra lygūs vienetui išskyrus pirmą, antrą, aštuntą ir dvidešimt ketvirtą tašką: p(1)=50; p(2)=100; p(8)=100; p(24)=100.

Tikslios funkcijos reikšmės pažymėtos raudonais apskritimėliais.

Žalios žvaigždutės žymi suglodintas aproksimuojamos funkcijos reikšmes. Per šias žvaigždutes išbrėžtas suglodinantis splainas.



5.26 pav. Funkcijos aproksimacija suglodinančiu splainu.

5.5.3. *MATLAB'inės aproksimavimo funkcijos*

MATLAB‘e patalpinta daug funkcijų, realizuojančių tiek anksčiau aprašytus, tiek ir šiame vadovėlyje nenagrinėtus funkcijų aproksimavimo metodus. Trumpai aptarkime pagrindines aproksimavimo funkcijas. Norėdami gauti daugiau informacijos, kreipkitės į MATLAB‘o pagalbos žinyną.

Interpoliavimo funkcijos. Tai funkcijos ***inerp1***, ***inerp2***, ***inerp3***, ***inerpn.***

Funkcija ***interp1*** apskaičiuoja vieno kintamojo funkcijos, nusakytos reikšmių lentele (interpoliavimo taškais), reikšmes nurodytuose taškuose. Į šią funkciją kreipiamasi taip:

yi=interp1(x,y,xi),

čia (x, y) - interpoliavimo taškai, *xi* – taškai, kuriuose norime apskaičiuoti funkcijos reikšmes, o *yi* – apskaičiuotos funkcijos reikšmės. Bendru atveju *x* ir *y* gali būti ne tik vektoriai, bet ir matricos.

Analogiškai funkcijos ***inerp2***, ***inerp3***, ***inerpn*** interpoliuoja funkcijas atitinkamai dvimatėje, trimatėje ir *n*-matėje erdvėje.

Splaininės interpoliavimo funkcijos. MATLAB‘as turi programinių priemonių komplektą (toolbox) ***spline***, skirtą splaininei funkcijų aproksimacijai.jame yra virš keturiasdešimties funkcijų. Tai funkcijos: ***spline, csape, csaps*** ir t.t. Be to, Akima splinams apskaičiuoti skirta funkcija ***pchip***.

Funkcija ***spline*** apskaičiuoja kubinį interpoliacinį splainą, kuris tenkina 5-as kraštines sąlygas (žr. 5.1.2.1 paragrafą). Angliškai šios sąlygos vadinamos „***not a knot***“.

Kreiptis į funkciją ***spline*** galima dviem būdais:

yy = spline (x,y,xx),

pp = spline(x,y),

čia (x, y) - interpoliavimo taškai, *xx* – taškai, kuriuose norime apskaičiuoti funkcijos reikšmes, o *yy* – apskaičiuotos funkcijos reikšmės.

Praktikoje pasitaiko atvejų, kai kiekviename intervale norime žinoti kubinio polinomo išraiškas. Šiuo atveju naudojamas antrasis kreipinys. Masyvas ***pp*** (taip vadinama *p-p* struktūra) apibrėžia kubinio polinomo koeficientus:

- pirmasis elementas *p*(1) – yra skaičius, nurodantis interpoliavimo intervalų skaičių, t.y. *n*-1;
- elementai, pradedant antruoju ir baigiant (*n*+1) (t.y. $p_2, p_3, \ldots, p_{n+1}$) apibrėžia *x* reikšmes;
- elementas *p*(n+2) = 4, t.y. nurodo kubinio polinomo koeficientų skaičių;
- likusių koeficientų ketvertai nuosekliai apibrėžia kiekvieno intervalo kubinio polinomo koeficientus.

Turint *p-p* struktūrą, funkcijos ***ppval*** pagalba galime apskaičiuoti splaino reikšmes norimuose taškuose. Kreipinys į ***ppval*** yra: ***yy=ppval (pp,xx)***, čia *pp* yra *p-p* struktūra, *xx* – taškai, kuriuose apskaičiuojamos splaino reikšmės ir *yy* – apskaičiuotos splaino reikšmės.

Funkcija ***csape*** apskaičiuoja kubinį splainą su įvairiomis kraštinėmis sąlygomis.

Kreiptis į funkciją ***csape*** galima dviem būdais:

pp = csape(x,y) arba ***pp = csape(x,y,conds)***.

Pirmuoju atveju, turint interpoliavimo taškus (x, y) apskaičiuojama anksčiau aptarta *p-p* struktūra, apibrėžiantį kubinį splainą. Pagal nutylėjimą elementai *y*(*1*) ir *y*(*end*) nurodo kubinio splaino pirmosios išvestinės reikšmes kraštiniuose mazguose.

Antruoju atveju, *p-p* struktūra apskaičiuojama pagal pasirinktas kraštines sąlygas. Pavyzdžiui, komanda ***p=csape***([-1 1],[3 -1 1 6],[1 2]) apskaičiuos kubinio splaino $s(x)$, einančio per taškus $(-1;1)$ ir $(1;1)$ ir tenkinančio kraštines sąlygas: $s'(-1)=3$, $s''(1)=6$, *p-p* struktūrą. Aišku, kad šis splainas sutaps su funkcija $y=x^3, x\in[-1;1]$.

Splaininės suglodinimo funkcijos. Funkcija ***csaps*** apskaičiuoja *p-p* struktūrą, kuri nusako suglodinantį kubinį splainą. Suglodinimo kriterijus truputį skiriasi nuo 5.5.2 paragrafe pateikto kriterijaus $\Phi(\varphi)$. Funkcijos ***csaps*** apskaičiuotas suglodinantysis splainas minimizuoja funkcionalą

$$\Phi(\varphi)=(1-p)\int_a^b(\varphi''(x))^2\,dx+p\sum_{i=0}^{N}w_i(\varphi(x_i)-\tilde{y}_i)^2,$$

čia parametras $p\in[0;1]$ nusako interpoliavimo ir mažiausių kvadratų metodo dedamųjų svorį.

Pilnas kreipinys į funkciją ***csaps*** yra

[***output,p***] = ***csaps(x,y,p,xx,w)***,

čia (*x,y*) - duotieji taškai, *p* – nusako interpoliavimo ir mažiausių kvadratų metodo dedamųjų svorį, *xx* – taškai, kuriuose norime apskaičiuoti suglodinančio splaino reikšmes, o *w* – svoriniai koeficientai.

Išėjimo parametras *output* gali būti tiek *p-p* struktūra (šiuo atveju parametro *xx* reikšmė yra tuščioji aibė, t.y. *xx*=[ ]), tiek splaino reikšmės ($xx\neq[\ ]$).

Išėjimo parametras *p* parodo tikrąją *p* reikšmę, jei *p* reikšmė nebuvo nurodyta, kaip įėjimo parametras.

Kreipiantis į funkciją ***csaps***, būtina nurodyti tik *x* ir *y* reikšmes. Kitos įėjimo parametrų reikšmės priimamos automatiškai pagal nutylėjimą.

Jei nenurodytos svorinių koeficientų w reikšmės, - jos pagal nutylėjimą prilyginamos vienetams.

Norint sužinoti kokia pagal nutylėjimą yra *p* reikšmė, į funkciją reikia kreiptis taip: **[*pp,p*] = *csaps*(*x,y*)**. Tada išėjimo parametro *p* reikšmė ir bus reikšmė pagal nutylėjimą. ir padės vartotojui orientuotis parenkant kitą *p* reikšmę.

Suglodinimo pobūdį galima keisti, jei skaliarą *p* pakeisime vektoriumi *p*, turintį tiek pat elementų, kaip ir vektorius *x*. Tada funkcija ***csaps*** minimizuoja funkcionalą

$$\Phi(\varphi)=(1-p)\int_a^b(\varphi''(x))^2\,dx+p\sum_{i=0}^{N}\lambda(x)w_i(\varphi(x_i)-\tilde{y}_i)^2.$$

Šiuo atveju parametras $p=p_1$, o likusios vektoriaus *p* komponentės $p_2,p_3,\ldots,p_N$ nusako laiptuotą funkciją $\lambda(x)$, kurios trūkio taškai yra *x* reikšmės. Funkcijos $\lambda(x)$ laiptelio reikšmė *i*-me intervale yra lygi p_{i+1}, t.y. $\lambda(x)=p_{i+1}, i=1,2,\ldots,N-1$.

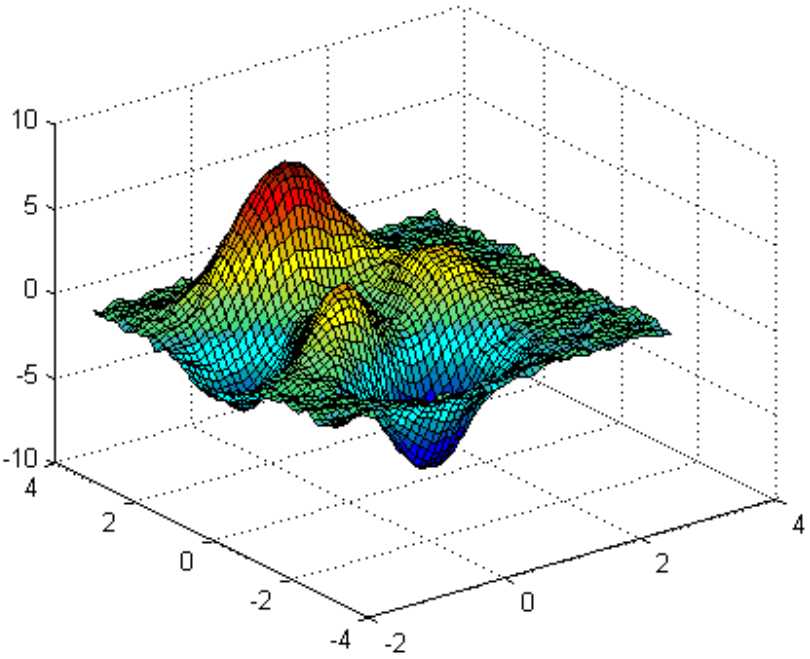
Funkcija ***csaps*** taikoma ir paviršiams (dviejų kintamųjų funkcijoms) glodinti.

Pavyzdžiui, įvykdžius seką komandų

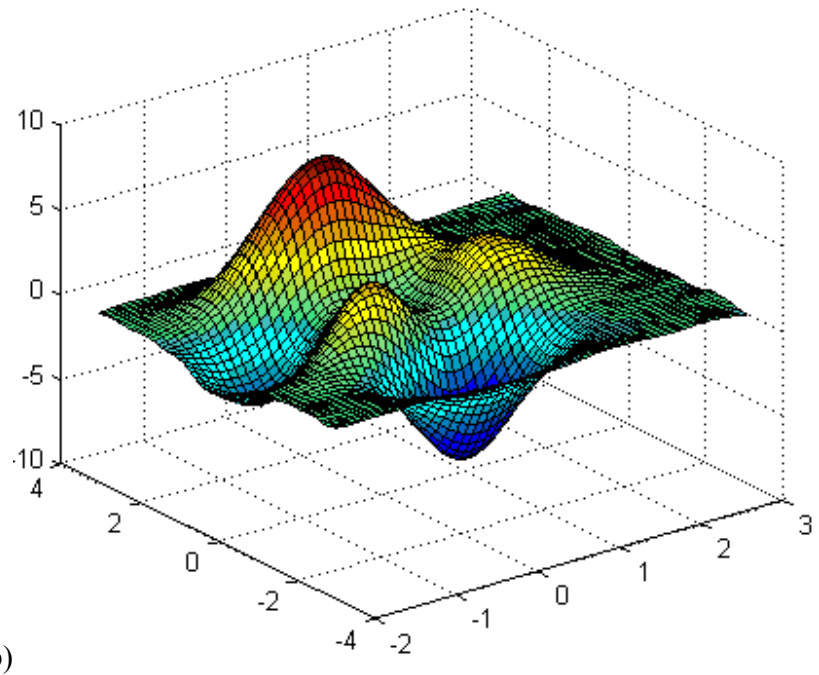
```
 x = {linspace(-2,3,51),linspace(-3,3,61)};
[xx,yy] = ndgrid(x{1},x{2}); y = peaks(xx,yy);
rand('state',0), noisy = y+(rand(size(y))-.5);
[smooth,p] = csaps(x,noisy,[],x);
surf(x{1},x{2},smooth.')
smoother = csaps(x,noisy,.996,x);
figure, surf(x{1},x{2},smoother.')
```

apskaičiuotume suglodintą paviršių „***peaks***“. Paviršiai grafiškai pavaizduoti 5.27 paveiksle (a) paviršius su triukšmu; b) suglodintas paviršius). Funkcija ***peaks*** yra vidinė MATLAB‘o funkcija, kuri apskaičiuoja dvimatį paviršių, gaunamą iš Gauso kreivės.

Be detaliau aptartos funkcijos ***csaps*** MATLAB‘as turi ir kitų suglodinančių funkcijų, kaip pavyzdžiui, ***spaps, tpaps, spap2***. Detaliau šių funkcijų čia nedetalizuosime. Plačią informaciją apie jas (kaip ir apie kitas MATLAB‘o funkcijas) galima rasti MATLAB‘o pagalbos žinyne.



a)



b)

*5.27 pav. Paviršius “**peaks”** :* a) *su triukšmu, b) suglodintas*