

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Magistrsko delo

**Hierarhični prostori zlepkov: od teorije do uporabe v
izogeometrični analizi**

(Hierarchical spline spaces: From theory to applications in isogeometric analysis)

Ime in priimek: *Aljaž Kosmač*

Študijski program: *Matematične znanosti, 2. stopnja*

Mentor: *prof. dr. Vito Vitrih*

Koper, avgust 2021

Ključna dokumentacijska informacija

Ime in PRIIMEK: Aljaž KOSMAČ

Naslov magistrskega dela: Hierarhični prostori zlepkov: od teorije do uporabe v izogeometrični analizi

Kraj: Koper

Leto: 2021

Število listov: 187

Število slik: 67

Število tabel: 21

Število prilog: 20

Število strani prilog: 41

Število referenc: 26

Mentor: prof. dr. Vito Vitrih

UDK: 511.4(043.2)

Ključne besede: hierarhični zlepki, aproksimacija, izogeometrična analiza

Math. Subj. Class. (2010): 41A15, 65D07, 65M50, 68U07

Izvleček:

V magistrskem delu predstavimo teorijo hierarhičnih prostorov zlepkov ter njihovo uporabo v aplikacijah. Prva dva poglavja sta namenjena spoznavanju nekaj nujno potrebne teorije B-zlepkov in škatlastih zlepkov. V teh dveh poglavjih predstavimo osnovne definicije in pomembne lastnosti obeh vrst zlepkov. Za tem sledi poglavje o hierarhičnih prostorih zlepkov. V tem poglavju definiramo lokalno zgoščevanje, ugnezdene prostore in domene ter konstrukcijo hierarhične baze B-zlepkov in škatlastih zlepkov. Glavni cilj tega poglavja je dokaz nekaterih lastnosti, s katerimi utemeljimo, da imajo tudi hierarhični prostori zlepkov enake lastnosti kot navadni ter da so primerni za aplikacije kot so aproksimacija in reševanje parcialnih diferencialnih enačb. Za tem sledi poglavje o implementaciji hierarhičnih zlepkov v okolju Matlab. Tukaj navedemo nekaj algoritmov in programerskih prijemov, da bralec lažje sledi Matlab skriptam v prilogah. Po poglavju o implementaciji pa sledi še zadnje poglavje o aplikacijah. Na začetku tega poglavja predstavimo nekaj nujno potrebne teorije iz področja numernične analize, ki jo bomo uporabili pri primerih aproksimacije po metodi najmanjših kvadratov in Galerkinovi metodi. Za tem z nekaj numeričnimi poskusi preverimo red konvergence obeh aplikacij pri uporabi obeh vrst zlepkov. Čisto na koncu pa naredimo še nekaj numeričnih poskusov obeh aplikacij, pri čemer uporabimo obe vrsti hierarhičnih zlepkov. Cilj teh poskusov je pokazati, da so prostori zlepkov ter numerične metode dobro implementirane, ter da je uporaba hierarhičnih prostorov zlepkov v primerjavi z globalnim zgoščevanjem bolj ekonomična v smislu, da za primerljiv red napake numeričnega približka v primeru lokalnega zgoščevanja potrebujemo manjše število baznih funkcij.

Key document information

Name and SURNAME: Aljaž KOSMAČ

Title of the thesis: Hierarchical spline spaces: From theory to applications in isogeometric analysis

Place: Koper

Year: 2021

Number of pages: 187

Number of figures: 67

Number of tables: 21

Number of appendices: 20

Number of appendix pages: 41

Number of references: 26

Mentor: Prof. Vito Vitrih, PhD

UDC: 511.4(043.2)

Keywords: hierarchical splines, approximation, isogeometric analysis

Math. Subj. Class. (2010): 41A15, 65D07, 65M50, 68U07

Abstract:

In this master thesis we present the theory of hierarchical spline spaces and their use in applications. The first two chapters cover some needed theory of B-splines and box splines. In this two chapters we present some basic definitions and properties of both aforementioned spline spaces. In the next chapter we present hierarchical spline spaces. In this chapter we define local refinement, nested domains and spaces for both B-splines and box splines. The main aim of this chapter is to show that the hierarchical spline spaces possess the same properties as the regular ones and that they are appropriate for applications such as approximation and solving partial differential equations. This is followed by a chapter about implementation of splines in Matlab environment. Here we describe a few algorithms and programming techniques so that the reader can follow the Matlab code in the appendices more effortlessly. Finally in the last chapter we present the applications of hierarchical spline spaces. The chapter starts with some theory about least squares approximation and the Galerkin method. Then we test the convergence of both applications with the use of both spline spaces with some numerical examples. At the very end we present some numerical experiments, where we test both applications with the use of both hierarchical spline spaces. The aim of these experiments is to show that the spline spaces and the methods are correctly implemented and that the use of hierarchical spline spaces with these methods is more economical in comparison to regular spline spaces in the sense that we need less basis functions to achieve comparable error of the numerical approximation.

Kazalo vsebine

1	UVOD	1
2	B-ZLEPKI	4
2.1	BAZA PROSTORA	4
2.1.1	Lastnosti baznih funkcij	9
2.2	PROSTOR B-ZLEPKOV	13
2.3	TENZORSKI PRODUKTI B-ZLEPKOV	17
3	ŠKATLASTI ZLEPKI	20
3.1	INDUKTIVNA DEFINICIJA IN LASTNOSTI	20
3.2	BERNSTEIN-BÉZIEREVA OBLIKA	24
3.3	PROSTOR PREMIKOV	36
4	HIERARHIČNI ZLEPKI	46
4.1	UGNEZDENI PROSTORI IN DOMENE B-ZLEPKOV	46
4.2	HIERARHIČNA BAZA B-ZLEPKOV	47
4.2.1	Dve lastnosti hierarhične baze	48
4.2.2	Širjenje domen hierarhije zlepkov	50
4.2.3	Utežena hierarhična baza	51
4.3	PRIREZANI HIERARHIČNI B-ZLEPKI	53
4.3.1	Lastnosti prirezane baze	54
4.4	HIERARHIČNI ŠKATLASTI ZLEPKI	57
4.4.1	Ugnezdene prostori in domene škatlastih zlepkov	57
4.4.2	Definicija hierarhične baze	58
4.4.3	Lastnosti hierarhične baze škatlastih zlepkov	60
5	IMPLEMENTACIJA	64
5.1	B-ZLEPKI	64
5.1.1	Matlab podlaga	64
5.1.2	Zapis v finejši bazi	70
5.1.3	Prirez zleпка	72
5.1.4	Hierarhična baza B-zlepkov	73

5.2	ŠKATLASTI ZLEPKI	75
5.2.1	Matlab podlaga	75
5.2.2	Implementacija zlepka B_{222}	77
5.2.3	Prostor premikov zlepka $B_{222}(\mathbf{x})$	80
5.2.4	Hierarhična baza škatlastih zlepkov	82
6	APLIKACIJE	85
6.1	TEORETIČNO OZADJE	85
6.1.1	Minimalna določitvena množica	85
6.1.2	Gaussova integracijska pravila	86
6.1.3	Aproksimacija po metodi najmanjših kvadratov	88
6.1.4	Galerkinova metoda	90
6.1.4.1	Nehomogen robni pogoj	92
6.1.4.2	Diskretna verzija Galerkinove metode	93
6.1.5	Prehod na novo domeno	93
6.1.5.1	Aproksimacija po metodi najmanjših kvadratov	94
6.1.5.2	Galerkinova metoda	95
6.2	NUMERIČNI POSKUSI	96
6.2.1	Red konvergence	97
6.2.2	Aproksimacija	110
6.2.3	Galerkinova metoda	120
7	ZAKLJUČEK	128
8	LITERATURA IN VIRI	130

Kazalo preglednic

1	Algoritem prireza tenzorskega produkt B-zlepkov.	72
2	Algoritem konstrukcije (prirežane) hierarhične baze tenzorskih produktov B-zlepkov.	74
3	Algoritem konstrukcije poljubnega premika škatlastega zlepka $B_{222}(\mathbf{x})$	80
4	Vozli in uteži Gaussovih integracijskih pravil na šestih točkah.	88
5	Primerjava različnih napak pri aproksimaciji funkcije $f(x) = e^{x^2} \sin(15x)$	100
6	Red konvergence aproksimacije funkcije $f(x) = e^{x^2} \sin(4x)$ v prostoru prostoru B-zlepkov stopenj 2, 3, 4, 5.	101
7	Red konvergence aproksimacije funkcije $f(x, y) = \cos(4x) \cdot \sin(4y)$ v prostoru tenzorskih produktov B-zlepkov stopenj 2 in 3.	102
8	Razlika v številu baznih funkcij prostorov, kjer je za bazne funkcije izpolnjen homogen robni pogoj in tistih, kjer ni.	102
9	Red konvergence Galerkinove metode s tenzorskimi produkti B-zlepkov stopenj 2 in 3 za Poissonovo enačbo (6.14).	103
10	Red konvergence aproksimacije po metodi najmanjših kvadratov s prostorom premikov škatlastega zlepka B_{222}	105
11	Primerjava števila baznih funkcij s faktorjem h skaliranega polnega prostora premikov zlepka B_{222} in tistega, kjer bazne funkcije izpolnjujejo homogen robni pogoj.	106
12	Primerjava števil baznih funkcij s faktorjem h skaliranega prostora premikov zlepka B_{222} v primeru vseh funkcij in homogenih funkcij, pridobljenih z algoritmom minimalne določitvene množice.	109
13	Določanje reda konvergence Galerkinove metode z uporabo homogene baze premikov škatlastega zlepka B_{222}	110
14	Rezultati numeričnih poskusov primera 6.15.	112
15	Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo tenzorskih produktov B-zlepkov in globalnim zgoščevanjem.	113
16	Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo škatlastih zlepkov in lokalnim zgoščevanjem.	115
17	Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo prostora premikov zlepka B_{222} in globalnim zgoščevanjem.	115

18	Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo obeh prostorov hierarhičnih zlepkov na reparametriziranem območju. Zadnja vrstica prikazuje rezultate v primeru globalnega zgoščevanja.	119
19	Rezultati Galerkinove metode z uporabo hierarhične prirezane baze tenzorskih produktov B-zlepkov.	122
20	Rezultati Galerkinove metode z uporabo hierarhične baze premikov zlepka B_{222}	123
21	Rezultati Galerkinove metode na poljubnem štirikotnem območju v \mathbb{R}^2 z uporabo hierarhičnega prostora tenzorskih produktov B-zlepkov. . . .	126

Kazalo slik in grafikonov

1	Ideja lokalnega zgoščevanja.	2
2	Grafi baznih funkcij $N_{i,0}$, $i = 0, 1, 2$	5
3	Trikotna shema.	6
4	Bazni funkciji $N_{0,1}$ (modra) ter $N_{1,1}$ (rdeča, črtkana).	7
5	Bazna funkcija $N_{0,2}$	7
6	Na sliki je označen nosilec $\text{supp}(N_{1,3}) = [u_1, u_5]$	8
7	Nenične funkcije stopnje 3 na razponih $[u_2, u_3]$, $[u_3, u_4]$, $[u_4, u_5]$	9
8	Bazne funkcije prostora kubičnih B-zlepkov nad intervalom $[0, 4]$	15
9	Primer odprte krivulje B-zlepkov.	16
10	Primer vpete krivulje B-zlepkov.	17
11	Trije primeri različnih mrež.	21
12	Škatlasti zlepek B_{111}	25
13	Nosilec zlepka B_{111} in pripadajoči Bernsteinovi koeficienti.	25
14	Baricentrična reprezentacija Kontrolnih točk Béziereve krpe na enem trikotniku na mreži tipa- I zlepka $B_n(\mathbf{x})$	28
15	Shema določanja Bernsteinovih koeficientov za poljuben škatlasti zlepek B_{ijk}	29
16	Nosilca zlepkov $B_{111}(\mathbf{x})$ in $B_{111}(\mathbf{x} + \mathbf{e}_1)$ z Bernsteinovimi koeficienti.	30
17	Računanje Bernsteinovih koeficientov zlepka $D_{\mathbf{e}_1}B_{211}(\mathbf{x})$	30
18	Dva primera trikotnikov iz nosilca zlepka $B_{211}(\mathbf{x})$	31
19	Nosilec zlepka $B_{211}(\mathbf{x})$ z Bernsteinovimi koeficienti.	31
20	Nosilec zlepka $D_{\mathbf{e}_1}B_{211}(\mathbf{x})$ z Bernsteinovimi koeficienti.	31
21	Zgled določanja enačbe v sistemu (3.7).	32
22	Bernsteinovi koeficienti zlepka $2B_{211}(\mathbf{x})$	33
23	Bernsteinovi koeficienti zlepka $24B_{222}(\mathbf{x})$	34
24	Indeksacija trikotnikov (modro) in oglišč (zeleno) nosilca zlepka $B_{111}(\mathbf{x})$. S črno so označene še kartezične koordinate oglišč.	34
25	Indeksacija kontrolnih točk v enem trikotniku zlepka $B_{111}(\mathbf{x})$	35
26	Dva primera pokritja prostora \mathbb{Z}^2	42

27	Primeri različnih pogojev na robovih domen. Z različnimi barvami so označena različna območja v hierarhiji. Osnovno območje je modro, znotraj modrega območja imamo rdeče, ki je finejše od modrega, in znotraj tega še zeleno območje, ki je še finejše.	47
28	Primer treh korakov konstrukcije hierarhične baze, kjer sta $(p^\ell, q^\ell) = (1, 1)$, $\ell = 0, 1, 2$	48
29	Primer povečevanja domene Ω^1	50
30	Primer treh korakov konstrukcije prirezane hierarhične baze kjer sta $(p^\ell, q^\ell) = (1, 1)$, $\ell = 0, 1, 2$. Konstrukcija je podobna kot na sliki 28, le da tukaj bazne funkcije iz nivoja ℓ , ki imajo neprazen presek z območjem Ω_ℓ prirežemo.	54
31	Primer treh različno gostih mrež tipa- I	58
32	Primer gnezdenih domen škatlastih zlepkov na mrežah tipa- I s krepkim robnim pogojem.	58
33	Gnezdeni domeni in nosilci baznih funkcij primera 4.12.	61
34	Nekateri koraki grajenja hierarhične baze škatlastih zlepkov iz primera 4.12.	61
35	Krivulja B-zlepkov ene spremenljivke primera 5.1.	66
36	Bazne funkcije prostora $\mathbb{S}_{2,U}$, kjer je $U = (0, 1, 2, 3, 4, 5)$	67
37	Bazna funkcija $N_{1,0}$ prostora $\mathbb{S}_{U,V}^{2,2}$	69
38	Ploskev primera 5.7.	70
39	Triangulacija primera 5.12.	76
40	Nosilec zlepka B_{222} z glavnimi oglišči v kartezični ravnini in njihovo indeksacijo ter indeksacija trikotnikov.	78
41	Primer dveh povezav med oglišči nosilca zlepka B_{222} in pripadajočimi Bernsteinovimi koeficienti. Bernsteinovi koeficienti so zaradi preglednosti pomnoženi s 24.	79
42	Kontrolne točke na enem trikotniku nosilca zlepka B_{222}	79
43	Shema napolnitve območja, ki je nosilec škatlastega zlepka B_{222}	82
44	Grafični prikaz prehoda na novo domeno Ω iz območja $[0, 1]^2$	94
45	Funkcija $f(x) = e^{x^2} \sin(15x)$ in njen aproksimant na intervalu $[0, 1]$	99
46	Funkcija $f(x) = e^{x^2} \sin(4x)$ na intervalu $[0, 1]$	100
47	Funkcija $f(x, y) = \cos(4x) \cdot \sin(4y)$ na območju $[0, 1]^2$	101
48	Funkcija $u(x, y) = \sin(\pi x) \sin(\pi y)(x - \frac{1}{2})(y - \frac{1}{2})$ na območju $\Omega = [0, 1]^2$	103
49	Grafi funkcij f_1, f_2 iz primera 6.12.	105
50	Unija nosilcev vseh premikov zlepka B_{222} , skaliranih s faktorjem $h = \frac{1}{8}$, ki so v celoti vsebovani v območju $\Omega = [0, 1]^2$	106
51	Kvadrat $[0, 1]^2$ pokrit na vsaki stranici s 17 točkami.	107

52	Primerjava števila ničelnih elementov jeder matrik, izračunanih na dva različna načina.	108
53	Grafi funkcij f_1, f_2, f_3 iz primera 6.15.	111
54	Prikaz lokalnega zgoščevanja domene za primer 6.15.	113
55	Grafi funkcij f_1, f_2, f_3 in f_4 iz primera 6.16.	114
56	Prikaz lokalnega zgoščevanja domene za primer 6.15.	116
57	Območje Ω primera 6.17.	117
58	Območje Ω^* primera 6.17.	118
59	Graf funkcije $f(x, y)$ primera 6.17.	118
60	Prikaz lokalnega zgoščevanja domene za primer 6.17.	119
61	Grafa rešitev problemov Poissonove enačbe primera 6.18.	121
62	Območja zgoščevanja pri uporabi Galerkinove metode v primeru 6.18.	122
63	Območja zgoščevanja pri uporabi Galerkinove metode v primeru 6.19.	124
64	Območje Ω primera 6.20.	124
65	Krivulja $G(x, y)$ s pripadajočo kontrolno mrežo \mathbf{P} primera 6.20.	125
66	Graf točne rešitve Poissonove enačbe primera 6.20.	125
67	Območja zgoščevanja pri uporabi Galerkinove metode v primeru 6.20.	126

Kazalo prilog

PRILOGA A Baza B-zlepkov

PRILOGA B Prirezana hierarhična baza tenzorskih produktov B-zlepkov

PRILOGA C Konstrukcija novega zlepka

PRILOGA D Hierarhična baza premika zlepka B_{222}

PRILOGA E Hierarhična baza premika zlepka B_{222} (Homogena)

PRILOGA F Nova domena

PRILOGA G Minimalna določitvena množica

PRILOGA H Gaussova integracijska pravila

PRILOGA I Implementacija zlepka B_{222}

PRILOGA J Implementacija odvoda zlepka B_{222} po smeri \mathbf{e}_1

PRILOGA K Implementacija odvoda zlepka B_{222} po smeri \mathbf{e}_2

PRILOGA L Aproksimacija po metodi najmanjših kvadratov (B-zlepki)

PRILOGA M Aproksimacija po metodi najmanjših kvadratov (Škatlasti zlepki)

PRILOGA N Reparametrizirana aproksimacija po metodi najmanjših kvadratov
(B-zlepki)

PRILOGA O Reparametrizirana aproksimacija po metodi najmanjših kvadratov
(Škatlasti zlepki)

PRILOGA P Reševanje homogene Poissonove enačbe (B-zlepki)

PRILOGA Q Reševanje homogene Poissonove enačbe (Škatlasti zlepki)

PRILOGA R Reševanje nehomogene Poissonove enačbe (B-zlepki)

PRILOGA S Reševanje nehomogene Poissonove enačbe (Škatlasti zlepki)

PRILOGA T Reševanje nehomogene Poissonove enačbe z reparametrizacijo (B-
zlepki)

Seznam kratic

tj. to je

npr. na primer

itd. in tako dalje

Zahvala

Zahvalo namenjam mentorju dr. Vitu Vitrihu za potrpežljivo vodenje, redne konzultacije in vse nasvete pri pisanju magistrskega dela.

Zahvaljujem se tudi mojima staršema in Katarini za vso spodbudo ter podporo v času študija.

1 UVOD

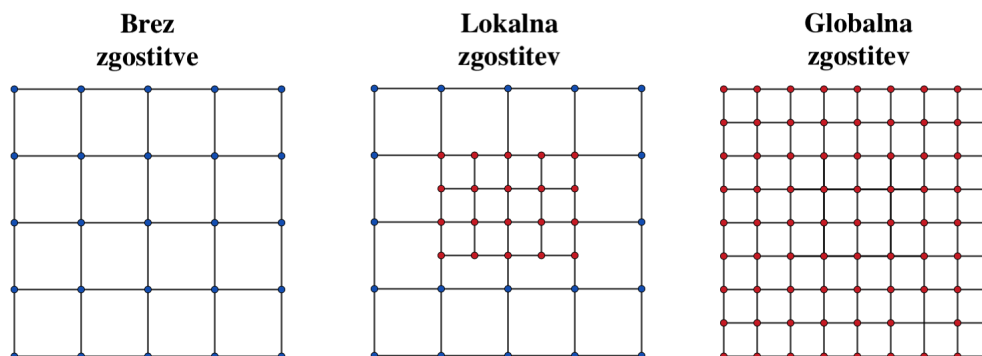
Izogeometrična analiza je ugledala luč sveta leta 2005 z objavo članka [9], kot metoda reševanja parcialnih diferencialnih enačb, ki združuje svetova analize končnih elementov in računalniško podprtega oblikovanja. Prednost reševanja parcialnih diferencialnih enačb z metodo izogeometrične analize je predvsem v tem, da domeno enačbe opišemo eksaktno, medtem ko v primeru metode končnih elementov domeno aproksimiramo z mrežo. Podrobno obravnavo izogeometrične analize bralec najde v [2]. Izogeometrična analiza je torej moderna in obetavna tema s področja numerične analize.

Bazne funkcije, ki nastopajo v izogeometrični analizi, so definirane le na majhnem intervalu oz. v višjih dimenzijah območju, so gladke in imajo lepe aproksimacijske lastnosti, zato magistrsko delo začnemo s poglavjem o B-zlepkih ene spremenljivke. Za začetek B-zlepke definiramo in pokažemo nekaj pomembnih lastnosti kot so gladkost, particija enote in linearna neodvisnost. B-zlepke dveh spremenljivk definiramo kot tenzorski produkt dveh zlepkov ene spremenljivke in tudi za njih pokažemo, da imajo lastnosti, ki sledijo iz B-zlepkov ene spremenljivke. B-zlepki so dobro opisano področje in obširno teorijo B-zlepkov najdemo v [3, 4, 22].

V naslednjem poglavju vpeljemo škatlaste zlepke. Škatlasti zlepki so posplošitev B-zlepkov, v zadnjem času pa so aktualni na področju aproksimacije in izogeometrične analize, saj zaradi svoje oblike omogočajo dober opis geometrije domene. V tem poglavju najprej definiramo škatlasti zlepek in njegove osnovne lastnosti. Tudi škatlasti zlepki so dobro opisani. Podrobnosti in nadaljnje posplošitve najdemo v [5, 21]. Za tem predstavimo Bernstein-Bézierevo obliko, s katero lahko škatlasti zlepek učinkovito predstavimo in z njim računamo. Poglavje končamo z definicijo prostora premikov škatlastega zlepka, kjer pokažemo, da so v posebnem primeru premiki škatlastega zlepka linearno neodvisni, tvorijo particijo enote ter da so B-zlepki le poseben primer škatlastih zlepkov.

Za poglavjem o škatlastih zlepkih pa pridemo do osrednje teme magistrskega delahierarhičnih prostorov zlepkov. Glavna ideja hierarhičnih zlepkov je lokalno zgoščevanje. Denimo, da imamo aplikacijo kot je aproksimacija ali reševanje parcialne diferencialne enačbe na nekem območju. Opazimo, da je na nekem delu napaka numeričnega približka večja od sprejemeljive predpisane vrednosti. Želeli bi orodje, s pomočjo katerega

bi lahko del območja zgostili in tam izračunali boljši približek, medtem ko bi preostalo območje ostalo nespremenjeno (glej sliko 1). Sama ideja lokalnega zgoščevanja niti ni



Slika 1: Ideja lokalnega zgoščevanja.

tako nova, kar lahko vidimo v [7]. V zadnjem času pa se hierarhični prostori uporabljajo v teoriji aproksimacije, kot npr. v [8, 11], kjer vidimo uporabo tako hierarhičnih prostorov B-zlepkov, kot uporabo hierarhičnih prostorov škatlastih zlepkov. Uporaba enakih hierarhičnih prostorov zlepkov pa je aktualna tudi na področju izogeometrične analize, kakor vidimo v [10, 19, 26]. Privlačnost hierarhičnih prostorov je v dejstvu, da z njimi prihranimo predvsem na prostorski zahtevnosti, saj imamo z lokalnim zgoščevanjem manj prostostnih stopenj v primerjavi z globalnim, medtem ko se nam pri napaki to ne pozna. Izkaže se, da lokalno zgoščevanje in definicija hierarhičnih prostorov ni trivialna naloga. Glavni izziv je pokazati, da hierarhični prostori zlepkov ohranjajo enake lastnosti kot navadni. V tem poglavju najprej definiramo ugnezdene prostore in domene B-zlepkov ter pokažemo omejitve, ki jih moramo ob tem sprejeti. Za tem definiramo hierarhično bazo prostora B-zlepkov in pokažemo, da so tudi elementi baze hierarhičnega prostora B-zlepkov linearno neodvisni, medtem ko particije enote nimamo nujno zagotovljene. Za tem definiramo še prirezano bazo prostora B-zlepkov in pokažemo, da so elementi te baze prav tako linearno neodvisni, poleg tega pa imajo tudi lastnost particije enote. V drugem delu tega poglavja definiramo še hierarhično bazo prostora premikov škatlastega zlepka. Opazimo, da so ideje podobne, vendar moramo v primeru škatlastih zlepkov sprejeti nekatere dodatne, strožje omejitve. Za določen tip škatlastih zlepkov zopet pokažemo, da so linearno neodvisni tudi v primeru hierarhične baze ter da v tem primeru ohranijo lastnost particije enote.

Hierarhične baze zlepkov želimo uporabiti v kakšni aplikaciji in predpogoj za to je implementacija v nekem programskem jeziku. V tem magistrskem delu izberemo okolje Matlab in v njem implementiramo hierarhične baze B-zlepkov in škatlastih zlepkov. Implementaciji namenimo svoje poglavje in v njem opišemo glavne programerske prijeme, da bralec lažje sledi programski kodi, zapisani v prilogah. Najprej opišemo im-

plementacijo (prirežane) hierarhične baze B-zlepkov. Na srečo imamo v okolju Matlab cel paket dobro implementiranih struktur ter funkcij za delo z B-zlepki, ki jih uporabljamo pri implementaciji. Podrobnosti in navodila za uporabo tega paketa najdemo v [6]. Naš cilj je učinkovita implementacija algoritmov zgoščevanja in algoritma za konstrukcijo hierarhične baze. Poleg B-zlepkov implementiramo tudi hierarhično bazo škatlastih zlepkov, kar pa je za razliko od B-zlepkov nekoliko zahtevnejša naloga, saj za primer škatlastih zlepkov ne obstaja veliko že vgrajenih funkcij. Vseeno so nam v veliko pomoč vgrajene funkcije, ki operirajo s triangulacijami (glej [17]). V primeru škatlastih zlepkov moramo najprej implementirati strukturo tega zlepka in šele nato se lahko lotimo algoritmov zgoščevanja in konstrukcije hierarhične baze.

Magistrsko delo končamo s poglavjem o aplikacijah. Želimo preveriti delovanje uporabe hierarhičnih prostorov zlepkov pri aproksimaciji po metodi najmanjših kvadratov in reševanju Poissonove enačbe na štirikotni domeni z Galerkinovo metodo. Na začetku poglavja najprej kratko zapišemo teoretično ozadje omenjenih aplikacij. Bolj izčrpno obravnavo teoretičnega ozadja aproksimacije in numeričnega reševanja parcialnih diferencialnih enačb bralec najde v [14, 20, 25]. Po opisu teoretičnega ozadja predstavimo rezultate numeričnih poskusov. Najprej preverimo red konvergence v drugi normi za implementirane aplikacije. Čisto na koncu pa naredimo še nekaj preprostih primerov aproksimacije in reševanja Poissonove enačbe z obema vrstama hierarhičnih baz zlepkov, kjer je glavni cilj pokazati, da so metode dobro implementirane. Poleg tega s temi primeri preverimo hipotezo, da je prostorska zahtevnost metod z lokalnim zgoščevanjem manjša v primerjavi z globalnim, ob tem pa se nam napaka numeričnega približka bistveno ne poveča.

2 B-ZLEPKI

V tem poglavju si ogledamo nekaj stvari iz teorije B-zlepkov ene spremenljivke, ki bodo služile kot nujno potrebna podlaga za nadaljnje delo. Najprej definiramo nekaj osnovnih pojmov, nato bazne funkcije prostora B-zlepkov, navedemo nekaj pomembnih lastnosti in kakšen primer, ter nazadnje še definicijo poljubne funkcije iz prostora B-zlepkov.

Za razliko od večine literature uporabljamo t.i. karakteristične funkcije. Uporaba takšnih funkcij precej intuitivna in z njo nekoliko lažje, vsaj v teoriji, računamo z B-zlepki.

Definicija 2.1. Naj bo $x \in \mathbb{R}$ in I interval na \mathbb{R} . Potem lahko definiramo karakteristično funkcijo $\mathbf{1} : \mathbb{R} \rightarrow \{0, 1\}$ na naslednji način:

$$\mathbf{1}_I(x) := \begin{cases} 1, & x \in I \\ 0, & \text{sicer} \end{cases}$$

Definicija 2.2. Naj bo $f : A \subset \mathbb{R}^d \rightarrow \mathbb{R}$. Potem je **nosilec funkcije** f (*angl. support*) množica

$$\text{supp}(f) = \{x \in A : f(x) \neq 0\}.$$

2.1 BAZA PROSTORA

Da bi ustrezno definirali B-zlepke potrebujemo vektor vozlov. Vektor vozlov in večkratnost vozla nam podajata naslednji definiciji.

Definicija 2.3. Naj bo $u_1 \leq u_2 \leq \dots \leq u_m$ naraščujoče zaporedje realnih števil. Elementu u_i iz zaporedja pravimo **vozel**, vektorju $U = (u_1, u_2, \dots, u_m)$ pa **vektor vozlov**. Polodprtemu intervalu $[u_i, u_{i+1})$, $u_i, u_{i+1} \in U$ pravimo **razpon vozla** u_i .

Definicija 2.4. Naj bo $U = (u_1, u_2, \dots, u_m)$ vektor vozlov. Če za $u_i \in U$ velja $u_i = u_{i+1} = \dots = u_{i+k-1}$ pravimo, da ime vozel u_i **večkratnost** k in to označimo z $\mu(u_i) = k$.

Za definicijo bazne funkcije prostora B-zlepkov potrebujemo še stopnjo bazne funkcije, ki jo označimo s p . Naslednja definicija (rekurzivno) definira i -to bazno funkcijo.

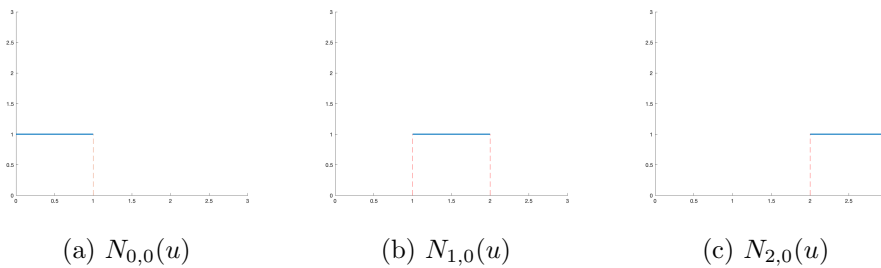
Definicija 2.5 (Cox - de Boor). Naj bo $U = (u_0, u_1, \dots, u_{m-1})$ vektor vozlov. Poljubno q -to bazno funkcijo definiramo na naslednji način.

$$N_{i,0}(u) = \mathbf{1}_{[u_i, u_{i+1})}(u), i = 0, 1, \dots, m-1$$

$$N_{i,q}(u) = \frac{u - u_i}{u_{i+q} - u_i} N_{i,q-1}(u) + \frac{u_{i+q+1} - u}{u_{i+q+1} - u_{i+1}} N_{i+1,q-1}(u), 1 \leq q \leq p.$$

Primer 2.6. Naj bo $U = (0, 1, 2, 3)$ in naj bo $p = 0$. Potem so vse bazne funkcije $N_{i,0}$, $i = 0, 1, 2$, enake

$$N_{0,0} = \mathbf{1}_{[0,1)}(u), N_{1,0} = \mathbf{1}_{[1,2)}(u), N_{2,0} = \mathbf{1}_{[2,3)}(u)$$



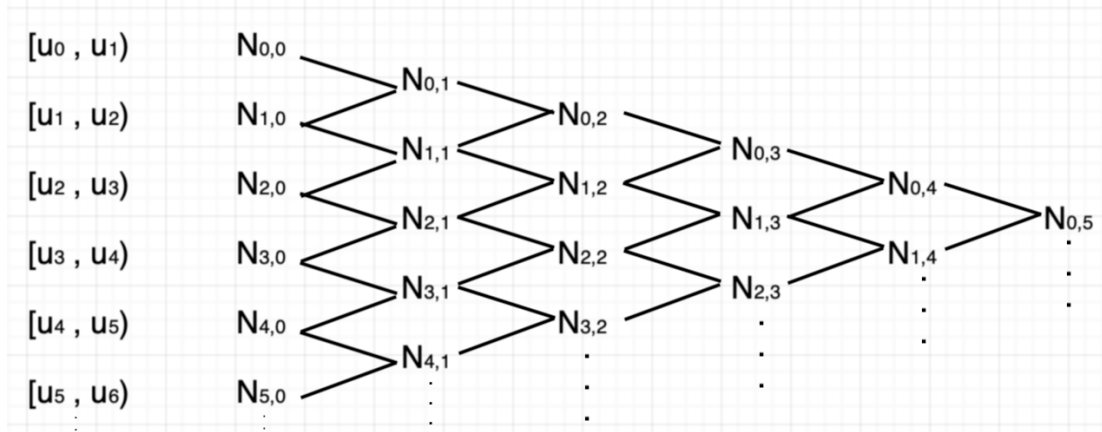
Slika 2: Grafi baznih funkcij $N_{i,0}$, $i = 0, 1, 2$.

Za lažje razumevanje računanja baznih funkcij uporabimo tako imenovano trikotno shemo. V prvi stolpec zapišemo vse razpone vozlov in v nadaljnje vse bazne funkcije, kot je prikazano na sliki 3.

Iz sheme je razvidno, da za izračun $N_{i,1}$ potrebujemo $N_{i,0}$ in $N_{i+1,0}$. Torej moramo najprej poračunati vse $N_{i,0}$, kar je preprosto, saj so te funkcije baza rekurzije. Šele nato lahko poračunamo vse potrebne $N_{i,1}$ in zatem še vse potrebne $N_{i,2}$. Postopek nadaljujemo dokler ne izračunamo vseh željenih $N_{i,q}$, $q \leq p$.

Primer 2.7. Naj bo zopet $U = (0, 1, 2, 3)$. Izračunajmo bazni funkciji $N_{0,1}$ in $N_{0,2}$.

Za izračun $N_{0,1}$ potrebujemo $N_{0,0}$ in $N_{1,0}$. Po definiciji je



Slika 3: Trikotna shema.

$$N_{0,1}(u) = \frac{u - u_0}{u_1 - u_0} N_{0,0}(u) + \frac{u_2 - u}{u_2 - u_1} N_{1,0}(u).$$

Ker so $u_0 = 0$, $u_1 = 1$ in $u_2 = 2$, dobimo

$$N_{0,1}(u) = u \cdot N_{0,0}(u) + (2 - u)N_{1,0}(u).$$

Bazni funkciji $N_{0,0}$ in $N_{1,0}$ pa sta znani iz definicije in tako dobimo

$$N_{0,1}(u) = u \cdot \mathbf{1}_{[0,1)}(u) + (2 - u)\mathbf{1}_{[1,2)}(u).$$

oziroma zapisano v bolj konvencionalni obliki

$$N_{0,1}(u) = \begin{cases} u, & u \in [0, 1) \\ 2 - u, & u \in [1, 2). \end{cases}$$

Podobno lahko izračunamo tudi $N_{1,1}$ in dobimo

$$N_{1,1}(u) = (u - 1)\mathbf{1}_{[1,2)}(u) + (3 - u)\mathbf{1}_{[2,3)}(u) = \begin{cases} u - 1, & u \in [1, 2) \\ 3 - u, & u \in [2, 3). \end{cases}$$

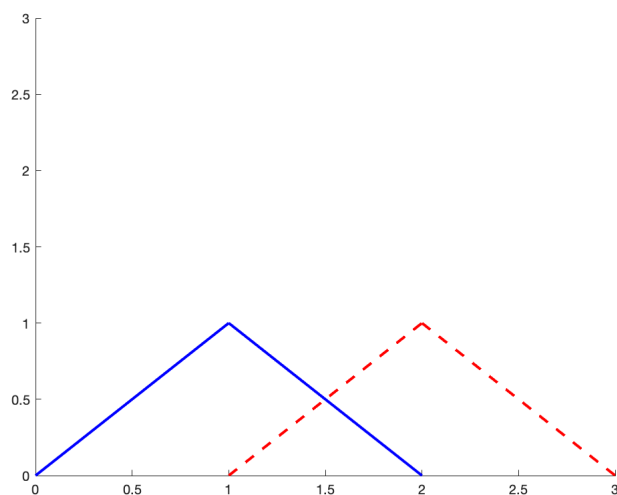
Slika 4 prikazuje bazni funkciji $N_{0,1}$ in $N_{1,1}$ na intervalu $[0, 3]$.

Izračunajmo na koncu še bazno funkcijo $N_{0,2}$. Potrebujemo bazni funkciji $N_{0,1}$, $N_{1,1}$ in dobimo

$$\begin{aligned} N_{0,2}(u) &= \frac{u - u_0}{u_2 - u_0} N_{0,1}(u) + \frac{u_3 - u}{u_3 - u_1} N_{1,1}(u) \\ &= \frac{1}{2}u \cdot N_{0,1}(u) + \frac{1}{2}(3 - u) \cdot N_{1,1}(u). \end{aligned}$$

Ker smo obe funkciji že poračunali, lahko direktno vstavimo predpis in dobimo

$$N_{0,2}(u) = \frac{1}{2}u(u\mathbf{1}_{[0,1)}(u) + (2 - u)\mathbf{1}_{[1,2)}(u)) + \frac{1}{2}(3 - u) \cdot ((u - 1)\mathbf{1}_{[1,2)}(u) + (3 - u)\mathbf{1}_{[2,3)}(u)).$$

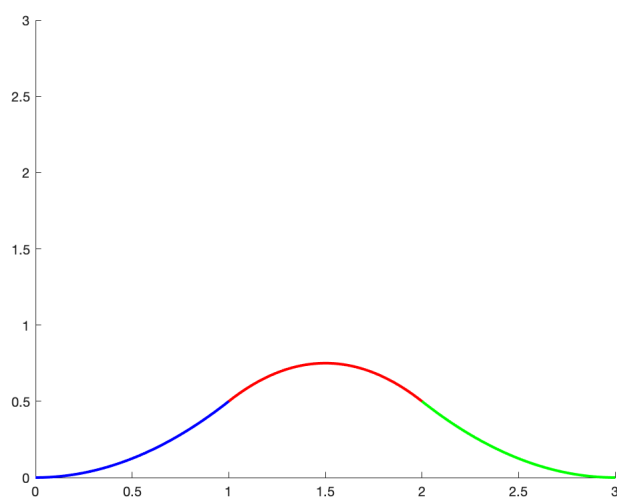
Slika 4: Bazni funkciji $N_{0,1}$ (modra) ter $N_{1,1}$ (rdeča, črtkana).

Z nekaj osnovnega računanja lahko vidimo, da velja

$$N_{0,2}(u) = \frac{1}{2}u^2 \cdot \mathbf{1}_{[0,1)}(u) + \frac{1}{2}(-3 + 6u - 2u^2) \cdot \mathbf{1}_{[1,2)}(u) + \frac{1}{2}(3 - u)^2 \cdot \mathbf{1}_{[2,3)}(u)$$

oziroma

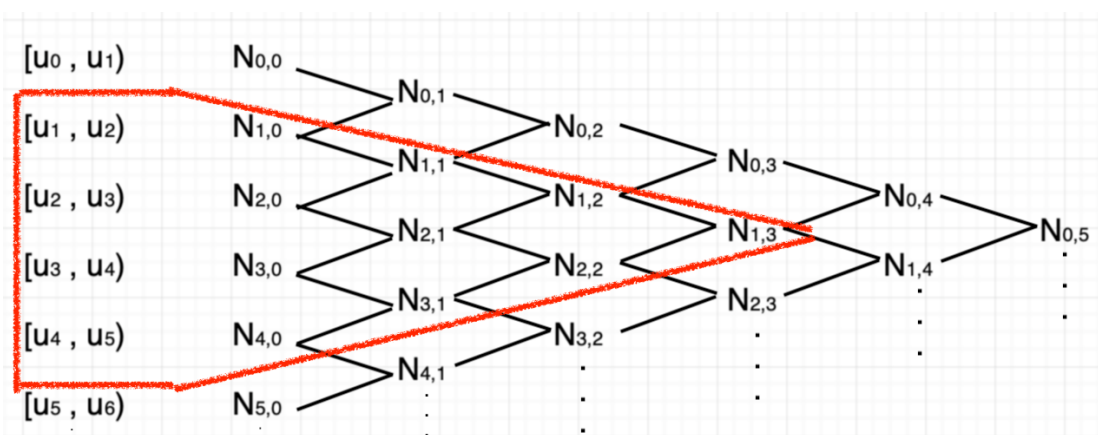
$$N_{0,2}(u) = \begin{cases} \frac{1}{2}u^2, & u \in [0, 1) \\ \frac{1}{2}(-3 + 6u - 2u^2), & u \in [1, 2) \\ \frac{1}{2}(3 - u)^2, & u \in [2, 3). \end{cases}$$

Slika 5: Bazna funkcija $N_{0,2}$.

Če narišemo grafe funkcij vseh treh predpisov na treh intervalih dobimo krivuljo na prejšnji sliki. Vsak odsek krivulje je označen z drugačno barvo, da vidimo, kako se

krivulje med seboj povežejo. Na sliki opazimo tudi, da je krivulja na intervalu $[0, 3]$ oz. v vozlih gladka. V nadaljevanju bomo videli, da v splošnem temu ni tako, če imamo nekatere vozle z večkratnostjo več kot 1.

Poglejmo si dve pomembni opombi, ki jih lahko opazimo že z opazovanjem trikotne sheme. Za izračun $N_{i,1}$ smo potrebovali funkciji $N_{i,0}$ ter $N_{i+1,0}$, ki sta neničelni na razponih $[u_i, u_{i+1})$ ter $[u_{i+1}, u_{i+2})$. Sledi, da je nosilec funkcije $N_{i,1}$ vsebovan na intervalu $[u_i, u_{i+2})$. Podobno za izračun $N_{i,2}$ potrebujemo $N_{i,1}$ ter $N_{i+1,1}$, ki sta neničelni na $[u_i, u_{i+2})$ ter $[u_{i+1}, u_{i+3})$. Sledi, da je nosilec funkcije $N_{i,2}$ vsebovan na intervalu $[u_i + 1, u_{i+3})$. Opazimo, da lahko nosilec vsake $N_{i,p}$ določimo iz trikotne sheme na način, da se iz bazne funkcije, ki nas zanima, pomikamo levo tako, da zajamemo vse funkcije, ki so bile potrebne za izračun, dokler ne pridemo do prvega stolpca, od koder preberemo vse intervale, na katerih je bazna funkcija neničelna. Primer je podan na naslednji sliki, kjer določamo nosilec funkcije $N_{1,3}$.



Slika 6: Na sliki je označen nosilec $\text{supp}(N_{1,3}) = [u_1, u_5)$.

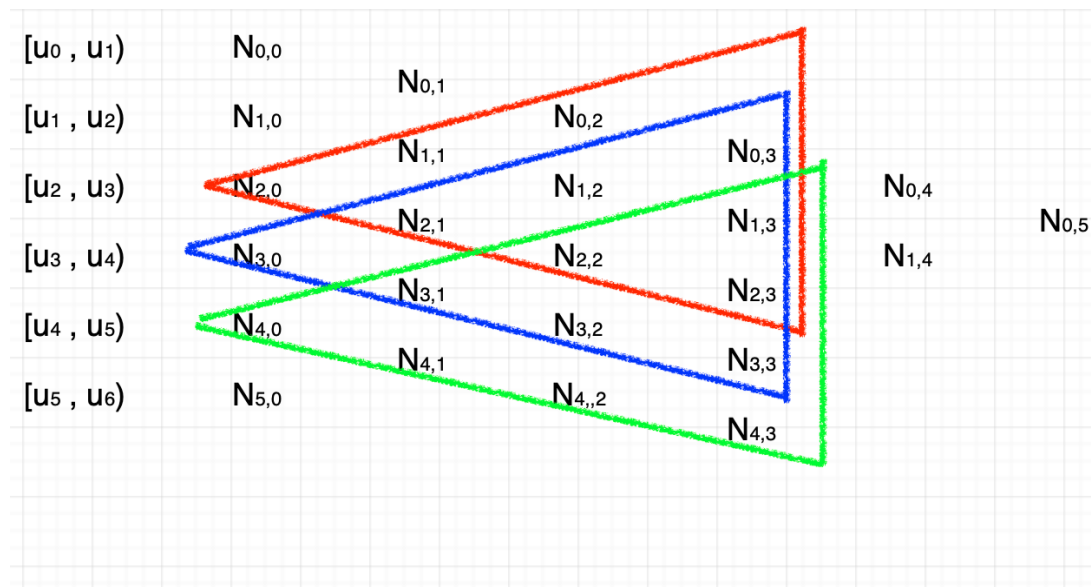
Opomba 2.8. Vsaka bazna funkcija $N_{i,p}$ je neničelna na intervalu $[u_i, u_{i+p+1})$.

Poglejmo si še obratno smer. Zanima nas, koliko baznih funkcije stopnje največ p je neničelnih na danem razponu $[u_i, u_{i+1})$.

V trikotni shemi lahko opazujemo, katere bazne funkcije uporabljajo bazno funkcijo $N_{i,0}$. To naredimo, kot je prikazano na spodnji sliki. Opazimo, da črte enake barve v smislu štetja funkcij tvorijo enakokraki trikotnik s $p + 1$ stolpci. Iz zadnjega stolpca preberemo funkcije, ki uporabljajo $N_{i,0}$ in so s tem neničelne na $[u_i, u_{i+1})$. Ker je trikotnik enakokraki ima lahko zadnji stolpec največ $p + 1$ elementov. To pa nas pripelje do naslednje opombe.

Opomba 2.9. Na vsakem razponu $[u_i, u_{i+1})$ je neničelnih funkcij stopnje p največ $p + 1$, in sicer

$$N_{i-p,p}(u), N_{i-p+1,p}(u), \dots, N_{i-1,p}(u), N_{i,p}(u).$$



Slika 7: Nenične funkcije stopnje 3 na razponih $[u_2, u_3)$, $[u_3, u_4)$, $[u_4, u_5)$.

2.1.1 Lastnosti baznih funkcij

- Funkcija $N_{i,p}$, je odsekoma polinomomska funkcija stopnje manj ali enako p .
- Velja $N_{i,p}(u) \geq 0$ za vsak $u \in \mathbb{R}$.
- **Lokalni nosilec.**

$$\text{supp}(N_{i,p}) = \{u \in \mathbb{R} : N_{i,p}(u) \neq 0\} = [u_i, u_{i+p+1}).$$

- Na vsakem intervalu $[u_i, u_{i+1})$ ima največ $p + 1$ baznih funkcij stopnje p neničelni nosilec.
- **Particija enote.** Vsota vseh baznih funkcij stopnje p na intervalu $[u_i, u_{i+1})$ je enaka 1.

Dokaz. Trdimo, da je $\sum_{j=0}^p N_{j,p}(u) = 1$, za vsak $u \in [u_i, u_{i+1})$. Brez škode za splošnost lahko predpostavimo, da je $i = p$ in potem trdimo $\sum_{j=0}^p N_{j,p}(u) = 1$, za vsak $u \in [u_p, u_{p+1})$. Dokaza se lotimo z indukcijo.

Za bazo indukcije vzamemo $p = 0$. Potem je

$$\sum_{j=0}^p N_{j,p}(u) = \sum_{j=0}^0 N_{j,0}(u) = N_{0,0}(u) = 1 \text{ za vsak } u \in [u_p, u_{p+1}) = [u_0, u_1).$$

Naj bo sedaj $p \in \mathbb{N}$. Imamo indukcijsko predpostavko, ki pravi, da je vsota iz trditve enaka 0 za vsa naravna števila manjša od p . Obravnavajmo vsoto

$$\begin{aligned} \sum_{j=0}^p N_{j,p}(u) &= N_{0,p}(u) + N_{1,p}(u) + \dots + N_{p,p}(u) = \\ &= \frac{u - u_0}{u_{p-1} - u_0} N_{0,p-1}(u) + \frac{u_p - u}{u_p - u_1} N_{1,p-1}(u) + \\ &+ \frac{u - u_1}{u_p - u_1} N_{1,p-1}(u) + \frac{u_{1+p} - u}{u_{1+p} - u_2} N_{2,p-1}(u) + \dots + \\ &+ \frac{u - u_p}{u_{2p-1} - u_p} N_{p-1,p-1}(u) + \frac{u_{2p} - u}{u_{2p} - u_{p+1}} N_{p+1,p-1}(u). \end{aligned}$$

Vemo, da je $\text{supp} N_{0,p-1} = [u_0, u_p)$ in $\text{supp} N_{p-1,p-1} = [u_{p+1}, u_{2p+1})$. Sledi, da sta obe funkciji enaki 0 za vsak $u \in [u_p, u_{p+1})$. Torej lahko prejšnjo vsoto zapišemo kot

$$\begin{aligned} &\left(\frac{u_p - u}{u_p - u_1} + \frac{u - u_1}{u_p - u_1} \right) N_{1,p-1}(u) + \\ &+ \left(\frac{u_{1+p} - u}{u_{1+p} - u_2} + \frac{u - u_2}{u_{1+p} - u_2} \right) N_{2,p-1}(u) + \dots + \\ &+ \left(\frac{u_{2p-1} - u}{u_{2p-1} - u_p} + \frac{u - u_p}{u_{2p-1} - u_p} \right) N_{p,p-1} = \\ &= N_{1,p-1}(u) + N_{2,p-1}(u) + \dots + N_{p,p-1}(u). \end{aligned}$$

Zadnja vsota pa je enaka 1 po indukcijski predpostavki.

□

- Formulo za j -ti bazni zlepek stopnje p lahko alternativno definiramo kot

$$N_{j,p}(u) = \sum_{i=j}^{j+p} N_{j,p}^{\{i\}}(u), \quad p \geq 0, \quad (2.1)$$

kjer je $N_{j,p}^{\{i\}}(u)$ polinom stopnje p in je enak 0, če za nek i velja $u_i = u_{i+1}$. Iz Definicije 2.5 ugotovimo, da sta prvi in zadnji polinom baznega zleпка $N_{j,p}^{\{i\}}(u)$ enaka

$$N_{j,p}^{\{j\}}(u) = \frac{(u - u_j)^p}{\prod_{i=1}^p (u_{j+i} - u_j)},$$

$$N_{j,p}^{\{j+p\}}(u) = \frac{(u_{j+p+1} - u)^p}{\prod_{i=1}^p (u_{j+p+1} - u_{j+i})}.$$

- **Odvod.** Naj bo $N_{j,p}(u)$ j -ti bazni zlepek nad vektorjem vozlov U . Odvod označimo z $D_+ N_{j,p}(u)$ in je enak

$$D_+ N_{j,p}(u) = p \left(\frac{N_{j,p-1}(u)}{u_{j+p} - u_j} - \frac{N_{j+1,p-1}(u)}{u_{j+p+1} - u_{j+1}} \right), \quad p \geq 1. \quad (2.2)$$

Dokaz. Dokaz trditve bralec najde v [16]. □

- Naj bodo bazne funkcije stopnje manj ali enako p in naj bo $|U| = m + 1$. Nadalje naj bo število baznih funkcij enako $n + 1$. Potem velja

$$m = n + p + 1.$$

Dokaz. Naj bo $N_{n,p}(u)$ zadnja bazna funkcija baze. Vemo, da je $N_{n,p}(u) > 0$, $u \in [u_n, u_{n+p+1})$. Ker je $N_{n,p}$ zadnja funkcija baze, mora biti $u_{n+p+1} = u_m \Rightarrow m = n + p + 1$. □

- Naj bo $u_i \in U$ in $\mu(u_i) = k$. Potem je $N_{i,p}(u)$ C^{p-k} zvezna v u_i .

Dokaz. Naj bo u vozlel, nastopajoč v nosilcu bazne funkcije $N_{j,p}(u)$ in naj velja $\mu(u) = p+1$. Vemo, da so vozli iz nosilca bazne funkcije enaki $[u_j, u_{j+1}, \dots, u_{j+p+1}]$, in ker za nek u velja $\mu(u) = p + 1$, ločimo dve možnosti

- (i) $u_j < u_{j+1} = \dots = u_{j+p+1}$. Potem je

$$N_{j,p} = \frac{(u - u_j)^p}{(u_{j+p+1} - u_j)^p} \mathbf{1}_{[u_j, u_{j+1})}(u).$$

Iz prejšnje enačbe takoj opazimo, da

$$\lim_{u \searrow u_{j+1}} N_{j,p}(u) \neq \lim_{u \nearrow u_{j+1}} N_{j,p}(u),$$

saj je $\lim_{u \searrow u_{j+1}} N_{j,p}(u) = 0$, medtem ko je $\lim_{u \nearrow u_{j+1}} N_{j,p}(u) = \frac{(u_{j+1} - u_j)^p}{(u_{j+p+1} - u_j)^p}$.

(ii) $u_j = u_{j+1} = u_{j+p} < u_{j+p+1}$. Potem je

$$N_{j,p} = \frac{(u_{j+p+1} - u)^p}{(u_{j+p+1} - u_j)^p} \mathbf{1}_{[u_j, u_{j+1})}(u).$$

Iz prejšnje enačbe takoj opazimo, da

$$\lim_{u \searrow u_j} N_{j,p}(u) \neq \lim_{u \nearrow u_j} N_{j,p}(u),$$

saj je $\lim_{u \searrow u_j} N_{j,p}(u) = \frac{(u_{j+p+1} - u_j)^p}{(u_{j+p+1} - u_j)^p}$, medtem ko je $\lim_{u \nearrow u_{j+1}} N_j(u) = 0$.

Iz obeh primerov vidimo, da za $\mu(u) \geq p+1$ zveznosti nimamo. Predpostavimo nadalje, da je $\mu(u) = p$ in da velja $u_j < u_{j+1} = \dots = u_{j+p} < u_{j+p+1}$. Potem je

$$N_{j,p}(u) = \frac{(u - u_j)^p}{(u_{j+p} - u_j)^p} N_{j,0}(u) + \frac{(u_{j+p+1} - u)^p}{(u_{j+p+1} - u_{j+1})^p} N_{j+p,0}(u).$$

Pogledamo limiti

$$\lim_{u \nearrow u_{j+1}} N_{j,p}(u) = \frac{(u_{j+1} - u_j)^p}{(u_{j+1} - u_j)^p} N_{j,0}(u) = \frac{1}{1} = 1,$$

$$\lim_{u \searrow u_{j+1}} N_{j,p}(u) = \frac{(u_{j+p+1} - u_{j+1})^p}{(u_{j+p+1} - u_{j+1})^p} N_{j+p,0}(u) = \frac{1}{1} = 1.$$

Opazimo, da imata funkciji enaki limiti. Ker sta obe funkciji monotoni sklepamo, da je zveznost v vozlu u_{j+1} enaka C^0 . Ostale primere pokažemo z indukcijo na p . Ločimo nekaj primerov.

(i) $p=1$. Potem je j -ta bazna funkcija stopnje 1 enaka

$$N_{j,1}(u) = \frac{u - u_j}{u_{j+1} - u_j} \mathbf{1}_{[u_j, u_{j+1})}(u) + \frac{u_{j+2} - u}{u_{j+2} - u_{j+1}} \mathbf{1}_{[u_{j+1}, u_{j+2})}(u).$$

Oba kosa zgornje funkcije sta odsekoma linearna. Če pogledamo levo in desno limito, vidimo, da se ujemata.

$$\lim_{u \nearrow u_{j+1}} N_{j,1}(u) = \frac{(u_{j+1} - u_j)}{(u_{j+1} - u_j)} N_{j,0}(u) = \frac{1}{1} = 1$$

in

$$\lim_{u \searrow u_{j+1}} N_{j,1}(u) = \frac{(u_{j+2} - u_{j+1})^p}{(u_{j+2} - u_{j+1})^p} N_{j+1,0}(u) = \frac{1}{1} = 1.$$

To pa pomeni, da je funkcija zveznosti C^0 v vozlu s stopnjo 1.

- (ii) V tej točki predpostavimo, da je za $p \geq 2$ bazna funkcija $N_{j,p-1} \in C^{p-1-\mu(u)}$ v nekem vozlu u . Potem si lahko pogledamo primer, ko je $u_j = \dots = u_{j+p-1} < u_{j+p} < u_{j+p+1}$ ¹. Potem je

$$N_{j,p}(u) = \frac{u - u_j}{u_{j+p} - u_j} N_{j,p-1}(u) + \frac{u_{j+p+1} - u}{u_{j+p+1} - u_{j+1}} N_{j+1,p-1}(u).$$

Vidimo, da je prvi člen zgornje enačbe enak 0, pri $u = u_j$, drugi pa je zvezen po indukcijski predpostavki, saj je stopnje $p - 1$. Torej je res $N_{j,p-1}(u) \in C^{p-1-\mu(u)}$ v nekem u , kjer je $\mu(u) = p - 1$.

- (iii) V zadnji točki le še združimo vse skupaj. Po enačbi za odvod in po točki (ii) vidimo, da je

$$D_+ N_{j,p}(u) \in C^{p-1-\mu(u)}.$$

Od tod pa lahko sklepamo, da je

$$N_{j,p}(u) \in C^{p-\mu(u)}. \quad \square$$

2.2 PROSTOR B-ZLEPKOV

V tem podpoglavju definiramo prostor B-zlepkov. Prostor B-zlepkov stopnje p nad vektorjem vozlov U označimo s $\mathbb{S}_{p,U}$.

Definicija 2.10. Naj bodo $\mathbf{P}_0, \dots, \mathbf{P}_n$ kontrolne točke ter $\mathbb{S}_{p,U}$ prostor B-zlepkov nad vektorjem vozlov U . Potem je poljubna funkcija $f \in \mathbb{S}_{p,U}$ definirana kot

$$f(u) = \sum_{i=0}^n \mathbf{P}_i \cdot N_{i,p}(u)$$

kjer je $N_{i,p}$ i -ta bazna funkcija stopnje p .

Za definicijo B-zlepkov potrebujemo torej $n + 1$ kontrolnih točk. Ker morajo n, p, m zadoščati enačbi $m = n + p + 1$ potrebujemo za definicijo funkcije iz prostora B-zlepkov podati $n + p + 2$ vozlov. Z drugimi besedami: s stopnjo prostora in številom kontrolnih točk imamo natančno določeno število vozlov (ali obratno). Krivulje B-zlepkov definirane s samimi enostavnimi vozli (vozli z večkratnostjo 1) imenujemo *odprte*. Zaradi lastnosti, da je na razponu $[u_i, u_{i+1})$ neničelnih največ $p + 1$ baznih funkcij stopnje p opazimo, da razponi $[u_0, u_p)$ ter $[u_{m-p}, u_m)$ nimajo "polnega nosilca" in jih lahko pri odprtih krivuljah ignoriramo. Od tod sledi pomembna opomba.

Opomba 2.11. Domena odprtih krivulj prostora $\mathbb{S}_{p,U}$ je interval $[u_p, u_{m-p}]$.

¹Enako bi postopali tudi, če bi imeli $p - 1$ enakih vozlov na desni.

V aplikacijah si pogosto želimo, da krivulja B-zlepkov interpolira prvo in zadnjo kontrolno točko, kar pa se pri odprtih krivuljah ne zgodi. Želeli bi si poln nosilec nad vsemi razponi vektorja vozlov. Recimo, da definiramo vektor vozlov nad nekim intervalom $[a, b]$. Da bi dosegli poln nosilec nad vsemi razponi, definiramo vektor U kot

$$U = (u_0, u_1, \dots, u_p, u_{p+1}, u_{p+2}, \dots, u_{p+m}, u_{p+m+1}, \dots, u_{m+2p+1}),$$

kjer velja

$$a := u_0 = u_1 = \dots = u_p < u_{p+1} < u_{p+2} < \dots < u_{p+m} < u_{p+m+1} = \dots = u_{m+2p+1} := b.$$

S tem povečamo število baznih funkcij. Čeprav na ta način izgubimo maksimalno možno gladkost na robu intervala $[a, b]$, se v njegovi notranjosti ta ohrani, saj imamo same enostavne vozle. Primer takšnega prostora nam podaja primer 2.13.

Opomba 2.12. Za vektor vozlov, ki je sestavljen na način, da sta prvi in zadnji vozle večkratnosti $p + 1$, v notranjosti pa so vsi vozli enostavni, pravimo, da je $(p + 1)$ -regularen.

Primer 2.13. Nad intervalom $[0, 4]$ želimo skonstruirati prostor kubičnih B-zlepkov. Želimo tudi, da so vozli razporejeni ekvidistantno v točkah $x = 1, 2, 3, 4$. To je torej prostor $\mathbb{S}_{p,U}$, kjer je $p = 3$ in

$$U = (0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4).$$

Vse bazne funkcije tega prostora so prikazane na sliki 8.

Krivulje B-zlepkov definirane z vektorji vozlov, ki v notranjosti vsebujejo same enostavne vozle, robni vozli pa so večkratnosti $p + 1$ imenujemo *zaprte*. Da zaprte krivulje res interpolirajo prvo in zadnjo kontrolno točko dokažemo v lemi 2.14.

Lema 2.14. *Naj bo p polinomska stopnja in*

$$U = (u_0, u_1, \dots, u_p, u_{p+1}, u_{p+2}, \dots, u_{p+m}, u_{p+m+1}, \dots, u_{m+2p+1})$$

$(p + 1)$ -regularen vektor vozlov. Krivulja f iz prostora $\mathbb{S}_{p,U}$ interpolira prvo in zadnjo kontrolno točko, torej točki \mathbf{P}_0 in \mathbf{P}_{m+p} .

Dokaz. Pokažemo samo za prvo točko. Naj bo p polinomska stopnja in BŠZS predpostavimo, da je

$$U = (a, a, \dots, a, x, b, b, \dots, b),$$

kjer so $\mu(a) = p + 1$, $\mu(b) = p + 1$ in $\mu(x) = 1$. Imamo torej le tri različne vozle. Takšna krivulja pa je zaprta. Krivuljo $f \in \mathbb{S}_{p,U}$ zapišemo kot

$$f(u) = \sum_{i=0}^{p+1} \mathbf{P}_i N_{i,p}(u), \tag{2.3}$$



Slika 8: Bazne funkcije prostora kubičnih B-zlepkov nad intervalom $[0, 4]$.

kjer so $N_{i,p}$ bazne funkcije prostora $\mathbb{S}_{p,U}$. Ker je krivulja f zaprta, je njena domena interval $[a, b]$. Trdimo, da velja

$$f(a) = \mathbf{P}_0.$$

Če vsoto na desni strani izraza (2.3) razpišemo, dobimo

$$f(u) = \mathbf{P}_0 N_{0,p}(u) + \mathbf{P}_1 N_{1,p}(u) + \cdots + \mathbf{P}_p N_{p,p}(u).$$

Vse bazne funkcije, razen funkcije N_0 , so na robovih svojega nosilca enake 0. Od tod sledi

$$f(a) = \mathbf{P}_0 N_{0,p}(a).$$

Sedaj moramo le še pokazati, da velja $N_{0,p}(a) = 1$. Po definiciji 2.5 je

$$\begin{aligned} N_{0,p}(a) &= \frac{a-a}{a-a} N_{0,p-1}(a) + \frac{x-a}{x-a} N_{1,p-1}(a) = \\ N_{1,p-1}(a) &= \frac{a-a}{a-a} N_{1,p-2}(a) + \frac{x-a}{x-a} N_{2,p-2}(a). \end{aligned}$$

Ta postopek lahko nadaljujemo do konca rekurzije, torej dokler ne dobimo vsote dveh konstantnih zlepkov. Opazimo, da bo na vsakem nivoju rekurzije prvi člen desne strani izraza v definiciji 2.5 enak 0, saj so za vse indekse $i = 0, \dots, p$ in stopnje $q = p - i$, vozli u_i, u_{i+1}, u_{i+q} enaki a . Ulomek pri drugem členu desne strani pa bo za vse indekse $i = 0, \dots, p$ in stopnje $q = p - i$ enak 1, saj bo u_{i+q+1} vedno enak x , u_i pa bo vedno enak a . Rekurzivni postopek nadaljujemo, dokler ne dobimo

$$N_{p-1,1}(a) = \frac{a-a}{a-a} N_{p-1,0}(a) + \frac{x-a}{x-a} N_{p,0}(a) = N_{p,0}(a).$$

Po definiciji 2.5 pa sledi, da je $N_{p,0}(a) = 1$. S tem smo pokazali, da krivulja f interpolira točko \mathbf{P}_0 . Enak razmislek uporabimo tudi za dokaz interpolacije zadnje točke.

□

Z naslednjim primerom si pogledjmo razliko med odprtimi in zaprtimi krivuljami B-zlepkov.

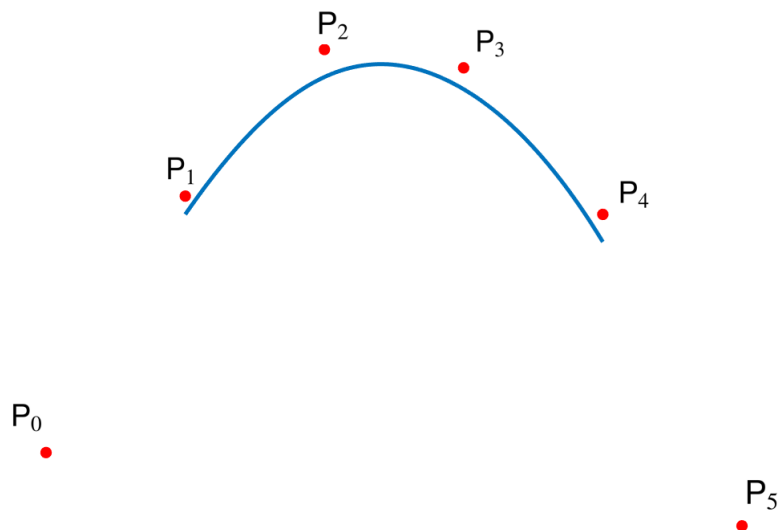
Primer 2.15. Naj bo $p = 3$ polinomska stopnja in

$$U = \left(0, \frac{1}{9}, \frac{2}{9}, \frac{1}{3}, \frac{4}{9}, \frac{5}{9}, \frac{2}{3}, \frac{7}{9}, \frac{8}{9}, 1 \right)$$

vektor vozlov. Z vektorjem vozlov U in stopnjo $p = 3$ definiramo prostor B-zlepkov $\mathbb{S}_{p,U}$. Za definicijo krivuljo prostora $\mathbb{S}_{p,U}$ potrebujemo torej 6 kontrolnih točk. Recimo, da so te enake

$$\mathbf{P} = \left\{ \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}, \begin{bmatrix} 1 \\ 2.9 \end{bmatrix}, \begin{bmatrix} 2 \\ 3.7 \end{bmatrix}, \begin{bmatrix} 3 \\ 3.6 \end{bmatrix}, \begin{bmatrix} 4 \\ 2.8 \end{bmatrix}, \begin{bmatrix} 5 \\ 1.1 \end{bmatrix} \right\}. \quad (2.4)$$

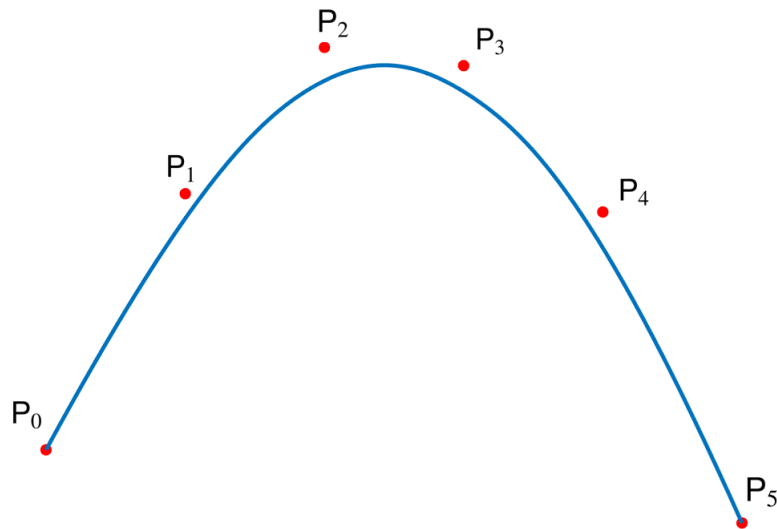
Slika 9 prikazuje krivuljo prostora $\mathbb{S}_{p,U}$, definirano s kontrolnimi točkami (2.4).



Slika 9: Primer odprte krivulje B-zlepkov.

Recimo, da bi želeli krivuljo stopnje 3 z istimi kontrolnimi točkami, ki bi interpolirala točki \mathbf{P}_0 ter \mathbf{P}_5 . Zaradi opombe 2.11 se to z odprtimi krivuljami ne more zgoditi, ker je domena krivulj premajhna. Za to definirajmo nov vektor vozlov

$$V = \left\{ 0, 0, 0, 0, \frac{1}{3}, \frac{2}{3}, 1, 1, 1, 1 \right\}$$



Slika 10: Primer vpete krivulje B-zlepkov.

in definirajmo nov prostor B-zlepkov stopnje 3, $\mathbb{S}_{p,V}$. Krivulja iz tega prostora z kontrolnimi točkami (2.4) je prikazana na sliki 10. Če primerjamo sliki 9 in 10 opazimo, da imata krivulji v notranjosti enako obliko. Sprememba je v tem, da ima vpeta krivulja poln nosilec nad vsemi razponi in ima posledično večjo domeno. Dosegli smo, da krivulja interpolira prvo in zadnjo kontrolno točko.

Za konec poglavja zapišimo še eno pomembno lastno B-zlepkov

Izrek 2.16. *Naj bo p polinomska stopnja in U $(p + 1)$ -regularen vektor vozlov, kjer označimo $m = |U|$. Potem so bazni zleпки prostora $\mathbb{S}_{p,U}$ linearno neodvisni na intervalu $[u_{p+1}, u_{m-p})$.*

Dokaz. Dokaz izreka opustimo. Bralec ga lahko najde v [16]. □

2.3 TENZORSKI PRODUKTI B-ZLEPKOV

V prejšnjem poglavju smo razdelali nekaj pomembnih definicij in lastnosti prostora $\mathbb{S}_{p,U}$. To nam da dobro osnovo za posplošitev na prostor B-zlepkov dveh spremenljivk, ki ga dobimo kot tenzorski produkt B-zlepkov ene spremenljivke. V tem poglavju bomo zapisali nekaj definicij in lastnosti takšnih zlepkov. Opazili bomo, da lastnosti sledijo iz primera ene spremenljivke. Uporabili bomo nekoliko drugačne oznake, ker se izkažejo za bolj uporabne tudi pri nadaljnjem delu. Pomembna je naslednja opomba.

Opomba 2.17. V prvem poglavju smo vektor vozlov označili kot $U = (u_0, \dots, u_m)$ in rekli, da ima U $m + 1$ elementov. V primeru dveh spremenljivk bi, kot bomo videli, to prineslo veliko zmede, saj bomo delali z dvema vektorjema. Od sedaj naprej bomo velikost vektorja izrazili s točnim številom elementov, kot je podano v lastnosti v poglavju 2.2.

Prostor tenzorskega produkta zlepkov označimo z \mathcal{B} . Prostor \mathcal{B} definiramo z dvema polinomskima stopnjama (p, q) ter z dvema vektorjema vozlov

$$U = (u_0, u_1, \dots, u_{n+p+1}), \quad V = (u_0, v_1, \dots, v_{m+q+1}).$$

Podobno kot v primeru z eno spremenljivko označimo večkratnost vozla v vektorju vozlov z $\mu(u, U)$ in $\mu(v, V)$. Bazna funkcija prostora \mathcal{B} je torej oblike

$$N_{i,j}^{p,q}(u, v) = N_i^p(u)N_j^q(v),$$

kjer sta $N_i^p(u)$ in $N_j^q(v)$ bazni funkciji prostorov B-zlepkov $\mathbb{S}_{p,U}$ in $\mathbb{S}_{q,V}$ definirani v poglavju 2.1, Definiciji 2.5.

Vsako funkcijo $f(u, v) \in \mathcal{B}$ lahko zapišemo kot

$$f(u, v) = \sum_{i=0}^n \sum_{j=0}^m \mathbf{d}_{i,j} N_{i,j}^{p,q}(u, v)$$

z

$$(u, v) \in [u_p, u_{n+1}] \times [v_q, v_{m+1}],$$

in kontrolnimi točkami $\mathbf{d}_{i,j}$, ki so posplošitev kontrolnih točk $\mathbf{P}_0, \dots, \mathbf{P}_n$ iz Definicije 2.10 in v primeru dveh spremenljivk tvorijo kontrolno mrežo.

Bazo prostora \mathcal{B} označimo z

$$\mathcal{N} = \{N_{i,j}^{p,q} : i = 0, 1, \dots, n, j = 0, 1, \dots, m\}.$$

Opomba 2.18. Zapisana definicija krivulje tenzorskih produktov B-zlepkov je funkcija, ki slika $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. V resnici v sklopu tega magistrskega dela takšne definicije ne bomo potrebovali velikokrat. Mnogo pogosteje nam bo prišla prav linearna kombinacija

$$f(x, y) = \sum_{i=0}^n \sum_{j=0}^m \alpha_{i,j} N_{i,j}^{p,q}(u, v),$$

kjer so $\alpha_{i,j} \in \mathbb{R}$. Takšna krivulja pa slika $\mathbb{R}^2 \rightarrow \mathbb{R}$. Na ta način bomo kasneje lažje dokazali lastnosti hierarhične baze, implementirali B-zlepke v ustreznem programskem jeziku, ter bazo tenzorskih produktov B-zlepkov uporabili v naših aplikacijah.

Iz primera ene spremenljivke lahko hitro posplošimo naslednje lastnosti

1. **Lokalni nosilec:**

$$\text{supp}(N_{i,j}^{p,q}) = \{(u, v) : N_{i,j}^{p,q}(u, v) \neq 0\} = [u_i, u_{i+p+1}] \times [v_j, v_{j+q+1}].$$

2. **Lokalna linearna neodvisnost:** Naj bo $\Omega' \subset \Omega$ odprta podmnožica. Potem so B-zlepki z neničelnim nosilcem na Ω' , na Ω' tudi linearno neodvisni.

3. **Pozitivnost:**

$$N_{i,j}^{p,q}(u, v) > 0 \text{ za vse } (u, v) \in \text{supp}(N_{i,j}^{p,q})$$

4. **Particija enote:**

$$\sum_{i=0}^n \sum_{j=0}^m N_{i,j}^{p,q}(u, v) = 1 \text{ za vse } (u, v) \in [u_p, u_{n+1}] \times [v_q, v_{m+1}].$$

5. Naj bosta $\mu(u_i, U) = k_1$ in $\mu(v_i, V) = k_2$. Potem je zlepek v točki (u_i, v_i) $\mathcal{C}^{p-\max\{k_1, k_2\}}$ zvezen.

3 ŠKATLASTI ZLEPKI

Škatlasti zlepki so posplošitev B-zlepkov. Dobro se obnesejo pri reševanju Parcialnih diferencialnih enačb, saj lahko z njimi bolje opišemo geometrijo problema. Videli bomo, da so prej definirani B-zlepki le poseben primer škatlastih zlepkov.

3.1 INDUKTIVNA DEFINICIJA IN LASTNOSTI

Naj bo $M \in \mathbb{Z}^{d \times n}$. Matriko M zapišemo kot $M := [\mathbf{v}_1, \mathbf{v}_2 \cdots \mathbf{v}_n]$, kjer so $\mathbf{v}_i \in \mathbb{Z}^d \setminus \mathbf{0}$, $i = 1, 2, \dots, n$. Predpostavimo, da vektorji matrike M napenjajo prostor \mathbb{R}^d oz. $\text{rang}(M) = d$. Z $M[0, 1)^d$ označimo vse točke v \mathbb{R}^d dobljene z množenjem matrike M z vsemi točkami iz $[0, 1)^{d1}$. Oznaka $X \subset M$ označuje, da je matrika X pridobljena z odstranitvijo nekaterih stolpcev matrike M in oznaka $M \cup X := [M \ X]$ označuje novo matriko, sestavljeno iz vseh stolpcev matrik M in X . Označimo še z $M \setminus X$ novo matriko, dobljeno na način, da iz matrike M po enkrat odstranimo vsak stolpec matrike X in z $\#M$ število stolpcev matrike M .

Definicija 3.1. Naj bo $M \in \mathbb{Z}^{d \times n}$. Potem lahko definiramo mrežo

$$\Delta(M) := \mathbb{H}(M) + \mathbb{Z}^d,$$

kjer je $\mathbb{H}(M)$ množica vseh hiperravnin, ki jih napenjajo vektorji (stolpci) matrike M , torej

$$\mathbb{H}(M) = \{ \langle X \rangle : X \subseteq M \text{ in } \text{rank}(X) = d - 1 \},$$

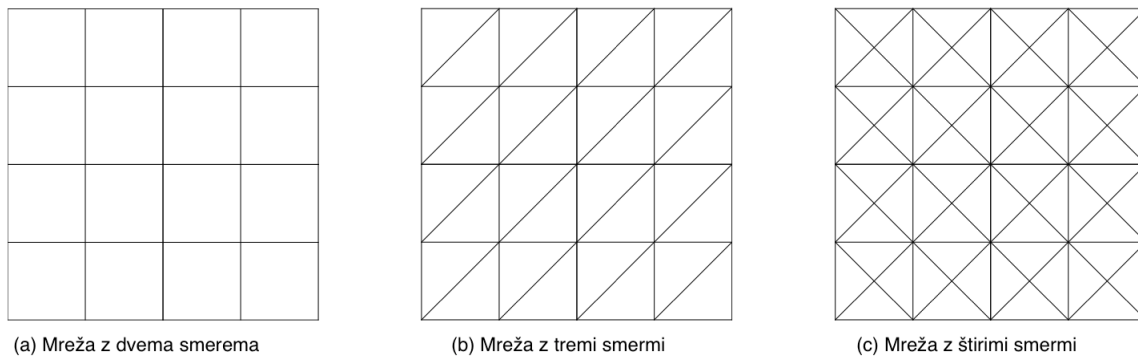
kjer $\langle X \rangle$ označuje realno linearno ogrinjačo, napeto z vektorji matrike M .

Primer 3.2. Naj bodo

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{v}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{v}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Če matriko M sestavljata le vektorja \mathbf{v}_1 in \mathbf{v}_2 , ima mreža $\Delta(M)$ le dve smeri (tenzorski produkt). Če M vsebuje tudi kakšen \mathbf{v}_3 , potem ima $\Delta(M)$ tri smeri (tenzorski produkt plus ena diagonala), in če M vsebuje tudi kakšen \mathbf{v}_4 , ima $\Delta(M)$ štiri smeri (tenzorski produkt plus dve diagonalni).

¹Tukaj predpostavimo, da je $M \in \mathbb{R}^{d \times d}$.



Slika 11: Trije primeri različnih mrež.

Opomba 3.3. Za mrežo, katero napenjaajo le vektorji $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, pravimo, da je *tipa-I*, za mrežo, katero napenjaajo le vektorji $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ pa pravimo, da je *tipa-II*.

Definicija 3.4. Naj bo $M \in \mathbb{Z}^{d \times n}$ in naj bo $M_d \subseteq M$, kjer $\det(M_d) \neq 0$. Potem lahko $n - ti$ škatlasti zlepek nad matriko M definiramo rekurzivno kot

$$B_M(\mathbf{x}) := \int_0^1 B_{M_n \setminus \mathbf{v}_n}(\mathbf{x} - t \cdot \mathbf{v}_n) dt$$

in

$$B_{M_d}(\mathbf{x}) := \begin{cases} \frac{1}{|\det M_d|}, & \mathbf{x} \in M_d[0, 1) \\ 0, & \text{sicer} \end{cases}.$$

Iz definicije 3.4 vidimo, da matrika M oz. smerni vektorji matrike M popolnoma določajo bazni škatlasti zlepek.

Primer 3.5. Naj bo $M = \begin{bmatrix} 1 & 1 \end{bmatrix}$. Izračunajmo škatlasti zlepek $B_M(x)$. Imamo torej škatlasti zlepek ene spremenljivke. Tukaj ni mogoče govoriti o več različnih smereh saj smo na premici. Glede na definicijo 3.4 bomo dobili odsekoma linearni polinom. Zapišimo

$$B_M(x) = \int_0^1 B_{\begin{bmatrix} 1 \end{bmatrix}}(x - t) dt, \quad (3.1)$$

kjer je

$$B_{\begin{bmatrix} 1 \end{bmatrix}}(x - t) = \begin{cases} 1, & x - t \in [0, 1) \\ 0, & \text{sicer} \end{cases}.$$

Zgornji izraz lahko ekvivalentno zapišemo kot

$$B_{\begin{bmatrix} 1 \end{bmatrix}}(x - t) = \begin{cases} 1, & t \in [x - 1, x) \\ 0, & \text{sicer} \end{cases}. \quad (3.2)$$

Analizirajmo sedaj integral (3.1). Da bo ta neničelen, mora za interval iz izraza (3.2) veljati

$$[x - 1, x) \subseteq [0, 1).$$

Ločimo dva primera, ko se to zgodi.

1. $0 \leq x < 1$:

Za vse takšne vrednosti x je $x - 1 < 0$, torej je funkcija iz izraza (3.2) neničelna na nekem podintervalu intervala $(-1, x]$, torej je v tem primeru integral iz (3.1) enak

$$B_M(x) = \int_0^1 B_{[1]}(x - t)dt = \int_0^x 1dt = x.$$

2. $1 \leq x < 2$:

V tem primeru pa je $x > 1$, torej je funkcija iz izraza (3.2) neničelna na nekem podintervalu intervala $(x - 1, 2]$ in s tem lahko integral (3.1) zapišemo kot

$$B_M(x) = \int_0^1 B_{[1]}(x - t)dt = \int_{x-1}^1 1dt = 2 - x.$$

V vseh ostalih primerih pa je integral (3.2) enak 0. Škatlasti zlepek lahko zapišemo bolj kompaktno kot

$$B_M(x) = \begin{cases} x, & x \in [0, 1) \\ 2 - x, & x \in [1, 2) \\ 0, & \text{sicer} \end{cases}.$$

Iz primera opazimo dve stvari. Škatlasti zlepek B_M se ujema z definicijo baznega B-zlepka $N_{0,1}$ iz primera 2.7. V nadaljevanju bomo videli, da so B-zlepki le posebni primeri škatlastih zlepkov. Druga stvar, ki jo opazimo, pa je, da je računanje škatlastih zlepkov s pomočjo definicije 3.4 zahtevno. Prejšnji primer je bil precej preprost, ampak smo vseeno potrebovali kar nekaj razmisleka. Koliko dela bi bilo potrebnega šele, če bi zlepku še zvišali stopnjo ali pa, če bi računali zlepke v višjih dimenzijah. Navedimo nekaj osnovnih lastnosti škatlastih zlepkov.

- Škatlasti zlepek B_M , definiran v definiciji 3.4, ni odvisen od vrstnega reda vektorjev matrike M .

- **Lokalni nosilec in pozitivnost.** Velja, da je $\text{supp}B_M = M[0, 1]^n$ in velja, da je $B_M(x) > 0$ za vsak $x \in M[0, 1]^n$. Nosilec škatlastega zleпка lahko zapišemo tudi kot

$$\text{supp}(B_M) = \sum_{j=1}^n t_j \mathbf{v}_j, \quad 0 \leq t_j < 1, \quad \mathbf{v}_j \subset M.$$

- **Simetričnost.** Zlepek B_M je simetričen glede na svojo sredinsko točko, določeno kot

$$\mu_M := \frac{1}{2} \sum_{i=1}^n \mathbf{v}_i.$$

- **Gladkost.** Zlepek B_M je $(\rho - 2)$ -krat zvezno odvedljiv, kjer je

$$\rho = \rho(M) = \min\{\#X : \mathcal{L}in(M \setminus X) \neq \mathbb{R}^d\}.$$

Dokaz. Bralec najde dokaz v [5]. □

- B_M je odsekoma polinomska funkcija d spremenljivk, stopnje $n - d$, definirana na mreži $\Delta(M)$.
- **Rekurzivna zveza.** Naj bodo škatlasti zleпки $B_{M \setminus \mathbf{v}_i}$, $i = 1, 2, \dots, n$, zvezni v točki $\mathbf{x} = \sum_{i=1}^k \alpha_k \mathbf{v}_k$. Potem velja

$$B_M(\mathbf{x}) = \frac{1}{n-d} \sum_{i=1}^n \alpha_k B_{M \setminus \mathbf{v}_k}(\mathbf{x}) + (1 - \alpha_k) B_{M \setminus \mathbf{v}_k}(\mathbf{x} - \mathbf{v}_k).$$

- Škatlasti zleпки so normirani, t.d. za poljubno matriko $M \in \mathbb{Z}^{d \times n}$ in pripadajoč škatlasti zlepek velja

$$\int_{\mathbb{R}^d} B_M(\mathbf{x}) d\mathbf{x} = 1.$$

Dokaz. Dokažimo z indukcijo za poljubni škatlasti zlepek, določen z matriko $M_n \in \mathbb{Z}^{d \times n}$. Za bazo indukcije vzamemo primer, ko imamo matriko M_k in $k = d$. Potem trditev očitno velja. Matrika M_k je kvadratna in njeni stolpci po definiciji napenjajo prostor \mathbb{R}^d . Predpostavimo lahko, da so stolpci enotski vektorji in potem je

$$B_{M_d}(\mathbf{x}) = \frac{1}{|\det M_d|} = 1, \quad \mathbf{x} \in M_d[0, 1),$$

in očitno tudi

$$\int_{\mathbb{R}^d} B_{M_d}(\mathbf{x}) d\mathbf{x} = 1.$$

Predpostavimo, da trditev velja za vse $k \in \mathbb{N}$ do vključno $k = n - 1$. Postavimo $k = n$ in integrirajmo

$$\int_{\mathbb{R}^d} B_{M_n}(\mathbf{x}) d\mathbf{x}.$$

Po definiciji je ta integral enak

$$\int_{\mathbb{R}^d} \left(\int_0^1 B_{M_n \setminus \mathbf{v}_n}(\mathbf{x} - t\mathbf{v}_n) dt \right) d\mathbf{x}.$$

Vrstni red integracije lahko zamenjamo in označimo $M_n \setminus \mathbf{v}_n = M_{n-1}$ in dobimo

$$\int_0^1 \left(\int_{\mathbb{R}^d} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_n) d\mathbf{x} \right) dt.$$

Po indukciji predpostavki pa je integral po \mathbf{x} enak 1 in s tem je tudi

$$\int_0^1 1 dt = 1. \quad \square$$

3.2 BERNSTEIN-BÉZIEREVA OBLIKA

Kot smo videli v primeru 3.5, je računanje škatlastih zlepkov po definiciji 3.4 zahtevno. Zaradi tega vpeljemo t.i. Bernstein-Bézierevo obliko, s katero lahko škatlaste zlepke računamo hitro in učinkovito. Od tukaj naprej se omejimo zgolj na mreže tipa- I . Za začetek potrebujemo nekaj dodatnih definicij. Označimo

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Ker smo se omejili na mreže tipa- I , definiramo

$$M_n = \left[\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n \right], \quad \mathbf{v}_i \in \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}.$$

Brez škode za splošnost, lahko vedno predpostavimo, da je $\mathbf{v}_1 = \mathbf{e}_1$, $\mathbf{v}_2 = \mathbf{e}_2$ in $\mathbf{v}_3 = \mathbf{e}_3$ in označimo

$$M_{n-1} = M_n \setminus \mathbf{v}_n.$$

Recimo, da se v matriki M_n vektor \mathbf{e}_1 pojavi i -krat, \mathbf{e}_2 j -krat in \mathbf{e}_3 k -krat, potem lahko zlepek $B_M(\mathbf{x})$ zapišemo kot

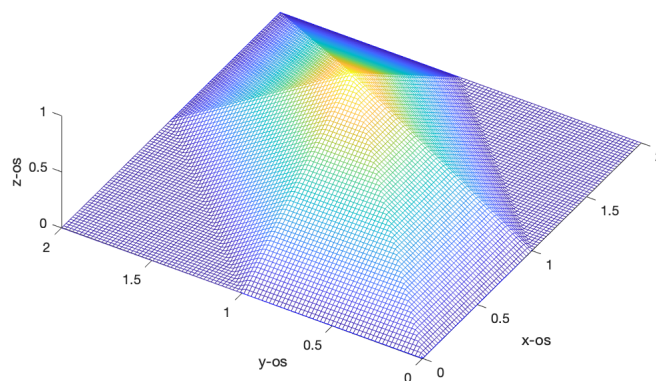
$$B_{ijk}(\mathbf{x}).$$

Definicija 3.6. Naj bo Δ mreža tipa- I , ki jo napenja matrika M_n . Prostor $\mathcal{S}_d^p(\Delta)$ vsebuje vse polinome dveh spremenljivk stopnje d , ki so na mreži Δ zveznosti ρ .

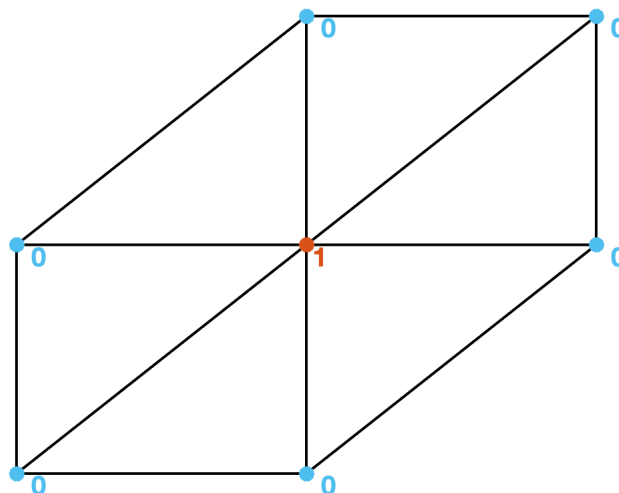
Za začetni primer vzamemo škatlasti zlepek B_{111} . To je torej zlepek, določen z matriko

$$M_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Ker so škatlasti zleпки normirani, določimo, da ima ta zlepek v točki $(1, 1)$ vrednost 1, na robovih svojega nosilca pa vrednost 0. Sliko tega zleпка in njegov nosilec prikazujeta sliki 12 in 13.



Slika 12: Škatlasti zlepek B_{111} .



Slika 13: Nosilec zleпка B_{111} in pripadajoči Bernsteinovi koeficienti.

Na sliki 13 vidimo tudi Bernsteinove koeficiente. Očitno lastnosti, definirane v prejšnjem poglavju, veljajo tudi za ta zlepek. Definicijo škatlastih zlepkov zapišimo na malo drugačen način, tako da za bazo rekurzije izhajamo iz zleпка B_{111} .

Definicija 3.7. Naj bo $M_n = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n]$, $n > 3$, matrika, ki opisuje mrežo tipa- I in naj bo $M_i = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_i]$ za $i = 3, \dots, n$. Potem lahko za vse $4 \leq i \leq n$ definiramo škatlasti zlepek B_{M_i} kot

$$B_{M_i}(\mathbf{x}) := \int_0^1 B_{M_{i-1}}(\mathbf{x} - t\mathbf{v}_i) dt,$$

pri čemer je

$$B_{M_3}(\mathbf{x}) = B_{111}(\mathbf{x}).$$

V nadaljevanju bomo potrebovali definicijo smernega odvoda zlepka. Naj bo $\mathbf{u} \in \mathbb{R}^2$ in $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. Potem smerni odvod funkcije f po vektorju \mathbf{u} označimo kot $D_{\mathbf{u}}f$. Za funkcijo f definirajmo še prednjo in zadnjo diferenco kot

$$\begin{aligned} \Delta_{\mathbf{u}}f(x) &= f(\mathbf{x} + \mathbf{u}) - f(\mathbf{x}) \\ \nabla_{\mathbf{u}}f(x) &= f(\mathbf{x}) - f(\mathbf{x} - \mathbf{u}). \end{aligned}$$

Naslednja lema nam podaja formulo za izračun smernega odvoda škatlastega zlepka.

Lema 3.8. Naj bo M_n matrika, ki določa mrežo tipa- I , in $n > 3$. Potem za vse $4 \leq j \leq n$ velja

$$D_{\mathbf{v}_j}B_{M_n}(\mathbf{x}) = \nabla_{\mathbf{v}_j}B_{M_n \setminus \mathbf{v}_j}(\mathbf{x}).$$

Dokaz. Naj bo $M_{n-1} := M_{M_n \setminus \mathbf{v}_j}$. Iz definicije 3.7 sledi

$$D_{\mathbf{v}_j}B_{M_n}(\mathbf{x}) = D_{\mathbf{v}_j} \int_0^1 B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) dt. \quad (3.3)$$

Trdimo, da je

$$D_{\mathbf{v}_j}B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) = -\frac{\partial}{\partial t}B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \quad (3.4)$$

Spomnimo se, da je $\mathbf{x} = [x_1, x_2]^T$ in $\mathbf{v}_j = [v_1, v_j]^T$. Potem lahko desno stran enačbe (3.4) razpišemo kot

$$-\frac{\partial}{\partial t}B_{M_{n-1}}(x_1 - tv_1, x_2 - tv_2).$$

Po verižnem pravilu dobimo, da je to enako

$$\begin{aligned} & - \left(\frac{\partial}{\partial x_1}B_{M_{n-1}}(x_1 - tv_1, x_2 - tv_2) \cdot \frac{\partial}{\partial t}(x_1 - tv_1) + \right. \\ & \left. \frac{\partial}{\partial x_2}B_{M_{n-1}}(x_1 - tv_1, x_2 - tv_2) \cdot \frac{\partial}{\partial t}(x_2 - tv_2) \right), \end{aligned}$$

kar pa lahko bolj kompaktno zapišemo kot

$$- \left(\frac{\partial}{\partial x_1}B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \cdot (-v_1) + \frac{\partial}{\partial x_2}B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \cdot (-v_2) \right). \quad (3.5)$$

Ker smo se omejili na mreže tipa- I , ločimo samo tri različne primere.

1. $\mathbf{v}_j = [1, 0]^T$. Potem lahko enačbo (3.5) zapišemo kot

$$-\left(\frac{\partial}{\partial x_1} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \cdot (-1)\right) = \frac{\partial}{\partial x_1} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) = D_{\mathbf{v}_j} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j).$$

2. $\mathbf{v}_j = [0, 1]^T$. Potem lahko enačbo (3.5) zapišemo kot

$$-\left(\frac{\partial}{\partial x_2} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \cdot (-1)\right) = \frac{\partial}{\partial x_2} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) = D_{\mathbf{v}_j} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j).$$

3. $\mathbf{v}_j = [1, 1]^T$. Potem lahko enačbo (3.5) zapišemo kot

$$\begin{aligned} & -\left(\frac{\partial}{\partial x_1} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \cdot (-1) + \frac{\partial}{\partial x_2} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \cdot (-1)\right) = \\ & \frac{\partial}{\partial x_1} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) + \frac{\partial}{\partial x_2} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) = D_{\mathbf{v}_j} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j). \end{aligned}$$

V vseh treh primerih smo videli, da enačba (3.4) drži. Nadaljujemo in enačbo (3.3) zapišemo kot

$$\begin{aligned} & -\int_0^1 \frac{\partial}{\partial t} B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) dt = \\ & -B_{M_{n-1}}(\mathbf{x} - t\mathbf{v}_j) \Big|_0^1 = \\ & -B_{M_{n-1}}(\mathbf{x} - \mathbf{v}_j) + B_{M_{n-1}}(\mathbf{x}) = \end{aligned}$$

$$\nabla_{\mathbf{v}_j} B_{M_n \setminus \mathbf{v}_j}(\mathbf{x}). \quad \square$$

Sedaj imamo podana vsa orodja za izračun Bernsteinovih koeficientov za poljuben $B_{ijk}(\mathbf{x})$. Koeficiente zleпка B_{111} imamo podane na sliki 13. Ker za poljubno M_n , ki določa zlepek B_{ijk} , velja, da je sestavljena iz vektorjev \mathbf{v}_i , za katere velja, da za vsak $\mathbf{v}_i \in \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, $i = 1, 2, \dots, n$, opazimo, da je za vsak \mathbf{v}_n smerni odvod $D_{\mathbf{v}_n} B_{ijk}(\mathbf{x})$, omejen na nek trikotnik v mreži, smerni odvod po eni stranici tega trikotnika. Naj bo

$$p_{n-2}(u, v, w) := \sum_{i+j+k=n-2} c_{i,j,k} \cdot \beta_{i,j,k}^{n-2}(u, v, w),$$

omejitev zleпка $B_{ijk}(\mathbf{x})$ v baricentričnih koordinatah glede na trikotnik T_0 s koordinatami $(0, 0)$, $(1, 0)$ in $(1, 1)$, kjer so $c_{i,j,k} \in \mathbb{R}^3$ kontrolne točke v baricentričnih koordinatah in $\beta_{i,j,k}^{n-2}$ Bernsteinovi bazni polinomi, definirani z definicijo 3.9 glede na trikotnik T_0 .

Definicija 3.9. Naj bo $n \in \mathbb{N}$ in $i, j, k \in \mathbb{N}^0$, pri čemer velja $i + j + k = n$. Bernsteinov bazni polinom stopnje skupne stopnje n je v baricentričnih koordinatah definiran kot

$$\beta_{i,j,k}^n(u, v, w) := \frac{n!}{i!j!k!} u^i v^j w^k, \quad u + v + w = 1.$$

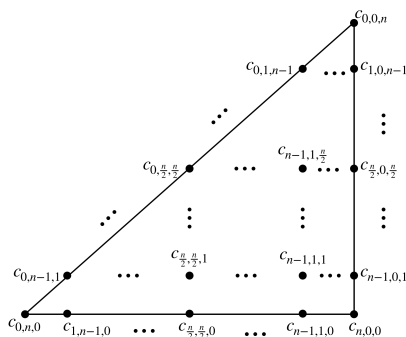
Opomba 3.10. Bernsteinov polinom iz definicije 3.9 je podan v baricentričnih koordinatah, definiranih na nekem trikotniku v ravnini. Od tod sledi, da je $u + v + w = 1$ in s tem lahko zapišemo polinom tudi kot $\beta_{ijk}^n : \mathbb{R}^2 \rightarrow \mathbb{R}$ s predpisom

$$\beta_{i,j,k}^n(u, v) := \frac{n!}{i!j!k!} u^i v^j (1 - u - v)^k.$$

Opomba 3.11. Iz definicije kontrolnih točk in Bernsteinovih polinomov ugotovimo, da imamo v baricentrični reprezentaciji na vsakem trikotniku nosilca zlepka $B_{ijk}(\mathbf{x})$, $i + j + k = n$, natanko $\binom{n}{2}$ kontrolnih točk, katerih indeksi so v vseh smereh ekvidistantno razporejeni. Primer prikazuje slika 14. Pri mrežah tipa- I so možni le takšni trikotniki, kot so prikazani na sliki 14 in trikotniki, ki so zrcalna slika preko hipotenuze takšnega trikotnika. Opozoriti velja, da je n na tej sliki mišljen kot $n - 2$, kjer je n dobljen iz števila vektorjev, ki določajo škatlasti zlepek. Dvournna oznaka je uporabljena zaradi preglednosti. Vsaka kontrolna točka $c_{i,j,k}$ je torej oblike

$$c_{i,j,k} = \begin{bmatrix} x_{ijk} \\ y_{ijk} \\ b_{ijk} \end{bmatrix},$$

kjer sta x_{ijk} in y_{ijk} kartezične koordinate kontrolne točke v ravnini, zadnja komponenta b_{ijk} pa je Bernsteinov koeficient.



Slika 14: Baricentrična reprezentacija Kontrolnih točk Béziereve krpe na enem trikotniku na mreži tipa- I zlepka $B_n(\mathbf{x})$.

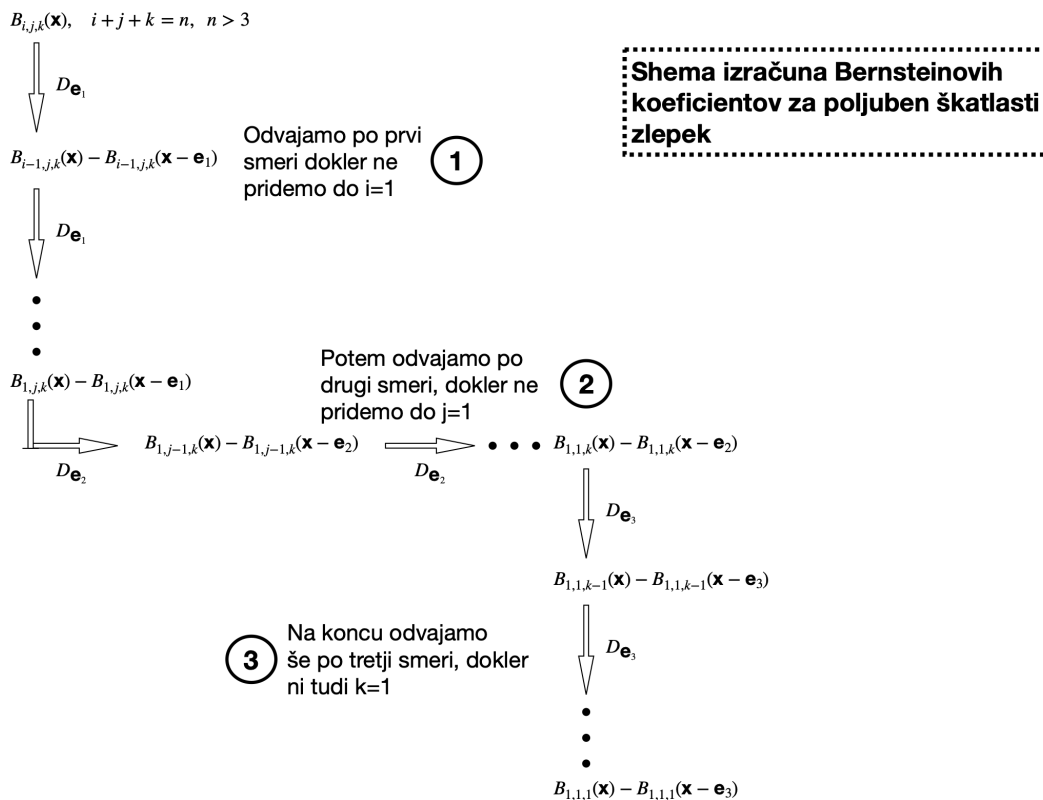
Naj bo p_{n-2} omejitvev zlepka $B_{i,j,k}(\mathbf{x})$ na trikotnik T_0 . Potem lahko izračunamo naslednje smerne odvode:

$$\begin{aligned}
 D_{\mathbf{e}_1} p_{n-2}(u, v) &= (n-2) \sum_{i+j+k=n-3} (c_{i,j+1,k} - c_{i+1,j,k}) \beta_{i,j,k}^{n-3}(u, v), \\
 D_{\mathbf{e}_2} p_{n-2}(u, v) &= (n-2) \sum_{i+j+k=n-3} (c_{i,j,k+1} - c_{i,j+1,k}) \beta_{i,j,k}^{n-3}(u, v), \\
 D_{\mathbf{e}_3} p_{n-2}(u, v) &= (n-2) \sum_{i+j+k=n-3} (c_{i,j,k+1} - c_{i+1,j,k}) \beta_{i,j,k}^{n-3}(u, v). \tag{3.6}
 \end{aligned}$$

Po lemi 3.8 vemo, da velja

$$D_{\mathbf{v}_n} B_{M_n}(\mathbf{x}) = \nabla_{\mathbf{v}_n} B_{M_n \setminus \mathbf{v}_n}(\mathbf{x}).$$

Predpostavimo, da za zlepek $B_{M_n \setminus \mathbf{v}_n}(\mathbf{x})$ poznamo vse Bernsteinove koeficiente. Potem poznamo tudi vse Bernsteinove koeficiente zleпка $\nabla_{\mathbf{v}_n} B_{M_n \setminus \mathbf{v}_n}(\mathbf{x})$, saj gre le za vsoto ustreznih koeficientov. Zlepku $B_{M_n \setminus \mathbf{v}_n}(\mathbf{x})$ prištejemo koeficiente tega istega zleпка, predstavljenega za vektor \mathbf{v}_n . Seštejemo torej koeficiente na istoležnih mestih. Koeficiente zleпка B_{111} poznamo (slika 13). Sledi, da lahko za poljuben B_{ijk} , $i + j + k = n$, izračunamo Bernsteinove koeficiente z reševanjem diferenčnih enačb prvega reda z začetnim pogojem, da so Bernsteinovi koeficienti na robu nosilca enaki 0. Slika 15 prikazuje grobo shemo določanja koeficientov. Potek iskanja pa prikazuje primer 3.12.

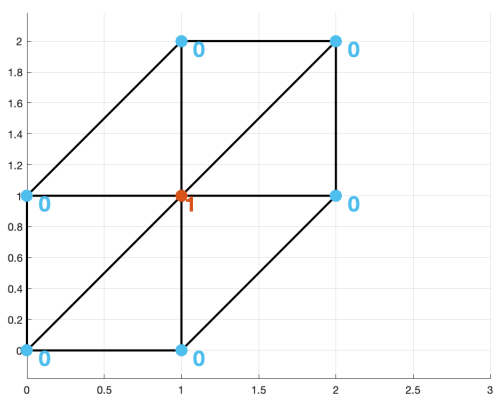


Slika 15: Shema določanja Bernsteinovih koeficientov za poljuben škatlasti zlepek B_{ijk} .

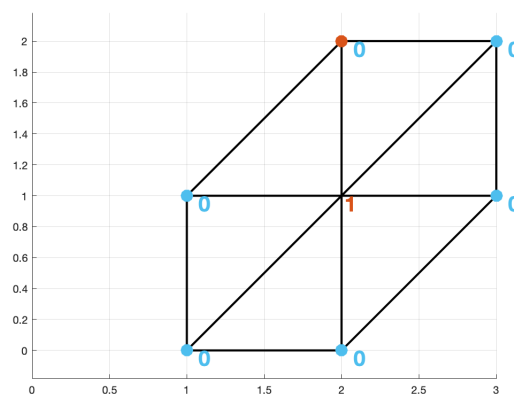
Primer 3.12. Določimo Bernsteinove koeficiente zlepka $B_{211}(\mathbf{x})$. To je torej zlepek podan z matriko $M_4 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$. Če ta zlepek odvajamo po \mathbf{e}_1 dobimo

$$D_{\mathbf{e}_1} B_{211}(\mathbf{x}) = B_{111}(\mathbf{x}) - B_{111}(\mathbf{x} + \mathbf{e}_1).$$

Najprej izračunajmo koeficiente odvoda $D_{\mathbf{e}_1} B_{211}(\mathbf{x})$. Če predpostavimo, da je oglišče skrajno levo-spodaj v nosilcu zlepka B_{111} postavljeno v izhodišče kartezične ravnine, dobimo zlepek na sliki 16. Vidimo, da je $B_{111}(\mathbf{x} + \mathbf{e}_1)$ enak kot zlepek $B_{111}(\mathbf{x})$ premaknjen za vektor \mathbf{e}_1 .



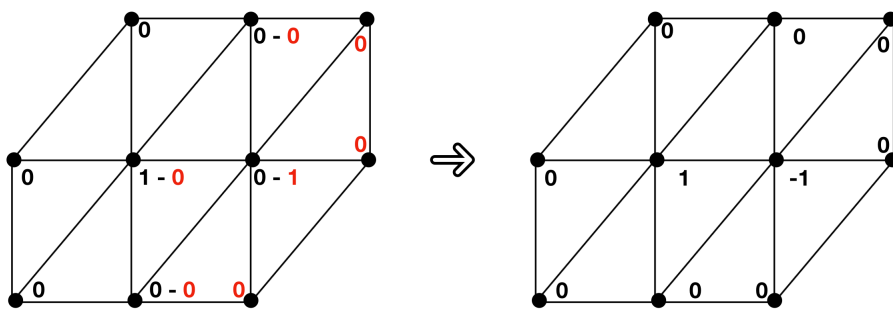
(a) Nosilec zlepka $B_{111}(\mathbf{x})$.



(b) Nosilec zlepka $B_{111}(\mathbf{x} + \mathbf{e}_1)$.

Slika 16: Nosilca zlepkov $B_{111}(\mathbf{x})$ in $B_{111}(\mathbf{x} + \mathbf{e}_1)$ z Bernsteinovimi koeficienti.

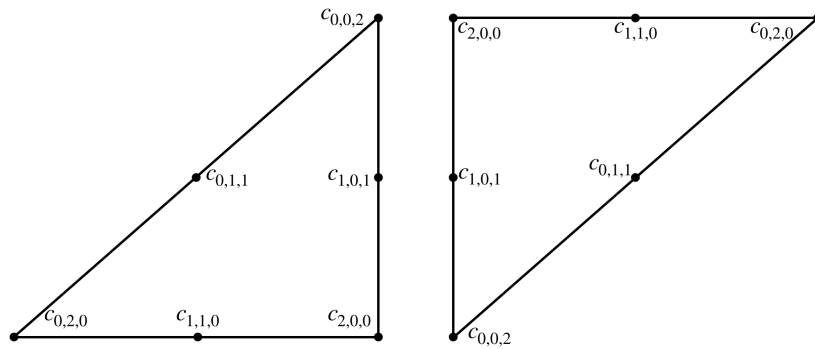
Zlepku $B_{111}(\mathbf{x})$ odštejemo istoležne komponente premaknjenega zlepka. Komponente, ki se ne prekrivajo, pustimo. Nosilec zlepka $D_{\mathbf{e}_1} B_{211}(\mathbf{x})$ s pripadajočimi Bernsteinovimi koeficienti vidimo na sliki 17, desno.



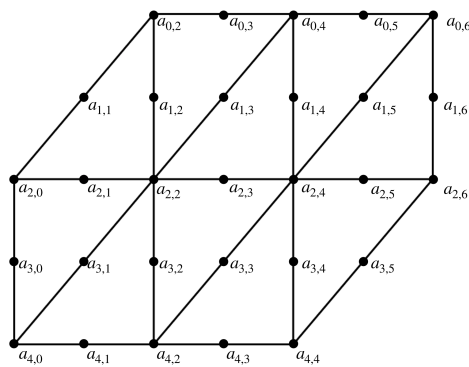
Slika 17: Računanje Bernsteinovih koeficientov zlepka $D_{\mathbf{e}_1} B_{211}(\mathbf{x})$.

Po opombi 3.11 ima vsak trikotnik v nosilcu zlepka B_{211} eno od dveh oblik, prikazanih na sliki 18.

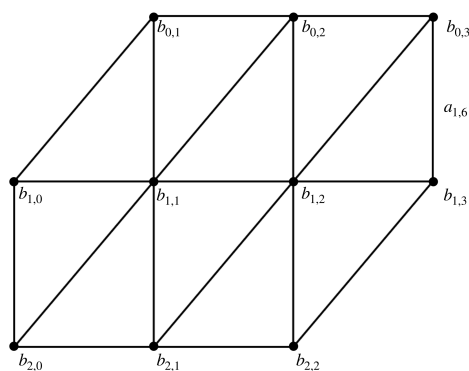
Celoten nosilec zlepka $B_{211}(\mathbf{x})$ prikazuje slika 19. Bernsteinove koeficiente označimo z $a_{i,j}$, z $b_{i,j}$ pa označimo vse koeficiente zlepka $D_{\mathbf{e}_1} B_{211}(\mathbf{x})$, kot je prikazano na sliki 20.



Slika 18: Dva primera trikotnikov iz nosilca zlepka $B_{211}(\mathbf{x})$.

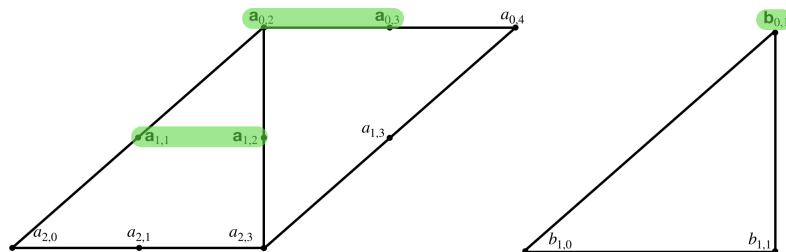


Slika 19: Nosilec zlepka $B_{211}(\mathbf{x})$ z Bernsteinovimi koeficienti.



Slika 20: Nosilec zlepka $D_{e_1} B_{211}(\mathbf{x})$ z Bernsteinovimi koeficienti.

Glede na enačbe (3.6) določimo sistem enačb (3.7). Zgled enačbe pri enem koeficientu prikazuje slika 21.



Slika 21: Zgled določanja enačbe v sistemu (3.7).

$$\begin{aligned}
 b_{01} &= a_{12} - a_{11} = a_{03} - a_{02} \\
 b_{02} &= a_{04} - a_{03} = a_{14} - a_{13} \\
 b_{03} &= a_{06} - a_{05} = a_{16} - a_{15} \\
 b_{10} &= a_{21} - a_{20} \\
 b_{11} &= a_{22} - a_{21} = a_{32} - a_{31} = a_{23} - a_{22} = a_{13} - a_{12} \\
 b_{12} &= a_{24} - a_{23} = a_{15} - a_{14} = a_{25} - a_{24} = a_{34} - a_{33} \\
 b_{13} &= a_{26} - a_{25} \\
 b_{20} &= a_{31} - a_{30} = a_{41} - a_{40} \\
 b_{21} &= a_{42} - a_{41} = a_{33} - a_{32} \\
 b_{22} &= a_{44} - a_{43} = a_{35} - a_{34}.
 \end{aligned} \tag{3.7}$$

Z upoštevanjem vrednosti koeficientov b_{ij} na sliki 20 in upoštevanjem, da mora biti vsak škatlasti zlepek enak 0 na robu svojega nosilca, dobimo, da so vsi koeficienti na robu nosilca $B_{211}(\mathbf{x})$ enaki 0, torej

$$\begin{aligned}
 a_{02} &= a_{03} = a_{04} = a_{05} = a_{06} = 0 \\
 a_{11} &= a_{16} = 0 \\
 a_{20} &= a_{26} = 0 \\
 a_{30} &= a_{35} = 0 \\
 a_{40} &= a_{41} = a_{42} = a_{43} = a_{44} = 0.
 \end{aligned}$$

Sistem enačb (3.7) zapišemo kot

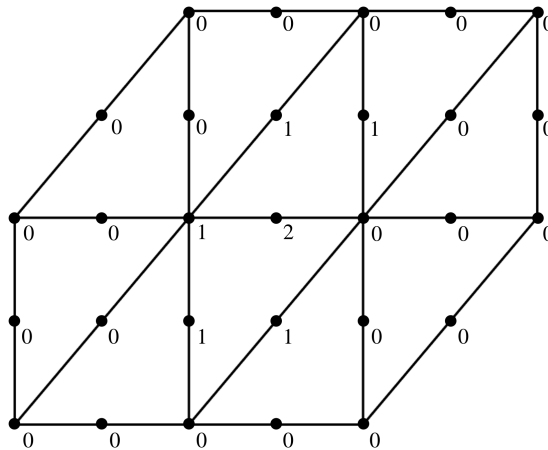
$$\begin{aligned}
b_{01} &= a_{12} = 0 \Rightarrow a_{12} = 0 \\
b_{02} &= 0 = a_{14} - a_{13} \Rightarrow a_{14} = a_{13} \\
b_{03} &= 0 = -a_{15} \Rightarrow a_{15} = 0 \\
b_{10} &= a_{21} \Rightarrow a_{21} = 0 \\
b_{11} &= a_{22} = a_{32} - a_{31} = a_{23} - a_{22} = a_{13} - a_{12} \Rightarrow a_{22} = 1 \\
b_{12} &= a_{24} - a_{23} = -a_{14} = a_{25} - a_{24} = a_{34} - a_{33} \Rightarrow a_{14} = 1 \\
b_{13} &= -a_{25} \Rightarrow a_{25} = 0 \\
b_{20} &= a_{31} = 0 \Rightarrow a_{31} = 0 \\
b_{21} &= 0 = a_{33} - a_{32} \Rightarrow a_{33} = a_{32} \\
b_{22} &= 0 = -a_{34} \Rightarrow a_{34} = 0.
\end{aligned} \tag{3.8}$$

Ostali so nam le še koeficienti v vrstici pri b_{11} in b_{12} . Ker je $a_{14} = a_{13}$, $a_{32} = a_{33}$, dobimo

$$b_{11} = 1 = a_{32} = a_{23} - 1 = a_{13} \Rightarrow a_{23} = 2, a_{14} = 1$$

$$b_{12} = a_{24} - 2 = -1 = -a_{24} = -a_{33} \Rightarrow a_{32} = 1, a_{24} = 1.$$

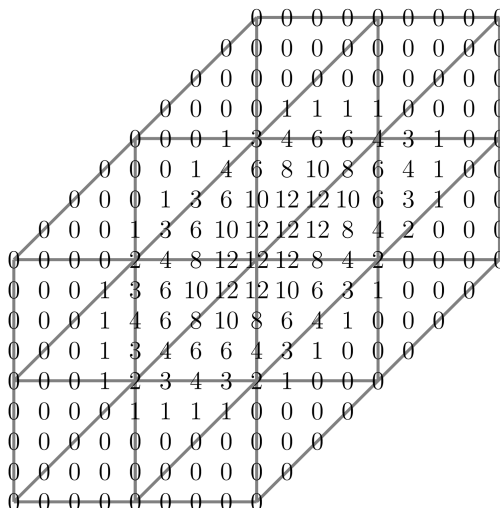
Nosilec z izračunanimi Bernsteinovimi koeficienti nam podaja slika 22.



Slika 22: Bernsteinovi koeficienti zlepka $2B_{211}(\mathbf{x})$.

Na analogen način kot v primeru 3.12 lahko izračunamo Bernsteinove koeficiente poljubnega zlepka. Slika 23 prikazuje Bernsteinove koeficiente zlepka $B_{222}(\mathbf{x})$, ki je lokalno polinom dveh spremenljivk, stopnje štiri. Zaradi preglednosti so koeficienti pomnoženi s 24.

Sedaj lahko zapišemo formalno definicijo Bernstein-Béziereve oblike škatlastega zlepka.



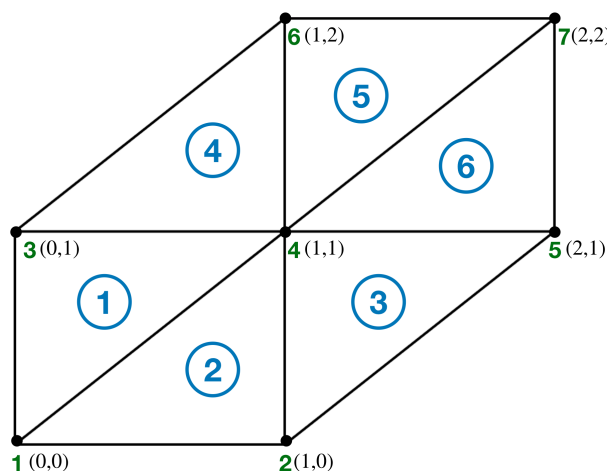
Slika 23: Bernsteinovi koeficienti zlepka $24B_{222}(\mathbf{x})$.

Definicija 3.13. Naj bo $B_{ijk}(\mathbf{x})$, $i + j + k = n$, škatlasti zlepek, definiran na mreži tipa- I , in naj $\text{supp}(B_{ijk})$ vsebuje N trikotnikov na mreži tipa- I . Potem imamo na vsakem trikotniku $\binom{n}{2}$ kontrolnih točk $c_{i,j,k}$ in Bernsteinovih polinomov $\beta_{i,j,k}^{n-2}(u, v)$ iz definicije 3.9. Množici polinomov

$$\mathcal{B} := \left\{ \sum_{i+j+k=n-2} c_{ijk}^{(\ell)} \cdot \beta_{ijk}^{n-2,(\ell)}(u, v), \quad \ell = 1, 2, \dots, N, \quad u, v \in [0, 1] \right\}$$

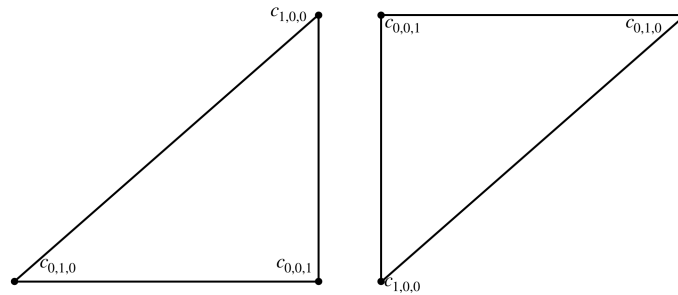
pravimo **Bernstein-Béziereva** oblika škatlastega zlepka $B_{ijk}(\mathbf{x})$.

Primer 3.14. Določimo Bernstein-Beziérove obliko škatlastega zlepka $B_{111}(\mathbf{x})$. Vse trikotnike in njihova oglišča v nosilcu indeksiramo kot prikazuje slika 24. Oglišče najbolj levo spodaj je postavljeno v izhodišče.



Slika 24: Indeksacija trikotnikov (modro) in oglišč (zeleno) nosilca zlepka $B_{111}(\mathbf{x})$. S črno so označene še kartezične koordinate oglišč.

Nad vsakim trikotnikom posebej določimo Bézierevo krpo. Indeksacijo kontrolnih točk v posameznem trikotniku prikazuje slika 25.



Slika 25: Indeksacija kontrolnih točk v enem trikotniku zlepk $B_{111}(\mathbf{x})$.

1.

$$\begin{aligned} p_1^{(1)}(u, v) &= c_{0,0,1}\beta_{0,0,1}^1(u, v) + c_{0,1,0}\beta_{0,1,0}^1(u, v) + c_{1,0,0}\beta_{1,0,0}^1(u, v) \\ &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot (1 - u - v) + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot v + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot u = \begin{bmatrix} v \\ 1 - u \\ v \end{bmatrix}. \end{aligned}$$

2.

$$\begin{aligned} p_1^{(2)}(u, v) &= c_{0,0,1}\beta_{0,0,1}^1(u, v) + c_{0,1,0}\beta_{0,1,0}^1(u, v) + c_{1,0,0}\beta_{1,0,0}^1(u, v) \\ &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot (1 - u - v) + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot v + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot u = \begin{bmatrix} 1 - v \\ u \\ u \end{bmatrix}. \end{aligned}$$

3.

$$\begin{aligned} p_1^{(3)}(u, v) &= c_{0,0,1}\beta_{0,0,1}^1(u, v) + c_{0,1,0}\beta_{0,1,0}^1(u, v) + c_{1,0,0}\beta_{1,0,0}^1(u, v) \\ &= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot (1 - u - v) + \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \cdot v + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot u = \begin{bmatrix} 1 - v \\ u + v \\ u \end{bmatrix}. \end{aligned}$$

4.

$$\begin{aligned} p_1^{(4)}(u, v) &= c_{0,0,1}\beta_{0,0,1}^1(u, v) + c_{0,1,0}\beta_{0,1,0}^1(u, v) + c_{1,0,0}\beta_{1,0,0}^1(u, v) \\ &= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot (1 - u - v) + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot v + \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \cdot u = \begin{bmatrix} 1 - v \\ 1 + u - v \\ 1 - u - v \end{bmatrix}. \end{aligned}$$

5.

$$\begin{aligned} p_1^{(5)}(u, v) &= c_{0,0,1}\beta_{0,0,1}^1(u, v) + c_{0,1,0}\beta_{0,1,0}^1(u, v) + c_{1,0,0}\beta_{1,0,0}^1(u, v) \\ &= \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \cdot (1 - u - v) + \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} \cdot v + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot u = \begin{bmatrix} 1 + v \\ 2 - u \\ u \end{bmatrix}. \end{aligned}$$

6.

$$\begin{aligned} p_1^{(6)}(u, v) &= c_{0,0,1}\beta_{0,0,1}^1(u, v) + c_{0,1,0}\beta_{0,1,0}^1(u, v) + c_{1,0,0}\beta_{1,0,0}^1(u, v) \\ &= \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \cdot (1 - u - v) + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot v + \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} \cdot u = \begin{bmatrix} 2 - v \\ 1 + u \\ v \end{bmatrix}. \end{aligned}$$

Bernstein-Béziereva oblika je potem enaka

$$\mathcal{B} = \left\{ \begin{bmatrix} v \\ 1 - u \\ v \end{bmatrix}, \begin{bmatrix} 1 - v \\ u \\ u \end{bmatrix}, \begin{bmatrix} 1 - v \\ u + v \\ u \end{bmatrix}, \begin{bmatrix} 1 - v \\ 1 + u - v \\ 1 - u - v \end{bmatrix}, \begin{bmatrix} 1 + v \\ 2 - u \\ u \end{bmatrix}, \begin{bmatrix} 2 - v \\ 1 + u \\ v \end{bmatrix} \right\}.$$

3.3 PROSTOR PREMIKOV

V praksi nas običajno zanimajo premiki (translacije) škatlastih zlepkov za nekatere vektorje iz \mathbb{Z}^n . Izkaže se, da so takšni zlepki primerni za aproksimacijo funkcij iz \mathbb{R}^n . Najprej zapišimo formalno definicijo prostora premikov.

Definicija 3.15. Naj bo $M \in \mathbb{Z}^{d \times n}$ in $B_M(\mathbf{x})$ škatlasti zlepek. Definiramo prostor

$$\mathcal{S} := \{B_M(\mathbf{x} - \mathbf{i})\}, \quad \mathbf{i} \in \mathbb{Z}^n.$$

Prostoru \mathcal{S} pravimo **prostor premikov** oz. prostor celoštevilskih premikov in vsebuje zlepke stopnje $n - 2$ v d spremenljivkah.

Opomba 3.16. Prostor premikov \mathcal{S} lahko tudi skaliramo na način, da ga definiramo kot

$$\mathcal{S} := \{B_M(h(\mathbf{x} - \mathbf{i}))\}, \quad \mathbf{i} \in \mathbb{Z}^n,$$

kjer je $h \in \mathbb{R}$. V praksi običajno vzamemo $h < 1$ in na ta način prostor premikov zgostimo, pri čemer ohranimo njegove lastnosti.

Da je \mathcal{S} primeren za aproksimacijo funkcij utemeljimo z naslednjima izrekoma. Prvi izrek govori o linearni neodvisnosti zlepkov prostora \mathcal{S} . Za nek $V \subseteq \mathbb{R}^n$ nas zanima linearna neodvisnost zlepkov $B_M(\mathbf{x} - \mathbf{v})$, kjer je $M \in \mathbb{Z}^{d \times n}$ in $\mathbf{v} \in V$.

Opomba 3.17. Včasih je bolj prikladno namesto $B_M(\mathbf{x})$, pisati $B_{\mathcal{M}}(\mathbf{x})$, kjer je \mathcal{M} končna podmnožica prostora \mathbb{Z}^d , za katero velja $|\mathcal{M}| = n$. Seveda še vedno predpostavimo, da je $M = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d]$, kjer je \mathbf{v}_j j -ti element množice \mathcal{M} . Na ta način bomo v izrekih lažje uporabili nekatere prijeme iz linearne algebre.

Trditvi, da so $B_M(\mathbf{x} - \mathbf{v})$ za $\mathbf{v} \in V \subseteq \mathbb{R}^d$ linearno neodvisni, je ekvivalentna trditev, da je linearna preslikava, ki preslika $a : V \rightarrow \mathbb{R}$ na način

$$a \mapsto \sum_{\mathbf{v}} a(\mathbf{v}) B_M(\mathbf{x} - \mathbf{v}), \quad \mathbf{v} \in V.$$

injektivna. Za poseben primer, ko je $\mathcal{M} \subseteq V = \mathbb{Z}^d$, lahko pokažemo, da so premiki zlepkov $B_M(\mathbf{x} - \mathbf{v})$ linearno odvisni razen če velja, da je $|\det Z| = 1$ za vse kvadratne matrike Z , katere stolpci so vektorji iz \mathcal{M} in predstavljajo bazo vektorskega prostora \mathbb{R}^d . Dokaz najdemo v [4]. Ta izrek nam torej da potreben pogoj za linearno neodvisnost. Z naslednjim izrekom, ki ga bomo tudi dokazali, pa pokažemo, da je pogoj, da so determinante vseh kvadratnih matrik, sestavljenih iz stolpcv množice \mathcal{M} enake ena, tudi zadosten.

Izrek 3.18. *Naj bo $\mathcal{M} \subseteq V = \mathbb{Z}^d$ in $\langle \mathcal{M} \rangle$ afina ogrinjača, ki napenja cel \mathbb{R}^d . Potem so škatlasti zlepki v množici*

$$\{B_M(\mathbf{x} - \mathbf{v}), \mathbf{v} \in V\}$$

linearno neodvisni, če in samo če velja

$$|\det Z| = 1$$

za vse kvadratne matrike Z sestavljene iz stolpcv množice \mathcal{M} , ki predstavljajo bazo prostora \mathbb{R}^d .

Dokaz. Dokaza se lotimo z indukcijo na $|\mathcal{M}|$. Za primer $|\mathcal{M}| = 1$ izrek očitno drži. V tem primeru imamo samo en vektor in samo en vektor lahko napenja le \mathbb{R} . Brez škode za splošnost predpostavimo, da je ta vektor kar 1 torej je tudi $\det Z = 1$ za vsak Z . Predpostavimo, da izrek velja za neko množico \mathcal{M}' t.d. $|\mathcal{M}'| < |\mathcal{M}|$. Zopet lahko brez škode za splošnost predpostavimo, da \mathcal{M} poleg poljubnih vektorjev vsebuje vse enotske vektorje \mathbf{e}_i , $i = 1, 2, \dots, d$, in da velja

$$\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_d\} \subset \mathcal{M}.$$

Namreč, po predpostavki v \mathcal{M} obstaja neka podmnožica vektorjev $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d\}$, ki tvori bazo prostora \mathbb{R}^d , in po predpostavki velja, da če jih zložimo v matriko Z , velja $|\det Z| = 1$. Naj bo \mathcal{A} linearna preslikava (v jeziku vektorskega prostora \mathbb{R}^d matrika A), ki preslika vsak vektor $\mathbf{v}_i \in \mathcal{M}$ v enotski vektor \mathbf{e}_i za $i = 1, 2, \dots, n$. Ker je $|\det Z| = 1$, velja, da \mathcal{A} preslika V v V . Še več, $|\det A| = 1$. Sledi, da je $\mathcal{A}\mathcal{M} \subseteq V$ in po trditvi v [4] velja reparametrizacija

$$B_M(\mathbf{x}) = |\det A| B_{AM}(\mathbf{x}) \circ \mathcal{A}(\mathbf{x}) = B_{AM}(\mathbf{x}) \circ \mathcal{A}(\mathbf{x}).$$

Od tod sledi, da so zlepki v $\{B_M(\mathbf{x} - \mathbf{v})\}$ linearno neodvisni, če so linearno neodvisni tudi v $\{B_{AM}(\mathbf{x} - \mathbf{v})\}$, in na ta način lahko vedno namesto s poljubnimi vektorji prostora \mathbb{R}^d delamo z enotskimi vektorji tega prostora. Ločimo dva primera.

1. Obstaja enotski vektor \mathbf{e}_k t.d $\langle \mathbf{e}_k \rangle \cap \langle \mathcal{M} \setminus \mathbf{e}_k \rangle = 0$. Brez škode za splošnost lahko predpostavimo, da je $k = d$ in $\langle \mathbf{e}_d \rangle \cap \langle \mathcal{M} \setminus \mathbf{e}_d \rangle = 0$. Potem je $\langle \mathcal{M} \setminus \mathbf{e}_d \rangle = \mathbb{R}^{d-1}$. Vsak vektor $\mathbf{v} \in V$ lahko zapišemo kot

$$\mathbf{v} = j\mathbf{e}_d + \mathbf{v}', \quad \text{kjer je } j \in \mathbb{Z} \text{ in } \mathbf{v}' \in \mathbb{Z}^{d-1}.$$

Ker je enotski vektor \mathbf{e}_d v kontekstu prostora \mathbb{R} enak 1, sledi $B_{\mathbf{e}_d}(x) = \mathbf{1}_{[0,1)}(x)$, $x \in \mathbb{R}$, in ker je $B_{\mathcal{M} \setminus \mathbf{e}_d} : \mathbb{R}^{d-1} \rightarrow \mathbb{R}$, je očitno

$$B_M(\mathbf{x}) = B_{\mathbf{e}_d}(x_d) \cdot B_{\mathcal{M} \setminus \mathbf{e}_d}(\mathbf{x}'),$$

kjer so $\mathbf{x} = [x_1, x_2, \dots, x_{d-1}, x_d]^T$, $\mathbf{x}' = [x_1, x_2, \dots, x_{d-1}]^T$ in $x_d \in \mathbb{R}$. Predpostavimo, da je

$$\sum_{\mathbf{v} \in V} a(\mathbf{v}) B_M(\mathbf{x} - \mathbf{v}) = 0$$

za neko preslikavo a . Potem je

$$\sum_{j \in \mathbb{Z}} \sum_{\mathbf{v}' \in \mathbb{Z}^{d-1}} a(j\mathbf{e}_d + \mathbf{v}') B_{\mathbf{e}_d}(x_d - j) B_{\mathcal{M} \setminus \mathbf{e}_d}(\mathbf{x}' - \mathbf{v}') = 0,$$

za vse $x_d \in \mathbb{R}$ in $\mathbf{x}' \in \mathbb{R}^{d-1}$. Izberimo nek $x_d = (i + \alpha)\mathbf{e}_d$ za nek $i \in \mathbb{Z}$ in $\alpha \in [0, 1)$. Izraz

$$B_{\mathbf{e}_d}(x_d - j\mathbf{e}_d)$$

je enak 0 za vse $i \in \mathbb{Z}$, razen za primer $i = j$. Upoštevamo še, da vektor \mathbf{e}_d v prostoru \mathbb{R} interpretiramo kot število 1 in tako je za $i = j$

$$B_{\mathbf{e}_d}(x_d - j\mathbf{e}_d) = B_{\mathbf{e}_d}(i + \alpha - i) = B_{\mathbf{e}_d}(\alpha).$$

Ker je $\alpha \in [0, 1)$, je po definiciji 3.7

$$B_{\mathbf{e}_d}(\alpha) = 1$$

in s tem tudi

$$\sum_{j \in \mathbb{Z}} B_{\mathbf{e}_d}(x_d - j\mathbf{e}_d) = 1.$$

Po indukcijski predpostavki so zleпки v $\{B_{\mathcal{M} \setminus \mathbf{e}_d}(\mathbf{x} - \mathbf{v}')\}$, $\mathbf{v}' \in \mathbb{Z}^{d-1}$, linearno neodvisni. Torej je $a(i\mathbf{e}_d + \mathbf{v}') = 0$ za vse $\mathbf{v}' \in \mathbb{Z}^{d-1}$ in $i \in \mathbb{Z}$. Sledi, da je $a = 0$ in s tem je izrek dokazan za primer 1.

2. V tem primeru pa predpostavimo, da je $\langle \mathcal{M} \setminus \mathbf{e}_k \rangle = \mathbb{R}^d$ za $k = 1, 2, \dots, d$. Naj bo

$$\sum_{\mathbf{v} \in V} a(\mathbf{v}) B_M(\mathbf{x} - \mathbf{v}) = 0$$

za neko preslikavo a . Potem po lemi 3.8 sledi, da je

$$\sum_{\mathbf{v} \in V} \nabla_{\mathbf{e}_k} a(\mathbf{v}) B_{\mathcal{M} \setminus \mathbf{e}_k}(\mathbf{x} - \mathbf{v}) = D_{\mathbf{e}_k} \left(\sum_{\mathbf{v} \in V} a(\mathbf{v}) B_M(\mathbf{x} - \mathbf{v}) \right) = 0$$

za $k = 1, 2, \dots, d$. Po indukcijski predpostavki so zlepki v $\{B_{\mathcal{M} \setminus \mathbf{e}_k}(\mathbf{x} - \mathbf{v}), \mathbf{v} \in V\}$ linearno neodvisni. Torej je

$$\nabla_{\mathbf{e}_k} a = 0, \quad k = 1, 2, \dots, d.$$

Sledi, da je

$$a(\mathbf{v}) = a(\mathbf{v} - \mathbf{e}_k),$$

in s tem

$$a(\mathbf{v}) = a(0), \quad \text{za vse } \mathbf{v} \in V.$$

Na koncu dobimo

$$0 = \sum_{\mathbf{v} \in V} a(\mathbf{v}) B_M(\mathbf{x} - \mathbf{v}) = a(0) \sum_{\mathbf{v} \in V} B_M(\mathbf{x} - \mathbf{v}),$$

pri čemer smo zaradi dejstva $a(\mathbf{v}) = a(0)$, za vse $\mathbf{v} \in V$, lahko $a(0)$ nesli pred vsoto.

Z upoštevanjem izreka 3.20 škatlasti zlepki tvorijo particijo enote in tako dobimo

$$a(0) = 0 \Rightarrow a(\mathbf{v}) = 0, \quad \text{za vsak } \mathbf{v} \in V.$$

S tem smo tudi v tem primeru dokazali linearno neodvisnost.

□

Iz izreka 3.18 sledi naslednja posledica.

Posledica 3.19. *Premiki škatlastih zlepkov, določeni na mreži tipa-II, niso linearno neodvisni.*

Dokaz. Trditev posledice preprosto dokažemo s protislovjem. Naj bo $B_{\mathcal{M}}$ škatlasti zlepek definiran na mreži tipa-III. Dokažimo posledico bolj splošno in predpostavimo zgolj, da je množica \mathcal{M} oblike

$$\mathcal{M} = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\},$$

kjer so vsi \mathbf{v}_1 do \mathbf{v}_4 iz \mathbb{R}^2 različni in paroma linearno neodvisni. Predpostavimo, da so premiki tega škatlastega zleпка linearno neodvisni. Potem po izreku 3.18 za vsak par $\mathbf{v}_i, \mathbf{v}_j, i = 1, 2, 3, 4, j = 1, 2, 3, 4, i \neq j$ velja

$$\det \begin{bmatrix} \mathbf{v}_i & \mathbf{v}_j \end{bmatrix} = \pm 1.$$

Zapišimo

$$\mathbf{v}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}.$$

Ker so premiki zlepkov linearno neodvisni, mora veljati

$$\det \begin{bmatrix} \mathbf{v}_i & \mathbf{v}_j \end{bmatrix} = x_1 y_2 - x_2 y_1 = \pm 1. \quad (3.9)$$

Ker \mathbf{v}_1 in \mathbf{v}_2 tvorita bazo prostora \mathbb{R}^2 , lahko preostala dva vektorja izrazimo kot linearno kombinacijo

$$\mathbf{v}_3 = \begin{bmatrix} \alpha x_1 + \beta x_2 \\ \alpha y_1 + \beta y_2 \end{bmatrix}, \quad \mathbf{v}_4 = \begin{bmatrix} \gamma x_1 + \delta x_2 \\ \gamma y_1 + \delta y_2 \end{bmatrix}, \quad \alpha, \beta, \gamma, \delta \in \mathbb{R}.$$

Potem mora veljati

$$\det \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_3 \end{bmatrix} = \pm 1.$$

Izračunana determinanta je enaka

$$\det \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_3 \end{bmatrix} = \alpha x_1 y_1 + \beta x_1 y_2 - \alpha x_1 y_1 - \beta x_2 y_1 = \beta(x_1 y_2 - x_2 y_1) = \pm 1.$$

Iz enačbe (3.9) sledi, da je

$$\beta = \pm 1.$$

Na enak način izračunamo tudi determinante $\det \begin{bmatrix} \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}$, $\det \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_4 \end{bmatrix}$, $\det \begin{bmatrix} \mathbf{v}_2 & \mathbf{v}_4 \end{bmatrix}$ in ugotovimo, da so tudi

$$\alpha = \pm 1, \quad \gamma = \pm 1, \quad \delta = \pm 1.$$

Za konec izračunajmo še

$$\begin{aligned} \det \begin{bmatrix} \mathbf{v}_3 & \mathbf{v}_4 \end{bmatrix} &= \\ &(\alpha x_1 + \beta x_2)(\gamma y_1 + \delta y_2) - (\alpha y_1 + \beta y_2)(\gamma x_1 + \delta x_2) = \\ &\alpha \gamma x_1 y_1 + \alpha \delta x_1 y_2 + \beta \gamma x_2 y_1 + \beta \delta x_2 y_2 - \\ &\alpha \gamma y_1 x_1 - \alpha \delta y_1 x_2 - \beta \gamma x_1 y_2 - \beta \delta x_2 y_2 = \\ &\alpha \delta (x_1 y_2 - y_1 x_2) - \beta \gamma (x_2 y_1 - x_1 y_2). \end{aligned}$$

Iz enačbe (3.9) sledi

$$\det \begin{bmatrix} \mathbf{v}_3 & \mathbf{v}_4 \end{bmatrix} = \pm\alpha\delta \pm \beta\gamma.$$

Vpeljemo oznako

$$\psi := \alpha\delta + \beta\gamma.$$

Ker smo za premike škatlastega zlepka predpostavili linearno neodvisnost, mora biti tudi $\psi = \pm 1$. To pa nas pripelje v protislovje s predpostavko, da so $\alpha, \beta, \gamma, \delta$ vsi enaki ± 1 , saj iz tega sledi, da je

$$\psi \in \{-2, 0, 2\}$$

in v nobenem primeru ne more biti enak ± 1 . □

Z naslednjim izrekom pokažemo, da premiki škatlastih zlepkov tvorijo particijo enote.

Izrek 3.20. *Naj bo \mathcal{S} prostor premikov škatlastega zlepka določenega z matriko $M \in \mathbb{Z}^{d \times n}$. Zlepki iz prostora \mathcal{S} tvorijo particijo enote.*

Dokaz. Naj bo $M_d \in \mathbb{Z}^{d \times d}$ in naj bodo stolpci matrike M_d linearno neodvisni, torej tvorijo bazo vektorskega prostora \mathbb{Z}^d . Očitno je, da se odsekoma konstantni zlepki $B_{M_d}(\mathbf{x} - \mathbf{j})$, za $\mathbf{j} \in M_d\mathbb{Z}^d$, seštejejo v

$$\gamma := \frac{1}{\det(M_d)}.$$

Prostor \mathbb{Z}^d lahko razcepimo v množice oblike $\mathbf{i} + M_d\mathbb{Z}^d$, $\mathbf{i} \in \mathbb{Z}^i$. Ker baza prostora \mathbb{Z}^d , vsebovana v matriki M_d , ni nujno sestavljena iz enotskih vektorjev, takšne množice prostor \mathbb{Z}^d pokrijejo večkrat. Prikaz pokrivanja takšnega prostora prikazuje slika 27. Recimo, da prostor s takšnimi množicami pokrijemo m -krat.

Sledi, da je

$$\sum_{\mathbf{i} \in \mathbb{Z}^d} B_{M_d}(\mathbf{x} - \mathbf{i}) = m\gamma. \quad (3.10)$$

Ker je $\int_0^1 m\gamma dt = 1$ in ker za poljuben $\mathbf{u} \in \mathbb{Z}^d$ po definiciji 3.4 velja

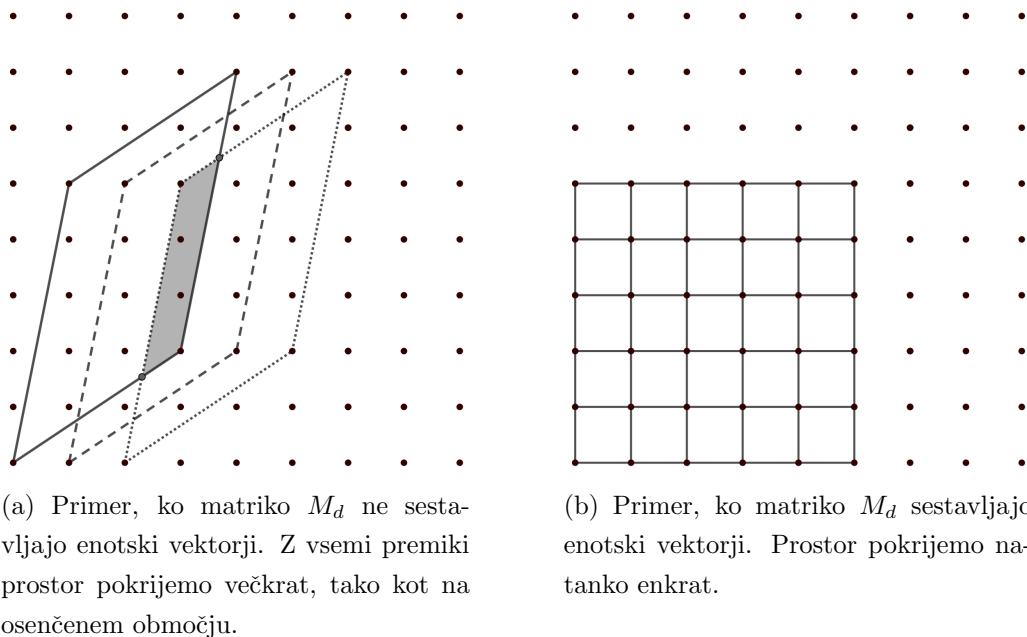
$$B_{M_d \cup \mathbf{u}}(\mathbf{x} - \mathbf{i}) = \int_0^1 B_{M_d}(\mathbf{x} - \mathbf{i} - t\mathbf{u}) dt,$$

lahko na obeh straneh enačbe (3.10) naredimo $k - d$ zaporednih konvolucij in dodamo $k - s$ poljubnih vektorjev iz prostora \mathbb{Z}^d na način

$$\int_0^1 m\gamma dt = m\gamma = \int_0^1 B_{M_d}(\mathbf{x} - \mathbf{i} - t\mathbf{v}_{s+1}) = B_{M_d \cup \mathbf{v}_{s+1}}(\mathbf{x} - \mathbf{i}).$$

Po končanih $k - d$ zaporednih konvolucijah označimo $M_k = \left[\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_d \mathbf{v}_{d+1} \cdots \mathbf{v}_k \right]$ in še vedno velja

$$m\gamma = B_{M_k}(\mathbf{x} - \mathbf{i}).$$



Slika 26: Dva primera pokritja prostora \mathbb{Z}^2 .

Na enak način naredimo še d konvolucij, kjer dodamo še d enotskih vektorjev prostora \mathbb{Z}^d . Označimo z

$$M = \left[\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_k \mathbf{e}_1 \mathbf{e}_2 \cdots \mathbf{e}_d \right].$$

Ker vrstni red stolpcev ni pomemben lahko zapišemo

$$M = \left[\mathbf{e}_1 \mathbf{e}_2 \cdots \mathbf{e}_d \mathbf{v}_1 \cdots \mathbf{v}_k \right].$$

Še vedno velja

$$m\gamma = B_M(\mathbf{x} - \mathbf{i}). \quad (3.11)$$

Poglejmo si, kaj se zgodi, če si namesto poljubne baze v matriki M_d vzamemo bazo sestavljeno iz enotskih vektorjev. Potem cel prostor pokrijemo natanko enkrat, sledi $m = 1$, determinanta $\det M_d = 1$, torej lahko enačbo (3.10) zapišemo kot

$$\sum_{\mathbf{i} \in \mathbb{Z}^d} B_{M_d}(\mathbf{x} - \mathbf{i}) = 1.$$

Zopet najprej naredimo $k-d$ zaporednih konvolucij, da dobimo $M_k = \left[\mathbf{e}_1 \mathbf{e}_2 \cdots \mathbf{e}_d \mathbf{v}_1 \cdots \mathbf{v}_k \right]$ in s tem

$$1 = m\gamma = B_{M_k}(\mathbf{x} - \mathbf{i}).$$

Ko naredimo še d dodatnih konvolucij, da dodamo še d poljubnih vektorjev, dobimo

$$1 = B_M(\mathbf{x} - \mathbf{i}). \quad (3.12)$$

Primerjemo enačbi (3.12) in (3.11) in vidimo, da smo na oba načina, ko smo začeli s poljubnimi in ko smo začeli z enotskimi vektorji, prišli do enake matrike. Sledi, da je

$$m\gamma = 1$$

in s tem, da celoštevilski premiki poljubnega škatlastega zlepka tvorijo particijo enote. \square

Z naslednjim izrekom eksplicitno pokažemo, da so tenzorski produkti B-zlepkov le poseben primer škatlastih zlepkov.

Izrek 3.21. Naj bo $M \in \mathbb{Z}^{2 \times (m+n)}$ sestavljena iz m enotskih vektorjev \mathbf{e}_1 in n enotskih vektorjev \mathbf{e}_2 , kjer sta

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Dalje naj bosta $p = m - 1$, $q = n - 1$ polinomski stopnji, ki skupaj z vektorjema vozlov $U = (0, 1, \dots, m)$ in $V = (0, 1, \dots, n)$ določata bazni zlepek prostora tenzorskih produktov B-zlepkov,

$$N_{0,0}^{p,q}(x, y).$$

Potem je ta bazni B-zlepek enak škatlastemu zlepku

$$B_M(\mathbf{x}).$$

Dokaz. Izrek dokažemo z indukcijo. Za bazni primer vzamemo $m = 1$, $n = 1$. Če izrek drži, potem je škatlasti zlepek določen z matriko

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

enak konstantnemu zlepku

$$N_{0,0}^{0,0}(x, y)$$

nad vektorjema vozlov $U = (0, 1)$, $V = (0, 1)$. Po definiciji 3.4 je

$$B_M(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in [0, 1]^2 \\ 0, & \text{sicer} \end{cases}.$$

Po definiciji 2.5 pa je

$$\begin{aligned} N_{0,0}^{0,0}(x, y) &= N_{0,0}^U(x) \cdot N_{0,0}^V(y) \\ &= \mathbf{1}_{[0,1]}(x) \cdot \mathbf{1}_{[0,1]}(y) = \mathbf{1}(x \in [0, 1] \wedge y \in [0, 1]) \\ &= \begin{cases} 1, & (x, y) \in [0, 1]^2 \\ 0, & \text{sicer} \end{cases}. \end{aligned}$$

Izrek v tem primeru očitno drži, saj lahko zapišemo $\mathbf{x} = (x, y)$. Predpostavimo, da trditev izreka velja za vse matrike sestavljene iz $m - 1$ vektorjev \mathbf{e}_1 in $n - 1$ vektorjev \mathbf{e}_2 , $m, n \in \mathbb{N}$. Takšno matriko označimo z M' . Eksplicitno zapisana indukcijska predpostavka se potem glasi

$$B_{M'}(\mathbf{x}) = N_{0,0}^{m-2,n-2}(x, y), \quad (3.13)$$

kjer je $N_{0,0}^{m-2,n-2}$ bazni zlepek prostora tenzorskih produktov B-zlepkov stopenj $m - 2, n - 2$ z vektorjema vozlov $U' = (0, 1, \dots, m - 1)$ in $V' = (0, 1, \dots, n - 1)$. Naj bo sedaj matrika M sestavljena iz m vektorjev \mathbf{e}_1 in n vektorjev \mathbf{e}_2 . Za škatlasti zlepek, določen s to matriko, po lemi 3.8 velja

$$\begin{aligned} D_{\mathbf{e}_2}(D_{\mathbf{e}_1}B_M(x, y)) &= D_{\mathbf{e}_2}(\nabla_{\mathbf{e}_1}B_M(x, y)) \\ &= D_{\mathbf{e}_2}(B_{M \setminus \mathbf{e}_1}(x, y) - (B_{M \setminus \mathbf{e}_1}(x - 1, y))) \\ &= \nabla_{\mathbf{e}_2}(B_{M \setminus \mathbf{e}_1}(x, y) - (B_{M \setminus \mathbf{e}_1}(x - 1, y))) \\ &= B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x, y) - B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x - 1, y) \\ &\quad - B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x, y - 1) + B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x - 1, y - 1). \end{aligned}$$

Zadnja vrstica ravno ustreza indukcijski predpostavki (3.13), torej lahko zapišemo

$$\begin{aligned} D_{\mathbf{e}_2}(D_{\mathbf{e}_1}B_M(x, y)) &= \\ &= N_{0,0}^{m-2,n-2}(x, y) - N_{0,0}^{m-2,n-2}(x - 1, y) - \\ &= N_{0,0}^{m-2,n-2}(x, y - 1) + N_{0,0}^{m-2,n-2}(x - 1, y - 1). \end{aligned} \quad (3.14)$$

Poglejmo še obratno smer, kjer definiramo polinomski stopnji $p = m - 1$ in $q = n - 1$ ter vektorja vozlov $U = (0, 1, \dots, m)$, $V = (0, 1, \dots, n)$. Potem za bazni zlepek $N_{0,0}^{m-1,n-1}$ prostora tenzorskih produktov B-zlepkov, definiranih s stopnjama p, q in vektorjema vozlov U, V po lastnosti (2.2) velja

$$\begin{aligned} \frac{\partial}{\partial y} \frac{\partial}{\partial x} N_{0,0}^{m-1,n-1}(x, y) &= \frac{\partial}{\partial y} \frac{\partial}{\partial x} (N_{0,m-1}^U(x) \cdot N_{0,n-1}^V(y)) \\ &= (N_{0,m-2}^U(x) - N_{1,m-2}^U(x)) \cdot (N_{0,n-2}^V(y) - N_{1,n-2}^V(y)). \end{aligned} \quad (3.15)$$

Ker je bazni zlepek $N_{1,m-2}^{U'}$ le za eno premaknjen bazni zlepek $N_{0,m-2}^{U'}$ lahko zapišemo

$$N_{1,m-2}^{U'}(x) = N_{0,m-2}^{U'}(x - 1).$$

Analogno velja tudi za bazni zlepek $N_{1,n-2}^{V'}$. Potem lahko izraz (3.15) nadalje zapišemo kot

$$\begin{aligned} \frac{\partial}{\partial y} \frac{\partial}{\partial x} N_{0,0}^{m-1,n-1}(x, y) &= N_{0,0}^{m-2,n-2}(x, y) - N_{0,0}^{m-2,n-2}(x - 1, y) - \\ &= N_{0,0}^{m-2,n-2}(x, y - 1) + N_{0,0}^{m-2,n-2}(x - 1, y - 1). \end{aligned}$$

Ta izraz zopet ustreza indukcijski predpostavki in lahko zapišemo

$$\begin{aligned} \frac{\partial}{\partial y} \frac{\partial}{\partial x} N_{0,0}^{m-1,n-1}(x,y) &= B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x,y) - B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x-1,y) \\ &\quad - B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x,y-1) + B_{M \setminus \{\mathbf{e}_1, \mathbf{e}_2\}}(x-1,y-1). \end{aligned} \quad (3.16)$$

Ker so smerni odvodi po enotskih vektorjih kar parcialni odvodi lahko s primerjanjem (3.14) in (3.16) zapišemo

$$\frac{\partial}{\partial y} \frac{\partial}{\partial x} B_M(x,y) = \frac{\partial}{\partial y} \frac{\partial}{\partial x} N_{0,0}^{m-1,n-1}(x,y). \quad (3.17)$$

Dobimo, da sta mešana druga odvoda enaka. Potem lahko obe strani enačbe (3.17) integriramo po spremenljivki x in dobimo

$$\begin{aligned} \int_0^x \frac{\partial}{\partial y} \frac{\partial}{\partial x} B_M(x,y) dx &= \int_0^x \frac{\partial}{\partial y} \frac{\partial}{\partial x} N_{0,0}^{m-1,n-1}(x,y) dx \\ \frac{\partial}{\partial y} B_M(x,y) - \frac{\partial}{\partial y} B_M(0,y) &= \frac{\partial}{\partial y} N_{0,0}^{m-1,n-1}(x,y) - \frac{\partial}{\partial y} N_{0,0}^{m-1,n-1}(0,y) \end{aligned} \quad (3.18)$$

Ker tako za škatlaste kot B-zlepke velja, da so bazne funkcije na robovih svojih nosilcev enake 0, lahko zadnjo vrstico enačbe (3.18) zapišemo kot

$$\frac{\partial}{\partial y} B_M(x,y) = \frac{\partial}{\partial y} N_{0,0}^{m-1,n-1}(x,y) \quad (3.19)$$

Obe strani enačbe (3.19) integriramo še po spremenljivki y in dobimo

$$\begin{aligned} \int_0^y \frac{\partial}{\partial y} B_M(x,y) dy &= \int_0^y \frac{\partial}{\partial y} N_{0,0}^{m-1,n-1}(x,y) dy \\ B_M(x,y) - B_M(x,0) &= N_{0,0}^{m-1,n-1}(x,y) - N_{0,0}^{m-1,n-1}(x,0) \\ B_M(x,y) &= N_{0,0}^{m-1,n-1}(x,y). \end{aligned} \quad (3.20)$$

□

Poglavje končamo s formalno definicijo funkcije prostora škatlastih zlepkov.

Definicija 3.22. Naj bo \mathcal{S} prostor premikov škatlastega zlepka $B_M(\mathbf{x} - \mathbf{i})$, kjer je $M \in \mathbb{Z}^{d \times n}$ in $\mathbf{i} \in \mathbb{Z}^d$. Vsako funkcijo $f \in \mathcal{S}$ zapišemo kot linearno kombinacijo

$$f(\mathbf{x}) = \sum_{\mathbf{i} \in \mathbb{Z}^d} \mathbf{P}_i \cdot B_M(\mathbf{x} - \mathbf{i}),$$

kjer so $\mathbf{P}_i \in \mathbb{R}^d$ kontrolne točke.

4 HIERARHIČNI ZLEPKI

V tem poglavju zapišemo nekaj teorije za glavni problem magistrskega dela. Tudi to poglavje se v grobem deli na dva dela. V enem obravnavamo hierarhične prostore B-zlepkov, v drugem pa hierarhične prostore škatlastih zlepkov. Pri B-zlepkih bomo najprej definirali ugnezdene prostore in domene ter navedli nekaj pomembnih pogojev, da res dobimo ugnezdenost. Nato bomo skonstruirali hierarhično bazo in navedli nekaj lastnosti. Sledi podpoglavje o širjenju ugnezdenih domen in poglavje o prirezani hierarhični bazi. Po tem poglavju na podoben način obravnavamo tudi hierarhično bazo škatlastih zlepkov, kjer jo najprej definiramo in navedemo nekaj lastnosti. Videli bomo, da se tudi v primeru hierarhične baze škatlastih zlepkov srečamo s podobnimi problemi in lastnostmi kot v primeru B-zlepkov. V primeru škatlastih zlepkov se omejimo še na primer navadne hierarhične baze, čeprav je s to vrsto zlepkov mogoče konstruirati tudi prirezano bazo, analogno tisti v primeru B-zlepkov. O prirezani hierarhični bazi najde bralec več informacij v [10].

4.1 UGNEZDENI PROSTORI IN DOMENE B-ZLEPKOV

Naj bo $(\mathcal{B}^\ell)_{\ell=0,1,\dots,N-1}$ končno zaporedje prostorov tenzorskih produktov B-zlepkov dveh spremenljivk, za katere predpostavimo, da so ugnezdeni. Torej,

$$\mathcal{B}^0 \subseteq \mathcal{B}^1 \subseteq \dots \subseteq \mathcal{B}^{N-1},$$

in naj bo $(\Omega^\ell)_{\ell=0,1,\dots,N-1}$ končno zaporedje omejenih odprtih množic, za katere velja

$$\Omega^{N-1} \subseteq \Omega^{N-2} \subseteq \dots \subseteq \Omega^0, \quad \Omega^N = \emptyset,$$

in definira ugnezdene domene za hierarhijo zlepkov. Začenši s stopnjama (p^0, q^0) , ki definirata prostor \mathcal{B}^0 , je vsak naslednji prostor $\mathcal{B}^{\ell+1} \supseteq \mathcal{B}^\ell$, $\ell = 0, 1, \dots, N-2$, definiran s polinomske stopnjama $(p^{\ell+1}, q^{\ell+1})$, za katera velja

$$p^{\ell+1} \geq p^\ell, \quad q^{\ell+1} \geq q^\ell, \quad \ell = 0, 1, \dots, N-2,$$

ter horizontalnim in vertikalnim vektorjem vozlov $U^{\ell+1}, V^{\ell+1}$, na katerih je definirana baza $\mathcal{N}^{\ell+1}$. Tudi za vektorja vozlov predpostavimo, da sta ugnezdena, torej

$$U^\ell \subseteq U^{\ell+1}, \quad V^\ell \subseteq V^{\ell+1}, \quad \ell = 0, \dots, N-2.$$

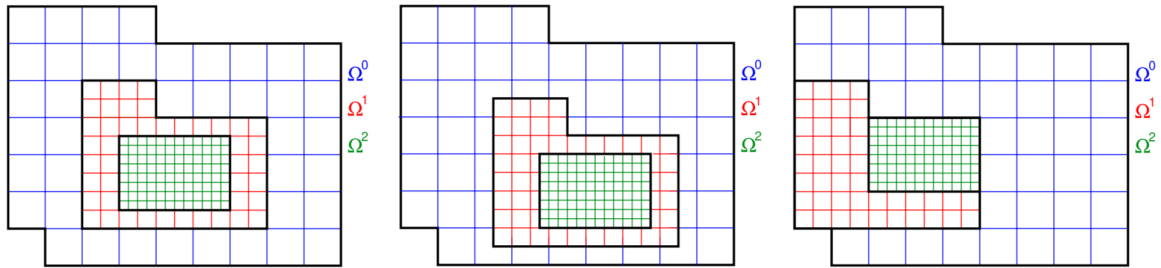
Da res dobimo ugnezdene prostore, moramo predpostaviti še pogoj

$$\mu(U^{\ell+1}, x) - \mu(U^\ell, x) \geq p^{\ell+1} - p^\ell,$$

$$\mu(V^{\ell+1}, y) - \mu(V^\ell, y) \geq q^{\ell+1} - q^\ell,$$

za vsak x, y in $\ell = 0, 1, \dots, N - 2$.

Na vsakem nivoju je rob $\partial\Omega^\ell$, $\ell = 0, 1, \dots, N - 1$, lahko poravnan z vozli prostora $\mathcal{B}^{\ell-1}$ (*krepki pogoj*) ali pa je poravnan z vozli prostora \mathcal{B}^ℓ (*šibki pogoj*). Če ni pogoj izrecno določen, vedno predpostavimo šibki pogoj. Različne primere pogojev prikazuje slika 27.



(a) Krepki pogoj na robovih do-(b) Šibki pogoj na robovih domen. (c) Primer $\partial\Omega^\ell \cap \partial\Omega^{\ell+1} \neq \emptyset$. men.

Slika 27: Primeri različnih pogojev na robovih domen. Z različnimi barvami so označena različna območja v hierarhiji. Osnovno območje je modro, znotraj modrega območja imamo rdeče, ki je finejše od modrega, in znotraj tega še zeleno območje, ki je še finejše.

4.2 HIERARHIČNA BAZA B-ZLEPKOV

Definicija 4.1. Naj bosta $(\mathcal{B}^\ell)_{\ell=0,1,\dots,N-1}$ in $(\Omega^\ell)_{\ell=0,1,\dots,N-1}$ zaporedja prostorov ugnezdenih tenzorskih produktov B-zlepkov in ugnezdenih domen, definiranih v podpoglavju 4.1. Hierarhično bazo označimo s \mathcal{K} in jo konstruiramo rekurzivno na naslednji način

(i) *Inicializacija:* $K^0 = \{\tau \in \mathcal{N}^0 : \text{supp}(\tau) \neq \emptyset\}$.

(ii) *Konstrukcija $K^{\ell+1}$ iz K^ℓ (rekurzivno):*

$$K^{\ell+1} = K_A^{\ell+1} \cup K_B^{\ell+1},$$

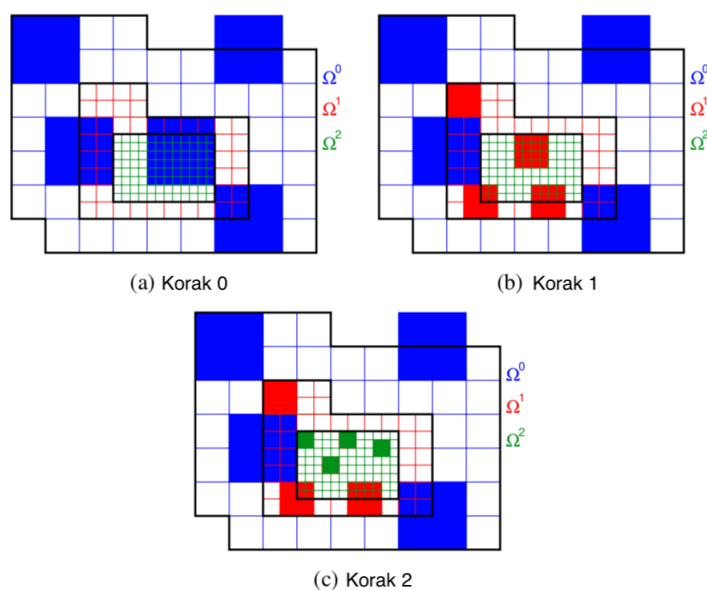
kjer sta

$$K_A^{\ell+1} = \{\tau \in K^\ell : \text{supp}(\tau) \not\subseteq \Omega^{\ell+1}\}$$

in

$$K_B^{\ell+1} = \{\tau \in \mathcal{N}^{\ell+1} : \text{supp}(\tau) \subseteq \Omega^{\ell+1}\}.$$

(iii) $\mathcal{K} = K^{N-1}$.



Slika 28: Primer treh korakov konstrukcije hierarhične baze, kjer sta $(p^\ell, q^\ell) = (1, 1)$, $\ell = 0, 1, 2$.

4.2.1 Dve lastnosti hierarhične baze

Lema 4.2. *Funkcije iz \mathcal{K} so linearno neodvisne.*

Dokaz. Obravnavajmo vsoto

$$\sum_{\tau \in \mathcal{K}} d_\tau \tau = 0,$$

ki jo lahko razpišemo kot

$$\sum_{\tau \in \mathcal{K}} d_\tau \tau = \sum_{\tau \in \mathcal{K} \cap \mathcal{N}^0} d_\tau \tau + \sum_{\tau \in \mathcal{K} \cap \mathcal{N}^1} d_\tau \tau + \cdots + \sum_{\tau \in \mathcal{K} \cap \mathcal{N}^{N-1}} d_\tau \tau = 0.$$

Želimo pokazati, da so $d_\tau = 0$. Zgornjo vsoto bomo obravnavali člen po člen. V prvem členu opazimo, da je $\mathcal{K} \cap \mathcal{N}^0 \subseteq \mathcal{N}^0$. Po drugi lastnosti v poglavju 2.3 sledi,

da so vse funkcije $\tau \in \mathcal{K} \cap \mathcal{N}^0$ linearno neodvisne. Opazimo tudi, da so na območju $\Omega^0 \setminus \Omega^1$ neničelne le funkcije $\tau \in \mathcal{K} \cap \mathcal{N}^0$. Sledi, da je $d_\tau = 0$ za vsak $\tau \in \mathcal{K} \cap \mathcal{N}^0$. Analogno lahko razmislimo tudi za vse vsote $\sum_{\tau \in \mathcal{K} \cap \mathcal{N}^\ell} d_\tau \tau$, $\ell = 1, 2, \dots, N-1$. Vsak $\mathcal{K} \cap \mathcal{N}^\ell \subseteq \mathcal{N}^\ell$, torej so vse $\tau \in \mathcal{K} \cap \mathcal{N}^\ell$ linearno neodvisne. Razen mogoče že prej obravnavanih funkcij so na območju $\Omega^\ell \setminus \Omega^{\ell+1}$ neničelne samo funkcije $\tau \in \mathcal{K} \cap \mathcal{N}^\ell$, ki pa so linearno neodvisne po enakem razmisleku kot v prvem delu. Od tod lahko sklepamo, da je $d_\tau = 0$ za vsak $\tau \in \mathcal{K} \cap \mathcal{N}^\ell$, $\ell = 0, 1, \dots, N-1$. \square

Lema 4.3. *Naj bodo K^0, K^1, \dots, K^{N-1} baze zlepkov, definirane v Definiciji 4.1. Potem je*

$$\mathcal{L}in K^\ell \subseteq \mathcal{L}in K^{\ell+1}.$$

Dokaz. Želimo pokazati, da je poljubna funkcija $f \in \mathcal{L}in K^\ell$ vsebovana tudi v $\mathcal{L}in K^{\ell+1}$. Vsako funkcijo $f \in \mathcal{L}in K^\ell$ lahko zapišemo kot

$$f = \sum_{\tau \in K^\ell} d_\tau \tau = \sum_{\tau \in K^\ell, \text{supp} \tau \not\subseteq \Omega^{\ell+1}} d_\tau \tau + \sum_{\tau \in K^\ell, \text{supp} \tau \subseteq \Omega^{\ell+1}} d_\tau \tau. \quad (4.1)$$

Prva vsota na desni strani izraza zajema vse funkcije v $K_A^{\ell+1}$. Z obravnavanjem prostorov $\mathcal{B}^0, \dots, \mathcal{B}^{N-1}$, lahko vsak $\tau \in \mathcal{N}^\ell$ zapišemo kot linearno kombinacijo funkcij baze $\mathcal{N}^{\ell+1}$ kot

$$\tau = \sum_{\sigma \in \mathcal{N}^{\ell+1}, \text{supp}(\sigma) \subseteq \text{supp}(\tau)} c_\sigma^{\ell+1}(\tau) \cdot \sigma, \quad (4.2)$$

kjer $c_\sigma^{\ell+1}(\tau)$ označuje koeficient pri σ v razpisu funkcije τ . Če vstavimo enačbo (4.2) v drugo vsoto enačbe (4.1), dobimo

$$f = \sum_{\tau \in K_A^{\ell+1}} d_\tau \tau + \sum_{\tau \in K^\ell, \text{supp} \tau \subseteq \Omega^{\ell+1}} d_\tau \left(\sum_{\sigma \in \mathcal{N}^{\ell+1}, \text{supp} \sigma \subseteq \text{supp} \tau} c_\sigma^{\ell+1}(\tau) \cdot \sigma \right).$$

Ker je $\text{supp} \sigma \subseteq \text{supp} \tau \subseteq \Omega^{\ell+1}$ in ker opazimo, da so dodatni koeficienti funkcij σ , katerih nosilec ni vsebovan v nosilcu funkcij τ iz enačbe (4.2), enaki 0, lahko sklepamo, da je $\text{supp} \sigma \subseteq \Omega^{\ell+1}$. Zamenjava vrstnega reda vsot v prejšnji enačbi nas pripelje do

$$\begin{aligned} f &= \sum_{\tau \in K_A^{\ell+1}} d_\tau \tau + \sum_{\sigma \in \mathcal{N}^{\ell+1}, \text{supp} \sigma \subseteq \Omega^{\ell+1}} \left(\sum_{\tau \in K^\ell, \text{supp} \tau \subseteq \Omega^{\ell+1}} d_\tau c_\sigma^{\ell+1}(\tau) \right) \sigma \\ &= \sum_{\tau \in K_A^{\ell+1}} d_\tau \tau + \sum_{\sigma \in K_B^{\ell+1}} d_\sigma \sigma, \end{aligned} \quad (4.3)$$

kjer je

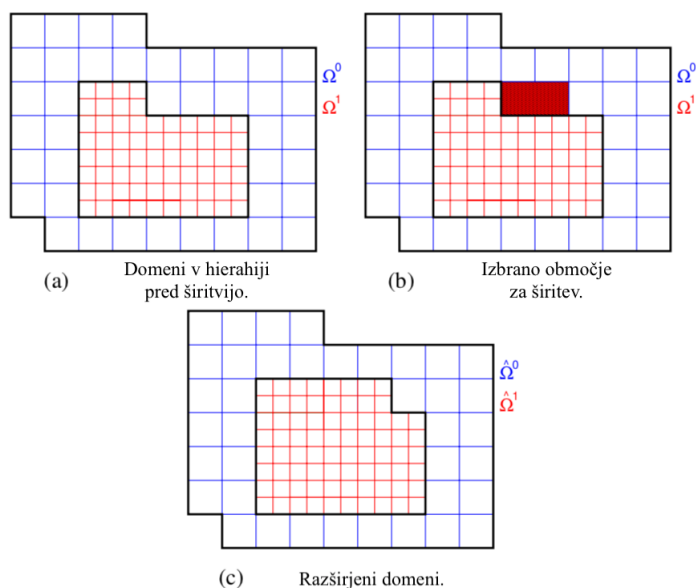
$$d_\sigma = \sum_{\tau \in K^\ell, \text{supp } \tau \subseteq \Omega^{\ell+1}} d_\tau c_\sigma^{\ell+1}(\tau).$$

Prva vsota v zadnji vrstici pripada linearni ogrinjači funkcij iz $K_A^{\ell+1}$, druga vsota pa linearni ogrinjači funkcij iz $K_B^{\ell+1}$. Sledi, da je $f \in \mathcal{L}in K^{\ell+1}$ in s tem $\mathcal{L}in K^\ell \subseteq \mathcal{L}in K^{\ell+1}$. \square

4.2.2 Širjenje domen hierarhije zlepkov

V samih aplikacijah hierarhičnih zlepkov je ključno omogočiti lokalno zgoščevanje na osnovi neke funkcije ocenjevanja napake. Začenši z osnovno mrežo, funkcija napake zazna območja mreže, ki jih je potrebno zgostiti.

Tipično začnemo z enakomerno diskretizacijo $\Omega^0 \supseteq \dots \supseteq \Omega^{N-1} = \emptyset$. Funkcija napake označi območja, ki jih je potrebno zgostiti in zgostimo Ω^0 . Na vsakem koraku lahko pričakujemo, da se maksimalni nivo zgoščevanja (največji ℓ t.d. $\Omega^\ell \neq \emptyset$) poveča za ena. Lahko pa se nam zgodi, da moramo zgostiti območje, ki ni v celoti vsebovano v domeni na maksimalnem nivoju. Primer podaja slika 9. V tem primeru lahko uvedemo novo zaporedje ugnezdenih domen $(\hat{\Omega}^\ell)_{\ell=0,1,\dots,N-1}$, ki omogoči zgostitev dodatnih območij, zaznanih s funkcijo napake. Naslednja lema nam zagotovi ohranjanje ugnezdene narave prostora zlepkov.



Slika 29: Primer povečevanja domene Ω^1 .

Lema 4.4. *Naj bosta*

$$\hat{\Omega}^{N-1} \subseteq \hat{\Omega}^{N-2} \subseteq \dots \subseteq \hat{\Omega}^0 \quad \text{in} \quad \Omega^{N-1} \subseteq \Omega^{N-2} \subseteq \dots \subseteq \Omega^0$$

zaporedji ugnezenih domen skupaj s pripadajočimi hierarhičnimi bazami K^ℓ in \hat{K}^ℓ , konstruiranimi kot v definiciji 4.1 za $\ell = 0, 1, \dots, N-1$. Če je $\Omega^\ell \subseteq \hat{\Omega}^\ell$ za $\ell = 0, 1, \dots, N-1$, potem je

$$\mathcal{L}in K^\ell \subseteq \mathcal{L}in \hat{K}^\ell.$$

Dokaz. Na vsakem nivoju ℓ , zaporedje $(\hat{\Omega}^\ell)_{\ell=0,1,\dots,N-1}$, povečano glede na zaporedje $(\Omega^\ell)_{\ell=0,1,\dots,N-1}$, pokriva večje območje domene funkcij iz naslednje baze pripadajočega zaporedja prostorov $\{\mathcal{N}_{\ell=0,1,\dots,N-1}^\ell\}$. Če velja $K^\ell = K_A^\ell \cup K_B^\ell$ in $\hat{K}^\ell = \hat{K}_A^\ell \cup \hat{K}_B^\ell$, dobimo

$$K_A^\ell \supseteq \hat{K}_A^\ell, \quad K_B^\ell \subseteq \hat{K}_B^\ell$$

in $\mathcal{L}in(K_A^\ell \setminus \hat{K}_A^\ell) \subseteq \mathcal{L}in \hat{K}_B^\ell$. Torej je prostor, ki ga napenjajo bazne funkcije iz K^ℓ , vsebovan v prostoru, ki ga napenjajo bazne funkcije iz \hat{K}^ℓ . \square

4.2.3 Utežena hierarhična baza

Ena ključnih lastnosti predstavitve krivulje/ploskve z B-zlepki je, da je krivulja/ploskev konveksna kombinacija kontrolnih točk. Da to dosežemo, lahko bazo \mathcal{K} predelamo v bazo \mathcal{W} , katera na vsakem nivoju tvori particijo enote. Ker so bazne funkcije B-zlepkov na nosilcu nenegativne, bo z definiranjem baze, ki tvori particijo enote, krivulja/ploskev konveksna kombinacija kontrolnih točk, kar implicira, da krivulja/ploskev leži znotraj *konveksne ovojnice* kontrolnih točk.

Če konstantna funkcija $f = 1$ pripada $\mathcal{L}in(K^0)$, jo lahko predstavimo v hierarhični bazi kot

$$1 = \sum_{\tau \in \mathcal{K}} w_\tau \tau. \quad (4.4)$$

Torej, če za vsako funkcijo $\tau \in \mathcal{K}$ definiramo funkcijo ω in predpostavimo $w_\tau \neq 0$, dobimo normalizirano hierarhično bazo

$$\mathcal{W} = \left\{ \omega = w_\tau \tau : \tau \in \mathcal{K} \wedge 1 = \sum_{\tau \in \mathcal{K}} w_\tau \tau \right\},$$

ki tvoji particijo enote, tj.

$$\sum_{\omega \in \mathcal{W}} \omega = 1.$$

Lema 4.5. *Vsaka utež w_τ , ki pripada $\tau \in \mathcal{K}$ v enačbi (4.4), je večja ali enaka 0.*

Dokaz. Ker je $1 \in K^0$, lahko zapišemo

$$1 = \sum_{\gamma \in \mathcal{N}^0} d_\gamma \gamma$$

z $d_\gamma \geq 0$ in potem

$$1 = \sum_{\gamma \in \mathcal{N}^0, \text{supp}\gamma \not\subseteq \Omega^1} d_\gamma \gamma + \sum_{\gamma \in \mathcal{N}^0, \text{supp}\gamma \subseteq \Omega^1} d_\gamma \gamma. \quad (4.5)$$

Prva vsota v enačbi (4.5) zajema vse bazne funkcije v K_A^1 . Kot smo že videli v dokazu leme ??, lahko vsako bazno funkcijo $\tau \in \mathcal{N}^0$ izrazimo kot linearno kombinacijo funkcij iz \mathcal{N}^1 kot

$$\gamma = \sum_{\sigma \in \mathcal{N}^1, \text{supp}\sigma \subseteq \Omega^1} c_\sigma^1(\gamma) \sigma,$$

in $c_\sigma^1 \geq 0$. Z zamenjavo zgornjega izraza za γ v enačbi (4.5) dobimo

$$1 = \sum_{\gamma \in K_A^1} d_\gamma \gamma + \sum_{\gamma \in \mathcal{N}^0, \text{supp}\gamma \subseteq \Omega^1} d_\gamma \left(\sum_{\sigma \in \mathcal{N}^1, \text{supp}\sigma \subseteq \Omega^1} c_\sigma^1(\gamma) \sigma \right).$$

Z menjavo vrstnega reda vsot v prejšnji enačbi dobimo

$$\begin{aligned} 1 &= \sum_{\gamma \in K_A^1} d_\gamma \gamma + \sum_{\sigma \in \mathcal{N}^1, \text{supp}\sigma \subseteq \Omega^1} \left(\sum_{\gamma \in \mathcal{N}^0, \text{supp}\gamma \subseteq \Omega^1} d_\gamma c_\sigma^1(\gamma) \right) \sigma \\ &= \sum_{\gamma \in K_A^1} d_\gamma \gamma + \sum_{\sigma \in K_B^1} d_\sigma \sigma, \end{aligned} \quad (4.6)$$

kjer je

$$d_\sigma = \sum_{\gamma \in \mathcal{N}^0, \text{supp}\gamma \subseteq \Omega^1} d_\gamma c_\sigma^1(\gamma) \geq 0.$$

Prva vsota druge vrstice enačbe (4.6) pripada linearni ogrinjači funkcij iz K_A^1 , druga pa linearni ogrinjači funkcij iz K_B^1 in vsi koeficienti so večji ali enaki 0. Enak razmislek lahko apliciramo na vse nadaljnje nivoje in vidimo, da lahko funkcijo $1 \in \mathcal{L}in(K^\ell)$, $\ell = 1, 2, \dots, N - 2$, izrazimo kot linearno kombinacijo funkcij iz $K^{\ell+1}$, $\ell = 1, 2, \dots, N - 2$, ter da so vsi koeficienti večji ali enaki 0.

□

Vse do sedaj v tem podpoglavju nam je povedalo in dokazalo le, da je hierarhično bazo, ki tvori particijo enote, mogoče skonstruirati. V resnici baze, tvorjene po definiciji 4.1 pogosto ne tvorijo particije enote. Zgodi se naprimer, da ob pomiku iz nivoja ℓ na nivo $\ell + 1$ v hierarhiji ne obstaja bazna funkcija $\tau \in \mathcal{K}$, z nosilcem v celoti vsebovanem na

$\Omega^{\ell+1}$. To pomeni, da je zgoščena domena manjša kot nosilci baznih funkcij, definirani na prejšnjem nivoju. V tem primeru bodo vse funkcije iz K^ℓ obdržane v hierarhični bazi in ker baza \mathcal{N}^ℓ tvori particijo enote zaradi lastnosti iz poglavja 2.3, bo utež vsake funkcije iz $\mathcal{N}^{\ell+1}$, katere nosilec je vsebovan na $\Omega^{\ell+1}$, enaka 0. Bolj natančno, ta primer se zgodi, če in samo če obstaja bazna funkcija $\tau \in \mathcal{K}_B^{\ell+1}$, katere nosilec ni vsebovan v nosilcu funkcij iz $K^\ell \setminus K^{\ell+1}$. Temu primeru se je sicer moč izogniti, če predpostavimo krepki pogoj na robovih domen in definiramo domeno $\Omega^{\ell+1}$ kot unijo nosilcev baznih funkcij $\tau \in \mathcal{N}^\ell$

$$\Omega^{\ell+1} = \overline{\bigcup_{\tau \in S} \text{supp}\tau},$$

kjer je $S \subseteq \mathcal{N}^\ell$.

Kot bomo videli v nadaljevanju, lahko prekrivanje baznih funkcij močno zmanjšamo z vpeljavo nove, tako imenovane *prirezane baze*, ki pa si zasluži svoje poglavje.

4.3 PRIREZANI HIERARHIČNI B-ZLEPKI

Hierarhična baza v prejšnjem poglavju nima lastnosti particije enote. Poleg tega se prekrivanje baznih funkcij, ki pripadajo določenemu nivoju, lahko hitro začne povečevati. V poglavju 4.2.3 smo zapisali motivacijo konstrukcije nove normalizirane baze, v tem poglavju pa jo bomo realizirali. Ključna ideja se nahaja v naslednji definiciji.

Definicija 4.6. Naj bo $\tau \in \mathcal{N}^\ell$ in naj bo

$$\tau = \sum_{\beta \in \mathcal{N}^{\ell+1}} c_\beta^{\ell+1}(\tau)\beta$$

zapis funkcije τ kot linearna kombinacija funkcij iz baze $\mathcal{N}^{\ell+1}$. Potem definiramo **prirez** funkcije τ glede na prostor $\mathcal{B}^{\ell+1}$ in domeno $\Omega^{\ell+1}$ kot

$$\text{trunc}^{\ell+1}(\tau) = \sum_{\beta \in \mathcal{N}^{\ell+1}, \text{supp}\beta \not\subseteq \Omega^{\ell+1}} c_\beta^{\ell+1}(\tau)\beta.$$

Z apliciranjem prirezovanja na bazne funkcije grobjih nivojev, lahko uvedemo prirezano bazo.

Definicija 4.7. Prirezana hierarhična baza (THB) \mathcal{T} je modifikacija klasične hierarhične baze \mathcal{K} iz definicije 4.1. Konstrukcijo modificiramo na naslednji način

- (i) *Inicializacija:* $T^0 = K^0$

(ii) *Konstrukcija $T^{\ell+1}$ iz T^ℓ (rekurzivno):*

$$T^{\ell+1} = T_A^{\ell+1} \cup T_B^{\ell+1}$$

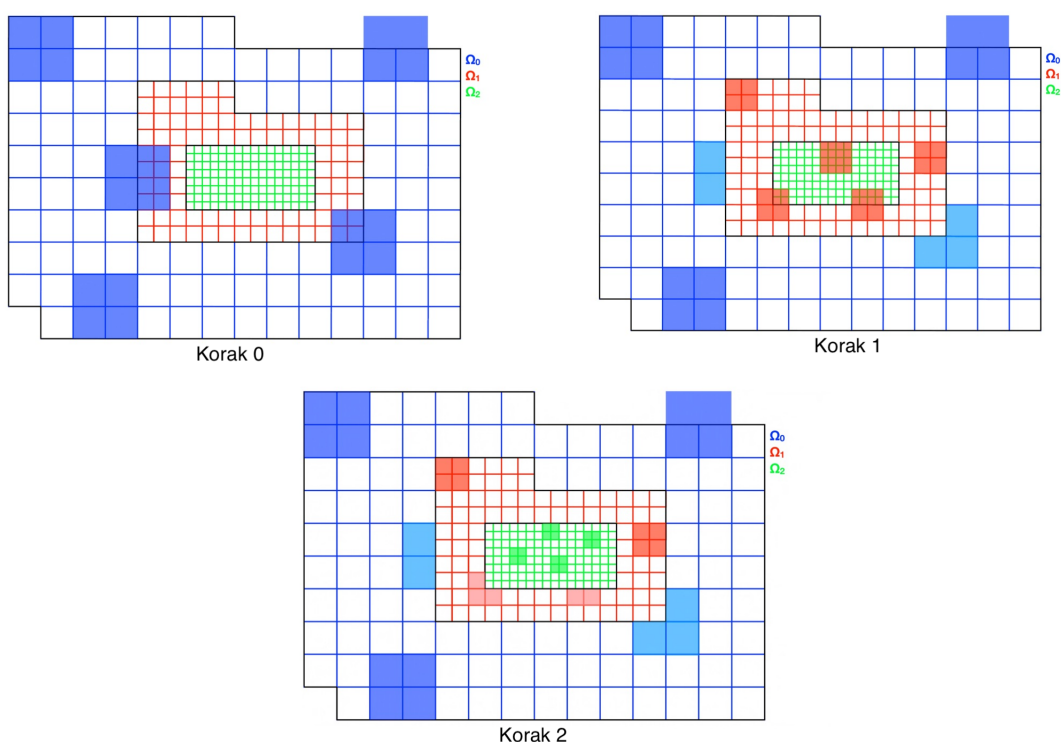
kjer sta

$$T_A^{\ell+1} = \{\text{trunc}^{\ell+1} \tau : \tau \in T^\ell \wedge \text{supp} \tau \not\subseteq \Omega^{\ell+1}\}$$

in

$$T_B^{\ell+1} = K_B^{\ell+1}$$

(iii) $\mathcal{T} = T^{N-1}$



Slika 30: Primer treh korakov konstrukcije prirezane hierarhične baze kjer sta $(p^\ell, q^\ell) = (1, 1)$, $\ell = 0, 1, 2$. Konstrukcija je podobna kot na sliki 28, le da tukaj bazne funkcije iz nivoja ℓ , ki imajo neprazen presek z območjem Ω_ℓ prirežemo.

V klasični definiciji hierarhične baze so bazne funkcije na nivoju ℓ , katerih nosilec je v celoti pokrit z baznimi funkcijami iz nivoja $\ell + 1$ (te dodamo v hierarhično bazo), zamenjane. Enaka zamenjava se zgodi tudi v primeru prirezane baze. Razlika je, da so v tem primeru bazne funkcije, katerih presek z domeno $\Omega^{\ell+1}$ ni prazen, prirezane.

4.3.1 Lastnosti prirezane baze

Veliko lastnosti sledi iz klasičnega primera hierarhične baze. Hitro lahko preverimo naslednje lastnosti.

- Za vsako bazno funkcijo τ , uvedeno na nivoju ℓ , obstaja ena funkcija $\beta \in \mathcal{N}^{\ell+1}$, da

$$\tau = \text{trunc}^{N-1}(\text{trunc}^{N-2} \dots (\text{trunc}^{\ell+1}(\beta) \dots))$$

in

$$\tau|_{\Omega^\ell \setminus \Omega^{\ell+1}} = \beta|_{\Omega^\ell \setminus \Omega^{\ell+1}}$$

- Funkcije v \mathcal{T} so linearno neodvisne.

Dokaz. Dokaz analogen tistemu za lemo 4.2. □

- $\mathcal{L}in T^\ell \subset \mathcal{L}in T^{\ell+1}$

Dokaz. Tudi ta dokaz je zelo podoben dokazu leme 4.3. □

- Število funkcij v \mathcal{T} je enako številu funkcij v \mathcal{K} .

Dokaz. Dokaz leme je precej očiten. Lahko je videti, da število funkcij v T_A^ℓ enako tistemu v K_A^ℓ in enako za T_B^ℓ in K_B^ℓ za vse $\ell = 0, \dots, N-1$. □

Pravo razliko med prirezano in klasično bazo nam podajata naslednja izreka.

Izrek 4.8. *Naj bosta \mathcal{K} in \mathcal{T} bazi, definirani v definicijah 4.1 in 4.7. Potem velja*

$$\mathcal{L}in(\mathcal{T}) = \mathcal{L}in(\mathcal{K}).$$

Dokaz. Dokaza se lotimo z indukcijo na ℓ . Želeli bi pokazati, da velja indukcijski korak

$$\mathcal{L}in K^\ell = \mathcal{L}in T^\ell \Rightarrow \mathcal{L}in K^{\ell+1} = \mathcal{L}in T^{\ell+1}$$

Za $\ell = 0$ sledi direktno iz definicije 4.7. Z upoštevanjem še definicije 4.1, indukcijske predpostavke, ter lastnosti $\mathcal{L}in T^\ell \subseteq \mathcal{L}in T^{\ell+1}$ vidimo, da

$$\mathcal{L}in K_A^{\ell+1} \subseteq \mathcal{L}in K^\ell = \mathcal{L}in T^\ell \subseteq \mathcal{L}in T^{\ell+1}.$$

Definicija 4.1 nam da

$$\mathcal{L}in K_B^{\ell+1} = \mathcal{L}in T_B^{\ell+1} \subseteq \mathcal{L}in T^{\ell+1}.$$

Potem je

$$\mathcal{L}in K_A^{\ell+1} \cup \mathcal{L}in K_B^{\ell+1} \subseteq \mathcal{L}in T^{\ell+1},$$

torej

$$\mathcal{L}in K^{\ell+1} \subseteq \mathcal{L}in T^{\ell+1}.$$

Ker so funkcije obeh baz linearno neodvisne in ker imata obe bazi enako število elementov, lahko sklepamo, da velja

$$\mathcal{L}in K^{\ell+1} = \mathcal{L}in T^{\ell+1}, \text{ za } \ell = 0, 1, \dots, N - 2. \quad \square$$

Izrek 4.9. *Prirezana hierarhična baza \mathcal{T} tvori particijo enote*

$$\sum_{\tau \in T^\ell} \tau = 1 \text{ na } \Omega^0, \quad \ell = 0, 1, \dots, N - 1.$$

Dokaz. Po lastnosti iz poglavja 2.3 vemo, da

$$\sum_{\beta \in \mathcal{N}^\ell} \beta = 1 \text{ na } \Omega^0, \quad \ell = 0, \dots, N - 1. \quad (4.7)$$

Tudi ta izrek lahko dokažemo z indukcijo. Baza indukcije sledi iz enačbe (4.7) za $\ell = 0$. Indukcijski korak

$$\sum_{\tau \in T^\ell} \tau = 1 \text{ na } \Omega^0 \Rightarrow \sum_{\tau \in T^{\ell+1}} \tau = 1 \text{ na } \Omega^0$$

lahko dokažemo z uporabo definicije 4.1 in preurejanjem naslednje vsote

$$\begin{aligned} 1 &= \sum_{\tau \in T^\ell} \tau = \sum_{\tau \in T^\ell} \sum_{\beta \in \mathcal{N}^{\ell+1}} c_\beta^{\ell+1}(\tau) \beta \\ &= \sum_{\tau \in T^\ell} \left(\sum_{\beta \in \mathcal{N}^{\ell+1}, \text{supp} \beta \not\subseteq \Omega^{\ell+1}} c_\beta^{\ell+1}(\tau) \beta + \sum_{\beta \in \mathcal{N}^{\ell+1}, \text{supp} \beta \subseteq \Omega^{\ell+1}} c_\beta^{\ell+1}(\tau) \beta \right) \\ &= \sum_{\tau \in T^\ell} \left(\sum_{\beta \in \mathcal{N}^{\ell+1}, \text{supp} \beta \not\subseteq \Omega^{\ell+1}} c_\beta^{\ell+1}(\tau) \right) + \sum_{\beta \in \mathcal{N}^{\ell+1}, \text{supp} \beta \subseteq \Omega^{\ell+1}} \left(\sum_{\tau \in T^\ell} c_\beta^{\ell+1}(\tau) \right) \beta. \end{aligned} \quad (4.8)$$

Izraz v oklepajih prve vsote zadnje vrstice enačbe (4.8) je enak $\text{trunc}^{\ell+1}(\tau)$. Glede na enačbo (4.7) in menjavo vrstnega reda vsot v prvi vrstici izražave (4.8), dobimo

$$\sum_{\beta \in \mathcal{N}^{\ell+1}} \beta = 1 \quad \text{in tudi} \quad 1 = \sum_{\beta \in \mathcal{N}^{\ell+1}} \left(\sum_{\tau \in T^\ell} c_\beta^{\ell+1}(\tau) \right) \beta.$$

na Ω^0 . Ko primerjamo koeficiente zgornjih vsot in upoštevamo linearno neodvisnost, vidimo, da

$$\sum_{\tau \in T^\ell} c_\beta^{\ell+1}(\tau) = 1,$$

za vse $\beta \in \mathcal{N}^{\ell+1}$ in še posebej za vse β , katerih $\text{supp}\beta \subseteq \Omega^{\ell+1}$. Iz enačbe (4.8) in upoštevanjem definicije 4.7 lahko zaključimo, da je

$$\begin{aligned} 1 &= \sum_{\tau \in T^\ell} \text{trunc}^{\ell+1}(\tau) + \sum_{\beta \in \mathcal{N}^{\ell+1}, \text{supp}\beta \subseteq \Omega^{\ell+1}} \beta \\ &= \sum_{\tau \in T_A^{\ell+1}} \tau + \sum_{\tau \in T_B^{\ell+1}} \tau = \sum_{\tau \in T^{\ell+1}} \tau. \end{aligned} \tag{4.9}$$

□

4.4 HIERARHIČNI ŠKATLASTI ZLEPKI

V tem poglavju bomo zgradili še hierarhično bazo škatlastih zlepkov dveh spremenljivk. Ker so škatlasti zlepki posplošitev B-zlepkov, bomo tudi v tem poglavju začeli z definicijo ugnezenih prostorov in domen in za tem konstruirali hierarhično bazo. Videli bomo, da je koncept hierarhije škatlastih zlepkov podoben tistemu pri B-zlepkih. Za razliko od B-zlepkov, se bomo tukaj omejili na navadno hierarhično bazo, medtem ko lahko bralec več podrobnosti o prirezani hierarhični bazi najde v [10]. Prav tako se pri hierarhičnih škatlastih zlepkih omejimo na tiste ki so definirani na mreži tipa-*I*. Mrež tipa-*II* se izognemo predvsem iz razloga, ker na njih ne moremo avtomatsko zagotoviti linearne neodvisnosti zlepkov iz prostora premikov.

4.4.1 Ugnezdeni prostori in domene škatlastih zlepkov

Preden začnemo z definicijami škatlastih zlepkov, vpeljimo nekaj oznak. Naj bo Δ mreža tipa-*I* in $\Omega \subseteq \mathbb{R}^2$. Dalje naj bo \mathcal{S} prostor premikov škatlastega zlepka dveh spremenljivk, določenega z matriko $M \in \mathbb{Z}^{2 \times n}$. Definiramo prostor

$$\mathcal{S} = \mathcal{L}in \{B_M \in \mathcal{S} \mid \text{supp}B_M \cap \Omega \neq \emptyset\}.$$

Ugnezdено zaporedje mrež tipa-*I* označimo z

$$\Delta^0 \subseteq \Delta^1 \subseteq \dots \subseteq \Delta^{N-1}.$$

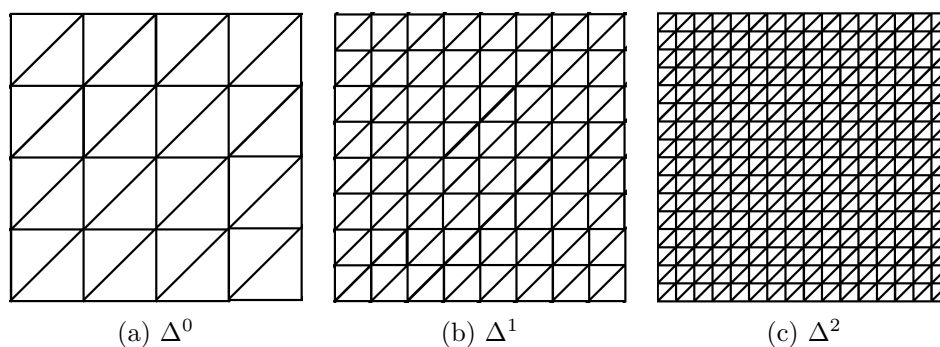
Za vsak $k = 0, 1, \dots, N-1$ dobimo mrežo Δ^{k+1} iz mreže Δ^k tako, da razdelimo vsak trikotnik mreže Δ^k na manjše trikotnike. Primer delitve je prikazan na sliki 31.

Naj bodo

$$S^0 \subseteq S^1 \subseteq \dots \subseteq S^{N-1}$$

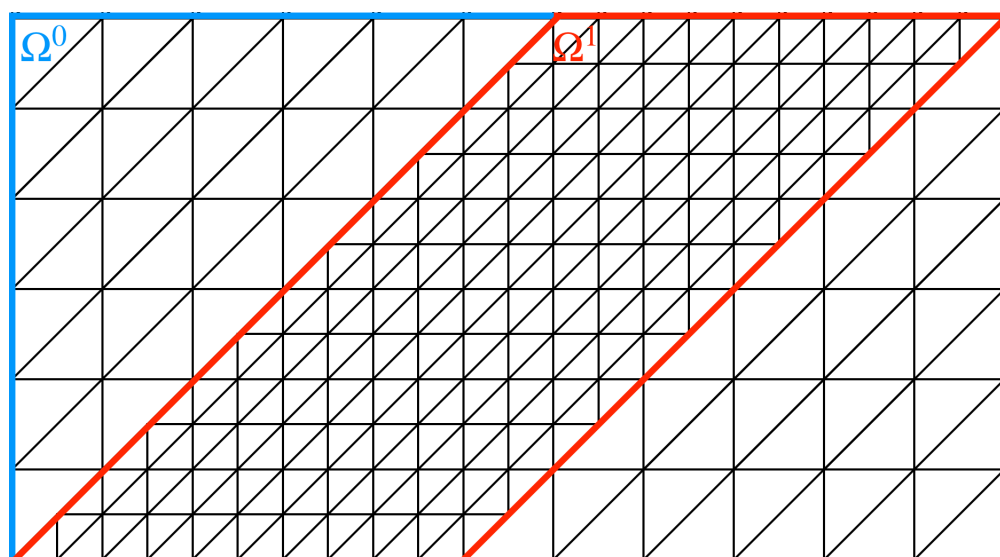
prostori škatlastih zlepkov, definirani na pripadajočih mrežah $\Delta^0, \Delta^1, \dots, \Delta^{N-1}$. Tem mrežam pripadajo tudi ugnezdene domene

$$\Omega_0 \supseteq \Omega^1 \supseteq \dots \supseteq \Omega^{N-1} \supseteq \Omega^N = \emptyset.$$



Slika 31: Primer treh različno gostih mrež tipa- I .

Z ugnezdenimi domenami predstavimo različna območja zgoščevanja v hierarhiji škatlastih zlepkov. Že na tej točki predpostavimo krepki pogoj, analogen tistemu pri B-zlepkih, da je rob k -tega območja $\partial\Omega^k$ poravnan z robovi mreže Ω^{k-1} . Primer dveh ugnezdenih domen najdemo na sliki 32.



Slika 32: Primer gnezdenih domen škatlastih zlepkov na mrežah tipa- I s krepkim robnim pogojem.

4.4.2 Definicija hierarhične baze

V tem delu formalno definiramo hierarhično bazo škatlastih zlepkov. Privzamemo vse oznake in definicije iz poglavja 4.4.1.

Definicija 4.10. Naj bo B_M škatlasti zlepek dveh spremenljivk, določen z matriko M na mreži tipa- I in naj bodo S^k ustrezno skalirani prostori premikov zleпка B_M na pripadajoči domeni Ω^k za $k = 0, 1, \dots, N$. Hierarhično bazo škatlastih zlepkov H definiramo rekurzivno kot:

1. Inicializacija: $H^0 = S^0$.
2. Rekurzivni korak: $H^{\ell+1} = H_A^{\ell+1} \cup H_B^{\ell+1}$, za $\ell = 0, 1, \dots, N-1$, kjer sta

$$H_A^{\ell+1} = \{\beta \in H^\ell : \text{supp } \beta \not\subseteq \Omega^{\ell+1}\}$$

in

$$H_B^{\ell+1} = \{\beta \in S^{\ell+1} : \text{supp } \beta \subseteq \Omega^{\ell+1}\}.$$

3. Zadnji korak: $H = H^N$.

Opomba 4.11. Tudi v primeru škatlastih zlepkov se srečamo s podobnimi problemi kot pri B-zlepkih, zaradi katerih smo potrebovali definicijo prirezane hierarhične baze. Na enak način lahko tudi pri škatlastih zlepkih uvedemo prirezano bazo, kjer na vsakem koraku zlepke $\beta \in H^\ell$, ki imajo neprazen presek nosilca (ampak niso vsebovani) z območjem $\Omega^{\ell+1}$, prirežemo. V sklopu te magistrske naloge se omejimo na navadno hierarhično bazo, za katero predpostavimo krepki pogoj na robovih domen. Še več, za vsak $\Omega^{\ell+1}$, $\ell = 0, 1, \dots, N-1$, predpostavimo, da velja

$$\Omega^{\ell+1} = \bigcup_{\beta \in D} \text{supp } \beta,$$

kjer je $D \subseteq S^\ell$. Predpostavljamo torej, da je domena $\Omega^{\ell+1}$ sestavljena iz nosilcev funkcij iz višjega nivoja. Na ta način močno zmanjšamo prekrivanje baznih funkcij in si enako kot v poglavju 4.2.3 zagotovimo particijo enote. O prirezani hierarhični bazi škatlastih zlepkov bralec najde več informacij v [10].

Z naslednjim primerom demonstrirajmo konstrukcijo preproste hierarhične baze.

Primer 4.12. Naj bo B_M škatlasti zlepek definiran z matriko

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Z drugim zapisom je to zlepek B_{222} , torej zlepek, ki je lokalno polinom stopnje 4. Za ta zlepek definiramo dva ustrezno skalirana prostora premikov in sicer

$$\mathcal{S}_0 = \{B_M(h_0 \mathbf{x} - \mathbf{i})\}, \mathbf{i} \in \mathbb{Z}^2$$

in

$$\mathcal{S}_1 = \{B_M(h_1 \mathbf{x} - \mathbf{i})\}, \mathbf{i} \in \mathbb{Z}^2,$$

kjer sta $h_0 = \frac{1}{4}$ in $h_1 = \frac{1}{8}$. Za prostora \mathcal{S}_0 in \mathcal{S}_1 definiramo prostora

$$S_0 = \{\beta \in \mathcal{S}_0 : \text{supp } \beta \cap \Omega^0 \neq \emptyset\}$$

in

$$S_1 = \{\beta \in \mathcal{S}_1 : \text{supp } \beta \cap \Omega^1 \neq \emptyset\},$$

kjer sta

$$\Omega^0 = [0, 1] \times [0, 1] \text{ in } \Omega^1 = \text{supp } B_M(h_0\mathbf{x}).$$

Ω^0 je torej kvadratno območje, določeno kot tenzorski produkt dveh intervalov, Ω^1 pa je nosilec zlepka prostora S_0 , pri čemer skrajno levo spodnje oglišče postavimo v izhodišče. Hierarhično bazo H konstruiramo v dveh korakih. V koraku inicializacije vzamemo $H^0 = S_0$. V naslednjem koraku pa določimo bazo H^1 , ki pa je po definiciji že H . Zadnji korak grajenja baze je torej sestavljen v dveh delih. V prvem delu odstranimo vse bazne funkcije iz H^0 , katerih nosilec je vsebovan v območju Ω^1 . Takšna je očitno samo ena bazna funkcija in sicer $B_M(h_0\mathbf{x})$, saj smo določili, da je Ω^1 kar njen nosilec. Torej

$$H_A^1 = S_0 \setminus B_M(h_0\mathbf{x}).$$

V drugem delu pa dodamo vse bazne funkcije prostora S_1 , katerih nosilci so v celoti vsebovani v Ω^1 . Eksplicitno zapisano je

$$H_B^1 = \{h_1 B_M(h_0(\mathbf{x} - \mathbf{i})), \mathbf{i} \in Q\},$$

kjer je

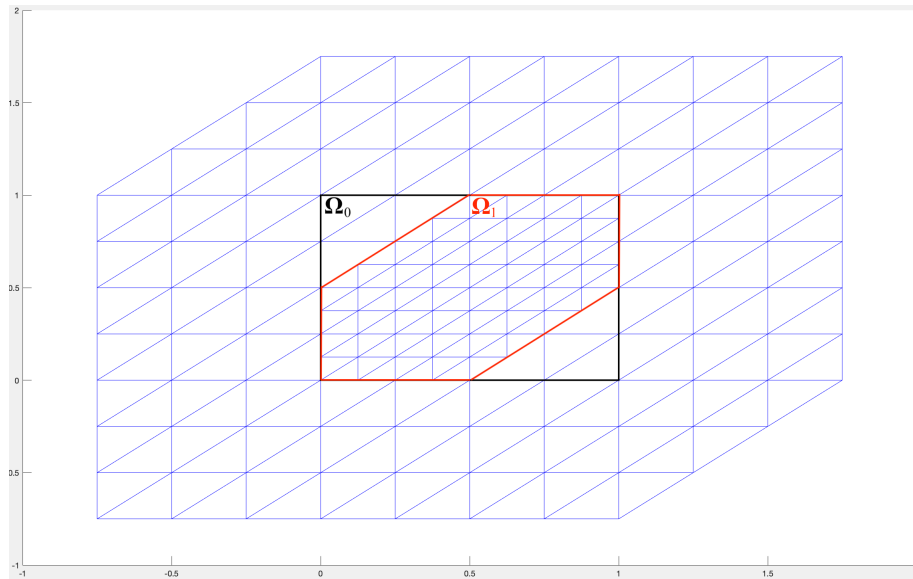
$$Q = \left\{ \begin{array}{l} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \\ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \end{bmatrix} \end{array} \right\}.$$

Hierarhično mrežo z nosilci baznih funkcij vidimo na sliki 33. S črno je označeno območje Ω^0 , z rdečo pa Ω^1 . Na sliki vidimo razliko, da za začetni prostor vzamemo vse zlepke prostora S_0 , ki imajo neprazen presek nosilca z območjem Ω^0 , medtem ko na Ω^1 vzamemo samo tiste zlepke prostora S_1 , ki so v celoti vsebovani v območju Ω^1 .

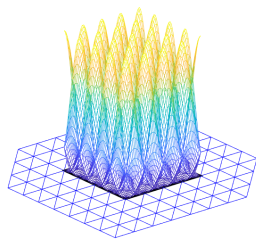
Za konec primera še narišemo vse bazne funkcije. Slika 34 prikazuje prostor H^0 , prostor H_B^1 in prostor H , ko iz prostora S_1 odstranimo ustrezen zlepek in ga nadomestimo z množico H_B^1 .

4.4.3 Lastnosti hierarhične baze škatlastih zlepkov

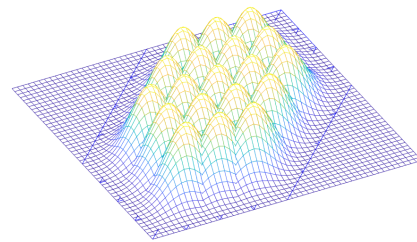
V tem delu zapišimo nekaj lastnosti hierarhičnih škatlastih zlepkov. Opazimo, da imajo hierarhični škatlasti zlepki podobne lastnosti kot (prirežani) hierarhični B-zlepki. Prvi dve lastnosti povzamemo v naslednji lemi.



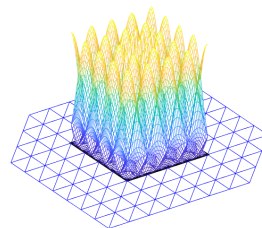
Slika 33: Gnezdeni domeni in nosilci baznih funkcij primera 4.12.



(a) H^0



(b) H_B^1



(c) H

Slika 34: Nekateri koraki grajenja hierarhične baze škatlastih zlepkov iz primera 4.12.

Lema 4.13. Naj bo H hierarhična baza škatlastih zlepkov, definirana z definicijo 4.10.

Potem za vsak bazni zlepek $\beta \in H$ velja

$$\beta(\mathbf{x}) > 0 \quad \text{za } \mathbf{x} \in \text{int}(\text{supp } \beta)$$

in

$$\beta(\mathbf{x}) = 0 \quad \text{povsod drugod.}$$

Dokaz leme opustimo, saj sledi neposredno iz lastnosti navadnih škatlastih zlepkov, definiranih v poglavju 3.1. Lema nam zagotavlja kompakten nosilec vsakega baznega zleпка $\beta \in H$ in nenegativnost le tega na celotnem \mathbb{R}^2 . Naslednja lema nam podaja linearno neodvisnost baznih zlepkov hierarhične baze.

Lema 4.14. *Naj bo H hierarhična baza, definirana kot v definiciji 4.10. Bazni zleпки v H so linearno neodvisni, če za matriko M , ki določa škatlasti zlepek B_M , velja*

$$|\det M| = 1.$$

Dokaz. Po izreku 3.18 so vsi zleпки iz prostorov premikov \mathcal{S}_k , $k = 0, 1, \dots, N$, in s tem tudi zleпки iz prostorov S_k , $k = 0, 1, \dots, N$, linearno neodvisni, če za matriko M , ki določa zlepek B_M , velja $|\det M| = 1$. Želimo pokazati, da so v vsoti

$$\sum_{\beta \in H} c_\beta \beta = 0 \tag{4.10}$$

vsi koeficienti c_β enaki 0. Vsoto (4.10) lahko zapišemo kot

$$\sum_{\beta \in H \cap S_0} c_\beta \beta + \sum_{\beta \in H \cap S_1} c_\beta \beta + \dots + \sum_{\beta \in H \cap S_N} c_\beta \beta.$$

Ker je $H \cap S_0 \subseteq S_0$, so vsi zleпки iz te množice po izreku 3.18 linearno neodvisni in ker so na območju $\Omega^0 \setminus \Omega^1$ neničelni le zleпки iz prostora S_0 lahko sklepamo da je $c_\beta = 0$ za vsak $\beta \in H \cap S_0$. Podobno sklepamo tudi za ostale $k = 1, 2, \dots, N$. Ker za vsak tak k velja $H \cap S_k \subseteq S_k$ so vsi zleпки iz $H \cap S_k$ linearno neodvisni po izreku 3.18. Tudi v tem primeru so na območju $\Omega^k \setminus \Omega^{k+1}$ neničelni le zleпки prostora S_k . Od tod sledi, da je

$$c_\beta = 0 \text{ za vsak } \beta \in H \cap S_k, \quad k = 0, 1, \dots, N.$$

□

Z naslednjo lemo pokažemo, da z gnezdenjem prostorov gnezdimo tudi linearno ogrinjačo baznih zlepkov.

Lema 4.15. *Naj bo H hierarhična baza, definirana z definicijo 4.10. Potem za vsak $k = 0, 1, \dots, N$ velja*

$$\mathcal{L}in(H^k) \subset \mathcal{L}in(H^{k+1}).$$

Dokaz. Vsako funkcijo $f \in \mathcal{L}in(H^k)$ lahko izrazimo kot linearno kombinacijo

$$\begin{aligned} f &= \sum_{\beta \in H^k} c_\beta \beta \\ &= \sum_{\beta \in H^k, \text{ supp } \beta \not\subseteq \Omega^{k+1}} c_\beta \beta + \sum_{\beta \in H^k, \text{ supp } \beta \subseteq \Omega^{k+1}} c_\beta \beta. \end{aligned} \tag{4.11}$$

Prva vsota v drugi vrstici enačbe (4.11) je po definiciji 4.10 ravno množica H_A^{k+1} . Vsak bazni zlepek $\beta \in S_k$ lahko vedno zapišemo kot linearno kombinacijo baznih zlepkov finejšega prostora S^{k+1} . Torej lahko drugo vsoto zadnje vrstice enačbe (4.11) zapišemo kot

$$\begin{aligned} \sum_{\beta \in H^k, \text{supp } \beta \subseteq \Omega^{k+1}} c_\beta \beta &= \sum_{\beta \in H^k, \text{supp } \beta \subseteq \Omega^{k+1}} c_\beta \left(\sum_{\alpha \in S^{k+1}, \text{supp } \alpha \subseteq \Omega^{k+1}} c_\alpha \alpha \right) \\ &= \sum_{\alpha \in S^{k+1}, \text{supp } \alpha \subseteq \Omega^{k+1}} \left(\sum_{\beta \in H^k, \text{supp } \beta \subseteq \Omega^{k+1}} c_\alpha c_\beta \right) \alpha. \end{aligned} \quad (4.12)$$

Na ta način smo bazne funkcije prostora S_k , katerih nosilci so vsebovani v Ω^{k+1} , zamenjali z baznimi funkcijami prostora S_{k+1} , katerih nosilec je vsebovan na Ω^{k+1} . Po definiciji pa je to ravno množica H_B^{k+1} . Začetno enačbo lahko torej zapišemo kot

$$\begin{aligned} f &= \sum_{\beta \in H_A^{k+1}} c_\beta \beta + \sum_{\alpha \in H_B^{k+1}} c_\alpha \alpha \Rightarrow \\ &f \in \mathcal{L}in(H^{k+1}). \end{aligned}$$

□

5 IMPLEMENTACIJA

V tem poglavju so predstavljene nekatere osnovne ideje implementacije teorije iz prejšnjih poglavij v programskem jeziku Matlab. Poglavje se v grobem deli na dva dela, in sicer na B-zlepke in škatlaste zlepke. V obeh delih si najprej pogledamo že obstoječo podlago v Matlabu, ki nam pomaga pri implementaciji. Za tem predstavimo potrebne algoritme in prijeme, s pomočjo katerih konstruiramo elemente hierarhične baze obeh vrst zlepkov. To poglavje je namenjeno bolj splošnemu opisu programerskih prijemov, s pomočjo katerih bralec lažje razume vse tehnične podrobnosti programske kode v prilogah. Vse tehnične podrobnosti uporabljenih Matlab funkcij najdemo v [6, 17], prikaz delovanja in uporabo implementiranih orodij pa v poglavju 9. Videli bomo, da se z implementacijo nekoliko odmaknemo od vse splošnosti teorije prejšnjih poglavij.

5.1 B-ZLEPKI

Poglejmo si, kako v Matlabu konstruiramo navadno in prirezano hierarhično bazo prostora tenzorskih produktov B-zlepkov. Najprej si bomo pogledali nekatere že vgrajene funkcije v Matlabu iz programskega paketa *Curve fitting toolbox* in videli, kako jih lahko uporabimo pri implementaciji baznih funkcij. Za tem spoznamo algoritem *Vstavi vozle* in algoritem za prirez bazne funkcije B-zlepka. Na koncu bomo zapisali še preglednejši algoritem za konstrukcijo (prirežane) hierarhične baze.

5.1.1 Matlab podlaga

Pri implementaciji nam je v veliko pomoč Matlabov paket *Curve fitting toolbox*. Za naše potrebe je najpomembnejša stvar iz tega paketa funkcija `spmak(knots, coefs)`. Ta funkcija kot argument sprejme vektor vozlov oz. vektorje vozlov kadar delamo v višjih dimenzijah in kontrolne točke. Rezultat funkcije `spmak` je struktura, ki predstavlja krivuljo/ploskev B-zlepkov. S to strukturo lahko s pomočjo drugih že vgrajenih funkcij implementiran B-zlepek narišemo, odvajamo, integriramo, izračunamo točko na krivulji oz. ploskvi, itd. Za začetek si poglejmo konstrukcijo in manipulacijo preproste enodimenzionalne krivulje.

Primer 5.1. Naj bo vektor $U = (0, 1, 2, 3, 4, 5)$ vektor vozlov in $P = (-0.3, 0.87, -1)$.

Stopnja te krivulje je torej enaka 2. V Matlabu želimo konstruirati krivuljo

$$f(x) = \sum_{i=0}^2 P_i \cdot N_{i,2}(x).$$

To naredimo z naslednjimi ukazi:

```
>>U = [0 1 2 3 4 5]
>>P = [-0.3 0.87 -1]
>>sp = spmak(U,P)
sp =
struct with fields:
form: 'B-'
knots: [0 1 2 3 4 5]
coefs: [-0.3 0.87 -1]
number: 3
order: 3
dim: 1
```

V zgornji kodi vidimo, katere pomembne podatke o zlepku hrani struktura `spmak`. Vidimo, da struktura pridobljena z ukazom `spmak` ne hrani podatka o stopnji krivulje, ampak podatek o redu krivulje/ploskve, ki je definiran kot $p+1$, kjer je p stopnja krivulje/ploskve.

Da dobimo vrednost točke na krivulji uporabimo ukaz `fnval`. Izračunamo lahko vrednost ene ali več točk na krivulji:

```
>>fnval(sp, [1 2 3])
ans =
-0.1500 0.2850 -0.0650
```

Izris krivulje na sliki 35 dobimo z ukazom:

```
>>fnplt(sp)
```

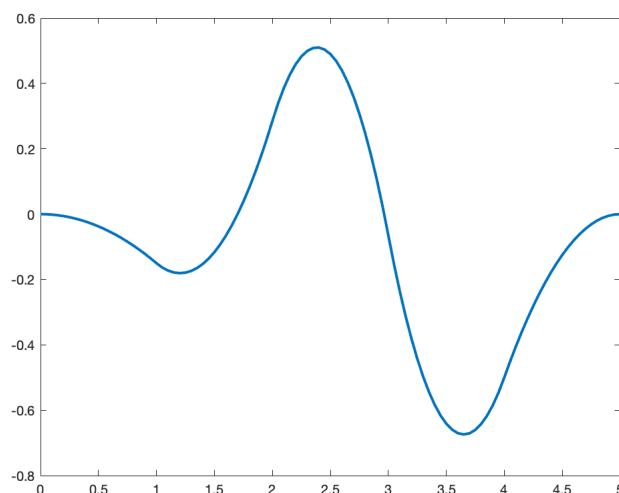
Krivuljo lahko odvajamo ali integriramo. To naredimo z ukazom `fnder(sp, dorder)`, kjer je `sp` B-zlepek dobljen z ukazom `spmak`, `dorder` pa stopnja odvoda. Če želimo prvi odvod zleпка uporabimo ukaz

```
>>fnder(sp, 1);
```

Z istim ukazom lahko krivuljo tudi integriramo. V tem primeru moramo uporabiti negativno stopnjo odvoda. Nedoločen integral $\int f(x)dx$ dobimo torej z ukazom

```
>>fnder(sp, -1);
```

Za konec primera omenimo še ukaz `fnbrk(sp, part1, ..., part2)`. Ta funkcije vrne



Slika 35: Krivulja B-zlepkov ene spremenljivke primera 5.1.

pomembne lastnosti strukture `sp`. Z ukazom

```
>>[U, P] = fnbrk(sp, 'knots', 'coeffs');
```

dobimo vektor vozlov U in kontrolne točke P krivulje B-zlepkov predstavljene s `sp`.

Za naše potrebe je pomembno, da lahko konstruiramo bazne funkcije prostora B-zlepkov. Tudi to vedno naredimo s funkcijo `spmak`. Naslednji primer prikazuje konstrukcijo baze B-zlepkov ene spremenljivke.

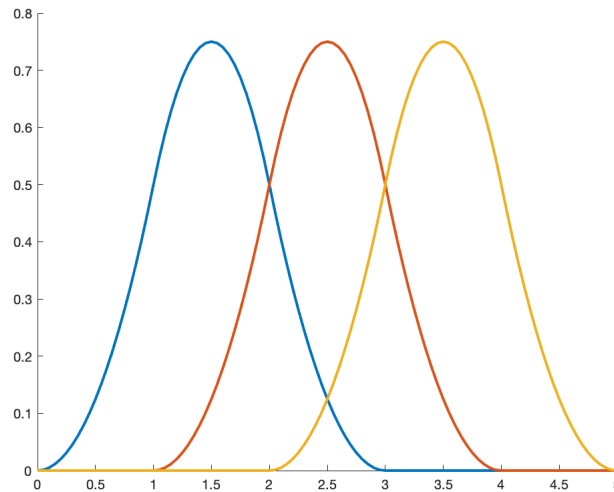
Primer 5.2. Naj bo zopet $U = (0, 1, 2, 3, 4, 5)$ vektor vozlov in $p = 2$ polinomska stopnja. Potem je baza prostora $\mathbb{S}_{p,U}$ enaka

$$\mathcal{N} = \{N_{0,2}, N_{1,2}, N_{2,2}\}.$$

V Matlabu bazne funkcije baze \mathcal{N} na najbolj osnoven in ekspliciten način predstavimo z ukazi

```
>>U = [0,1,2,3,4,5];
>>P0 = [1 0 0];
>>P1 = [0 1 0];
>>P2 = [0 0 1];
>>sp0 = spmak(U, P0);
>>sp1 = spmak(U, P1);
>>sp2 = spmak(U, P2);
```

Vse tri bazne funkcije prikazuje slika 36.



Slika 36: Bazne funkcije prostora $\mathbb{S}_{2,U}$, kjer je $U = (0, 1, 2, 3, 4, 5)$.

Z istimi ukazi lahko implementiramo in uporabljamo tudi B-zlepke dveh spremenljivk. V nadaljevanju bomo spoznali intuitiven način implementacije, pri katerem imamo ves čas v mislih, da so B-zlepki dveh spremenljivk tenzorski produkti B-zlepkov ene spremenljivke. Najprej definirajmo poseben produkt dveh vektorjev. Ta produkt nam bo prišel prav pri implementaciji kontrolnih mrež zlepkov dveh spremenljivk.

Definicija 5.3. Naj bo $\mathbf{u} \in \mathbb{R}^{1 \times m}$ in $\mathbf{v} \in \mathbb{R}^{n \times 1}$, kjer sta

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}$$

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix},$$

Za vektorja u, v definiramo poseben produkt \circ , katerega rezultat je matrika velikosti $m \times n$, pridobljena na način

$$\mathbf{u} \circ \mathbf{v} = \begin{bmatrix} u_1 v & u_2 v & \dots & u_m v \end{bmatrix}.$$

Malo drugačno definicijo funkcije iz prostora tenzorskih produktov B-zlepkov nam podaja definicija 5.4.

Definicija 5.4. Naj bosta $U = (u_0, u_1, \dots, u_m)$ in $V = (v_0, v_1, \dots, v_n)$ vektorja vozlov in $p, q \in \mathbb{N}$ polinomski stopnji. Potem lahko vsako funkcijo $f \in \mathbb{S}_{U,V}^{p,q}$ zapišemo kot

$$f(u, v) = \sum_{i=0}^{m+p} \sum_{j=0}^{n+q} A_{i,j} N_i^p(u) N_j^q(v),$$

kjer sta $N_i^p \in \mathbb{S}_{p,U}$ in $N_j^q \in \mathbb{S}_{q,V}$ s pripadajočima vrstičnima vektorjema kontrolnih točk \mathbf{P} in \mathbf{Q} . Matrika $A \in \mathbb{R}^{m \times n}$ pa je določena kot

$$A = \mathbf{Q} \circ \mathbf{P}^T.$$

Lema 5.5. *Definicija ploskve iz prostora tenzorskih produktov B-zlepkov $\mathbb{S}_{U,V}^{p,q}$, definirana v definiciji 5.4, je ekvivalentna definiciji v poglavju 2.3.*

Dokaz. Vse kar moramo pokazati je, da velja

$$A_{ij} = P_i \cdot Q_j.$$

Če to velja, dobimo enako definicijo poljubne funkcije prostora $\mathbb{S}_{U,V}^{p,q}$, kot v poglavju 2.3. Eksplicitno zapišimo vektorja kontrolnih točk P in Q .

$$\mathbf{P} = [P_0, P_1, \dots, P_{m-p}]$$

$$\mathbf{Q} = [Q_0, Q_1, \dots, Q_{n-q}]$$

Potem je produkt $\mathbf{Q} \circ \mathbf{P}^T$ enak

$$\left[Q_0 \cdot \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{m-p} \end{bmatrix} \quad Q_1 \cdot \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{m-p} \end{bmatrix} \quad \cdots \quad Q_{n-q} \cdot \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{m-p} \end{bmatrix} \right].$$

Do konca izračunana matrika je potem enaka

$$\mathbf{Q} \circ \mathbf{P}^T = \begin{bmatrix} P_0 Q_0 & P_0 Q_1 & \cdots & P_0 Q_{n-q} \\ P_1 Q_0 & P_1 Q_1 & \cdots & P_1 Q_{n-q} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m-p} Q_0 & P_{m-p} Q_1 & \cdots & P_{m-p} Q_{n-q} \end{bmatrix}. \quad (5.1)$$

Iz izraza (5.1) je očitno, da velja

$$A_{i,j} = (\mathbf{Q} \circ \mathbf{P}^T)_{i,j} = P_i Q_j. \quad \square$$

Pripravljeno imamo vse za implementacijo tenzorskih produktov B-zlepkov v Matlabu. Poglejmo si naslednji preprost primer.

Primer 5.6. Naj bosta $U = V = (0, 1, 2, 3, 4)$ vektorja vozlov ter $p = q = 2$ polinomski stopnji. V Matlabu želimo konstruirati kakšno bazno funkcijo prostora $\mathbb{S}_{U,V}^{p,q}$. Naj bo ta funkcija $N_{1,0}(u, v)$. Enodimenzionalna zlepka, ki določata tenzorski produkt te funkcije

sta določena z istima vektorjema vozlov U, V in kontrolnimi točkami $\mathbf{P} = (0, 1)$ ter $\mathbf{Q} = (1, 0)$. Potem je matrika, ki določa kontrolne točke tenzorskega produkta enaka

$$A = \mathbf{Q} \circ \mathbf{P}^T = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

V Matlabu lahko to zapišemo z naslednjimi ukazi

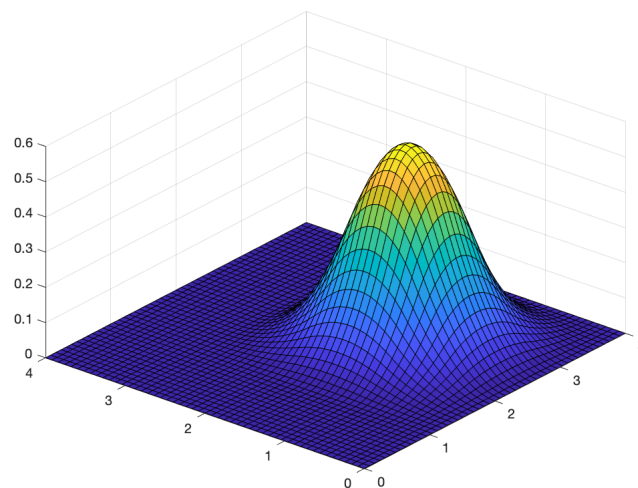
```
>>U=[0 1 2 3 4]; V=[0 1 2 3 4];
```

```
>>P=[0 1]; Q=[1 0];
```

```
>>A=Q .* P';
```

```
>>bsp = spmak({U,V},A);
```

Z ukazom `fnplt(bsp)` narišemo našo bazno funkcijo. Prikaz na sliki 37.



Slika 37: Bazna funkcija $N_{1,0}$ prostora $\mathbb{S}_{U,V}^{2,2}$.

Poglejmo še primer, kako s podano bazo tenzorskih produktov B-zlepkov in podanimi kontrolnimi točkami sestavimo krivuljo in z njo operiramo.

Primer 5.7. Naj bo \mathcal{B} baza prostora $\mathbb{S}_{U,V}^{p,q}$, kjer je ponovno $U = V = (0, 1, 2, 3, 4)$ in $p = q = 2$. Predpostavimo, da smo bazo uspešno implementirali v Matlabu in jo shranili v podatkovno strukturo *cell array* z imenom `baza`. Podana naj bo tudi matrika kontrolnih točk

$$A = \begin{bmatrix} -1.2 & 0.18 \\ 2 & -0.3 \end{bmatrix}.$$

Naslednji ukazi prikazujejo, kako v Matlabu iz podanih stvari učinkovito konstruiramo funkcijo, podano s kontrolnimi točkami, kot linearno kombinacijo baznih funkcij.

```
>>A = [-1.2 0.18; 2 -0.3];
```

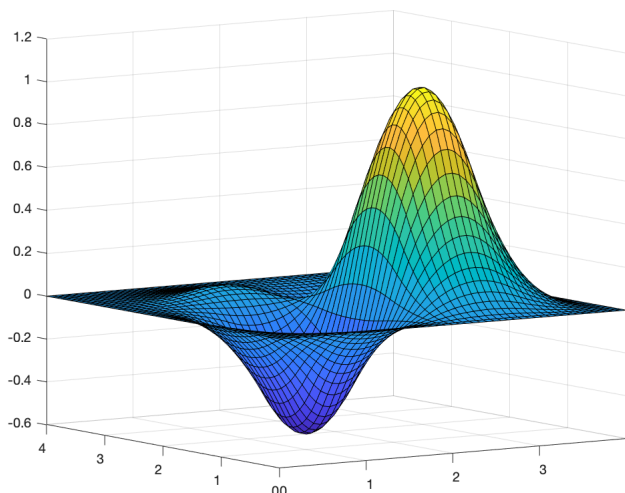
```
>>A = reshape(A,[1 numel(A)]); % Matriko preoblikujemo v en sam vektor.
```

```

>>M = zeros(4); % Tako velike so v našem primeru matrike kontrolnih točk.
>>for i=1:length(baza)
[coef, velikost] = fnbrk(bsp, 'coeff', 'number');
M = M + A(i) * reshape(coef, velikost);
end
>> bsp = spmak({U,V}, M );

```

Vidimo torej, da tudi če je baza podana kot seznam baznih funkcij, tj. struktur narejenih z ukazom `spmak`, lahko krivuljo, ki je linearna kombinacija le-teh, predstavimo z eno samo strukturo, ki je tudi generirana z ukazom `spmak`. Ploskev tega primera prikazuje slika 38.



Slika 38: Ploskev primera 5.7.

Za konec pogledjmo le še, kako dobimo gradient kakšnega zleпка. Predpostavimo, da je `bsp` implementacija neke krivulje iz prostora $\mathbb{S}_{U,V}^{p,q}$. Gradient preprosto izračunamo z naslednjimi ukazi:

```

>>ddx = fnder(bsp, [1 0])
>>ddy = fnder(bsp, [0 1])
>>grad = [ddx ; ddy]

```

5.1.2 Zapis v finejši bazi

Pri implementaciji hierarhične baze in še posebej pri implementaciji prirezane hierarhične baze, bomo morali določen zlepek, definiran nad vektorjem vozlov $U = (u_0, u_1, \dots, u_m)$, zapisati v dvakrat finejši bazi. To pomeni, da mora zlepek ohraniti stopnjo in imeti enako obliko, le zapisan mora biti nad vektorjem vozlov

$$\hat{U} = \left(u_0, \frac{u_0 + u_1}{2}, u_1, \dots, \frac{u_{m-1} + u_m}{2}, u_m \right).$$

Da bi to lahko storili, moramo najprej spoznati algoritem "Vstavi vozle". S tem algoritmom vstavimo poljuben vozle t v obstoječ vektor vozlov U , tako da ne spremenimo oblike krivulje. Zaradi identitete $m = n + p + 1$ je očitno, da če povečamo vektor vozlov, moramo povečati tudi število kontrolnih točk. Očitno pa je tudi, da bomo morali nekatere kontrolne točke spremeniti. Kar naredi algoritem učinkovit je dejstvo, da točke modificiramo le lokalno. Kot bomo videli v algoritmu 1, upoštevamo le p kontrolnih točk in sicer $P_{k-p}, P_{k-p+1}, \dots, P_k$. Dokaz pravilnosti algoritma najdemo v [16].

Algoritem 1: Vstavi vozle

Vhod: vektor vozlov U , kontrolne točke \mathbf{P} , nov vozle t .

Izhod: nov vektor vektor vozlov \hat{U} , nove kontrolne točke $\hat{\mathbf{P}}$.

- 1 Poišči $k \in \{0, 1, \dots, m\}$, tako da $t \in [u_k, u_{k+1})$.
 - 2 **za** $i = k - p, \dots, k$
 - 3 $Q_i = (1 - a_i)P_{i-1} + a_iP_i;$
 - 4 $a_i = \frac{t - u_i}{u_{i+p} - u_i};$
 - 5 $\hat{U} = (u_0, u_1, \dots, u_k, t, u_{k+1}, \dots, u_m);$
 - 6 $\hat{\mathbf{P}} = (P_0, P_1, \dots, P_{k-p}, Q_{k-p+1}, \dots, Q_k, P_k, \dots, P_n);$
 - 7 **vrni** $\hat{U}, \hat{\mathbf{P}};$
-

Primer 5.8. Naj bo $U = (0, 1, 2, 3, 4, 5)$ in $\mathbf{P} = (0, 1, 0)$. Potem je to bazna funkcija $N_{0,2}$ prostora $\mathbb{S}_{2,U}$. V vektor vozlov vstavimo nov vozle $t = \frac{5}{2}$. Hitro vidimo, da

$$t \in [2, 3) = [u_2, u_3) \Rightarrow k = 2.$$

Delamo torej s kontrolnimi točkami

$$P_0 = 0, P_1 = 1, P_2 = 2.$$

Izračunamo novi kontrolni točki

$$\begin{aligned} Q_1 &= (1 - a_1)P_0 + a_1P_1 \\ a_1 &= \frac{\frac{5}{2} - 1}{3 - 1} = \frac{3}{4} \Rightarrow \\ Q_1 &= \frac{3}{4}. \end{aligned}$$

in

$$\begin{aligned} Q_2 &= (1 - a_2)P_1 + a_2P_2 \\ a_2 &= \frac{\frac{5}{2} - 2}{4 - 2} = \frac{1}{4} \Rightarrow \\ Q_2 &= \frac{3}{4}. \end{aligned}$$

Nov vektor vozlov je potem enak $\hat{U} = (0, 1, 2, \frac{5}{2}, 3, 4, 5)$, nove kontrolne točke pa so enake $\hat{\mathbf{P}} = (0, \frac{3}{4}, \frac{3}{4}, 0)$.

Algoritem "Vstavi vozle" lahko uporabimo večkrat zaporedoma in tako zapišemo poljuben zlepek nad vektorjem vozlov U , v dvakrat finejši bazi \hat{U} .

5.1.3 Prirez zleпка

Prirez B-zleпка nam podaja definicija 4.6. V tem podpoglavju bomo konstruirali bolj pregleden algoritem te definicije. V grobem je ideja izbran bazni zlepek zapisati kot linearno kombinacijo baznih zlepkov dvakrat finejše baze in kontrolne točke vseh baznih zlepkov, katerih nosilec je vsebovan v območju, glede na katerega prirezujemo, postaviti na 0. Območje, glede na katerega prirezujemo, označimo z Ω . Predpostavimo, da je Ω pravokotno območje v \mathbb{R}^2 , torej $\Omega = [a, b] \times [c, d]$, $a, b, c, d \in \mathbb{R}$. Predpostavimo še omejitve, da so bazne funkcije tenzorskih produktov B-zlepkov, definirane na območju Ω , enake stopnje kot zlepek, ki ga prirezujemo z dvakrat finejšim vektorjem vozlov. Algoritem za tenzorske produkte B-zlepkov je povzet v tabeli 1. Primer z eno spremenljivko je analogen primeru z dvema.

Tabela 1: Algoritem prireza tenzorskega produkt B-zlepkov.

Algoritem PRIREZ:

Podatka: zlepek $B \in \mathbb{S}_{U,V}^{p,q}$, območje $\Omega = [a, b] \times [c, d]$, $a, b, c, d \in \mathbb{R}$.

1. Zlepek B zapišemo v dvakrat finejši bazi in ga označimo z \hat{B} . Temu zlepku pripadata dvakrat finejša vektorja vozlov \hat{U} in \hat{V} , ter matrika kontrolnih točk \hat{A} .
2. Ker za ustrezna $\hat{\mathbf{P}}$ in $\hat{\mathbf{Q}}$ velja, $\hat{A} = \hat{\mathbf{Q}} \circ \hat{\mathbf{P}}^T$, koeficient $\hat{A}_{i,j}$ določa bazni zlepek

$$N_{i,j}^{p,q}(u, v) = \hat{A}_{i,j} N_{i,p}(u) N_{j,q}(v) = \hat{P}_i \hat{Q}_j N_{i,p}(u) N_{j,q}(v),$$

z nosilcema

$$\begin{aligned} \text{supp}(N_{i,p}) &= [\hat{u}_i, \hat{u}_{i+p+1}) \\ \text{supp}(N_{j,q}) &= [\hat{v}_j, \hat{v}_{j+q+1}) \end{aligned}$$

prirežemo zlepek B , tako da za vsak $i = 1, 2, \dots, |\hat{\mathbf{P}}|$ in vsak $j = 1, 2, \dots, |\hat{\mathbf{Q}}|$ popravimo elemente matrike \hat{A} po naslednjem pravilu:

$$\hat{A}_{i,j} = \begin{cases} \hat{A}_{i,j}; & [\hat{u}_i, \hat{u}_{i+p+1}) \not\subseteq [a, b] \wedge [\hat{v}_j, \hat{v}_{j+q+1}) \not\subseteq [c, d] \\ 0; & \text{sicer.} \end{cases}$$

3. Vrni zlepek določen z vektorjema vozlov \hat{U} , \hat{V} in matriko \hat{A} .

Opomba 5.9. Algoritem, opisan v tabeli 1, implicira kvadratično časovno zahtevnost. V

praksi algoritem pohitrino na način, da vnaprej poiščemo indekse neničelnih kontrolnih točk in potem gledamo vsebovanost nosilcev baznih funkcij samo pri teh točkah.

5.1.4 Hierarhična baza B-zlepkov

V tem delu bolj podrobno opišemo implementacijo navadne in prirezane hierarhične baze iz definicij 4.1 in 4.7. Obe bazi združimo v en algoritem, saj sta si definiciji obeh baz precej podobni. Pri implementaciji sprejmemo določene omejitve. Predpostavimo, da je vsako območje Ω v gnezdenem zaporedju domen pravokotno območje tj. tenzorski produkt dveh intervalov $\Omega = [a, b] \times [c, d]$. Predpostavimo tudi, da se stopnji na vseh nivojih ohranjata in da je vektor vozlov na območju $\Omega^{\ell+1}$ dvakrat finejši v primerjavi z vektorjem vozlov na območju Ω^ℓ . Torej, če je

$$U^\ell = (u_0, u_1, \dots, u_m),$$

je

$$U^{\ell+1} = \left(u_0, \frac{u_0 + u_1}{2}, u_1, \dots, \frac{u_{m-1} + u_m}{2} \right).$$

Predpostavimo tudi, da se robni vozli v začetnih vektorjih vozlov ujemaajo z robovi prve Ω^0 . Če sta

$$U^0 = (u_0, u_1, \dots, u_m), \quad V^0 = (v_0, v_1, \dots, v_n),$$

za $\Omega^0 = [a, b] \times [c, d]$, velja

$$a = u_0, b = u_m, \quad c = v_0, d = v_n.$$

Algoritem 2 je zapisan za konstrukcijo baze tenzorskih produktov B-zlepkov. Za primer v eni dimeziji uporabimo enak algoritem.

Za povečanje učinkovitosti algoritma 2 navedimo dve bolj tehnični opombi pri implementaciji v jeziku Matlab.

Opomba 5.10. Zaradi omejitve, da so vsa območja zgoščevanja v hierarhični bazi pravokotna, lahko razvrščanje baznih zlepkov v tri množice v drugem koraku nekoliko pohitrino. V jeziku Matlab velja splošno pravilo, da vedno poskušamo nadomestiti zanke z matričnimi operacijami. Zato bo hitreje, sploh v primerih, ko hierarhična baza T vsebuje že veliko zlepkov, kot se sprehoditi čez vse funkcije in za vsako posebej določiti kam spada, se še vedno sprehoditi čez vse zlepke in samo izluščiti njihove nosilce, ki so prav tako pravokotna območja v \mathbb{R}^2 , in jih shraniti v dolgo matriko. Naj bodo $[x_i, x_{i+p+1}] \times [y_j, y_{j+q+1}]$, za vse $i = 0, \dots, m$, $j = 0, \dots, n$ nosilci vseh baznih zlepkov baze T . Nosilce potem shranimo v matriko

$$A = \begin{bmatrix} x_i & x_{i+p+1} & y_j & y_{j+q+1} \end{bmatrix}, \quad i = 0, \dots, m, \quad j = 0, \dots, n. \quad (5.3)$$

Tabela 2: Algoritem konstrukcije (prirežane) hierarhične baze tenzorskih produktov B-zlepkov.

Algoritem HIERAHIČNA BAZA:

Podatki: Zaporedje gnezdenih domen $\{\Omega^\ell\}_{\ell=0,1,\dots,N}$, začetna vektorja vozlov U^0 in V^0 , stopnji p, q .

1. Z začetnima vektorjema vozlov skonstruiraj navadno bazo tenzorskih produktov B-zlepkov za prostor $\mathbb{S}_{U^0, V^0}^{p,q}$. Bazo označi s T .
2. Sprehodi se čez bazo T in razvrsti bazne zlepke v tri množice.

$$\begin{aligned} T_A &= \{\beta \in T; \beta \not\subseteq \Omega^1\} \\ T_B &= \{\beta \in T; \beta \subseteq \Omega^1\} \\ T_C &= \{\beta \in T; \beta \not\subseteq \Omega^1 \wedge \beta \cap \Omega^1 \neq \emptyset\} \end{aligned} \tag{5.2}$$

3. Bazo T preuredi na naslednji način. Vse zlepke iz T_A pusti takšne kot so, vse zlepke iz T_B izbriši. Če delaš prirezano bazo, zlepke iz množice T_C prireži tako kot v algoritmu 1 glede na Ω^1 , če pa delaš navadno bazo pa tudi zlepke iz T_C pusti takšne kot so.
4. Na območju Ω^1 ustvari novo bazo za prostor navadnih tenzorskih produktov B-zlepkov z dvakrat finejšima vektorjema vozlov in dodaj te zlepke v bazo T .
5. Ponovi korake 2 - 4, za vse ostale Ω^ℓ , $\ell = 2, 3, \dots, N$.
6. Vrni bazo T .

Implementacija takšne matrike je v programskem jeziku Matlab trivialna. Ker so vsa območja v hierarhiji pravokotna, lahko izpeljemo preproste pogoje, kdaj je zlepek v celoti vsebovan in kdaj je presek z območjem prazna množica. Kakšno operacijo več zahteva preverjanje, kdaj je bazni zlepek v množici T_C , torej v množici, ko ni v celoti vsebovan ampak presek tudi ni prazen. Dobra stvar v Matlabu je, da lahko za takšno matriko določimo, kateri zleпки pripadajo določeni množici za vse zlepke naenkrat. Zato najprej preverimo, kateri so v T_A , zatem kateri v T_B in kar nam ostane zagotovo pripada množici T_C .

Opomba 5.11. Pri imeplementaciji hierarhične baze strmimo k temu, da na koncu krivuljo, ki je linearna kombinacija baznih zlepkov iz hierarhične baze, zapišemo s strukturo **spmak**. Seveda krivulje, ki je določena z baznimi zleпки iz hierarhične baze, ne bo moč zapisati samo z eno strukturo, saj delamo z različno velikimi vektorji vozlov in

kontrolnimi matrikami. Največ, kar lahko naredimo je, da krivuljo zapišemo s toliko strukturami spmak , koliko je nivojev v hierarhiji. Zato vsako bazno funkcijo na vsakem nivoju določimo z vektorjem vozlov, ki izgleda, kot če bi bil zapisan čez največje območje Ω^0 . Na ta način si zagotovimo v hierarhični bazi z N nivoji, natanko N različnih vektorjev vozlov.

5.2 ŠKATLASTI ZLEPKI

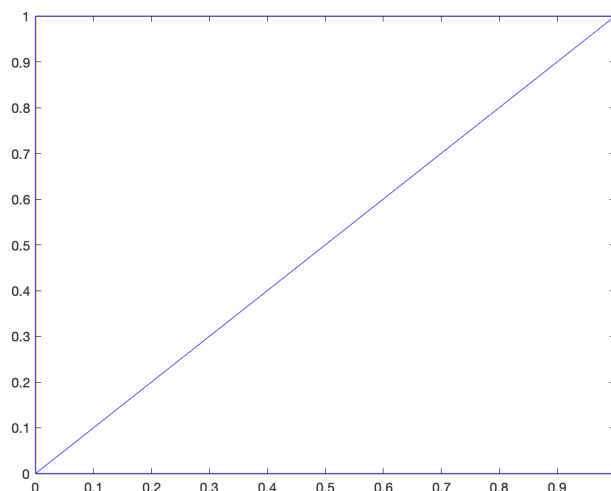
Tudi hierarhično bazo prostora premikov škatlastih zlepkov implementiramo v okolju Matlab. Implementacija te vrste zlepkov je nekoliko drugačna od tiste z B-zlepki. Tudi tukaj si najprej pogledamo Matlab podlago, ki nam omogoča lažjo implementacijo. Za razliko od B-zlepkov Matlab nima že vgrajenih orodij za delo s škatlastimi zlepki, zato jih moramo implementirati sami. V veliko pomoč pri tem so nam že vgrajene funkcije za delo s triangulacijami in baricentričnimi koordinatami iz paketa orodij *Triangulation Representations*. Tudi v primeru škatlastih zlepkov se bomo nekoliko oddaljili od vse splošnosti teorije škatlastih zlepkov iz poglavij in 4. Ogleдали si bomo implementacijo zlepka B_{222} iz definicije 3.7, zatem si bomo ogledali implementacijo prostora premikov določenega z zlepkom B_{222} in na koncu še konstrukcijo hierarhične baze prostorov premikov določenih s tem zlepkom.

5.2.1 Matlab podlaga

Implementacija škatlastih zlepkov zahteva delo s triangulacijami in baricentričnimi koordinatami. Na srečo ravno za te namene obstajajo nekatere že vgrajene matlab funkcije, ki olajšajo in optimizirajo delovanja funkcij v prilogah I,J,K,L,M,O,Q,S. Prva pomembna funkcija je `triangulation(T,P)`. Ta funkcija kot argument sprejme matriko T , velikosti $n \times 3$, kjer vsaka vrstica pove, na katerih treh ogliščih stoji i -ti trikotnik v triangulaciji. Poleg matrike T , funkcija kot argument sprejme tudi matriko P , velikosti $m \times 2$, s katero določimo m oglišč v triangulaciji. V vsaki vrstici sta podani kartezični koordinati i -tega oglišča. Trikotniki v strukturi `triangulation` so indeksirani po vrsticah matrike T .

Primer 5.12. Recimo, da želimo v okolju Matlab implementirati triangulacijo na sliki 39. V okolju Matlab naredimo to na način, da najprej definiramo matriki

```
>>P = [0 0; ...
      1 0; ...
      0 1; ...
      1 1];
```



Slika 39: Triangulacija primera 5.12.

```
>>T = [1 2 4;...
       3 1 4];
>>TR = triangulation(T,P)
TR =
triangulation with properties:
  Points: [4x2 double]
  ConnectivityList: [2x3 double].
```

Spremenljivka TR je torej struktura tipa `triangulation`, iz katere lahko vedno dobimo podatka o vozliščih in povezavah te triangulacije. Triangulacijo lahko narišemo z ukazom

```
>>triplot(TR)
```

in dobimo sliko 39.

Naslednja pomembna že vgrajena funkcija je `pointLocation(TR, QP)`. Argumenta funkcije sta spremenljivka TR, ki je struktura, pridobljena z ukazom `triangulation` in QP, ki je matrika velikosti $n \times 2$, kjer so v vsaki vrstici kartezične koordinate točk v ravnini. Funkcija vrne dve vrednosti. Za vsako točko vrne indeks trikotnika iz triangulacije TR, v katerem se točka nahaja, in baricentrične koordinate točke glede na trikotnik iz triangulacije, v katerem se točka nahaja.

Primer 5.13. Naj bo TR triangulacija iz primera 5.12. Za točke $(0.5, 0.3)$, $(0.5, 0.7)$, $(0.5, 1.5)$ želimo preveriti, v katerem trikotniku triangulacije TR se nahajajo in pripadajoče baricentrične koordinate glede na ta trikotnik. To naredimo z naslednjim ukazom.

```
>>[ID, BC] = pointLocation(TR, [0.5 0.3; 0.5 0.7; 0.5 1.5])
ID =
```

```

1
2
NaN

```

```
BC =
```

```

0.5000    0.2000    0.3000
0.2000    0.3000    0.5000
NaN       NaN       NaN

```

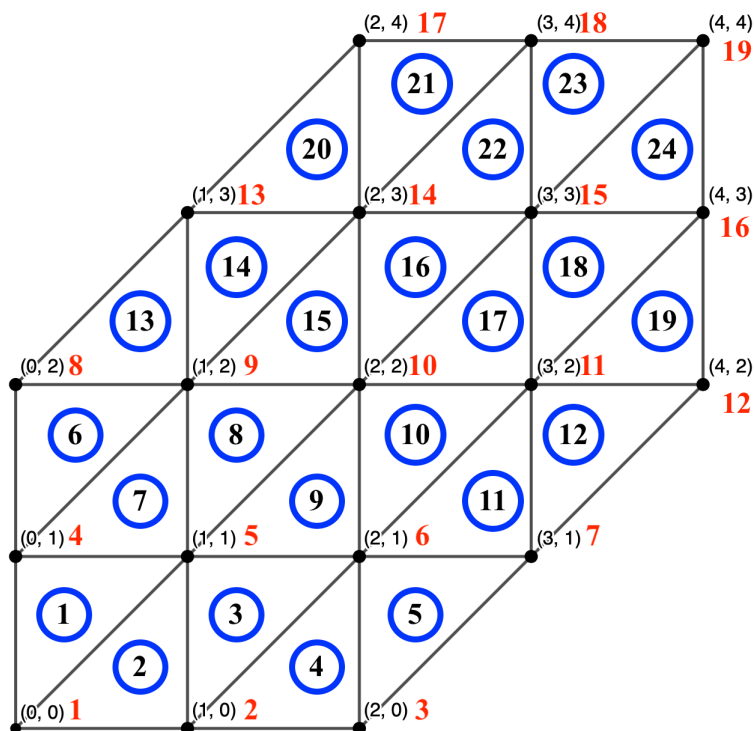
Rezultat nam pove, da prva točka leži v prvem trikotniku triangulacije, druga točka v drugem, tretja pa ne leži v nobenem od dveh trikotnikov, zato ji Matlab pripiše vrednost NaN. Enako tudi pri baricentričnih koordinatah.

5.2.2 Implementacija zlepka B_{222}

V nadaljevanju se omejimo na prostor premikov in kasneje hierarhično bazo zlepkov iz prostora premikov, ki so določeni z zlepkom $B_{222}(\mathbf{x})$. Spomnimo se, to je zlepek $B_M(\mathbf{x})$, določen z matriko

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Zlepek bomo predstavili z njegovo Bernstein-Bézierevo obliko v okolju Matlab. Cilj je napisati funkcijo, ki sprejme podatek o premiku (translaciji) ter faktorju skaliranja in vrne strukturo, katera nam omogoča računanje točk na zlepku, predstavljenim z Bernstein-Bézierevo obliko, ter smiselno triangulacijo nosilca tega zlepka. V tem delu zapišemo nekaj glavnih idej in programerskih prijemov za lažje razumevanje implementacije v prilogi I. Zaradi narave škatlastih zlepkov pri implementaciji uporabimo dobro mero principa *hard coding*. Vedno se zanašamo na osnovni primer, kjer je škatlasti zlepek B_{222} v kartezični ravnini postavljen tako, da ima skrajno levo-spodnje oglišče postavljeno v izhodišče. Potem v matriko, recimo ji P , velikosti 19×2 , shranimo vsa oglišča iz slike 40. Na isti sliki imamo z odebeljenimi rdečimi številkami prikazano indeksacijo oglišč, z obkroženimi modrimi pa indeksacijo trikotnikov. Oglišča shranjujemo v matriko na način, da vsaka vrstica ustreza x in y kordinati glede na indeksacijo slike 40. Triangulacijo potem preprosto ustvarimo še na način, da naredimo matriko T , velikosti 24×3 , kjer na smislen način glede na indeksacijo trikotnikov zapišemo vseh 24 trikotnikov v triangulaciji. Naslednja stvar, ki jo moramo določiti, so Bernsteinovi koeficienti. Te imamo podane na sliki 23. Mislimo si, da so tej koeficienti določeni z matriko velikosti 17×17 . Približno 70% elementov te matrike je enake 0. Tisti, ki niso, jih ročno spremenimo na ustrezen koeficient iz slike 23. Koeficiente moramo še deliti s 24. Potrebujemo še eno matriko, v kateri shranimo povezavo med vsakim ogliščem



Slika 40: Nosilec zlepka B_{222} z glavnimi oglišči v kartezični ravnini in njihovo indeksacijo ter indeksacija trikotnikov.

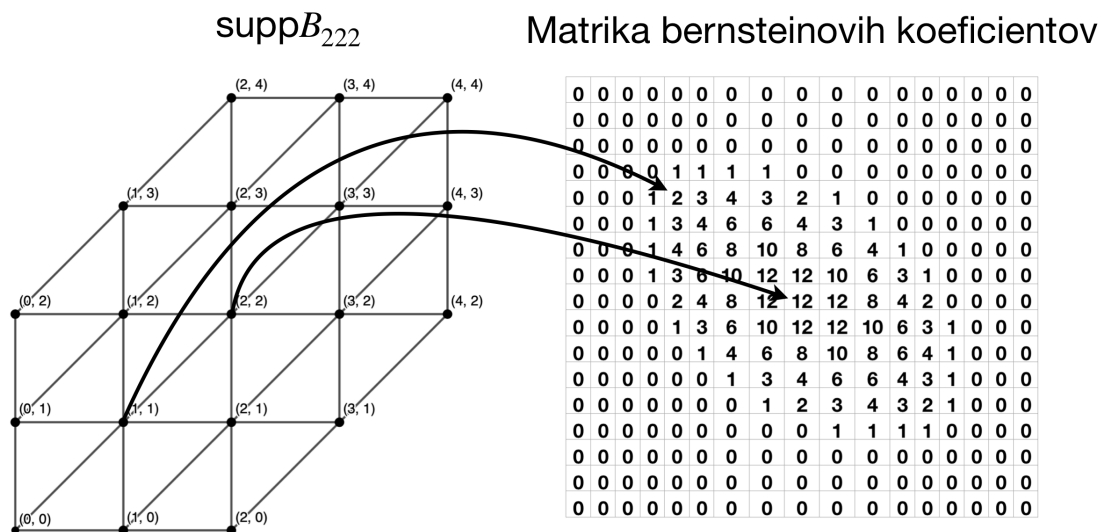
slike 40 in ustreznim indeksom v matriki koeficientov. To pomeni, da ustvarimo še eno matriko velikosti 17×2 , kjer vrstice predstavljajo oglišča iz indeksacije na sliki 40 in nosijo informacijo o indeksih tega oglišča v matriki Bernsteinovih koeficientov. Demonstracijo dveh takšnih povezav vidimo na sliki 41. Ker so kontrolne točke in s tem Bernsteinovi koeficienti po nosilcu razporejeni ekvidistantno, lahko s temi podatki izračunamo tudi vse ostale kontrolne točke. S temi predpostavkami lahko z algoritmom 3 izračunamo Bernstein-Bézierevo obliko poljubnega premika zlepka B_{222} .

Primer 5.14. Poglejmo si kako izračunamo točko C_{202} na sliki 42, če poznamo le točke v ogliščih trikotnika, torej C_{400} , C_{040} in C_{004} . Ker so točke ekvidistantno razporejene, leži točka C_{202} ravno med točkama C_{004} in C_{400} . Označimo kartezične koordinate točke C_{004} z x_{004} in y_{004} , koordinate točke C_{400} pa z x_{400} in y_{400} . Potem so kartezične koordinate točke C_{202} enake

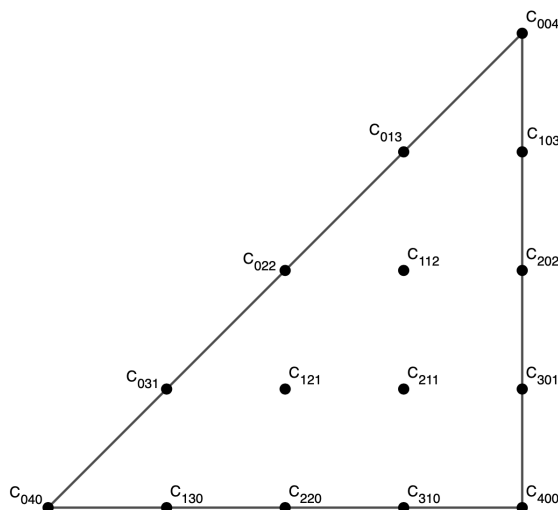
$$x_{202} = \frac{1}{2}(x_{400} + x_{004}), \quad y_{202} = \frac{1}{2}(y_{400} + y_{004}).$$

Zadnja komponenta vsake kontrolne točke je Bernsteinov koeficient. Tudi koeficient b_{202} je shranjen v matriki Bernsteinovih koeficientov. Ker poznamo povezavo med oglišči trikotnika z indeksi v matriki Bernsteinovih koeficientov, lahko izračunamo indekse tudi za točko C_{202} . Naj bosta

$$C_{400}^{ij} = (i_{400}, j_{400})$$



Slika 41: Primer dveh povezav med oglišči nosilca zlepka B_{222} in pripadajočimi Bernsteinovimi koeficienti. Bernsteinovi koeficienti so zaradi preglednosti pomnoženi s 24.



Slika 42: Kontrolne točke na enem trikotniku nosilca zlepka B_{222} .

in

$$C_{004}^{ij} = (i_{004}, j_{004})$$

indeksa v matriki Bernsteinovih koeficientov za točki C_{400} in C_{004} . Potem je

$$C_{202}^{ij} = \frac{1}{2} (C_{400}^{ij} + C_{004}^{ij}).$$

Ko imamo točko C_{202} , lahko izračunamo tudi točki C_{301} in C_{103} in postopoma izračunamo vse kontrolne točke na sliki 42.

Tabela 3: Algoritem konstrukcije poljubnega premika škatlastega zlepka $B_{222}(\mathbf{x})$.

Algoritem BB-OBLIKA PREMIKA ZLEPKA $B_{222}(\mathbf{x})$:

Podatki: Vektor premika zlepka $\mathbf{v} = \begin{bmatrix} x & y \end{bmatrix}^T$, faktor skaliranja h .

1. Implementiraj oglišča nepremaknjega nosilca z matriko P , trikotnike v triangulaciji z matriko T , Bernsteinove koeficiente z matriko B in matriko povezav med oglišči in indeksi Bernsteinovih koeficientov z matriko M .

2. Nosilec skaliraj in ga prestavi po pravilu

$$P_{ij} = P_{ij} \cdot h$$

$$P_{i1} = P_{i1} + x$$

$$P_{i2} = P_{i2} + y, \quad \text{za } i = 1, 2, \dots, 19, j = 1, 2.$$

3. Ustvari triangulacijo \mathcal{T} glede na modificirana oglišča P in predpise trikotnikov T .
4. Za k -ti trikotnik v triangulaciji določi vse kontrolne točke iz slike 42. Za točko

$$C_{ijk} = \begin{bmatrix} x_{ijk} \\ y_{ijk} \\ b_{ijk} \end{bmatrix},$$

velja, da sta x_{ijk} in y_{ijk} kartezični koordinati kontrolne točke, b_{ijk} pa pripadajoč Bernsteinov koeficient. Vrednosti kontrolnih točk v oglišču vsakega trikotnika dobimo neposredno s pomočjo matrike M , ki nam pove indeks koeficienta v matriki B . Ker so vse kontrolne točke razporejene ekvidistantno, lahko tudi te preprosto izračunamo. Zgled izračuna ene točke najdemo v primeru 5.14. Ko izračunaš vse kontrolne točke, na vsakem trikotniku določi Bézierevo krpo

$$B(u, v) = \sum_{i+j+k=4} C_{ijk} \beta_{ijk}(u, v)$$

in jo shrani v množico \mathcal{B} .

5. Vrni množico \mathcal{B} in triangulacijo \mathcal{T} .

5.2.3 Prostor premikov zlepka $B_{222}(\mathbf{x})$

Po implementaciji Bernstein-Bézierove oblike škatlastega zlepka $B_{222}(\mathbf{x})$ implementiramo prostor premikov škatlastega zlepka, definirane v definiciji 3.15. Omejimo se na neko kvadratno območje v \mathbb{R}^2 in se spomnimo, da lahko za prostor premikov \mathcal{S} iz

definicije 3.15 in območje Ω definiramo prostor

$$S = \{\beta \in \mathcal{S} \mid \text{supp } \beta \cap \Omega \neq \emptyset\}.$$

V prostoru S so torej tisti premiki zlepkov, ki vsebujejo na območju Ω vsaj nekaj svojega nosilca. Ker predpostavimo, da je prostor \mathcal{S} določen z zlepkom B_{222} in ker se omejimo na kvadratna območja v \mathbb{R}^2 , se nam stvari nekoliko poenostavijo. Za začetek si pogledajmo primer, ko je $\Omega = [0, 1] \times [0, 1]$ in naj bo h faktor skaliranja. Od tukaj naprej postavimo pogoj, da je $h \in \mathbb{N}$. Potem velja

$$S = \left\{ B_{222} \left(\frac{1}{h} \cdot \left(\mathbf{x} + \begin{bmatrix} i \\ j \end{bmatrix} \right) \right) \mid (i, j) \in Q \right\}$$

$$Q = \{(i, j) \neq (h-1, -3), (-3, h-1) \mid \\ i = -3, -2, \dots, h-1, j = -3, -2, \dots, h-1\}.$$

Hitro lahko preverimo, da na ta način ravno poberemo vse zlepke, ki vsebujejo nekaj nosilca na območju $[0, 1] \times [0, 1]$. Prostor S posplošimo v smislu, da je Ω poljubno kvadratno območje v \mathbb{R}^2 , torej $\Omega = [a, b] \times [a, b]$. To naredimo na način, da območje $[0, 1] \times [0, 1]$ ustrezno razširimo in premaknemo. Naj bo $m = (b - a - 1) \cdot h$. Potem je

$$S = \left\{ B_{222} \left(\begin{bmatrix} a \\ a \end{bmatrix} + \frac{1}{h} \cdot \left(\mathbf{x} + \begin{bmatrix} i \\ j \end{bmatrix} \right) \right) \mid (i, j) \in Q \right\}$$

$$Q = \{(i, j) \neq (m+h-1, -3), (-3, m+h-1) \mid \\ i = -3, -2, \dots, m+h-1, j = -3, -2, \dots, m+h-1\}.$$

Algoritem 2 prikazuje psevdokodo konstrukcije baze premikov zlepka $B_{222}(\mathbf{x})$.

Algoritem 2: Prostor premikov zlepka $B_{222}(\mathbf{x})$

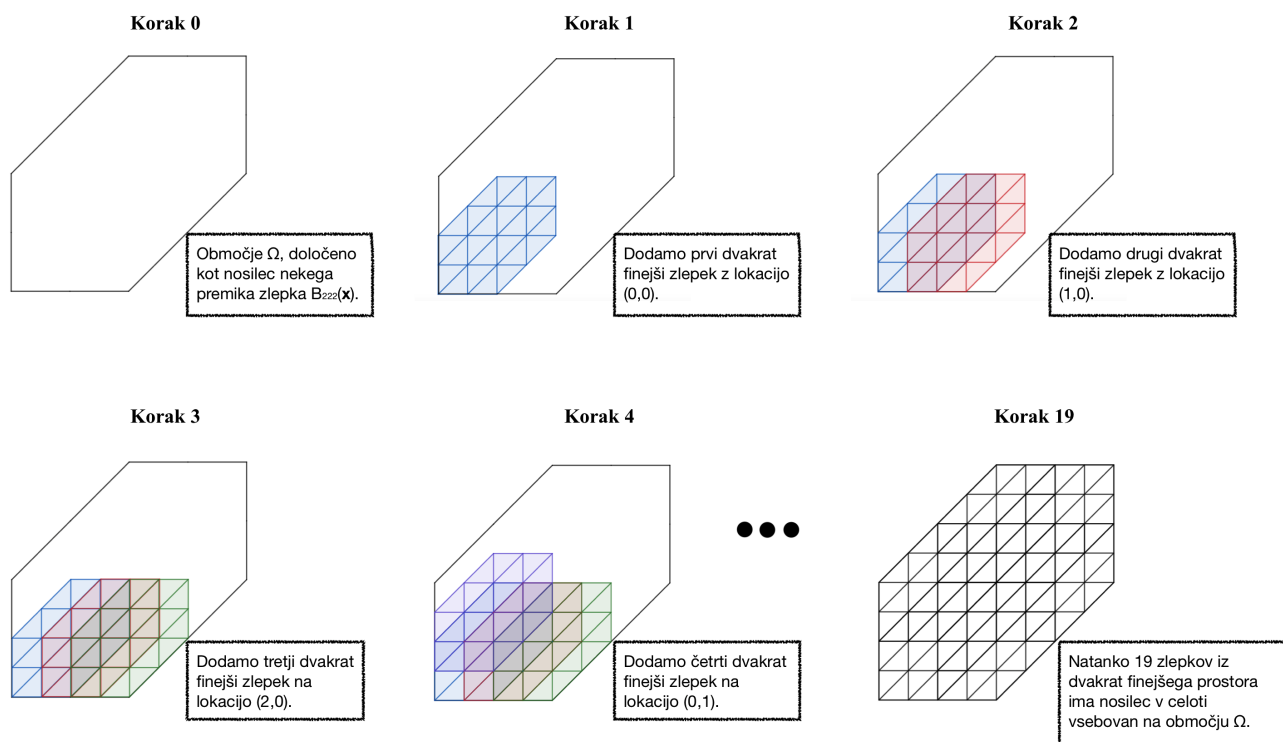
Vhod: območje $\Omega = [a, b] \times [a, b]$, faktor skaliranja h .

Izhod: množica premikov zlepka $B_{222} \mathcal{B}$.

- 1 Definiraj $m = (b - a - 1) \cdot h$.
 - 2 **za** $i = -3, -2, \dots, m + h - 1$
 - 3 **za** $j = -3, -2, \dots, m + h - 1$
 - 4 **če** $(i, j) \neq (-3, m + h - 1), (m + h - 1, -3)$ **potem**
 - 5 $\mathbf{v} = \begin{bmatrix} a + \frac{1}{h}i \\ a + \frac{1}{h}j \end{bmatrix};$
 - 6 Ustvari škatlasti zlepek $B_{222}(\frac{1}{h}(\mathbf{x} - \mathbf{v}))$ z algoritmom 3;
 - 7 Dodaj zlepek $B_{222}(\frac{1}{h}(\mathbf{x} - \mathbf{v}))$ v množico \mathcal{B} ;
 - 8 **vrni** \mathcal{B} ;
-

5.2.4 Hierarhična baza škatlastih zlepkov

Ostala nam je samo še implementacija baze hierarhičnih zlepkov iz definicije 4.10. Spomnimo se, da smo sprejeli omejitev, da je v zaporedju gnezdenih domen $\Omega^0 \supseteq \Omega^1 \supseteq \dots \supseteq \Omega^N$, Ω^0 kvadratno območje v \mathbb{R}^2 , ostala območja pa so tvorjena iz unije nosilcev zlepkov iz višjega nivoja, saj na ta način v hierarhični bazi ohranimo lastnost particije enote. Prav tako predpostavimo, da za nek faktor skaliranja na i -tem nivoju h_i velja, da za faktor skaliranja na naslednjem nivoju velja $h_{i+1} = 2h_i$ za $i = 0, 1, \dots, N-1$. S tem si na vsakem koraku tvorjenja hierarhične baze zagotovimo predvidljivo menjavo baznih funkcij iz višjega nivoja s finejšimi baznimi funkcijami iz enega nivoja nižje. Vpeljimo dogovor, da lokacijo nekega škatlastega zlepka označimo s kartezičnimi koordinatami oglišča nosilca, ki je skrajno levo spodaj. Tako je npr. lokacija zlepka $B_{222}(\mathbf{x})$ enaka $(0, 0)$. Potem lahko neko območje Ω , ki je nosilec nekega premika škatlastega zlepka (brez škode za splošnost naj bo to kar, iz izhodišča nepremaknjen, zlepek $B_{222}(\frac{1}{h}\mathbf{x})$) napolnimo s funkcijami dvakrat finejšega prostora na način kot je prikazano na sliki 43. Na takšnem območju imamo torej natanko 19 škatlastih zlepkov iz dvakrat finejšega prostora. V Algoritmu 3 je prikazana pseudokoda tvorjenja množice premikov



Slika 43: Shema napolnitve območja, ki je nosilec škatlastega zlepka B_{222} .

škatlastih zlepkov, definiranih nad območjem nosilca zlepka $B_{222}(\frac{1}{h}\mathbf{x})$. Algoritem je zapisan za iz izhodišča nepremaknjen zlepek.

Sedaj imamo pripravljeno vse za zapis pseudokode tvorjenja hierarhične baze škatlastih zlepkov. Bazo tvorimo podobno kot tisto v primeru B-zlepkov, torej imamo eno

Algoritem 3: Tvorjenje finejše množice premikov škatlastih zlepkov, definiranih nad nosilcem zlepka $B_{222}(\frac{1}{h_0}\mathbf{x})$

Vhod: Območje $\Omega = \text{supp } B_{222}(\frac{1}{h_0}\mathbf{x})$, faktor sklairanja h_0 .

Izhod: Množica premikov zlepka $B_{222}(\frac{1}{h_1}\mathbf{x})$ \mathcal{B} .

1 Definiraj $h_1 = 2h_0$;

2 **za** $i = 0, 1, \dots, 4$

3 **za** $j = 0, 1, \dots, 4$

4 **če** $(i, j) \neq (0, 3), (0, 4), (1, 4), (3, 0), (4, 0), (4, 1)$ **potem**

5 $\mathbf{v} = [i, j]^T$;

6 Ustvari škatlasti zlepek $B_{222}(\frac{1}{h_1}(\mathbf{x} - \mathbf{v}))$ z algoritmom 3;

7 Dodaj zlepek $B_{222}(\frac{1}{h_1}(\mathbf{x} - \mathbf{v}))$ v množico \mathcal{B} ;

8 **vrni** \mathcal{B} ;

množico, ki jo po korakih posodabljam. Na vsakem koraku nekatere bazne funkcije obdržimo, nekatere izbrišemo in dodamo nove. Pseudokodo najdemo v Algoritemu 4.

Algoritem 4: Hierarhična baza premikov škatlastih zlepkov $B_{222}(\mathbf{x})$.

Vhod: Gnezdeno zaporedje domen $\{\Omega^\ell\}_{\ell=0,1,\dots,N}$, začetni faktor skaliranja h_0 .

Izhod: Hierarhična baza škatlastih zlepkov H , množica triangulacij vseh zlepkov \mathcal{T} .

1 Napolni množico H z zlepkami po Algoritmu 8, glede na območje Ω^0 in začetni faktor skaliranja h_0 ;

2 **za** $\ell = 2, 3, \dots, N$

3 $h_\ell = 2h_{\ell-1}$;

4 Najdi množico zlepkov

$$B = \{\text{supp } \beta \in \mathcal{S}^\ell\},$$

tako da,

$$\bigcup_{\beta \in B} \beta = \Omega^\ell;$$

5 Posodobi množico H tako, da

$$H = H \setminus B;$$

6 **za** $k = 1, 2, \dots, |B|$

7 Z Algoritmom 8 konstruiraj množico \mathcal{B}_k glede na območje $\text{supp } \beta_k$ in faktor skaliranja h_ℓ ;

8 Posodobi množico H tako, da

$$H = H \cup \mathcal{B}_k;$$

9 **vrni** H ;

6 APLIKACIJE

V tem poglavju predstavimo uporabo in prikaz delovanja do sedaj zapisane teorije. Hierarhično bazo B-zlepkov in škatlastih zlepkov uporabimo za aproksimacijo funkcij po metodi najmanjših kvadratov ter za reševanje Poissonove enačbe z Galerkinovo metodo. V prvem delu tega poglavja bomo na kratko in večinoma brez dokazov zapisali nekaj nujno potrebne teorije iz aproksimacije in numeričnega reševanja parcialnih diferencialnih enačb, nato pa si bomo ogledali nekaj rezultatov implementacije v okolju Matlab. Ker je uporaba hierarhične baze zlepkov nekaj relativno novega in v Matlabu posebej za hierarhijo ne obstajajo že vgrajene funkcije, moramo algoritme implementirati sami, kot smo videli v poglavju 5. Ob implementiranju obeh vrst zlepkov naletimo na tehnične izzive, ki so specifični za vsako vrsto zlepka posebej. Lahko trdimo, da je implementacija škatlastih zlepkov zahtevnejša, saj je v Matlabu za to vrsto zlepkov na voljo precej manj že vgrajenih funkcij in ukazov kot za primer B-zlepkov. Posledično nam pri implementaciji obeh vrst zlepkov ostane še nekaj prostora za optimizacijo, poleg tega ne moremo trditi, da sta obe vrsti zlepkov implementirani enako učinkovito. Ravno zaradi teh razlogov, bi bila resnejša primerjava časovnih in prostorskih zmogljivosti implementiranih algoritmov pristranska in se ji v večji meri izognemo.

6.1 TEORETIČNO OZADJE

V tem podpoglavju zapišemo nekaj teorije, ki jo bomo nujno potrebovali v drugem delu, kjer si ogledamo rezultate implementacij v okolju Matlab.

6.1.1 Minimalna določitvena množica

Poglavje začnemo s temo, za katero se na prvi pogled zdi, da nima neposredne povezave s problemi, ki jih obravnavamo v magistrskem delu. Ogledali si bomo algoritem minimalne določitvene množice iz članka [12] (v nadaljevanju tudi MDS algoritem). Ta algoritem uporabimo pri iskanju jedra matrike, ki je sestavljeno iz vektorjev, ki imajo čim več ničelnih elementov. Naj bo $T \in \mathbb{R}^{m \times n}$. Iščemo takšne vektorje \mathbf{x} , za katere velja

$$T\mathbf{x} = \mathbf{0}. \quad (6.1)$$

Označimo $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$. Ideja je sledeča: želimo poiskati najmanjšo indeksno množico I , za katero velja, da če x_i postavimo na 0 za $i \in I$, morajo biti tudi vsi koeficienti, x_i $i \notin I$, enaki 0, da dobimo rešitev sistema (6.1). Za množico I očitno

velja $|I| = \dim(\ker(T))$. Ko enkrat imamo množico I , izračunamo jedro na način, da po enega izmed koeficientov $x_i, i \in I$ postavimo na 1, ostale elemente $x_j, j \in I, j \neq i$ pa na 0. Potem pa s temi omejitvami poračunamo še ostale koeficiente. Postopek iskanja minimalne določitvene množice nam podaja Algoritem 5. Implementacijo najdemo v prilogi G.

Algoritem 5: Algoritem Minimalne določitvene množice

Vhod: Matrika T iz sistema (6.1).

Izhod: Minimalna določitvena množica M .

- 1 Definiraj $\tilde{I} = \{\}$ in $\bar{I} = \{1, 2, \dots, n\}$.
 - 2 **za** $i = 1, 2, \dots, n$
 - 3 Reši sistem $T\mathbf{x} = \mathbf{0}$ z omejitvami $x_i = 1$ in $x_j = 0$ za $j \in \tilde{I} \cup (\bar{I} \setminus \{i\})$;
 - 4 **če** za sistem rešitev $\tilde{\mathbf{x}}$ obstaja, **potem**
 - 5 najdi tak minimalni indeks r , da $|\tilde{x}_r| = \max_{i=\{1,2,\dots,n\}} |\tilde{x}_i|$;
 - 6 $\tilde{I} = \tilde{I} \cup \{r\}$;
 - 7 $\bar{I} = \bar{I} \setminus \{i\}$;
 - 8 **vrni** \tilde{I} ;
-

6.1.2 Gaussova integracijska pravila

Pri aplikacijah bomo pogosto potrebovali izračun kakšnega integrala z večjo natančnostjo in čeprav v okolju Matlab že obstajajo vgrajene funkcije za njihov izračun, bi si vseeno želeli lastno implementacijo kakšne metode, ki bo pisana na kožo Matlab strukturam, s katerimi implementiramo B-zlepke. Zato na tem mestu na kratko povzamemo nekaj teorije o Gaussovih integracijskih pravilih. Gaussova integracijska pravila temeljijo na ideji, da podobno kot pri Newton-Cotesovih pravilih (Glej stran 148 v [14]) za nenegativno utež $\rho(x)$ integral izračunamo kot

$$\int_a^b f(x)\rho(x)dx = \sum_{i=0}^n \alpha_i f(x_i) + R(f). \quad (6.2)$$

Razlika med Gaussovimi in Newton-Cotesovimi pravili pa je v tem, da pri Gaussovih pravilih vozlov x_i ne določimo vnaprej. Iz tega razloga so Gaussova pravila primerna za numerično integracijo funkcij, ki jih poznamo v zaključeni obliki. Uteži α_i na desni strani izraza 6.2 dobimo z integracijo $\alpha_i = \int_a^b L_{n,i}(x)\rho(x)dx$, kjer so $L_{n,i}$ Lagrangevi bazni polinomi. S primerno izbiro vozlov x_i pa lahko dosežemo, da je pravilo (6.2) točno za polinome stopnje $2n + 1$. To naredimo s pomočjo ortogonalnih polinomov. Najprej za funkciji f in g na intervalu $[a, b]$ definiramo skalarni produkt

$$\langle f, g \rangle = \int_a^b f(x)g(x)\rho(x)dx.$$

Za funkciji f in g pravimo, da sta ortogonalni če velja $\langle f, g \rangle = 0$. Naj bo

$$P_0(x), P_1(x), \dots, P_n(x)$$

ortonormirana baza prostora polinomov stopnje n . Za vsak $0 \leq i \leq n$ velja, da je $P_i(x)$ stopnje i . Naj bo $P_{n+1}(x)$ tak polinom, da je ortogonalen na vse polinome q stopnje največ n . Polinom P_{n+1} lahko zapišemo v ničelni obliki kot

$$P_{n+1}(x) = a_{n+1}(x - x_0)(x - x_1) \cdots (x - x_n) = a_{n+1}\omega(x).$$

Ničle polinoma $P_{n+1}(x)$ vzamemo za vozle pri Gaussovem integracijskem pravilu. Naj bo sedaj funkcija f polinom stopnje $2n + 1$. Funkcijo lahko zapišemo kot $f(x) = p(x)\omega(x) + r(x)$, kjer sta p in r polinoma stopnje največ n . Polinoma p in ω sta očitno ortogonalna, saj je $P_{n+1}(x) = a_{n+1}\omega(x)$ ortogonalen na vse polinome stopnje $\leq n$. Potem lahko za nenegativno utež $\rho(x)$ zapišemo

$$\int_a^b f(x)\rho(x)dx = \int_a^b \omega(x)q(x)\rho(x)dx + \int_a^b r(x)\rho(x)dx.$$

Prvi integral na desni strani je enak 0, saj sta p in ω ortogonalna. Drugi integral pa zapišemo kot

$$\int_a^b r(x)\rho(x)dx = \sum_{i=0}^n \alpha_i r(x_i) + R(r) = \sum_{i=0}^n \alpha_i r(x_i) = \sum_{i=0}^n \alpha_i f(x_i).$$

Pri tem smo uporabili, da je polinom r stopnje največ n , zato ga lahko ekstatno integriramo s koeficienti α_i . Torej so Gaussova pravila res točna za polinome stopnje $2n + 1$. Za Gaussova pravila lahko pokažemo, da so uteži α_i vedno pozitivne (Glej izrek 4.7 v [14]). Prej smo videli, da koeficiente α_i lahko računamo z integriranjem Lagrangevih polinomov, v praksi pa običajno uporabimo Christoffel–Darbouxjevo formulo

$$\alpha_i = \frac{1}{\sum_{j=0}^n P_j^2(x_i)} \quad i = 0, 1, \dots, n,$$

kjer so P_j enako kot prej ortogonalni polinomi. Gaussova integracijska pravila najdemo tabelirana za kar nekaj različnih družin ortogonalnih polinomov (Legendrova, Čebiševa, Hermitova, itd.). V naših aplikacijah za računanje vozlov in uteži uporabimo Legendrove polinome, ki so definirani na intervalu $[-1, 1]$. V tabeli 4 imamo prikazane že izračunane vozle in uteži za Gaussova integracijska pravila na šestih točkah. Vozli v tabeli 4 so sicer izračunani za interval $[-1, 1]$, na poljuben interval $[a, b]$ pa jih lahko preslikamo na naslednji način. Naj bo $x_i \in [-1, 1]$. Tega preslikamo na poljuben interval s preslikavo

$$x_i \mapsto \frac{b-a}{2}x_i + \frac{a+b}{2} := \tilde{x}_i.$$

Gaussovo kvadraturno pravilo pa potem s temi koeficienti postane

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{i=0}^5 \alpha_i f(\tilde{x}_i).$$

Tabela 4: Vozli in uteži Gaussovih integracijskih pravil na šestih točkah.

x_i	α_i
-0.932469514203152	0.171324492379170
-0.661209386466264	0.360761573048139
-0.238619186083197	0.467913934572691
0.238619186083197	0.467913934572691
0.661209386466264	0.360761573048139
0.932469514203152	0.171324492379170

6.1.3 Aproksimacija po metodi najmanjših kvadratov

V tem delu predstavimo nujno potrebno teorijo za konstrukcijo najboljšega aproksimanta po metodi najmanjših kvadratov. Zaradi preglednosti bomo večino teorije zapisali za primer ene spremenljivke, katero bomo kasneje enostavno posplošili na primer dveh. Naj bo $X = \mathcal{L}^2([a, b])$ prostor normiran z drugo normo $\| \cdot \|_2 = \sqrt{\langle x, y \rangle}$ in S končna podmnožica prostora X . V prostoru S imamo zagotovljen obstoj in enoličnost najboljšega aproksimanta (Glej izrek 1.3 v [14]). Za začetek definirajmo dve vrsti skalarnega produkta.

Definicija 6.1. Naj bo $X = \mathcal{L}^2([a, b])$ in $f, g \in X$. Za funkciji f, g ter utež ρ definiramo **zvezni** skalarni produkt

$$\langle f, g \rangle = \int_a^b f(x)g(x)\rho(x) dx.$$

Definicija 6.2. Naj bo $X = \mathcal{L}^2([a, b])$, $f, g \in X$ in $\mathbf{x} = \{x_0, x_1, \dots, x_m\}$. Za funkciji f, g ter utež ρ definiramo **diskretni** skalarni produkt

$$\langle f, g \rangle = \sum_{i=0}^m f(x_i)g(x_i)\rho(x_i).$$

Opomba 6.3. Definiciji 6.1 in 6.2 lahko smiselno posplošimo na dve dimenziji, tako da predpostavimo $X = \mathcal{L}^2([a, b] \times [c, d])$, $f, g \in X$ ter $\mathbf{x} = \{x_0, x_1, \dots, x_m\}$ in $\mathbf{y} = \{y_0, y_1, \dots, y_m\}$. Potem definiramo zvezni skalarni produkt kot

$$\langle f, g \rangle = \int_a^b \int_c^d f(x, y)g(x, y)\rho(x, y) dy dx$$

in diskretni skalarni produkt kot

$$\langle f, g \rangle = \sum_{i=0}^m \sum_{j=0}^m f(x_i, y_j)g(x_i, y_j)\rho(x_i, y_j).$$

Naslednji izrek nam podaja konstrukcijo najboljšega aproksimanta po metodi najmanjših kvadratov za eno spremenljivko.

Izrek 6.4. Naj bo $X = \mathcal{L}^2([a, b])$ in $S \subseteq X$. Element $\tilde{f} \in S$ je najboljši aproksimant po metodi najmanjših kvadratov za funkcijo $f \in X$, če in samo če velja

$$f - \tilde{f} \perp S.$$

Dokaz. Izrek moramo pokazati v obe smeri. Spomnimo se, da je druga norma inducirana s skalarnim produktom iz definicij 6.1 in 6.2. V dokazu zaradi preglednosti privzamemo zvezno verzijo, čeprav bi očitno enako delovalo tudi v primeru diskretne. Naj bo $f - \tilde{f} \perp S$. Potem je

$$\begin{aligned} \|f - s\|_2^2 &= \|f - \tilde{f} + \tilde{f} - s\|_2^2 = \int_a^b (f - \tilde{f} + \tilde{f} - s)^2 dx \\ &= \int_a^b (f - \tilde{f})^2 dx + 2 \int_a^b (f - \tilde{f})(\tilde{f} - s) dx + \int_a^b (\tilde{f} - s)^2 dx \quad (6.3) \\ &= \langle f - \tilde{f}, f - \tilde{f} \rangle + 2\langle f - \tilde{f}, \tilde{f} - s \rangle + \langle \tilde{f} - s, \tilde{f} - s \rangle \\ &= \|f - \tilde{f}\|_2^2 + \|\tilde{f} - s\|_2^2. \end{aligned}$$

Za zadnjo vrstico pa očitno velja $\|f - \tilde{f}\|_2^2 + \|\tilde{f} - s\|_2^2 \geq \|f - \tilde{f}\|_2^2$. Torej je \tilde{f} res najboljši aproksimant za f .

Poglejmo si še obratno smer. Naj bo \tilde{f} najboljši aproksimant za funkcijo f . Pokažimo, da je $f - \tilde{f}$ pravokoten na S . Nadalje naj bo $s \in S$ poljuben in $\lambda > 0$. Ker je \tilde{f} najboljši aproksimant, $\tilde{f} + \lambda s$ kvečjemu slabše aproksimira f . Potem velja

$$\begin{aligned} 0 &\leq \|f - \tilde{f} + \lambda s\|_2^2 - \|f - \tilde{f}\|_2^2 \\ &= \|f - \tilde{f}\|_2^2 + 2\lambda \langle f - \tilde{f}, s \rangle + \lambda^2 \|s\|_2^2 - \|f - \tilde{f}\|_2^2 \\ &= 2\lambda \langle f - \tilde{f}, s \rangle + \lambda^2 \|s\|_2^2 = \lambda(2\langle f - \tilde{f}, s \rangle + \lambda \|s\|_2^2) \end{aligned}$$

Vedno lahko izberemo dovolj majhen $\lambda > 0$, da bo $\langle f - \tilde{f}, s \rangle$ po velikosti prevladal nad $\lambda \|s\|_2^2$. Od tod sledi, da je $\langle f - \tilde{f}, s \rangle \geq 0$. Ker pa je bil $s \in S$ poljuben in ker je tudi $-s \in S$ velja

$$\begin{aligned} \langle f - \tilde{f}, -s \rangle &= -\langle f - \tilde{f}, s \rangle \geq 0 \Rightarrow \\ \langle f - \tilde{f}, s \rangle &\leq 0 \end{aligned}$$

Od tod pa sledi, da mora veljati $\langle f - \tilde{f}, s \rangle = 0$ in s tem, da je najboljši aproksimant pravokoten na S . \square

Sedaj imamo pripravljeno vse za konstrukcijo aproksimanta po metodi najmanjših kvadratov. Naj bo $\{s_i\}$, $i = 1, 2, \dots, n$, baza za S . Sledi, da lahko aproksimant $\tilde{f} \in S$ zapišemo kot linearno kombinacijo

$$\tilde{f} = \sum_{i=1}^n \alpha_i s_i, \quad \alpha_i \in \mathbb{R}.$$

Ker je \tilde{f} najboljši aproksimant, mora po izreku 6.4 veljati

$$f - \sum_{i=1}^n \alpha_i s_i \perp S.$$

Za ugotovitev ali je \tilde{f} pravokoten na S , je dovolj preveriti ali je pravokoten na vse bazne funkcije prostora S , torej

$$\langle f - \sum_{i=1}^n \alpha_i s_i, s_j \rangle = 0, \quad j = 1, 2, \dots, n,$$

kar pa lahko ekvivalentno zapišemo kot

$$\sum_{i=1}^n \alpha_i \langle s_i, s_j \rangle = \langle f, s_j \rangle. \quad (6.4)$$

Ta izraz pa lahko zapišemo v matrični obliki kot

$$G \cdot \mathbf{a} = \mathbf{b},$$

kjer je

$$G = \begin{bmatrix} \langle s_1, s_1 \rangle & \langle s_2, s_1 \rangle & \cdots & \langle s_n, s_1 \rangle \\ \langle s_1, s_2 \rangle & \langle s_2, s_2 \rangle & \cdots & \langle s_n, s_2 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle s_1, s_n \rangle & \langle s_2, s_n \rangle & \cdots & \langle s_n, s_n \rangle \end{bmatrix}$$

in

$$\mathbf{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \langle f, s_1 \rangle \\ \langle f, s_2 \rangle \\ \vdots \\ \langle f, s_n \rangle \end{bmatrix}.$$

Matriko G imenujemo Gramova matrika oz. v primeru aproksimacije po metodi najmanjših kvadratov pogosto tudi **masna** matrika.

6.1.4 Galerkinova metoda

Druga uporaba hierarhičnih zlepkov pa je reševanje parcialnih diferencialnih enačb. V tem poglavju si pogledamo nekaj teorije za reševanje parcialnih diferencialnih enačb z Galerkinovo metodo. V tem magistrskem delu se bomo precej omejili in si pogledali le en tip eliptične parcialne diferencialne enačbe in sicer Poissonovo enačbo, čeprav lahko ta princip reševanja razširimo na druge tipe parcialnih diferencialnih enačb tako kot npr. v [10]. Za začetek si bomo pogledali reševanje Poissonove enačbe s homogenim robnih pogojem. Takšna enačba se glasi

$$\begin{aligned} -\Delta u(\mathbf{x}) &= f & \mathbf{x} \in \Omega \\ u(\mathbf{x}) &= 0 & \mathbf{x} \in \partial\Omega, \end{aligned} \quad (6.5)$$

kjer je $\Omega \subset \mathbb{R}^2$. Cilj Galerkinove metode je najti približek za rešitev u . Prvi korak je zapisati Poissonovo enačbo v šibki obliki. Predpostavimo, da je V prostor funkcij, ki izpolnjujejo homogen Dirichletov robni pogoj in predpostavimo, da tudi rešitev problema (6.5) leži v prostoru V . Enačbo (6.5) pomnožimo z neko funkcijo $v \in V$ (to funkcijo običajno imenujemo testna funkcija) in integriramo po območju Ω . Potem dobimo

$$\int_{\Omega} \Delta u \cdot v \, d\mathbf{x} = \int_{\Omega} f \cdot v \, d\mathbf{x}. \quad (6.6)$$

Levo stran enačbe (6.6) lahko integriramo po delih. Zanašamo se na Greenovo formulo (Glej poglavje 7 v [25]) in homogenost Dirichletovega pogoja, da dobimo

$$\int_{\Omega} \langle \nabla u, \nabla v \rangle \, d\mathbf{x} = \int_{\Omega} f \cdot v \, d\mathbf{x},$$

kjer ∇u predstavlja gradient funkcije u , skalarni produkt $\langle \cdot, \cdot \rangle$ pa vzamemo iz definicije 6.1. Na tej točki pridemo do ključne ideje Galerkinove metode. Prostor V zamenjamo z nekim končno-dimenzionalnim prostorom V_h , t.d. $V_h \subset V$. Potem se v prostoru V_h nahaja tudi najboljši aproksimant za rešitev enačbe (6.5). Najboljši aproksimant označimo z u_h . Recimo, da je dimenzija prostora V_h enaka n in naj bo $\{\varphi_i\}$, $i = 1, 2, \dots, n$, baza prostora V_h . Potem lahko aproksimant u_h zapišemo kot linearno kombinacijo

$$u_h = \sum_{i=1}^n c_i \varphi_i, \quad c_i \in \mathbb{R}.$$

To linearno kombinacijo vstavimo v enačbo (6.5) in dobimo

$$\Delta \sum_{i=1}^n c_i \varphi_i = f.$$

Laplacov operator lahko nesemo v vsoto in dobimo

$$\sum_{i=1}^n c_i \Delta \varphi_i = f. \quad (6.7)$$

Da dobimo šibko obliko moramo (6.7) pomnožiti s testno funkcijo prostora V_h in integrirati po Ω . V resnici enačbo (6.7) pomnožimo z vsemi baznimi funkcijami prostora V_h . Sledi

$$\sum_{i=1}^n c_i \int_{\Omega} \Delta \varphi_i \cdot \varphi_j \, d\mathbf{x} = \int_{\Omega} f \cdot \varphi_j \, d\mathbf{x}, \quad j = 1, 2, \dots, n.$$

Uporabimo enak razmislek kot prej, da dobimo

$$\sum_{i=1}^n c_i \int_{\Omega} \langle \nabla \varphi_i, \nabla \varphi_j \rangle \, d\mathbf{x} = \int_{\Omega} f \cdot \varphi_j \, d\mathbf{x}, \quad j = 1, 2, \dots, n. \quad (6.8)$$

To pa lahko zapišemo v matrični obliki

$$G \cdot \mathbf{a} = \mathbf{b},$$

kjer so

$$G = \begin{bmatrix} \int_{\Omega} \langle \nabla \varphi_1, \nabla \varphi_1 \rangle d\mathbf{x} & \int_{\Omega} \langle \nabla \varphi_2, \nabla \varphi_1 \rangle d\mathbf{x} & \cdots & \int_{\Omega} \langle \nabla \varphi_n, \nabla \varphi_1 \rangle d\mathbf{x} \\ \int_{\Omega} \langle \nabla \varphi_1, \nabla \varphi_2 \rangle d\mathbf{x} & \int_{\Omega} \langle \nabla \varphi_2, \nabla \varphi_2 \rangle d\mathbf{x} & \cdots & \int_{\Omega} \langle \nabla \varphi_n, \nabla \varphi_2 \rangle d\mathbf{x} \\ \vdots & \vdots & \ddots & \vdots \\ \int_{\Omega} \langle \nabla \varphi_1, \nabla \varphi_n \rangle d\mathbf{x} & \int_{\Omega} \langle \nabla \varphi_2, \nabla \varphi_n \rangle d\mathbf{x} & \cdots & \int_{\Omega} \langle \nabla \varphi_n, \nabla \varphi_n \rangle d\mathbf{x} \end{bmatrix}$$

in

$$\mathbf{a} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \int_{\Omega} f \cdot \varphi_1 d\mathbf{x} \\ \int_{\Omega} f \cdot \varphi_2 d\mathbf{x} \\ \vdots \\ \int_{\Omega} f \cdot \varphi_n d\mathbf{x} \end{bmatrix}.$$

Matriko G v primeru Galerkinove metode imenujemo **togostna** matrika.

6.1.4.1 Nehomogen robni pogoj

V tem delu si pogledjmo še način, kako rešimo Poissonovo enačbo z nehomogenim robnim pogojem. Problem definiramo podobno kot v poglavju 6.1.4. Naj bo $\Omega \subset \mathbb{R}^2$. Potem se Poissonova enačba z nehomogenim robnim pogojem glasi

$$\begin{aligned} -\Delta u(\mathbf{x}) &= f & \mathbf{x} \in \Omega \\ u(\mathbf{x}) &= g & \mathbf{x} \in \partial\Omega. \end{aligned} \tag{6.9}$$

Želeli bi priti do enačbe oz. sistema enačb, ki nam bo dala šibko rešitev, kot linearno kombinacijo testnih funkcij. Predpostavimo, da je šibka rešitev problema (6.9) sestavljena iz dveh delov in sicer

$$u_h = u_H + u_R,$$

kjer je u_H rešitev homogene različice problema (6.9), u_R pa je rešitev problema (6.9) na robu domene Ω . Podobno, kot v poglavju 6.1.4, bomo šibko rešitev iskali v nekem končno-dimenzionalnem prostoru. V resnici v primeru nehomegene enačbe vzamemo dva končnodimenzionalna prostora. Naj bo V_H končno-dimezionalni prostor, katerega elementi so funkcije, ki izpolnjujejo homogen Dirichletov robni pogoj na Ω in V_R končno-dimenzionalni prostor, katerega elementi so funkcije, ki tega pogoja ne izpolnjujejo nujno. Definirajmo množici $I_R = \{1, 2, \dots, \dim V_R\}$ in $I_H = \{1, 2, \dots, \dim V_H\}$ in naj bosta $\{\varphi_i\}_{i \in I_R}$ in $\{\psi_i\}_{i \in I_H}$ bazi prostorov V_R in V_H . Šibka oblika Poissonove enačbe se v tem primeru glasi

$$\int_{\Omega} \nabla u_H \cdot \nabla v_H d\mathbf{x} = \int_{\Omega} f \cdot v_H d\mathbf{x} - \int_{\Omega} \nabla u_R \cdot \nabla v_H d\mathbf{x}, \tag{6.10}$$

kjer je v_H testna funkcija prostora V_H . Rešitev na robu u_R dobimo z aproksimacijo funkcije g po metodi najmanjših kvadratov, enako kot v poglavju 6.1.3. Ker je funkcija

g na robu območja Ω neničelna, iščemo aproksimant po metodi najmanjših kvadratov v prostoru V_R . Potem lahko rešitev na robu zapišemo kot linearno kombinacijo

$$u_R = \sum_{i \in I_R} c_i \psi_i.$$

Tudi gradient takšne funkcije zlahka izračunamo. Pripravljeno imamo vse, da poiščemo tudi homogeno rešitev problema (6.9). Ta rešitev pa se bo nahajala v prostoru V_H , torej jo lahko zapišemo kot linearno kombinacijo

$$u_H = \sum_{i \in I_H} c_i \varphi_i,$$

kar vstavimo v enačbo (6.10). Za testne funkcije vzamemo kar bazne funkcije prostora V_H in dobimo sistem

$$\sum_{i \in I_H} c_i \int_{\Omega} \nabla \varphi_i \cdot \nabla \varphi_j d\mathbf{x} = \int_{\Omega} f \cdot \varphi_j d\mathbf{x} - \int_{\Omega} \nabla u_R \cdot \nabla \varphi_j d\mathbf{x}, \text{ za } j \in I_H.$$

6.1.4.2 Diskretna verzija Galerkinove metode

Integrale skalarnih produktov testnih funkcij je težko učinkovito implementirati. Zato si na tej točki pogledimo še diskretno verzijo Galerkinove metode, ki je podobna diskretni verziji aproksimacije po metodi najmanjših kvadratov. Enačbe bomo zapisali le za primer homogenih robnih pogojev in se omejili na kvadratna območja $\Omega \subset \mathbb{R}^2$, tako da jih diskretiziramo z vektorjema $\mathbf{x} = \{x_0, x_1, \dots, x_m\}$ in $\mathbf{y} = \{y_0, y_1, \dots, y_m\}$. Posplošitev diskretne verzije za nehomogene robne pogoje je trivialna, metodo za poljubna območja pa bomo spoznali v nadaljevanju. Za elemente množic \mathbf{x} in \mathbf{y} predpostavimo, da so ekvidistantno razporejene, torej $x_{i+1} - x_i = y_{i+1} - y_i = h$, za vse $x_i, y_i, i = 1, 2, \dots, m$. Potem definiramo diskretno območje $\Omega = \mathbf{x} \times \mathbf{y}$. Integral po območju Ω nadomestimo z dvema vsotama in končni rezultat Galerkinove metode (enačbo (6.7)) zapišemo kot

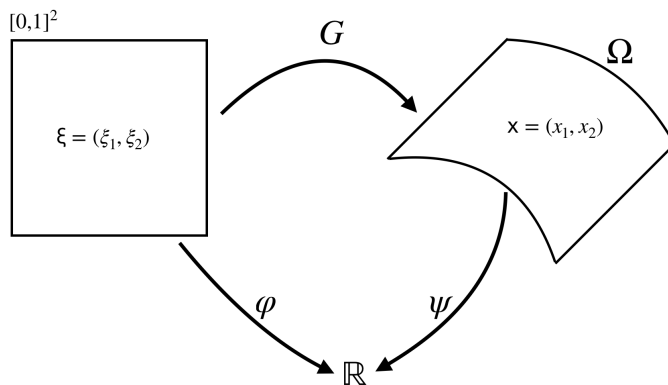
$$\sum_{i=1}^n c_i \sum_{k=0}^m \sum_{l=0}^m \langle \nabla \varphi_i(x_k, y_l), \nabla \varphi_j(x_k, y_l) \rangle = \sum_{k=0}^m \sum_{l=0}^m f(x_k, y_l) \cdot \varphi_j(x_k, y_l), \quad j = 1, 2, \dots, n,$$

kjer na levi strani zgornje enačbe za skalarni produkt vzamemo diskretno različico iz definicije 6.2. Na ta način smo se sicer omejili na posebna območja v ravnini, vendar že v naslednjem poglavju zapišemo nekaj teorije, kako lahko tako Galerkinovo metodo, kot aproksimacijo po metodi najmanjših kvadratov izvedemo na poljubnem območju v \mathbb{R}^2 .

6.1.5 Prehod na novo domeno

V poglavjih 6.1.3 in 6.1.4 smo se omejili na posebna območja v ravnini nad katerimi izvedemo posamezno aplikacijo. V primeru aproksimacije po metodi najmanjših kvadratov smo se omejili na pravokotna območja, medtem ko smo v primeru Galerkinove

metode sprejeli še močnejšo omejitev in se omejili na kvadratna območja v ravnini. Da bi lahko izvedli aplikacije na poljubni štirikotni domeni, moramo poiskati način, kako funkcije, definirane na tej domeni, reparametriziramo na neki domeni, ki jo znamo uporabiti. Naj bo ta domena enaka $[0, 1]^2$, poljubno domeno v \mathbb{R}^2 pa označimo z Ω . Definirajmo preslikavo $G : [0, 1]^2 \rightarrow \Omega$, ki preslika vsako točko $(\xi_1, \xi_2) \in [0, 1]^2$ v točko $(x_1, x_2) \in \Omega$. Definirajmo še preslikavi $\varphi : [0, 1]^2 \rightarrow \mathbb{R}$ in $\psi : \Omega \rightarrow \mathbb{R}$. Grafični prikaz domen prikazuje slika 44.



Slika 44: Grafični prikaz prehoda na novo domeno Ω iz območja $[0, 1]^2$.

Ker tako φ kot ψ slikata iz svoje domene v množico realnih števil, domeni pa sta med seboj povezani z preslikavo G , lahko namesto direktne preslikave naredimo ovinek, kot je to prikazano na shemi 44. Potem je očitno

$$\psi(x_1, x_2) = (\varphi \circ G^{-1})(x_1, x_2) = \varphi(G^{-1}(x_1, x_2)) = \varphi(\xi_1, \xi_2).$$

Poglejmo si, kako lahko takšen prehod na novo domeno uporabimo na prej opisanih aplikacijah.

6.1.5.1 Aproksimacija po metodi najmanjših kvadratov

Recimo, da za aproksimacijo po metodi najmanjših kvadratov ne gledamo več funkcij iz prostora $\mathcal{L}^2([a, b] \times [c, d])$, ampak iz prostora $\mathcal{L}^2(\Omega)$, kjer je Ω poljubno štirikotno območje v \mathbb{R}^2 . Potem lahko glavni rezultat (6.4) iz poglavja 6.1.3, v smislu zveznega skalarnega produkta zapišemo kot

$$\sum_{i=1}^n \alpha_i \langle \tilde{s}_i, \tilde{s}_j \rangle = \langle f, \tilde{s}_j \rangle$$

$$\sum_{i=1}^n \alpha_i \int_{\Omega} \tilde{s}_i(x_1, x_2) \cdot \tilde{s}_j(x_1, x_2) d\mathbf{x} = \int_{\Omega} f(x_1, x_2) \cdot \tilde{s}_j(x_1, x_2) d\mathbf{x},$$
(6.11)

kjer $\tilde{s}_i : \Omega \rightarrow \mathbb{R}$. Osredotočimo se na oba integrala zgornje enačbe in ju reparametriziramo z neko preslikavo $G : [0, 1]^2 \rightarrow \Omega$ na območje $[0, 1]^2$. Naj ima preslikava

G obliko $G(\xi_1, \xi_2) = (G_1(\xi_1, \xi_2), G_2(\xi_1, \xi_2))$, potem lahko integral na levi strani enačbe (6.11) zapišemo kot

$$\int_{\Omega} \tilde{s}_i(x_1, x_2) \cdot \tilde{s}_j(x_1, x_2) d\mathbf{x} = \int_{[0,1]^2} s_i(\xi_1, \xi_2) \cdot s_j(\xi_1, \xi_2) \cdot |\det J(\xi_1, \xi_2)| d\boldsymbol{\xi},$$

kjer je $s_\ell(\xi_1, \xi_2) = s_\ell(G^{-1}(\xi_1, \xi_2)) = \tilde{s}_\ell(\xi_1, \xi_2)$, $\ell = i, j$ in $J(\xi_1, \xi_2)$ Jakobijeva matrika preslikave G , torej

$$J(\xi_1, \xi_2) = \begin{bmatrix} \frac{\partial}{\partial \xi_1} G_1(\xi_1, \xi_2) & \frac{\partial}{\partial \xi_1} G_2(\xi_1, \xi_2) \\ \frac{\partial}{\partial \xi_2} G_1(\xi_1, \xi_2) & \frac{\partial}{\partial \xi_2} G_2(\xi_1, \xi_2) \end{bmatrix}. \quad (6.12)$$

Integral na desni strani enačbe (6.11) pa lahko reparametriziramo kot

$$\int_{\Omega} f(x_1, x_2) \cdot \tilde{s}_j(x_1, x_2) d\mathbf{x} = \int_{[0,1]^2} f(G(\xi_1, \xi_2)) \cdot s_j(\xi_1, \xi_1) \cdot |\det J(\xi_1, \xi_2)| d\boldsymbol{\xi}.$$

6.1.5.2 Galerkinova metoda

Tudi pri Galerkinovi metodi lahko uporabimo podoben razmislek kot pri aproksimaciji po metodi najmanjših kvadratov. Galerkinovo metodo smo v poglavju 6.1.4 definirali na poljubnem območju Ω , vendar je to iz podobnih razlogov kot pri aproksimaciji v praksi težko realizirati. Zato si zopet pomagamo z neko preslikavo $G : [0, 1]^2 \rightarrow \Omega$. Lema 6.5 nam pove, kako enačbo (6.8) iz poljubnega območja Ω reparametriziramo na območje $[0, 1]^2$.

Lema 6.5. *Naj bodo $\psi_i : \Omega \rightarrow \mathbb{R}$ elementi končno dimenzionalnega prostora V_h in naj bodo $\varphi_i : [0, 1]^2 \rightarrow \mathbb{R}$ tudi elementi končno dimenzionalnega prostora V_h , za $i = 1, 2, \dots, \dim(V_h)$. Naj bo še $G : [0, 1]^2 \rightarrow \Omega$. Potem velja*

$$\int_{\Omega} \nabla \psi_i^T \cdot \nabla \psi_j d\mathbf{x} = \int_{[0,1]^2} (J(\boldsymbol{\xi})^{-1} \nabla \varphi_i)^T \cdot (J(\boldsymbol{\xi})^{-1} \nabla \varphi_j) \cdot |\det J(\boldsymbol{\xi})| d\boldsymbol{\xi},$$

kjer je $J(\boldsymbol{\xi})$ Jakobijeva matrika za preslikavo G , enako kot v (6.12).

Dokaz. Vse, kar moramo pokazati, je, da velja

$$\nabla \varphi(\boldsymbol{\xi}) = J(\boldsymbol{\xi}) \nabla \psi(\mathbf{x}).$$

Poglejmo si gradient $\nabla \varphi(\boldsymbol{\xi})$.

$$\nabla \varphi(\boldsymbol{\xi}) = \begin{bmatrix} \frac{\partial}{\partial \xi_1} \varphi(\xi_1, \xi_2) \\ \frac{\partial}{\partial \xi_2} \varphi(\xi_1, \xi_2) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \xi_1} \psi(G(\xi_1, \xi_2)) \\ \frac{\partial}{\partial \xi_2} \psi(G(\xi_1, \xi_2)) \end{bmatrix}.$$

Uporabimo verižno pravilo in dobimo

$$\begin{bmatrix} \frac{\partial}{\partial \xi_1} \psi(G(\xi_1, \xi_2)) \\ \frac{\partial}{\partial \xi_2} \psi(G(\xi_1, \xi_2)) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} \psi(x_1, x_2) \cdot \frac{\partial x_1}{\partial \xi_1} + \frac{\partial}{\partial x_2} \psi(x_1, x_2) \cdot \frac{\partial x_2}{\partial \xi_1} \\ \frac{\partial}{\partial x_1} \psi(x_1, x_2) \cdot \frac{\partial x_1}{\partial \xi_2} + \frac{\partial}{\partial x_2} \psi(x_1, x_2) \cdot \frac{\partial x_2}{\partial \xi_2} \end{bmatrix}. \quad (6.13)$$

Ker je $(x_1, x_2) = G(\xi_1, \xi_2)$

$$\frac{\partial x_i}{\partial \xi_j} = \frac{\partial}{\partial \xi_j} G_i, \text{ za } i = 1, 2.$$

Desno stran enačbe (6.13) lahko zapišemo kot

$$\begin{bmatrix} \frac{\partial}{\partial x_1} \psi(x_1, x_2) \cdot \frac{\partial}{\partial \xi_1} G_1(\xi_1, \xi_2) + \frac{\partial}{\partial x_2} \psi(x_1, x_2) \cdot \frac{\partial}{\partial \xi_1} G_2(\xi_1, \xi_2) \\ \frac{\partial}{\partial x_1} \psi(x_1, x_2) \cdot \frac{\partial}{\partial \xi_2} G_1(\xi_1, \xi_2) + \frac{\partial}{\partial x_2} \psi(x_1, x_2) \cdot \frac{\partial}{\partial \xi_2} G_2(\xi_1, \xi_2) \end{bmatrix},$$

kar pa lahko zapišemo v matrični obliki kot

$$\begin{bmatrix} \frac{\partial}{\partial \xi_1} G_1(\xi_1, \xi_2) & \frac{\partial}{\partial \xi_1} G_2(\xi_1, \xi_2) \\ \frac{\partial}{\partial \xi_2} G_1(\xi_1, \xi_2) & \frac{\partial}{\partial \xi_2} G_2(\xi_1, \xi_2) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_1} \psi(x_1, x_2) \\ \frac{\partial}{\partial x_2} \psi(x_1, x_2) \end{bmatrix},$$

to pa je ravno enako

$$J(\xi_1, \xi_2) \nabla \psi(x_1, x_2). \quad \square$$

V primeru Galerkinove metode desno stran enačbe (6.8) reparametriziramo enako kot v primeru aproksimacije po metodi najmanjših kvadratov in dobimo

$$\begin{aligned} \int_{\Omega} f(x_1, x_2) \cdot \psi_j(x_1, x_2) d\mathbf{x} = \\ \int_{[0,1]^2} f(G(\xi_1, \xi_2)) \cdot \varphi_j(\xi_1, \xi_2) \cdot |\det J(\xi_1, \xi_2)| d\xi. \end{aligned}$$

6.2 NUMERIČNI POSKUSI

V tem delu si ogledamo nekaj numeričnih poskusov aproksimacije po metodi najmanjših kvadratov in reševanja Poissonove enačbe s hierarhičnimi prostori B-zlepkov in škatlastih zlepkov. Glavni namen numeričnih poskusov je pokazati, da je reševanje problemov opisanih v poglavju 6.1 s hierarhičnimi bazami učinkovito. To pokažemo na način, da vsak obravnavan primer rešimo s hierarhično bazo zlepkov in hierarhično rešitev primerjamo z navadno tj. rešitev z navadno bazo, kjer zgostimo celotno domeno naše aplikacije. Na tem mestu velja še enkrat opozoriti na vse sprejete omejitve, opisane v poglavju 5, s katerimi se nekoliko odmaknemo od vse splošnosti opisane teorije. Poleg tega implementacija ne vsebuje samodejne adaptivnosti, opisane v 4.2.2. To pomeni, da napisani Matlab programi nimajo implementirane funkcije, ki bi lahko glede na neko predpisano napako sama določala območja zgoščevanja. To pa ne pomeni, da tega ne moremo narediti ročno, kot bomo videli v nadaljevanju.

Poglavje ima sledečo strukturo. Najprej obravnavamo red konvergence. S tem preverimo, ali imajo implementirane metode enak red konvergence k točni rešitvi kot bi ga pričakovali v teoriji. Za tem pa se poglavje zopet razdeli na dva dela. Najprej obravnavamo B-zlepke in z njimi rešimo nekaj preprostih problemov iz aproksimacije in reševanja PDE. Enako storimo tudi s škatlastimi zlepki.

6.2.1 Red konvergence

Zanima nas red konvergence aproksimacije po metodi najmanjših kvadratov ter Galerkinove metode za reševanje Poissonove enačbe v L^2 normi. Na tem mestu še ne uporabljamo hierarhičnih prostorov, saj želimo preveriti konvergenco ob zgoščevanju celotnega območja. Začnemo z B-zlepki in tenzorskimi produkti B-zlepkov. Za B-zlepke stopnje p pričakujemo red konvergence pri aproksimaciji po metodi najmanjših kvadratov enak $r = p + 1$. Več podrobnosti o teoriji reda aproksimacije bralec najde v [3, 14]. Red konvergence želimo preveriti z numeričnim poskusom. Za začetek definirajmo L^2 napako.

Definicija 6.6. Naj bosta f in g realni funkciji in Ω območje. Normirano L^2 napako definiramo kot

$$\|f - g\|_{L^2} := \left(\frac{\int_{\Omega} (f(x) - g(x))^2 dx}{\int_{\Omega} f^2(x) dx} \right)^{\frac{1}{2}}.$$

Potem izberemo funkcijo f , ki jo bomo aproksimirali. V primerih kasneje bomo obravnavali tako funkcije ene kot funkcije dveh spremenljivk, postopek pa bomo na tem mestu zapisali za primer dveh spremenljivk. Naj bo $h \in \mathbb{R}$ in $\Omega = [a, b] \times [c, d]$. Dalje naj bosta $U_h = (a := u_0, u_1, \dots, u_{n+2p+1} := b)$ in $V_h = (c := v_0, v_1, \dots, v_{n+2p+1} := d)$, kjer je p polinomska stopnja. Za vektorja U_h in V_h naj velja, da je prvih in zadnjih $p + 1$ vozlov enakih, medtem ko za vozle v notranjosti velja, da so ekvidistantno razporejeni, torej

$$u_{i+1} - u_i = h, \quad v_{i+1} - v_i = h,$$

za vse $i = p + 1, \dots, p + n$. Z vektorjema vozlov U_h, V_h ter polinomsko stopnjo p definiramo prostor $\mathbb{S}_{U_h, V_h}^{p, q}$. V tem prostoru po metodi najmanjših kvadratov poiščemo aproksimant za funkcijo f . Označimo ga z \tilde{f}_h . Napako aproksimacije določimo po definiciji 6.6. Označimo

$$e_{L^2}^h = \|f - \tilde{f}_h\|_{L^2}.$$

Nadalje lahko za isto polinomsko stopnjo p definiramo dvakrat finejša vektorja vozlov $U_{\frac{h}{2}}$ in $V_{\frac{h}{2}}$, za katera še vedno velja, da imata prvih in zadnjih $p + 1$ vozlov enakih, v notranjosti pa so enako kot prej razporejeni ekvidistantno z razmakom $\frac{h}{2}$. Zopet

definiramo prostor zlepkov in v njem poiščemo aproksimacijo funkcije f po metodi najmanjših kvadratov. Ta aproksimant označimo z $\tilde{f}_{\frac{h}{2}}$ in izračunamo napako

$$e_{L^2}^{\frac{h}{2}} = \|f - \tilde{f}_{\frac{h}{2}}\|_{L^2}.$$

Napaki $e_{L^2}^h$ in $e_{L^2}^{\frac{h}{2}}$ lahko zapišemo kot izraza v odvisnosti od r -ja kot

$$\begin{aligned} e_{L^2}^h &= Ch^r + O(h^{r+1}) \\ e_{L^2}^{\frac{h}{2}} &= C \left(\frac{h}{2}\right)^r + O\left(\frac{h^{r+1}}{2}\right), \end{aligned}$$

kjer je $C \in \mathbb{R}$. Pri obeh zanemarimo drugi člen na desni strani in izraza med seboj delimo. Potem dobimo

$$\begin{aligned} \frac{e_{L^2}^h}{e_{L^2}^{\frac{h}{2}}} &= \frac{(h)^r}{\left(\frac{h}{2}\right)^r} = 2^r \Rightarrow \\ r_h &:= r = \log_2 \left(\frac{e_{L^2}^h}{e_{L^2}^{\frac{h}{2}}} \right). \end{aligned}$$

Na enak način lahko definiramo še finejše vektorje vozlov in s tem izračunamo napake $e_{L^2}^{\frac{h}{4}}, e_{L^2}^{\frac{h}{8}} \dots$ ter $r_{\frac{h}{2}}, r_{\frac{h}{4}}, \dots$. Pričakujemo, da zaporedje teh r -jev konvergira proti $p+1$ torej

$$r_h, r_{\frac{h}{2}}, r_{\frac{h}{4}} \cdots \longrightarrow p+1.$$

V nadaljevanju si pogledamo dva primera reda konvergence v L^2 normi za primer B-zlepkov ene spremenljivke in tenzorskih produktov B-zlepkov. V obeh primerih si izberemo funkcijo, ki jo želimo aproksimirati. V primeru obeh dimenzij vzamemo preprosto območje, in sicer $\Omega = [0, 1]$ v primeru ene spremenljivke ter $\Omega = [0, 1]^2$ v primeru dveh. Pri primeru tenzorskih produktov B-zlepkov si zadevo še nekoliko poenostavimo in za polinomski stopnji p in q privzamemo, da velja $p = q$. Izračunamo nekaj zaporednih približkov in v tabeli poročamo rezultate za vsako razmerje. Pri računanju je zaradi medsebojnega deljenja relativno majhnih števil potrebna nekoliko večja natančnost. V ta namen uporabimo Gaussova kvadratura pravila s šestimi točkami iz poglavja v 6.1.2, ki ji implementiramo, kot je prikazano v prilogi H. Še pred primeroma reda konvergence zapišimo pomembno opombo.

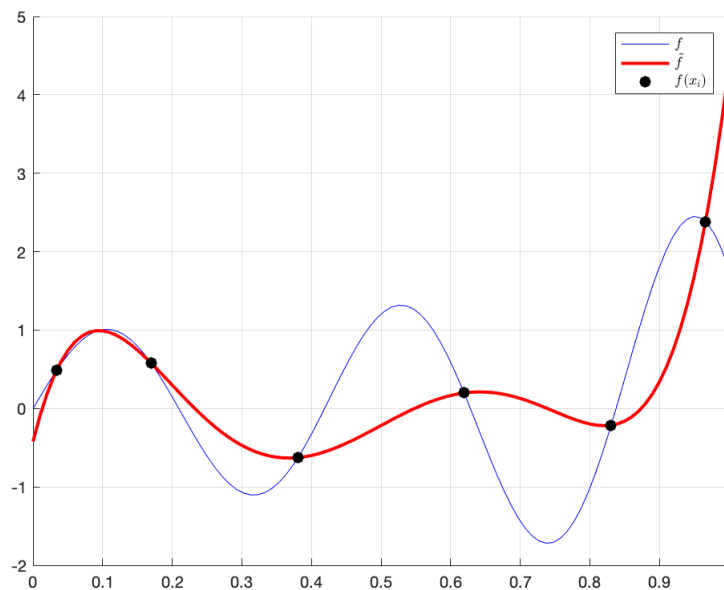
Opomba 6.7. Naj bo $f : \mathbb{R} \rightarrow \mathbb{R}$ zvezna na intervalu $[a, b]$ in naj bo \tilde{f} aproksimant za f po metodi najmanjših kvadratov iz prostora polinomov stopnje vsaj 5, kjer smo za izračun vseh skalarnih produktov uporabili Gaussovo integracijsko pravilo na šestih točkah x_i iz tabele 4, kot je opisano v poglavju 6.1.2. Potem \tilde{f} očitno interpolira funkcijo f v vseh šestih točkah Gaussovega pravila x_i . Nadalje izračunajmo še L^2 normo iz definicije 6.6, kjer za izračun integralov uporabimo enako Gaussovo pravilo

na istih šestih točkah. Sledi

$$\begin{aligned} \|f - \tilde{f}\|_{L^2} &= \left(\frac{\int_a^b (f(x) - \tilde{f}(x))^2 dx}{\int_a^b f^2(x) dx} \right)^{\frac{1}{2}} \approx \\ &\approx \left(\frac{\sum_{i=0}^5 (f(x_i) - \tilde{f}(x_i))^2}{\sum_{i=0}^5 f^2(x_i)} \right)^{\frac{1}{2}}. \end{aligned}$$

Ker \tilde{f} interpolira f v vseh točkah x_i , je $\|f - \tilde{f}\|_{L^2} = 0$, kar pa ni vedno nujno res, niti smiselno kot nam ilustrira primer 6.8. Za ta namen, kadar uporabimo Gaussovo metodo na šestih točkah za aproksimacijo s polinomi stopnje vsaj 5, za izračun L^2 napake vzamemo raje kakšno drugo metodo integriranja.

Primer 6.8. Naj bo $f(x) = e^{x^2} \sin(15x)$. Funkcijo aproksimiramo nad intervalom $[0, 1]$ po metodi najmanjših kvadratov s funkcijo iz prostora B-zlepkov stopnje 5, definirane nad vektorjem vozlov $U = (0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1)$, kar je dejansko polinomski primer in po opombi 6.7 je L^2 norma napake, če za integracijo uporabimo Gaussovo pravilo na šestih točkah, enaka 0.



Slika 45: Funkcija $f(x) = e^{x^2} \sin(15x)$ in njen aproksimant na intervalu $[0, 1]$.

Čeprav je L^2 norma, izračunana na ta način, enaka 0, slika 45 jasno pokaže, da temu ni tako. Za realno oceno napake izračunajmo L^2 napako še enkrat s tem, da tokrat uporabimo Matlabov že vgrajen ukaz `integrate`. Definiramo tudi napako v smislu neskončne norme na intervalu $[a, b]$ kot

$$\|f - \tilde{f}\|_{L^\infty} := \sup_{x \in [a, b]} |f(x) - \tilde{f}(x)|.$$

Podatke treh različnih norm zberemo v tabeli 5. Neskončno normo izračunamo diskretno kot absolutno maksimalno razliko med funkcijo in aproksimantom na diskretizaciji intervala $[a, b]$.

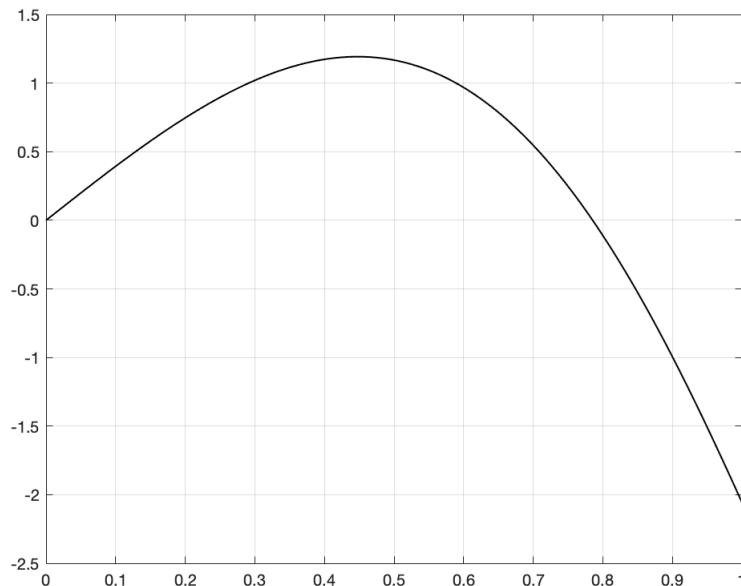
Tabela 5: Primerjava različnih napak pri aproksimaciji funkcije $f(x) = e^{x^2} \sin(15x)$.

$\ f - \tilde{f}\ _{L^2}$ (Gaussova integ.)	$\ f - \tilde{f}\ _{L^2}$ (Vgrajena integ.)	$\ f - \tilde{f}\ _{L^\infty}$
1.458642324041090e-14	0.792856063165304	2.610034353353433

Iz tabele 5 je očitno, da je napaka, izračunana s pomočjo vgrajene integracijske metode bolj smiselna kot tista z implementiranim Gaussovimi pravilom na šestih točkah. Potrebno bi bilo izbrati Gaussovo pravilo na vsaj $p + 2$ točkah, potem do te težave ne bi prišlo.

Sedaj imamo pripravljeno vse, da si pogledamo dva primera konvergence reda aproksimacije v L^2 normi.

Primer 6.9. Naj bo $\Omega = [0, 1]$ in $f(x) = e^{x^2} \sin 4x$ (slika 46). Funkcijo aproksimiramo z B-zlepki stopenj 2, 3, 4 in 5. Rezultate zberemo v tabeli 6.



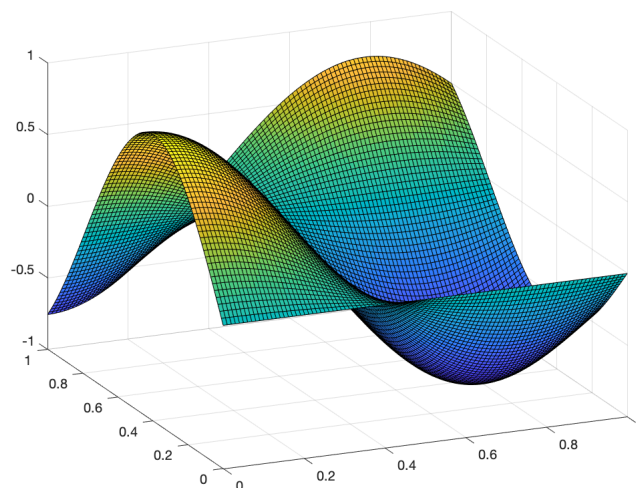
Slika 46: Funkcija $f(x) = e^{x^2} \sin(4x)$ na intervalu $[0, 1]$.

Z naslednjim primerom preizkusimo še red konvergence v primeru tenzorskih produktov B-zlepkov. Zaradi večje časovne zahtevnosti se tukaj nekoliko omejimo in naredimo le za primere polinomskih stopenj 2 in 3 ter izračunamo manj členov zaporedja.

Tabela 6: Red konvergence aproksimacije funkcije $f(x) = e^{x^2} \sin(4x)$ v prostoru prostoru B-zlepkov stopenj 2, 3, 4, 5.

Ocena redov konvergence				
h	r_h pri $p = 2$	r_h pri $p = 3$	r_h pri $p = 4$	r_h pri $p = 5$
1	1.9540	1.5918	2.0982	-37.3894
$\frac{1}{2}$	2.4482	3.5754	3.5152	3.0857
$\frac{1}{4}$	3.0616	3.5055	4.5961	5.9177
$\frac{1}{8}$	2.9469	3.7833	4.9010	5.9795
$\frac{1}{16}$	2.9405	3.9075	4.9582	5.9821
$\frac{1}{32}$	2.9631	3.9577	4.9794	5.9882

Primer 6.10. Naj bo $\Omega = [0, 1]^2$ in funkcija, ki jo aproksimiramo $f(x, y) = \cos(4x) \cdot \sin(4y)$ (slika 47). Funkcijo aproksimiramo s tenzorskimi produkti B-zlepkov stopenj $p = q = 2$ in $p = q = 3$. Rezultate reda konvergence zberemo v tabeli 7.

Slika 47: Funkcija $f(x, y) = \cos(4x) \cdot \sin(4y)$ na območju $[0, 1]^2$.

Naslednja stvar, ki jo želimo preveriti, je red konvergence Galerkinove metode za reševanje Poissonove enačbe s homogenim robnim pogojem, kjer za prostor testnih funkcij vzamemo tenzorske produkte B-zlepkov. Sicer pričakujemo enak red konvergence kot v primeru aproksimacije po metodi najmanjših kvadratov, vendar nekaj dvoma vseeno ostaja. Testne funkcije morajo izpolnjevati homogen robni pogoj na območju, nad katerim rešujemo Poissonovo enačbo. Pri tenzorskih produktih B-zlepkov to seveda ne velja avtomatsko. Spomnimo se poglavja 2.2, kjer smo definirali zaprte krivulje iz prostora B-zlepkov, tako da smo v vektorju vozlov prvi in zadnji vozlov ponovili $(p + 1)$ -krat. S tem pa smo v prostoru B-zlepkov dobili dve bazni funkciji, ki

Tabela 7: Red konvergence aproksimacije funkcije $f(x, y) = \cos(4x) \cdot \sin(4y)$ v prostoru tenzorskih produktov B-zlepkov stopenj 2 in 3.

Ocena redov konvergence		
h	r_h pri $p = q = 2$	r_h pri $p = q = 3$
1	1.5920	2.3203
$\frac{1}{2}$	2.6099	2.8376
$\frac{1}{4}$	2.9159	4.2815
$\frac{1}{8}$	2.9536	4.1444
$\frac{1}{16}$	2.9784	4.0418

ne izpolnjujeta homogenega robnega pogoja (glej sliko 8). Homogene bazne funkcije dobimo na način, da ti dve funkciji iz baze enostavno odstranimo, vendar smo s tem izgubili dve bazni funkciji v primeru B-zlepkov. V primeru tenzorskih produktov B-zlepkov pa še več. Razliko v številu baznih funkcij polne baze in homogene baze za nekaj primerov zberemo v tabeli 8. S p označimo stopnjo B-zlepkov in privzamemo, da je $p = q$. S k označimo število notranjih vozlov v vektorjih vozlov, ki določata prostor tenzorskih produktov B-zlepkov. Zopet predpostavimo, da so notranji vozli enostavni, vsak robni vozlel pa ponovimo $(p + 1)$ -krat. Tako imamo skupaj $k + 2p$ vozlov. V tabeli z B_H označimo število baznih funkcij, ki izpolnjujejo homogen robni pogoj, z B_P pa označimo število vseh baznih funkcij. Z naslednjim primerom preverimo red

Tabela 8: Razlika v številu baznih funkcij prostorov, kjer je za bazne funkcije izpolnjen homogen robni pogoj in tistih, kjer ni.

	$k = 1$		$k = 3$		$k = 7$		$k = 15$		$k = 31$		$k = 63$	
p	B_P	B_H	B_P	B_H	B_P	B_H	B_P	B_H	B_P	B_H	B_P	B_H
2	16	4	36	16	100	64	324	256	1156	1024	4356	4096
3	25	9	49	25	121	81	361	289	1225	1089	4489	4225
4	36	16	64	36	144	100	400	324	1296	1156	4624	4356
5	49	25	81	49	169	121	441	361	1369	1225	4761	4489

konvergence Galerkinove metode v L^2 normi.

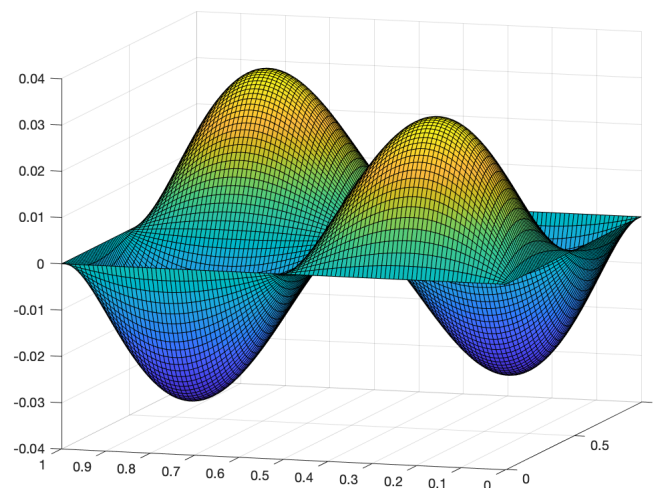
Primer 6.11. Rešujemo homogeno Poissonovo enačbo

$$\begin{aligned}
 -\Delta u(x, y) = & -2\pi^2 \sin(\pi x) \sin(\pi y) \left(\frac{1}{2} - x\right) \left(\frac{1}{2} - y\right) \\
 & + 2\pi \sin(\pi y) \cos(\pi x) \left(\frac{1}{2} - y\right) \\
 & + 2\pi \sin(\pi x) \cos(\pi y) \left(\frac{1}{2} - x\right), \quad (x, y) \in [0, 1]^2, \\
 u(x, y) = & 0, \quad (x, y) \in \partial[0, 1]^2.
 \end{aligned} \tag{6.14}$$

Točna rešitev, ki jo iščemo, je enaka

$$u(x, y) = \sin(\pi x) \sin(\pi y) \left(x - \frac{1}{2}\right) \left(y - \frac{1}{2}\right)$$

in je prikazana na sliki 48. Enačbo rešimo še s pomočjo Galerkinove metode z uporabo tenzorskih produktov B-zlepkov stopenj 2 in 3. Tudi tukaj se bomo lotili enakega postopka, kot pri aproksimaciji po metodi najmanjših kvadratov v primerih 6.9 in 6.10. Razlika je, da bomo tokrat uporabili prostor tenzorskih produktov B-zlepkov, čigar funkcije izpolnjujejo homogen robni pogoj na območju $\Omega = [0, 1]^2$. Tudi tukaj bomo s povsem enakim postopkom računali zaporedje, ki bi po pričakovanjih morale konvergirati k pričakovanemu redu konvergence $p + 1$, kjer je p polinomska stopnja prostora tenzorskih produktov B-zlepkov. Tudi v tem primeru predpostavimo, da za vektorja vozlov U, V in polinomske stopnji p, q velja $U = V$ in $p = q$. Rezultate zberemo v tabeli 9.



Slika 48: Funkcija $u(x, y) = \sin(\pi x) \sin(\pi y) \left(x - \frac{1}{2}\right) \left(y - \frac{1}{2}\right)$ na območju $\Omega = [0, 1]^2$.

Tabela 9: Red konvergence Galerkinove metode s tenzorskimi produkti B-zlepkov stopenj 2 in 3 za Poissonovo enačbo (6.14).

Ocena redov konvergence		
h	$p = 2$	$p = 3$
1	3.1136	0.0137
$\frac{1}{2}$	2.2230	4.0771
$\frac{1}{4}$	3.6351	4.1039
$\frac{1}{8}$	3.2202	4.1368
$\frac{1}{16}$	3.0513	4.0384

Iz tabele 9 je vidno, da ima tudi Galerkinova metoda s tenzorskimi produkti B-zlepkov stopnje p red konvergence $p + 1$.

Red konvergence bi želeli na enak način preveriti tudi za primer škatlastih zlepkov. V tem sklopu magistrskega dela se nekoliko omejimo in naredimo nekaj poskusov le za primer ustrezno skaliranega prostora premikov zlepka B_{222} . Ker je tudi škatlasti zlepek B_{222} lokalno gledano polinom stopnje 4 in ker so škatlasti zlepki posplošitev tenzorskih produktov B-zlepkov, bi nekako pričakovali, da ima tudi ta prostor enak red konvergence kot prostor tenzorskih produktov B-zlepkov. To pričakovanje še nekoliko bolj utemeljimo z referenco na poglavje 4.2 v članku [1]. Avtorji so v tem članku z numeričnimi poskusi, kjer so gledali interpolacijo funkcij dveh spremenljivk, nakazali, da imajo tako tenzorski produkti B-zlepkov kot škatlasti zlepki enake aproksimacijske lastnosti. To bomo poskušali potrditi v naslednjem primeru.

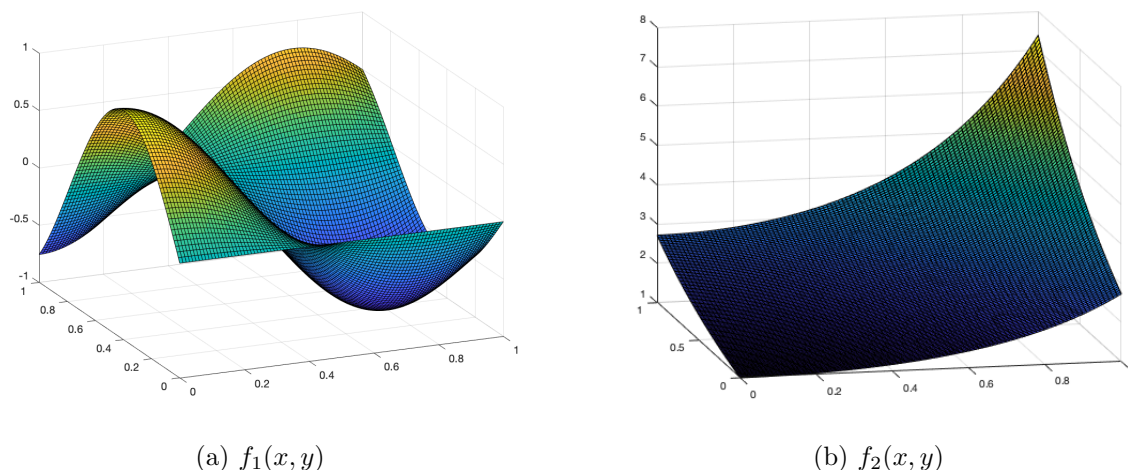
Primer 6.12. Naj bo $\Omega = [0, 1]^2$ in \mathcal{S} prostor premikov škatlastega zlepka B_{222} , katerih nosilci imajo z območjem Ω neprazen presek. Tudi s temi zlepkami na enak način kot v primeru 6.9 poskusimo pokazati red konvergence. Za ta namen najprej definiramo dve funkciji, ki jih bomo aproksimirali. Pri izbiri funkcij je potrebno nekaj previdnosti. Paziti moramo, da funkcija ni preveč „grda“, saj bi za tako funkcijo potrebovali precej korakov, preden bi dobili konvergenco (glej funkcijo f_2 na strani 320 v [1]). Po drugi strani pa funkcija ne sme biti preveč lepa, saj lahko že na prvem koraku dosežemo dovolj dobro aproksimacijo s tako majhno L_2 napako, da računanje v Matlabovi privzeti natančnosti ne bo vrnilo smiselnih rezultatov. Primer takšne funkcije najdemo v Example 1 v članku [11]. To funkcijo bomo kasneje srečali še enkrat pri primerih iz hierarhične baze škatlastih zlepkov. V tem primeru pa definirajmo funkciji

$$f_1(x, y) = \cos(4x) \cdot \sin(4y), \quad f_2(x, y) = e^{-(x^2-y)}.$$

Red konvergence potem ocenjujemo tako, da si izberemo faktor skaliranja h . Začnemo s faktorjem $h = 1$ in izračunamo aproksimant za izbrano funkcijo po metodi najmanjših kvadratov. Za to aproksimacijo potem izračunamo L_2 napako. Za razliko od primera B-zlepkov v tem primeru uporabimo nekoliko drugačno definicijo L_2 napake, saj definicije 6.6 ne moremo tako enostavno posplošiti nad polinome definirane nad trikotniki, kot smo to lahko naredili v primeru tenzorskih produktov. Odločimo se raje za diskretno definicijo L_2 napake po vzoru norme, definirane na strani 322 v članku [1], ki jo definiramo kot

$$\|f - \tilde{f}\|_{L_2, d} := \left(\frac{1}{16} h^2 \sum_{i=1}^m \sum_{j=1}^n (f(x_i, y_j) - \tilde{f}(x_i, y_j))^2 \right)^{\frac{1}{2}}, \quad (6.15)$$

kjer so x_i in y_i točke diskretizacije kvadratnega območja $\Omega = [0, 1]^2$. Enako naredimo tudi za faktorje skaliranja $h = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$. Ko imamo enkrat te napake, izračunamo

Slika 49: Grafi funkcij f_1 , f_2 iz primera 6.12.

zaporedje r -jev enako kot v primeru tenzorskih produktov B-zlepkov. Rezultate poskusa najdemo v tabeli 10.

Tabela 10: Red konvergence aproksimacije po metodi najmanjših kvadratov s prostorom premikov škatlastega zlepka B_{222} .

	f_1	f_2
h	r	r
1	2.824800659363723	3.344766148460345
$\frac{1}{2}$	4.035487411527265	4.211380475037919
$\frac{1}{4}$	5.510343638120944	5.049274968360816
$\frac{1}{8}$	5.192456993927154	5.023369790440292
$\frac{1}{16}$	4.926018681762204	4.868191272324112

Iz tabele 10 bi lahko sklepali, da je tudi v primeru prostora premikov škatlastega zlepka B_{222} red konvergence enak 5.

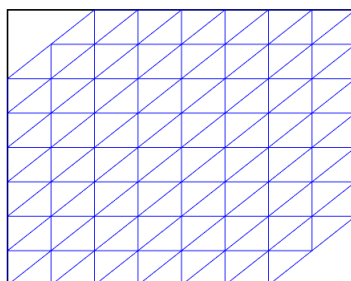
Želeli bi obravnavati še reševanje Poissonove enačbe s homogenim robnim pogojem z Galerkinovo metodo, kjer bi za prostor testnih funkcij uporabili prostor premikov zlepka B_{222} . Izkaže se, da to ni tako enostavno, kot je bilo v primeru tenzorskih produktov B-zlepkov, kjer smo v vsaki smeri odstranili le dve bazni funkciji kar pa ni vplivalo na red konvergence, ki je bila enaka kot pri aproksimaciji po metodi najmanjših kvadratov. V prostoru premikov zlepka B_{222} nad območjem Ω se nahajajo vsi premiki, ki imajo z Ω neprazen presek. Če bi iz tega prostora odstranili vse, ki ne izpolnjujejo robnih pogojev, bi nam ostale le funkcije, katerih nosilec je v celoti vsebovan v območju Ω , kar pa je bistveno premalo. Naj bo $\Omega = [0, 1]^2$. Če nad Ω definiramo prostor premikov brez skaliranja in odstranimo funkcije, ki ne izpolnjujejo homogenih robnih pogojev,

v tem prostoru nimamo več nobene funkcije. Enako se nam zgodi tudi v primeru, če zlepke skaliramo s faktorjem $h = \frac{1}{2}$. Šele v primeru skaliranja z eno četrtnino imamo v prostoru eno samo bazno funkcijo. Tudi z nadaljnjim definiranjem finejših prostorov imamo še vedno relativno malo funkcij, kot lahko razberemo iz tabele 11. Število vseh funkcij označimo z B_P , število funkcij, ki izpolnjujejo homogen robni pogoj, pa z B_H .

Tabela 11: Primerjava števila baznih funkcij s faktorjem h skaliranega polnega prostora premikov zlepka B_{222} in tistega, kjer bazne funkcije izpolnjujejo homogen robni pogoj.

Število baznih funkcij		
h	B_P	B_H
1	14	0
$\frac{1}{2}$	23	0
$\frac{1}{4}$	47	1
$\frac{1}{8}$	119	25
$\frac{1}{16}$	359	169
$\frac{1}{32}$	1223	841
$\frac{1}{64}$	4487	3721

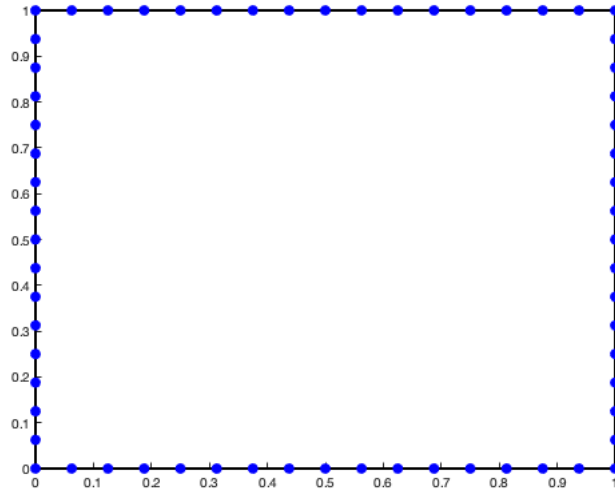
Iz tabele 11 opazimo, da je baznih funkcij res malo. Celo tako malo, da imamo do faktorja skaliranja $h = \frac{1}{32}$ manj kot polovico funkcij, ki na območju $[0, 1]^2$ izpolnjujejo homogen robni pogoj. Poleg tega na tak način z baznimi funkcijami ne moremo nikoli, ne glede na to, kako fino delitev vzamemo, v celoti pokriti nobenega pravokotnega območja. To se zgodi iz preprostega razloga, ker imajo nosilci zlepka B_{222} obliko šestkotnika in z njimi, ne glede na to, kako majhni so, ne moremo nikoli pokriti pravokotnika. Vedno nam ostane nekaj nepokritega prostora v zgornjem levem in spodnjem desnem kotu, kot je to prikazano na sliki 50, za faktor skaliranja $h = \frac{1}{8}$.



Slika 50: Unija nosilcev vseh premikov zlepka B_{222} , skaliranih s faktorjem $h = \frac{1}{8}$, ki so v celoti vsebovani v območju $\Omega = [0, 1]^2$.

V nadaljevanju predpostavimo, da je faktor skaliranja $h < 1$, območje $\Omega = [0, 1]^2$, prostor premikov zlepka B_{222} skaliranega s faktorjem h definirane nad Ω pa označimo

s \mathcal{S} . Ideja je pridobiti še nekaj dodatnih funkcij, ki bi izpolnjevale homogen robni pogoj kot linearne kombinacije nekaterih funkcij prostora \mathcal{S} . To naredimo tako, da na robu območja Ω izberemo nekaj točk. Za faktor skaliranja h izberemo na vsaki stranici območja $(4\frac{1}{h} + 1)$ točk. Prikaz točk za faktor $h = \frac{1}{4}$ vidimo na sliki 51.



Slika 51: Kvadrat $[0, 1]^2$ pokrit na vsaki stranici s 17 točkami.

Vse točke na robu območja označimo z X . Sedaj želimo najti vse možne linearne kombinacije funkcij prostora \mathcal{S} , ki bi imele v vseh točkah X vrednost 0. Ker je zlepek B_{222} lokalno gledano polinom stopnje 4 in ker smo na vsaki stranici izbrali $(4\frac{1}{h} + 1)$ točk, sledi, da bodo te linearne kombinacije ničelne na celotnem robu območja Ω . Problem formuliramo v jeziku matrik. Naj bo $\{\varphi_i\}_{i=1,2,\dots,\dim \mathcal{S}}$ baza prostora \mathcal{S} in naj bo $T \in \mathbb{R}^{|X| \times \dim \mathcal{S}}$ enaka

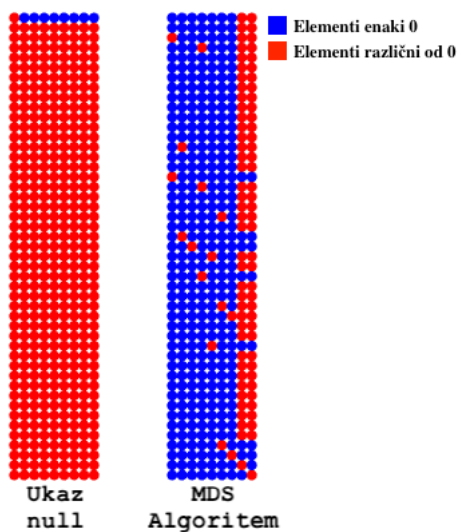
$$T = \begin{bmatrix} \varphi_1(X) & \varphi_2(X) & \cdots & \varphi_{\dim \mathcal{S}}(X) \end{bmatrix}.$$

Vsak stoplec matrike T vsebuje vrednosti določene bazne funkcije, ovrednotene v točkah množice X . Ker morajo linearne kombinacije izpolnjevati homogen robni pogoj, iščemo torej takšne vektorje $\mathbf{x} \in \mathbb{R}^{\dim \mathcal{S}}$, ki so rešitev sistema

$$T\mathbf{x} = \mathbf{0}. \quad (6.16)$$

Iščemo torej jedro matrike T . Tukaj pa nastopi tehnična težava. V Matlabu vgrajen ukaz, ki izračuna jedro matrike `null` je sicer učinkovit, vendar ta rešitev ni najlepša. Želeli bi si, da v linearnih kombinacijah nastopa čim manj baznih funkcij s čim lepšimi koeficienti. V ta namen uporabimo implementiran algoritem za iskanje minimalne določitvene množice, opisan v poglavju 6.1.1. Za ilustracijo si pogledajmo primer 6.13, kjer si pogledamo razliko med jedrom, izračunanim z vgrajenim ukazom, in jedrom, izračunanim s pomočja algoritma minimalne določitvene množice.

Primer 6.13. Naj bo $h = \frac{1}{4}$ faktor skaliranja. Nad območjem $\Omega = [0, 1]^2$ želimo konstruirati prostor premikov zlepkov B_{222} s homogenimi robnimi pogoji. Do matrike T pridemo z zgoraj opisanim postopkom, potem pa izračunamo linearne kombinacije kot jedro matrike T na dva različna načina. Najprej z ukazom `null` in potem še s pomočjo implementiranega algoritma minimalne določitvene množice. Matrika T je velikosti 64×47 . Vseh točk na robu območja je torej 64, vseh baznih funkcij prostora \mathcal{S} pa je 47. Homogena baza bo velikosti dimezije jedra matrike T . V konkretnem primeru se izkaže, da je $\dim(\ker(T)) = 9$. Na sliki 52 vidimo primerjavo med dvema načinoma izračuna jedra matrike. Iz slike 52 je očitno, da ima jedro, pridobljeno z



Slika 52: Primerjava števila ničelnih elementov jeder matrik, izračunanih na dva različna načina.

MDS algoritmom, veliko lepšo strukturo z veliko ničelnimi elementi, kar nam ob še dodatnem normiranju predstavlja dodatno numerično stabilnost.

Primer lahko ponovimo še za druge manjše faktorje skaliranja in opazimo, da vsi primeri sledijo enaki strukturi. Algoritem najde vse bazne funkcije, ki so v celoti vsebovane v območju Ω , in jih izpostavi kot linearne kombinacije z enim samim neničelnim členom. Večina ostalih linearnih kombinacij pa je vsota največ nekaj različnih funkcij, kar nakazuje, da za večino linearnih kombinacij obstajajo preprosta pravila, kako konstruirati homogene bazne funkcije tudi brez algoritma. Izstopata običajno le zadnji dve funkciji, ki pa kot linearni kombinaciji vključujeta večino baznih funkcij. Tudi te dve funkciji vključimo v homogeno bazo. V tabeli 12 za nekaj različnih faktorjev skaliranja zberemo podatke o številu baznih funkcij, konstruiranih z algoritmom minimalne določitvene množice. Zopet z B_P označimo vse bazne funkcije v prostoru \mathcal{S} , z $B_H(MDS)$ vse homogene bazne funkcije, pridobljene z implementiranim algoritmom, za primerjavo pa dodamo še število funkcij iz tabele 11, ki jih ponovno označimo z B_H .

Tabela 12: Primerjava števil baznih funkcij s faktorjem h skaliranega prostora premikov zlepka B_{222} v primeru vseh funkcij in homogenih funkcij, pridobljenih z algoritmom minimalne določitvene množice.

Število baznih funkcij			
h	B_P	B_H (MDS)	B_H (notranje)
1	14	0	0
$\frac{1}{2}$	23	1	0
$\frac{1}{4}$	47	9	1
$\frac{1}{8}$	119	49	25
$\frac{1}{16}$	359	225	169
$\frac{1}{32}$	1223	961	841
$\frac{1}{64}$	4487	3969	3721

V tabeli opazimo, da se nam je s pomočjo algoritma število baznih funkcij nekoliko povečalo. To je izrazito predvsem za nekoliko večje faktorje skaliranja, medtem ko se z manjšanjem faktorja zmanjšuje tudi razlika med funkcijami, izračunanimi z algoritmom in le številom notranjih funkcij. Imamo pa na ta način vedno vsaj z nosilci pokrito celotno kvadratno območje $\Omega = [0, 1]^2$. Sedaj lahko pričakujemo, da bomo s takšno bazo za dovolj majhen faktor skaliranja pridobili konvergenco k pravi rešitvi pri reševanju Poissonove enačbe z Galerkinovo metodo, čeprav bo ta verjetno počasnejša kot pri aproksimaciji po metodi najmanjših kvadratov. Hipotezo preverimo v naslednjem primeru.

Primer 6.14. Poskušamo določiti red konvergence Galerkinove metode za reševanje Poissonove enačbe pri uporabi skaliranega prostora premikov škatlastega zlepka B_{222} s homogenimi robnimi pogoji. Rešujemo torej enačbo

$$-\Delta u(x, y) = f_i(x, y), \quad (x, y) \in [0, 1]^2 \text{ in } i = 1, 2, 3, 4$$

$$u(x, y) = 0, \quad (x, y) \in \partial[0, 1]^2,$$

kjer so

$$f_1(x, y) = 2(x - 1)x + 2(y - 1)y,$$

$$f_2(x, y) = 2\pi^2 \sin(\pi x) \sin(\pi y),$$

$$f_3(x, y) = 2\pi \sin(\pi y) \left(\cos(\pi x) - \pi \left(x - \frac{1}{2}\right) \sin(\pi x) \right),$$

$$f_4(x, y) = -2\pi^2 \sin(\pi x) \sin(\pi y) \left(\frac{1}{2} - x \right) \left(\frac{1}{2} - y \right)$$

$$+ 2\pi \sin(\pi y) \cos(\pi x) \left(\frac{1}{2} - y \right)$$

$$+ 2\pi \sin(\pi x) \cos(\pi y) \left(\frac{1}{2} - x \right).$$

Za vse štiri Poissonove probleme poznamo točne rešitve. Označimo jih z u_i , kjer vsaka rešitev pripada ustrezni funkciji f_i . Te rešitve so enake

$$\begin{aligned} u_1(x, y) &= xy(x-1)(y-1), & u_2(x, y) &= \sin(\pi x) \sin(\pi y), \\ u_3(x, y) &= \sin(\pi x) \sin(\pi y) \left(x - \frac{1}{2}\right), & u_4(x, y) &= \sin(\pi x) \sin(\pi y) \left(x - \frac{1}{2}\right) \left(y - \frac{1}{2}\right). \end{aligned}$$

Vse štiri enačbe rešimo s pomočjo baze, pridobljene z algoritmom minimalne določitvene množice, na enak način kot do sedaj in enako poskusimo določiti red konvergence v smislu L^2 napake. Rezultate zberemo v tabeli 13.

Tabela 13: Določanje reda konvergence Galerkinove metode z uporabo homogene baze premikov škatlastega zlepka B_{222} .

Zaporedje r -jev				
h	f_1	f_2	f_3	f_4
1	1.9128	1.7968	1	0.9172
$\frac{1}{2}$	3.8137	3.4379	3.6791	2.5369
$\frac{1}{4}$	2.1170	2.3253	2.5637	2.9301
$\frac{1}{8}$	1.8012	1.7595	1.7530	2.3675
$\frac{1}{16}$	2.9211	2.5976	2.4515	2.5208

Iz tabele 13 opazimo vzorec, ki morda nakazuje na konvergenco in že dejstvo, da so vsa števila pozitivna, nakazuje, da se na vsakem koraku napaka manjša. Najbolj merodajni sta zadnji dve vrstici, saj smo imeli v prostorih z manjšimi faktorji skaliranja relativno malo baznih funkcij oz. v primeru skaliranja s faktorjem $\frac{1}{2}$ le eno samo bazno funkcijo, v primeru brez skaliranja pa baznih funkcij sploh nismo imeli. Zaključimo lahko samo, da metoda deluje in z njo lahko rešujemo Poissonovo enačbo, ni pa ta metoda morda optimalnega reda. Izboljšave te metode za reševanje Poissonovega problema z uporabo prostorov premikov škatlastih zlepkov imamo podane v [10, 26].

6.2.2 Aproksimacija

V tem delu si pogledamo nekaj numeričnih poskusov uporabe hierarhičnih prostorov zlepkov pri aproksimaciji po metodi najmanjših kvadratov. Poglavje začnemo z uporabo hierarhičnih tenzorskih produktov B-zlepkov za aproksimacijo nekaj funkcij dveh spremenljivk. Za tem iste funkcije aproksimiramo še z uporabo hierarhičnega prostora premikov zlepka B_{222} . Poglavje končamo s primerjavo učinkovitosti obeh vrst zlepkov in pokažemo primer, kako funkcijo preslikamo iz kvadratne na poljubno štirikotno domeno v \mathbb{R}^2 . Pri primerih uporabe tenzorskih produktov B-zlepkov vedno uporabimo prirezano hierarhično bazo. Pri vseh primerih nas bo zanimala napaka v smislu diskretne neskončne norme na celotnem območju aplikacije, število potrebnih baznih funkcij

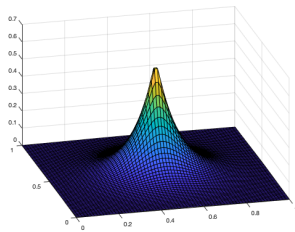
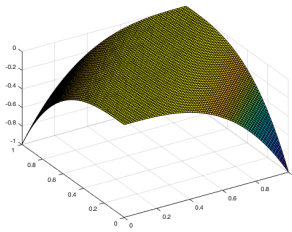
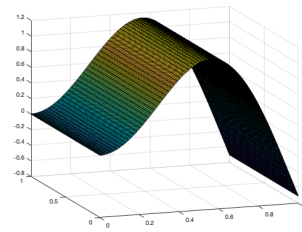
ter območje, kjer je bila potrebna zgostitev. V naslednjem primeru si pogledamo aproksimacijo treh funkcij z uporabo prirezane baze tenzorskih produktov B-zlepkov.

Primer 6.15. Naj bo $\Omega = [0, 1]^2$. Na Ω definiramo naslednje funkcije

$$f_1(x, y) = \frac{2}{3e^{\sqrt{10(x-\frac{1}{2})^2+10(y-\frac{1}{2})^2}}},$$

$$f_2(x, y) = \begin{cases} (x-y)^3, & x \leq y, \\ -(x-y)^3, & x > y \end{cases},$$

$$f_3(x, y) = \begin{cases} -\cos(\frac{3\pi}{2}x) + \cos(\frac{3\pi}{4}x) - x^3 \sin(x - \frac{1-3\pi}{3}) \sin(x - \frac{2-3\pi}{3}), & \frac{1}{3} < x < \frac{2}{3}, \\ -\cos(\frac{3\pi}{2}x) + \cos(\frac{3\pi}{4}x), & \text{sicer} \end{cases}.$$

(a) $f_1(x, y)$ (b) $f_2(x, y)$ (c) $f_3(x, y)$

Slika 53: Grafi funkcij f_1 , f_2 , f_3 iz primera 6.15.

Vse tri funkcije imamo prikazane na sliki 53. Problem rešujemo na sledeči način. Območje Ω diskretiziramo, izberemo stopnjo in določimo začetni vektor vozlov. Vektor vozlov izberemo zopet takšen, da je prvi vozle enak 0, zadnji vozle pa 1. Prvi in zadnji vozle ponovimo $(p+1)$ -krat, vozle v sredini pa razporedimo ekvidistantno. Za začetek si izberemo tak vektor vozlov, da na sredini nimamo nobenega vozla. Potem sestavimo hierarhično bazo in pogledamo napako v neskončni normi na celotnem območju. Za tem pogledamo območje, kjer je neskončna napaka večja od izbrane tolerance ϵ . To območje zgostimo, tako, da je vektor vozlov nad tem območjem dvakrat finejši. Postopek ponavljamo, dokler ni napaka na celotnem območju manjša od predpisane napake ϵ . Privzamemo tudi, da je $U = V$ ter $p = q$ ter toleranca napake $\epsilon = 10^{-4}$. Rezultate poskusov zberemo v tabeli 14. Poskuse izvedemo za vse tri funkcije s polinomskimi stopnjami $p = q = 3$ ter $p = q = 4$. Izpustimo le aproksimacijo funkcije f_3 z zlepki stopnje 4, zaradi prevelike računske zahtevnosti. V tabeli označimo napako v smislu neskončne norme nad celotnim območjem Ω z $\|f - \tilde{f}\|_{L^\infty}$, območje, ki ga je potrebno zgostiti z $\hat{\Omega}$, ter število baznih funkcij z $\dim(H)$. Lokalna zgostitev mrež za vsak primer posebej je prikazana na sliki 54.

V tabeli vidimo, da implementirani algoritem prirezane hierarhične baze deluje, ter da se napaka na vsakem koraku zmanjšuje. Poskuse ponovimo še enkrat z uporabo

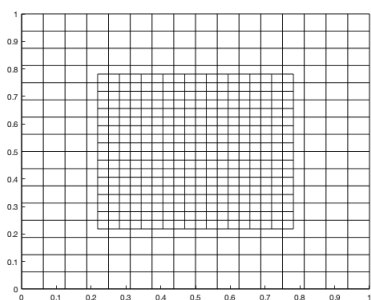
Tabela 14: Rezultati numeričnih poskusov primera 6.15.

$f_1(x, y)$, pri $\epsilon = 10^{-4}$						
h	$\ f - \tilde{f}\ _{L^\infty}$	$p = 3$		$p = 4$		
		$\hat{\Omega}$	$\dim(H)$	$\ f - \tilde{f}\ _{L^\infty}$	$\hat{\Omega}$	$\dim(H)$
1	0.4585	$[0, 1]^2$	16	0.3585	$[0, 1]^2$	25
$\frac{1}{2}$	0.3261	$[0, 1]^2$	25	0.3585	$[0, 1]^2$	36
$\frac{1}{4}$	0.2398	$[0, 1]^2$	49	0.2959	$[0, 1]^2$	64
$\frac{1}{8}$	0.1193	$[0, 1]^2$	121	0.1863	$[0, 1]^2$	144
$\frac{1}{16}$	0.0378	$[0, 1]^2$	361	0.0773	$[0, 1]^2$	400
$\frac{1}{32}$	0.0046	$[0, 1]^2$	1225	0.0184	$[0, 1]^2$	1296
$\frac{1}{64}$	$8.8064 \cdot 10^{-7}$	$[0.28, 0.72]^2$	1729	$1.0797 \cdot 10^{-4}$	$[0.22, 0.78]^2$	2124
$\frac{1}{128}$	-	-	-	$2.3273 \cdot 10^{-8}$	$[0.39, 0.61]^2$	2600
$f_2(x, y)$, pri $\epsilon = 10^{-4}$						
h	$\ f - \tilde{f}\ _{L^\infty}$	$p = 3$		$p = 4$		
		$\hat{\Omega}$	$\dim(H)$	$\ f - \tilde{f}\ _{L^\infty}$	$\hat{\Omega}$	$\dim(H)$
1	0.4585	$[0, 1]^2$	16	0.0034	$[0, 1]^2$	25
$\frac{1}{2}$	0.3261	$[0, 1]^2$	25	0.0018	$[0, 1]^2$	36
$\frac{1}{4}$	0.0014	$[0, 1]^2$	49	$7.5833 \cdot 10^{-4}$	$[0, 1]^2$	64
$\frac{1}{8}$	$2.1623 \cdot 10^{-4}$	$[0, 1]^2$	121	$1.4484 \cdot 10^{-4}$	$[0, 1]^2$	144
$\frac{1}{16}$	$1.6544 \cdot 10^{-4}$	$[0.13, 0.88]^2$	193	$7.5823 \cdot 10^{-5}$	$[0.13, 0.88]^2$	204
$\frac{1}{32}$	$8.5066 \cdot 10^{-5}$	$[0.13, 0.88]^2$	553	-	-	-
$f_3(x, y)$, pri $\epsilon = 10^{-4}$						
h	$\ f - \tilde{f}\ _{L^\infty}$	$\hat{\Omega}$	$\dim(H)$			
1	0.1800	$[0, 1]^2$	16			
$\frac{1}{2}$	0.0426	$[0, 1]^2$	25			
$\frac{1}{4}$	0.0047	$[0, 1]^2$	49			
$\frac{1}{8}$	0.0013	$[0, 1]^2$	121			
$\frac{1}{16}$	$6.0878 \cdot 10^{-4}$	$[0, 1]^2$	361			
$\frac{1}{32}$	$3.0495 \cdot 10^{-4}$	$[0.56, 0.81] \times [0, 1]$	517			
$\frac{1}{64}$	$8.5509 \cdot 10^{-5}$	$[0.63, 0.72] \times [0, 1]$	718			

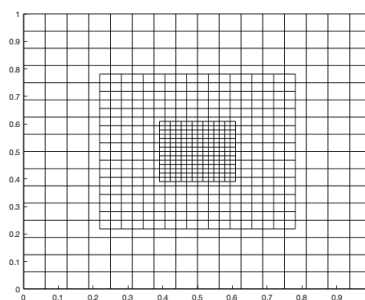
globalnega zgoščevanja domene. Za vsak primer izračunamo le na nivoju, kjer je bila napaka manjša od predpisane tolerance¹. Rezultati so podani v tabeli 15.

Po primerjanju rezultatov tabel 14 in 15 sklepamo, da je implementacija učinkovita, ter da s hierarhično bazo zlepkov ne izgubimo bistveno pri redu napake, čeprav imamo precej manjše število baznih funkcij.

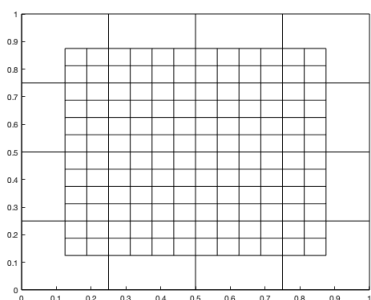
¹Zaradi prevelike računske zahtevnosti poskusa za funkcijo $f_1(x, y)$ pri $p = 4$ ne izvedemo do konca. Prikazano je le število baznih funkcij.



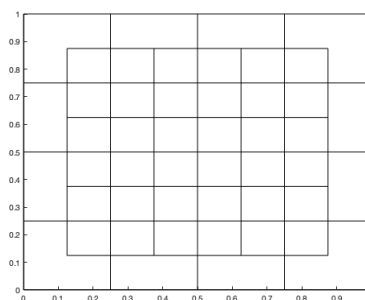
(a) $f_1(x, y)$, pri $p = 3$



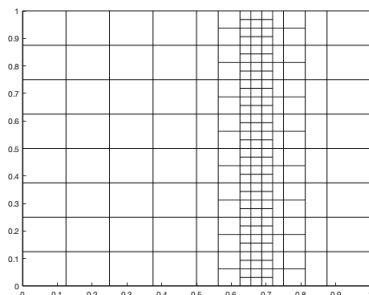
(b) $f_1(x, y)$, pri $p = 4$



(c) $f_2(x, y)$, pri $p = 3$



(d) $f_2(x, y)$, pri $p = 4$



(e) $f_3(x, y)$, pri $p = 3$

Slika 54: Prikaz lokalnega zgoščevanja domene za primer 6.15.

Tabela 15: Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo tenzor-skih produktov B-zlepkov in globalnim zgoščevanjem.

		h	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$
$f_1(x, y)$	$p = 3$	$\frac{1}{64}$	$7.6154 \cdot 10^{-15}$	8593
	$p = 4$	$\frac{1}{128}$	-	17161
$f_2(x, y)$	$p = 3$	$\frac{1}{32}$	$3.4087 \cdot 10^{-6}$	1225
	$p = 4$	$\frac{1}{16}$	$1.9024 \cdot 10^{-5}$	400
$f_3(x, y)$	$p = 3$	$\frac{1}{64}$	$5.0641 \cdot 10^{-5}$	4489

V nadaljevanju si pogledamo še aproksimacijo po metodi najmanjših kvadratov z

uporabo hierahične baze premikov škatlastega zleпка $B_{222}(\mathbf{x})$.

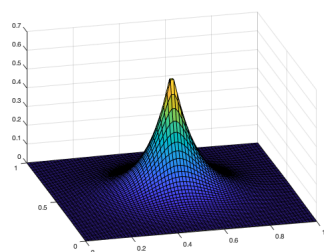
Primer 6.16. Naj bo $\Omega = [0, 1]^2$. Na Ω definiramo naslednje štiri funkcije (slika 55):

$$f_1(x, y) = \frac{2}{3 \exp(\sqrt{10(x - \frac{1}{2})^2 + 10(y - \frac{1}{2})^2})},$$

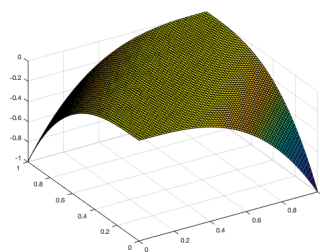
$$f_2(x, y) = \begin{cases} (x - y)^3, & x \leq y, \\ -(x - y)^3, & x > y \end{cases},$$

$$f_3(x, y) = \begin{cases} -\cos(\frac{3\pi}{2}x) + \cos(\frac{3\pi}{4}x) - x^3 \sin(x - \frac{1-3\pi}{3}) \sin(x - \frac{2-3\pi}{3}), & \frac{1}{3} < x < \frac{2}{3}, \\ -\cos(\frac{3\pi}{2}x) + \cos(\frac{3\pi}{4}x), & \text{sicer} \end{cases},$$

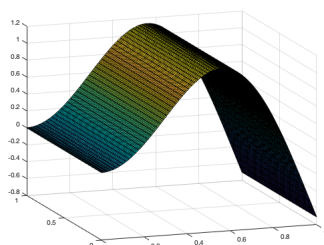
$$f_4(x, y) = \begin{cases} 100(x^3 - \frac{3}{2}x + \frac{1}{2}x - \frac{1}{9})(y^3 - \frac{5}{4}y + \frac{1}{2}y - \frac{1}{16}), & \frac{1}{3} \leq x \leq \frac{2}{3}, \frac{1}{4} \leq y \leq \frac{1}{2}, \\ 0, & \text{sicer} \end{cases}.$$



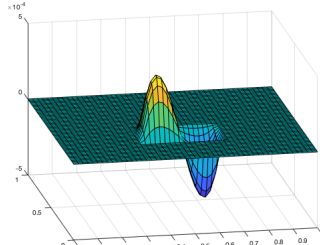
(a) $f_1(x, y)$



(b) $f_2(x, y)$



(c) $f_3(x, y)$



(d) $f_4(x, y)$

Slika 55: Grafi funkcij f_1 , f_2 , f_3 in f_4 iz primera 6.16.

Reševanja se lotimo podobno kot v prejšnjem primeru. Za vsak primer si izberemo začetni faktor skaliranja h (v naših primerih vedno 1) in nad območjem Ω zgradimo prostor premikov zleпка B_{222} in s pomočje baze tega prostora rešimo problem aproksimacije po metodi najmanjših kvadratov. Za izbrano toleranco ϵ pogledamo, na katerih območjih je napaka v neskončni normi večja od predpisane tolerance. To območje zgo-stimo na način, da na njem zgradimo finejšo bazo premika škatlastih zlepkov s faktorjem skaliranja $\frac{h}{2}$ in postopek ponovimo. Spomnimo se poglavja 5.2.4, kjer smo privzeli, da so območja finejših nivojev enaka uniji nosilcev iz prejšnjih. Postopek ponavljamo, dokler na celotnem območju ni neskončna norma napake manjša od predpisane tolerance.

Na vsakem koraku algoritma nas zanima napaka v smislu neskončne norme ter število baznih funkcij. Rezultate poskusov zberemo v tabeli 16, prikaz hierarhične mreže pa vidimo na sliki ???. V tabeli ohranimo enake oznake kot v primeru 6.15.

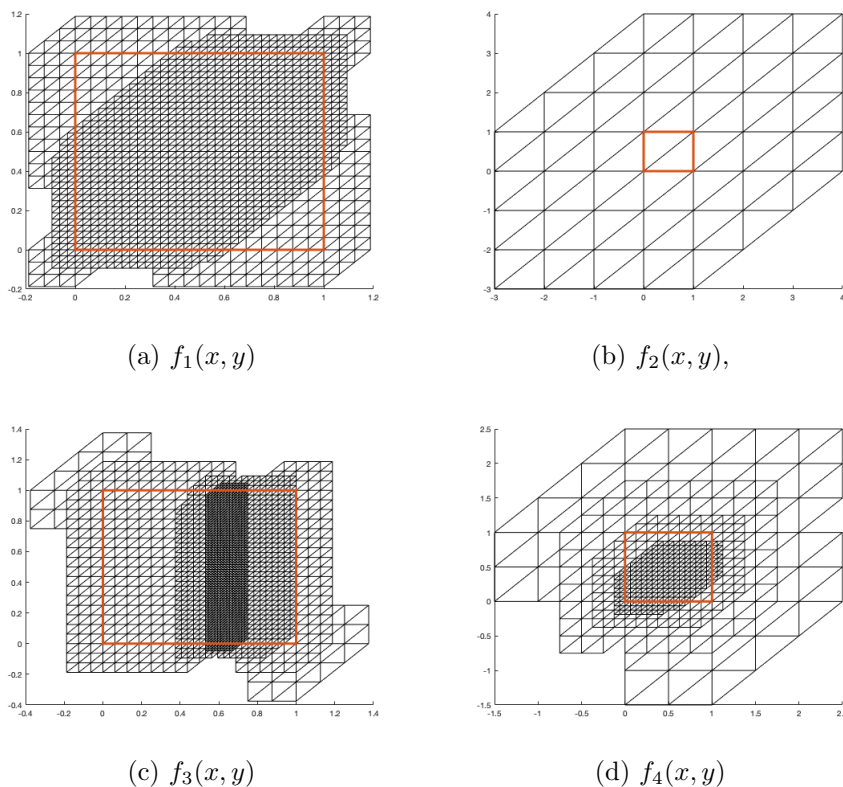
Tabela 16: Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo škatlastih zlepkov in lokalnim zgoščevanjem.

$f_1(x, y)$			$f_2(x, y)$	
h	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$
1	0.5217	14	$8.1247 \cdot 10^{-14}$	14
$\frac{1}{2}$	0.4184	23	-	-
$\frac{1}{4}$	0.3175	47	-	-
$\frac{1}{8}$	0.1847	119	-	-
$\frac{1}{16}$	0.0902	359	-	-
$\frac{1}{32}$	0.0278	1067	-	-
$f_3(x, y)$			$f_4(x, y)$	
h	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$
$\frac{1}{2}$	0.2626	14	$4.0134 \cdot 10^{-4}$	14
$\frac{1}{4}$	0.0573	23	$3.8967 \cdot 10^{-4}$	23
$\frac{1}{8}$	0.0048	47	$3.4224 \cdot 10^{-4}$	54
$\frac{1}{16}$	$9.2606 \cdot 10^{-4}$	119	$1.7407 \cdot 10^{-4}$	137
$\frac{1}{32}$	$3.5634 \cdot 10^{-4}$	363	$6.4328 \cdot 10^{-5}$	290
$\frac{1}{32}$	$1.1811 \cdot 10^{-4}$	816	-	-
$\frac{1}{64}$	$6.2595 \cdot 10^{-5}$	1407	-	-

V tabeli 16 vidimo, da je tudi implementacija hierarhične baze škatlastih zlepkov učinkovita, ter da se nam napaka na vsakem koraku zmanjšuje. Nekoliko izstopa le funkcija $f_1(x, y)$, ki pa je računsko toliko zahtevna, da jo je bilo potrebno ustaviti preden smo dosegli sprejemljivo napako. Vse štiri primere rešimo še enkrat, vendar uporabimo globalno zgostitev na nivoju, kjer smo postopek reševanja s hierarhično bazo ustavili. Rezultate zberemo v tabeli 17. Tudi v primeru škatlastih zlepkov vidimo, da smo s hierarhičnim

Tabela 17: Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo prostora premikov zlepka B_{222} in globalnim zgoščevanjem.

	h	$\ f - \tilde{f}\ _{L^\infty}$	$\dim H$
$f_1(x, y)$	$\frac{1}{32}$	0.0278	1223
$f_2(x, y)$	1	$8.1247 \cdot 10^{-14}$	14
$f_3(x, y)$	$\frac{1}{64}$	$6.2595 \cdot 10^{-5}$	4487
$f_4(x, y)$	$\frac{1}{32}$	$6.4327 \cdot 10^{-14}$	359



Slika 56: Prikaz lokalnega zgoščevanja domene za primer 6.15.

pristopom dosegli enak red napake, kot če bi domeno zgoščevali globalno, vendar pa smo na ta način bistveno izboljšali prostorsko zahtevnost, saj v primerjavi z globalnim zgoščevanjem potrebujemo precej manj baznih funkcij.

Z naslednjim primerom si pogledamo le še aproksimacijo po metodi najmanjših kvadratov z obema bazama hierarhičnih zlepkov na poljubni štirikotni domeni v \mathbb{R}^2 .

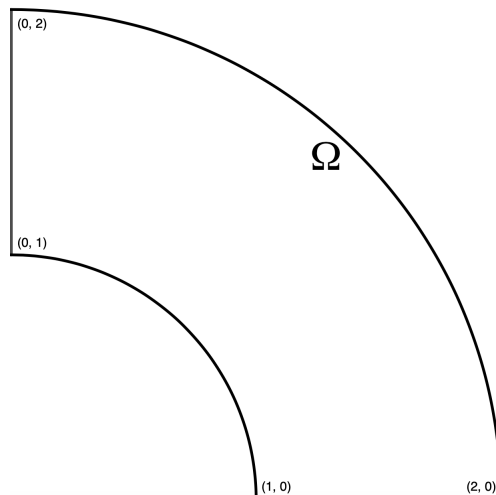
Primer 6.17. V tem primeru želimo pokazati aproksimacijo na poljubni štirikotni domeni v \mathbb{R}^2 . Želeli bi konstruirati podobno domeno kot na sliki 57. Domeno na sliki 57 lahko ekstatno opišemo kot

$$\Omega = \{(x, y) \in \mathbb{R}^2 : 1 \leq x^2 + y^2 \leq 4 \text{ in } x, y \geq 0\}. \quad (6.17)$$

Sedaj potrebujemo funkcijo \mathbf{G} , ki preslika točke območja $[0, 1]^2$ v točke območja Ω . Da si računanje nekoliko poenostavimo območje Ω aproksimiramo s tenzorskimi produkti B-zlepkov. Naj bo $p = 3$ in

$$U = (0, 0, 0, 0, \frac{1}{9}, \frac{2}{9}, \dots, \frac{8}{9}, 1, 1, 1, 1).$$

Določimo tudi $U = V$ in $p = q$. Potem vemo, da imamo v prostoru $\mathbb{S}_{U,V}^{p,q}$ 144 baznih funkcij. Območje Ω aproksimiramo s ploskvijo iz prostora $\mathbb{S}_{U,V}^{p,q}$, ki interpolira 144

Slika 57: Območje Ω primera 6.17.

vnaprej izbranih kontrolnih točk na Ω . Kontrolne točke dobimo na naslednji način. Naj bosta

$$K_1 = \{(x, y) : x^2 + y^2 = 1, \quad x, y \geq 0\}$$

$$K_2 = \{(x, y) : x^2 + y^2 = 4, \quad x, y \geq 0\}$$

S tema krožnima lokoma opišemo rob domene Ω . Potem iz vsake množice K_1 in K_2 izberemo 12 točk na način

$$K_1^* = \left\{ \left(\cos\left(\frac{\pi}{22}i\right), \sin\left(\frac{\pi}{22}i\right) \right), \quad i = 0, 1, \dots, 11 \right\}$$

$$K_2^* = \left\{ \left(2 \cos\left(\frac{\pi}{22}i\right), 2 \sin\left(\frac{\pi}{22}i\right) \right), \quad i = 0, 1, \dots, 11 \right\}$$

Kontrolno mrežo sestavimo iz točk

$$\mathbf{P}_{i,j} = \begin{bmatrix} \cos\left(\frac{\pi}{22}i\right) + jh_i \\ \sin\left(\frac{\pi}{22}i\right) + jh_i \\ 0 \end{bmatrix},$$

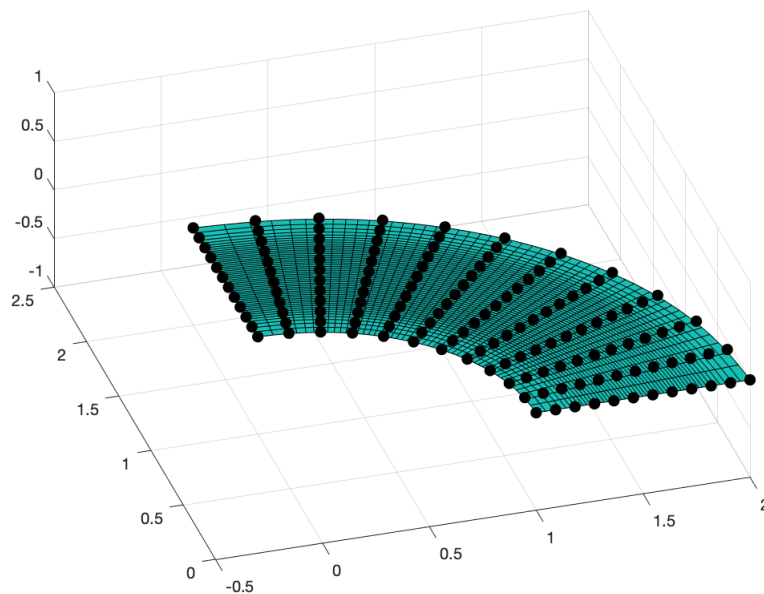
za $i = 0, 1, \dots, 11$, $j = 0, 1, \dots, 11$ ter $h_i \in \mathbb{R}$ t.d. $\cos\left(\frac{\pi}{22}i\right) + 11h_i = 2 \cos\left(\frac{\pi}{22}i\right)$ in $\sin\left(\frac{\pi}{22}i\right) + 11h_i = 2 \sin\left(\frac{\pi}{22}i\right)$. Funkcija $\mathbf{G} \in \mathbb{S}_{U,V}^{p,q}$ s predpisom

$$\mathbf{G}(u, v) = \sum_{i=0}^{11} \sum_{j=0}^{11} \mathbf{P}_{i,j} N_{i,j}^{p,q}(u, v)$$

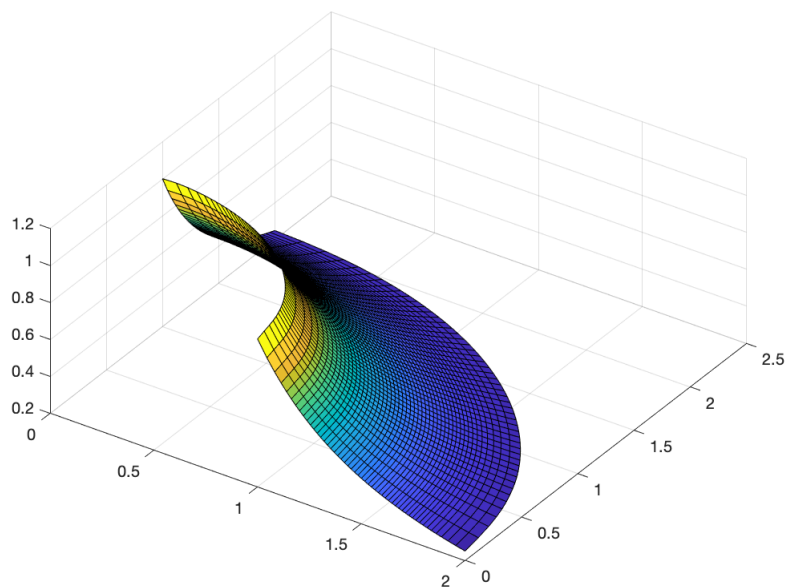
preslika vsako točko območja $[0, 1]^2$ na aproksimacijo območja Ω , ki jo označimo z Ω^* . Območje Ω^* skupaj s kontrolno mrežo imamo prikazano na sliki 58.

Na območju Ω^* definiramo funkcijo

$$f(x, y) = \frac{1}{x^2 + y^2}.$$



Slika 58: Območje Ω^* primera 6.17.



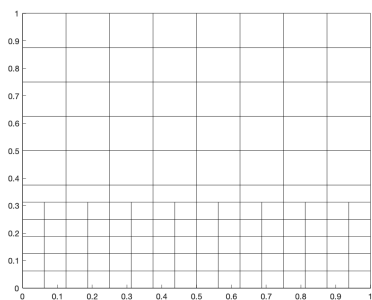
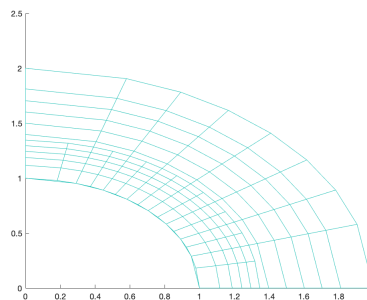
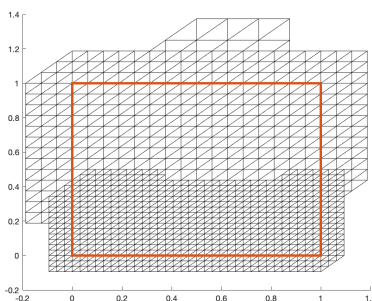
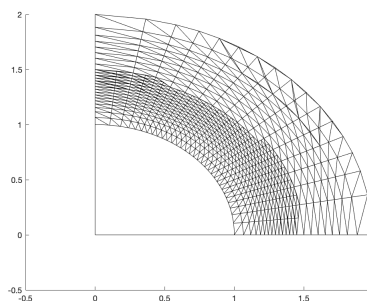
Slika 59: Graf funkcije $f(x, y)$ primera 6.17.

Funkcijo vidimo na sliki 59. V tem primeru želimo najti aproksimacijo po metodi najmanjših kvadratov iz prostora hierarhičnih tenzorskih produktov B-zlepkov ter hierarhičnega prostora premikov škatlastega zlepka B_{222} . Reševanja se lotimo podobno kot v primerih 6.15 in 6.16 z upoštevanjem teorije zapisane v 6.1.5.1. V primeru tenzorskih produktov B-zlepkov za začetni prostor uporabimo stopnji $p = q = 4$, ter začetna vektorja vozlov $U = V = (0, 0, 0, 0, 0, 1, 1, 1, 1, 1)$, v primeru prostora premika zlepka

B_{222} pa uporabimo začetni faktor skaliranja $h = 1$. Rezultate poskusa zberemo v tabeli 18. Oba poskusa izvedemo pri toleranci $\epsilon = 10^{-4}$. Območja zgoščevanja tako na domeni $[0, 1]^2$, kot na domeni Ω^* vidimo na sliki 18.

Tabela 18: Rezultati aproksimacije po metodi najmanjših kvadratov z uporabo obeh prostorov hierarhičnih zlepkov na reparametriziranem območju. Zadnja vrstica prikazuje rezultate v primeru globalnega zgoščevanja.

h	Škatlasti		B-zlepki	
	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$
1	0.0513	25	0.0792	14
$\frac{1}{2}$	0.0277	36	0.0591	23
$\frac{1}{4}$	0.0077	64	0.0214	47
$\frac{1}{8}$	0.0010	144	0.0035	119
$\frac{1}{16}$	$1.9384 \cdot 10^{-4}$	400	$3.3060 \cdot 10^{-4}$	361
$\frac{1}{32}$	$9.1665 \cdot 10^{-5}$	660	$7.0308 \cdot 10^{-5}$	742
GLOBALNO	$9.4354 \cdot 10^{-6}$	1296	$3.4509 \cdot 10^{-5}$	1223

(a) Domena $(0, 1)^2$ (B-zlepki).(b) Domena Ω^* (B-zlepki).(c) Domena $(0, 1)^2$ (B_{222}).(d) Domena Ω^* (B_{222}).

Slika 60: Prikaz lokalnega zgoščevanja domene za primer 6.17.

Tudi v primeru preslikave na novo domeno dobimo enake rezultate kot v prejšnjih primerih. Opazimo, da je implementacija algoritmov učinkovita, kar pomeni, da se na-

paka na vsakem koraku algoritma zmanjšuje. Tudi v tem primeru z uporabo lokalnega zgoščevanja dosežemo primerljiv red napake približka za obe vrsti zlepkov z bistveno manjšim številom baznih funkcij.

Ogledali smo si nekaj preprostih primerov aproksimacije po metodi najmanjših kvadratov z uporabo hierarhičnih prostrov tenzorskih produktov B-zlepkov ter hierarhičnih prostorov škatlastih zlepkov. V obeh primerih se je pokazalo, da implementirani algoritmi delujejo, ter da je prostorska zahtevnost aplikacije v primerjavi z globalnim zgoščevanjem manjša. Iz rezultatov pa ne moremo direktno trditi, da je ena vrsta zlepkov avtomatično bolj uporabna od druge. Kot vidimo v tabelah je vse odvisno od primera do primera. Za funkcijo f_1 izgleda, da je za aproksimacijo na domeni $[0, 1]^2$ bolje uporabiti tenzorske produkte B-zlepkov, saj lahko z njimi učinkoviteje opišemo območja zgoščevanja in napaka celotnega območja kaj hitro pade pod vrednost vnaprej določene tolerance, medtem ko v primeru škatlastih zlepkov napaka pada tako počasi, da je po primerljivem številu korakov napaka na celotnem območju še vedno bistveno večja od predpisane tolerance. Po drugi strani pa imamo funkcijo f_2 , kjer s škatlastimi zlepkami pademo precej pod predpisano toleranco že na prvem koraku, brez potrebnega zgoščevanja, medtem ko v tem primeru B-zlepki potrebujejo kar nekaj korakov, da padejo pod predpisano toleranco.

6.2.3 Galerkinova metoda

V tem poglavju z nekaj numeričnimi poskusi preverimo delovanje Galerkinove metode z uporabo obeh vrst hierarhičnih zlepkov. Velja omeniti, da je naloga reševanje Poissonove enačbe z Galerkinovo metodo težja kot aproksimacija po metodi najmanjših kvadratov in moramo zato še nekoliko spustiti pričakovanja glede rezultatov. To pomeni, da bomo naredili manj primerov pri še višji toleranci kot v prejšnjem poglavju. Tudi tukaj poglavje začnemo s primerom uporabe prirezane hierarhične baze tenzorskih produktov B-zlepkov in zatem nadaljujemo še s primerom hierarhične baze premikov škatlastega zlepka B_{222} . Za razliko od prejšnjega poglavja, kjer smo naredili nekaj primerov aproksimacije po metodi najmanjših kvadratov, smo v tem poglavju nekoliko skromnejši s številom primerov. Za vsako vrsto zlepkov naredimo en sam primer reševanja Poissonove enačbe tako s homogenim kot nehomogenim robnim pogojem. Tudi tokrat končamo poglavje, kot pri aproksimaciji, s primerom preslikave iz domene $[0, 1]^2$ na poljubno štirikotno domeno, le da se tukaj zaradi velike računske zahtevnosti omejimo le na primer tenzorskih produktov B-zlepkov.

Primer 6.18. Rešujemo Poissonovo enačbo na območju $\Omega = [0, 1]^2$. Poznamo točno rešitev homogene Poissonove enačbe

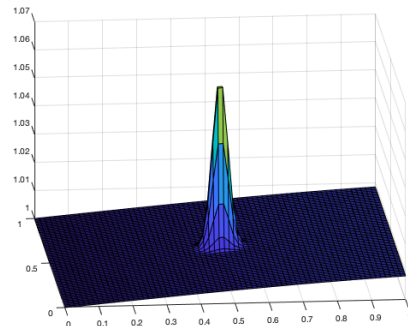
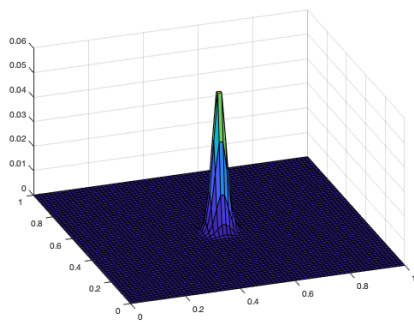
$$u(x, y) = xy(x - 1)(y - 1) \frac{1}{e^{1000((x-\frac{1}{2})^2 + (y-\frac{1}{2})^2)}}$$

in definiramo² $f(x, y) = \Delta u(x, y)$. Potem definiramo dve Poissonovi enačbi. Prvo enačbo, s homogenim robnim pogojem, definiramo kot

$$\begin{aligned} -\Delta u(x, y) &= f(x, y), & (x, y) \in \Omega, \\ u(x, y) &= 0, & (x, y) \in \partial\Omega, \end{aligned} \quad (6.18)$$

drugo enačbo, ki je sicer podobna prvi, z nehomogenim robnim pogojem pa definiramo kot

$$\begin{aligned} -\Delta u(x, y) &= f(x, y), & (x, y) \in \Omega, \\ u(x, y) &= \frac{1}{100} \sin(x) + 1, & (x, y) \in \partial\Omega. \end{aligned} \quad (6.19)$$



(a) Graf točne rešitve problema (6.18). (b) Graf točne rešitve problema (6.19).

Slika 61: Grafa rešitev problemov Poissonove enačbe primera 6.18.

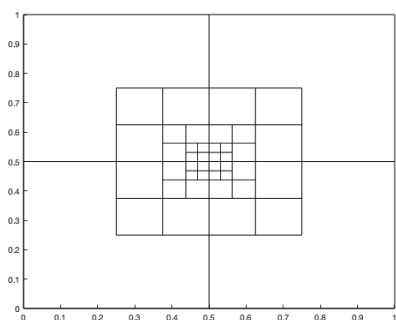
Rešitvi obeh primerov nam prikazuje slika 61. Enačbi rešimo s pomočjo teorije zapisane v poglavju 6.1.4, in uporabo hierarhičnega prostora tenzorskih produktov B-zlepkov. Za oba problema izberemo začetne prostore tenzorskih produktov B-zlepkov definiranih s polinomskima stopnjama $p = q = 2$ ter vektorjema vozlov $U = V = (0, 0, 0, 1, 1, 1)$. Ideja adaptivnega zgoščevanja domene pa je enaka tisti pri aproksimaciji po metodi najmanjših kvadratov v 6.15. V obeh primerih si postavimo toleranco $\epsilon = 10^{-3}$. Rezultate zberemo v tabeli 19, kjer uporabljamo enake oznake kot v primeru 6.15. V tabeli najdemo tudi primerjavo prostorske zahtevnosti reševanja, če bi enak problem reševali z globalnim zgoščevanjem.

Iz tabele 19 je razvidno, da je metoda reševanja implementirana dobro in učinkovito. Na vsakem koraku se nam napaka zmanjšuje in v primerjavi z globalnim zgoščevanjem je prostorska zahtevnost mnogo manjša, pri čemer red napake v smislu neskončne norme praktično ohranimo. Velja sicer poudariti, da sta problema (6.18) in (6.19) pisana na kožo hierarhičnemu reševanju, saj je bila rešitev $u(x, y)$ konstruirana z namenom, da je na večini območja Ω pohlavna, medtem ko se njena računaska zahtevnost poveča le na sredini območja Ω . Prikaz lokalnega zgoščevanja v obeh primerih vidimo na sliki 62.

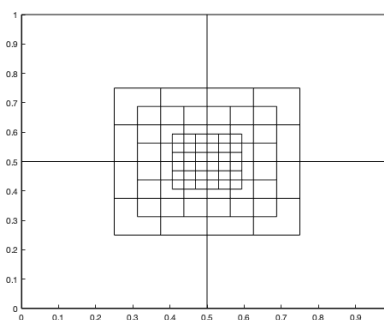
²Laplacov operator bi seveda lahko določili tudi eksaktno, vendar je hitreje in nekoliko stabilnejše vzeti Matlabov numerični približek.

Tabela 19: Rezultati Galerkinove metode z uporabo hierarhične prirezane baze tenzorskih produktov B-zlepkov.

h	Homogen primer			Nehomogen primer		
	$\ f - \tilde{f}\ _{L^\infty}$	$\hat{\Omega}$	$\dim(H)$	$\ f - \tilde{2}\ _{L^\infty}$	$\hat{\Omega}$	$\dim(H)$
1	0.0557	$[0, 1]^2$	1	0.0561	$[0, 1]^2$	1
$\frac{1}{2}$	0.0557	$[0, 1]^2$	4	0.0561	$[0, 1]^2$	4
$\frac{1}{4}$	0.0548	$[0, 1]^2$	16	0.0551	$[0, 1]^2$	16
$\frac{1}{8}$	0.0509	$[\frac{1}{4}, \frac{3}{4}]^2$	20	0.0512	$[\frac{1}{4}, \frac{3}{4}]^2$	20
$\frac{1}{16}$	0.0359	$[\frac{1}{4}, \frac{3}{4}]^2$	52	0.0362	$[\frac{1}{4}, \frac{3}{4}]^2$	52
$\frac{1}{32}$	0.0076	$[\frac{3}{8}, \frac{5}{8}]^2$	84	0.0079	$[\frac{5}{16}, \frac{11}{16}]^2$	136
$\frac{1}{64}$	$5.3026 \cdot 10^{-4}$	$[\frac{7}{16}, \frac{9}{16}]^2$	116	$8.3106 \cdot 10^{-4}$	$[\frac{13}{32}, \frac{38}{64}]^2$	220
GLOBALNO	$5.2810 \cdot 10^{-4}$	-	4096	$8.3336 \cdot 10^{-4}$	-	4096



(a) Homogen primer.



(b) Nehomogen primer.

Slika 62: Območja zgoščevanja pri uporabi Galerkinove metode v primeru 6.18.

Z naslednjim primerom poskušamo rešiti Poissonovo enačbo z Galerkinovo metodo, pri čemer uporabimo hierarhični prostor premikov zlepka B_{222} .

Primer 6.19. Iz primera 6.18 vzamemo funkcijo f in definiramo Poissonovo enačbo s homogenim robnim pogojem

$$\begin{aligned} -\Delta u(x, y) &= f(x, y), & (x, y) \in \Omega. \\ u(x, y) &= 0, & (x, y) \in \partial\Omega. \end{aligned} \tag{6.20}$$

Definiramo še enačbo z nehomogenim robnim pogojem

$$\begin{aligned} -\Delta u(x, y) &= -f(x, y), & (x, y) \in \Omega, \\ u(x, y) &= \frac{1}{100}x, & (x, y) \in \partial\Omega. \end{aligned} \tag{6.21}$$

Hierarhični prostor tenzorskih produktov B-zlepkov nadomestimo s hierarhičnim prostorom ustrezno skaliranega prostora premikov škatlastega zlepka B_{222} . Tudi v tem

primeru pričakujemo konvergenco k pravi rešitvi, čeprav lahko iz primera 6.14 sklepamo, da bo ta počasnejša kot v primeru tenzorskih produktov B-zlepkov. Tudi poskusa se lotimo nekoliko drugače. Za razliko od vseh primerov s tenzorskimi produkti B-zlepkov, ter aproksimacije z uporabo škatlastih zlepkov, kjer smo na začetnem nivoju za osnovno delitev območja Ω vzeli faktor skaliranja $h = 1$ oz. v primeru B-zlepkov vektor vozlov sestavljen le iz robnih vozlov, v tem primeru začnemo s faktorjem skaliranja $h = \frac{1}{16}$. To naredimo iz razloga, ker je do tega faktorja skaliranja večina baznih funkcij sestavljenih kot linearna kombinacija dveh ali treh drugih funkcij, kar pa bi nam precej otežilo upoštevanje predpostavljenega krepkega pogoja, da so območja zgoščevanja na finejših nivojih sestavljena iz nosilcev baznih funkcij prejšnjega nivoja. Ravno v ta namen izberemo tako velik začetni faktor skaliranja ter funkcijo, ki je pisana na kožo hierarhičnemu načinu reševanja, kjer pričakujemo, da bo napaka previsoka le v sredini celotnega območja in bomo brez težav lahko zgostili območje z upoštevanjem krepkih robnih pogojev. Tudi v tem primeru predpišemo toleranco $\epsilon = 10^{-3}$. Rezultate obeh poskusov zberemo v tabeli 20, kjer zopet upoštevamo enake oznake kot v prejšnjih primerih. Tako v primeru homogene kot v primeru nehomogene Poissonove enačbe se

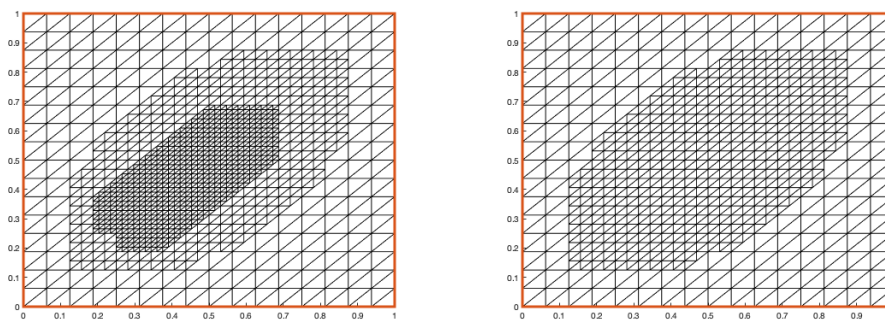
Tabela 20: Rezultati Galerkinove metode z uporabo hierarhične baze premikov zlepka B_{222} .

	Homogen primer		Nehomogen primer	
h	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$	$\ f - \tilde{2}\ _{L^\infty}$	$\dim(H)$
$\frac{1}{16}$	0.0192	225	0.0182	225
$\frac{1}{32}$	0.0051	531	0.0062	531
$\frac{1}{64}$	0.0018	969	-	-

nam napaka na vsakem koraku zmanjšuje, domena pa se zgoščuje lokalno, kot lahko vidimo na sliki 63. Opazimo pa tudi, da je konvergenca metode počasnejša v primerjavi s prejšnjimi metodami in po treh korakih oz. po dveh v primeru nehomogene enačbe še ne pademo pod predpisano toleranco. Nadaljnih korakov poskusa ne moremo izvesti zaradi same računske zahtevnosti implementirane metode. Prav tako sta obe metodi prezahtevni, da bi lahko izvedli globalno zgostitev s tako majhnim faktorjem skaliranja. Izračunamo lahko le prostorsko zahtevnost, ki v obeh primerih znaša 3969 baznih funkcij.

Z naslednjim primerom si pogledjmo še Galerkinovo metodo z uporabo hierarhičnega prostora tenzorskih produktov B-zlepkov na poljubni štirikotni domeni v \mathbb{R}^2 . Hierarhičnemu prostoru premikov zlepka B_{222} pa se v tem primeru zaradi prevelike računske zahtevnosti izognemo.

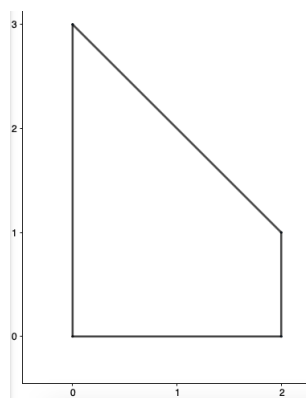
Primer 6.20. Naj bo Ω območje v \mathbb{R}^2 , kot je prikazano na sliki 64. Takšno območje



(a) Homogen problem.

(b) Nehomogen problem.

Slika 63: Območja zgoščevanja pri uporabi Galerkinove metode v primeru 6.19.



Slika 64: Območje Ω primera 6.20.

eksaktno opišemo kot

$$\Omega = \{(x, y); \quad 0 \leq x \leq 2, 0 \leq y \leq -x + 3\}.$$

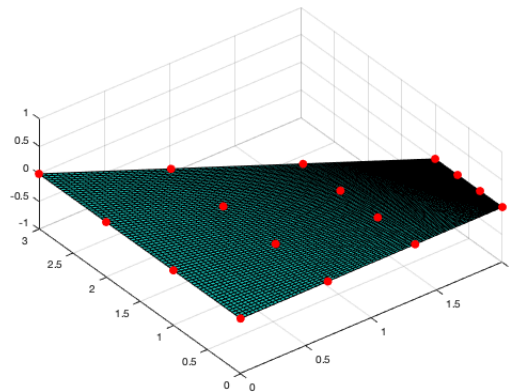
Območje podobno kot v primeru 6.17 predstavimo kot ploskev iz prostora tenzorskih produktov B-zlepkov. Za ta namen izberemo prostor določen z vektorjema vozlov $U = V = (0, 0, \frac{1}{3}, \frac{2}{3}, 1, 1)$ in stopnjama $p = q = 1$. V tem prostoru imamo 16 baznih funkcij. Določimo še kontrolno mrežo, tvorjeno s šestnajstimi kontrolnimi točkami

$$\mathbf{P}_{i,j} = \begin{bmatrix} \frac{2}{3}i \\ \frac{2}{3}i + jh_i \\ 0 \end{bmatrix},$$

za $i = 0, 1, 2, 3$ in $j = 0, 1, 2, 3$ ter takšne h_i , da velja $\frac{2}{3}i + 3h_i = 3 - \frac{2}{3}i$. Potem preslikava $\mathbf{G} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, določena s predpisom

$$\mathbf{G}(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 \mathbf{P}_{i,j} N_{i,j}^{p,q}(u, v)$$

preslika vsako točko enotskega kvadrata na območje Ω . Območje Ω , predstavljeno kot slika ploskve \mathbf{G} skupaj s kontrolno mrežo, vidimo na sliki 65. Nad območjem Ω bi

Slika 65: Krivulja $G(x, y)$ s pripadajočo kontrolno mrežo \mathbf{P} primera 6.20.

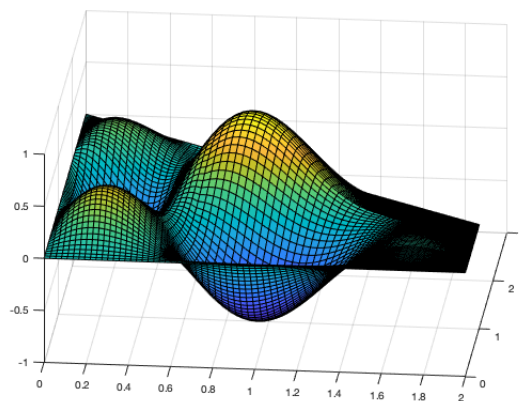
želeli rešiti sledečo Poissonovo enačbo. Naj bo

$$\begin{aligned}
 f(x, y) = & -(-6(x-2)xy \sin(3x) \sin(3y) - 6(x-2)y(x+y-3) \sin(3x) \sin(3y) - \\
 & 6xy(x+y-3) \sin(3x) \sin(3y) + 6(x-2)xy \cos(3x) \cos(3y) + \\
 & 6(x-2)x(x+y-3) \cos(3x) \cos(3y) + 2(x-2)x \cos(3x) \sin(3y) + \\
 & 2(x-2)y \cos(3x) \sin(3y) + 2xy \cos(3x) \sin(3y) - \\
 & 18(x-2)xy(x+y-3) \cos(3x) \sin(3y) + 2y(x+y-3) \cos(3x) \sin(3y)).
 \end{aligned}$$

Potem definiramo Poissonovo enačbo kot

$$\begin{aligned}
 -\Delta u(x, y) &= f(x, y), \quad (x, y) \in \Omega, \\
 u(x, y) &= 0, \quad (x, y) \in \partial\Omega.
 \end{aligned} \tag{6.22}$$

Poznamo točno rešitev problema (6.22) $u(x, y) = xy(x-2)(y+x-3) \cos(3x) \sin(3y)$, njen graf pa vidimo na sliki 66. Enačbo na območju Ω rešimo s pomočjo teorije zapisane



Slika 66: Graf točne rešitve Poissonove enačbe primera 6.20.

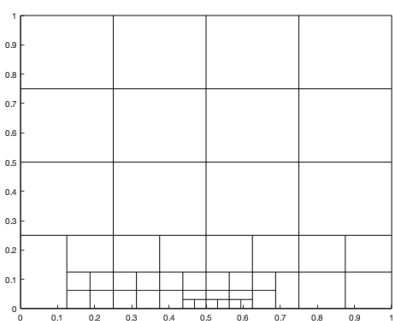
v poglavju 6.1.5.2 z uporabo hierarhičnega prostora tenzorskih produktov B-zlepkov,

stopenj $p = q = 3$. Enako kot v vseh primerih do sedaj si izberemo začetni vektor vozlov $U = V = (0, 0, 0, 0, 1, 1, 1, 1)$ in potem gostimo območje tam, kjer je napaka večja od predpisane tolerance. Zaradi velike računske zahtevnosti si v tem primeru izberemo toleranco $\epsilon = 0.02$. Rezultate poskusa zberemo v tabeli 21, območja zgoščevanja tako na domeni Ω , kot na domeni $[0, 1]^2$ pa vidimo na sliki 67. V tabeli 21 vidimo, da je

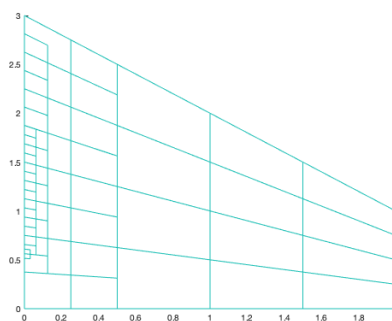
Tabela 21: Rezultati Galerkinove metode na poljubnem štirikotnem območju v \mathbb{R}^2 z uporabo hierarhičnega prostora tenzorskih produktov B-zlepkov.

h	$\ f - \tilde{f}\ _{L^\infty}$	$\dim(H)$
1	0.6226	4
$\frac{1}{2}$	0.5246	9
$\frac{1}{4}$	0.2106	25
$\frac{1}{8}$	0.0464	81
$\frac{1}{16}$	0.0379	114
$\frac{1}{32}$	0.0263	153
$\frac{1}{64}$	0.0199	162
GLOBALNO	-	4096

tudi ta metoda dobro implementirana, saj se napaka na vsakem koraku zmanjšuje. Zaradi velike računske zahtevnosti ne moremo izračunati rešitve v primeru globalnega zgoščevanja, lahko pa izračunamo prostorsko zahtevnost, ki pa je tudi v tem primeru mnogo večja v primerjavi z uporabo hierarhičnega prostora.



(a) Območje $(0, 1)^2$.



(b) Območje Ω .

Slika 67: Območja zgoščevanja pri uporabi Galerkinove metode v primeru 6.20.

Ogledali smo si nekaj preprostih primerov Galerkinove metode za reševanje Poissonove enačbe z uporabo hierarhičnih prostorov zlepkov. Za razliko od aproksimacije smo tukaj zaradi velike računske zahtevnosti naredili bistveno manj primerov. Kljub temu pa lahko v teh nekaj skromnih zgledih, enako kot v primeru aproksimacije, vidimo, da

implementirani algoritmi delujejo ter da se napaka na vsakem koraku zmanjšuje. Tudi v primeru Galerkinove metode je prostorska zahtevnost algoritmov, ki uporabljajo hierarhične prostore zlepkov, mnogo manjša kot v primeru globalnega zgoščevanja, pri čemer pa se red napake bistveno ne poveča, kar smo tudi eksplicitno potrdili za tenzorske produkte B-zlepkov. Sklepamo lahko, da enako velja tudi za škatlaste zlepke, kar pa nam zaradi velike računske zahtevnosti ni uspelo pokazati eksplicitno.

7 ZAKLJUČEK

V magistrskem delu smo predstavili hierarhične prostore zlepkov in njihovo uporabo v dveh različnih aplikacijah. Definirali smo navadne prostore B-zlepkov ene spremenljivke in pokazali nekaj pomembnih lastnosti. Za tem smo definirali B-zlepke dveh spremenljivk kot tenzorski produkt dveh B-zlepkov ene spremenljivke in pokazali, da imajo tenzorski produkti B-zlepkov enake lastnosti kot B-zlepki ene spremenljivke. Za tem smo definirali škatlaste zlepke in pokazali, da imajo tudi škatlasti zlepki podobne lastnosti kot B-zlepki. Ker je bilo operiranje s škatlastimi zlepkami definiranimi z definicijo 3.4 zapleteno, smo spoznali Bernstein-Bézierevo obliko škatlastega zlepka, s pomočje katere smo lahko škatlaste zlepke kasneje tudi implementirali. Potem smo definirali še prostor premikov škatlastega zlepka in pokazali, da tudi za te prostore velja, da so elementi tega prostora pod določenimi pogoji linearno neodvisni ter tvorijo particijo enote. Videli smo tudi, da so škatlasti zlepki posplošitev B-zlepkov. Sledilo je poglavje o hierarhičnih prostorih zlepkov. Za obe vrsti zlepkov smo definirali ugnezdene prostore in domene ter konstruirali hierarhično bazo, za katero smo dokazali linearno neodvisnost. Videli smo tudi, da lastnost particije enote pri hierarhični bazi ni avtomatično zagotovljena, zato smo v primeru B-zlepkov konstruirali priznano hierarhično bazo, v primeru škatlastih zlepkov pa sprejeli strožje robne pogoje, pod katerimi ohranimo particijo enote. Za teorijo o hierarhičnih prostorih zlepkov smo navedli nekaj algoritmov in programerskih prijemov, s pomočjo katerih smo lažje implementirali obe vrsti hierarhičnih zlepkov v okolju Matlab. V zadnjem poglavju pa smo naredili še nekaj primerov aplikacij z uporabo hierarhičnih prostorov zlepkov. Na začetku poglavja smo zapisali nekaj nujno potrebne teorije o aproksimaciji po metodi najmanjših kvadratov ter reševanju Poissonove enačbe z Galerkinovo metodo. Poleg tega smo na kratko povzeli še teorijo Gaussovih integracijskih pravil, algoritma minimalne določitvene množice ter reparametrizacije problema aproksimacije po metodi najmanjših kvadratov in Galerkinove metode iz območja $[0, 1]^2$ na poljubno štirikotno domeno v \mathbb{R}^2 . Za tem smo naredili nekaj numeričnih poskusov. Na začetku smo preverili red konvergence obeh metod z uporabo obeh vrst zlepkov v primeru globalnega zgoščevanja. Videli smo, da se redi konvergence ujemajo z zapisano teorijo v [1, 3, 14]. Za vse metode in vrste zlepkov smo dobili enak red konvergence, razen pri Galerkinovi metodi z uporabo škatlastih zlepkov, kjer smo pričakovano dobili nižji red. Čisto na koncu smo naredili še nekaj numeričnih poskusov aproksimacije po metodi najmanjših kvadratov ter reševanja Poissonove enačbe z Galerkinovo metodo, pri čemer

smo uporabili obe vrsti hierarhičnih prostorov zlepkov. S temi poskusi smo preverjali hipotezo, da so hierarhični zlepki ter metode dobro implementirane, ter da je uporaba hierarhičnih prostorov zlepkov v določenih primerih bolj ekonomična, v smislu, da v primerjavi z globalnim zgoščevanjem dosežemo primerljivo napako numeričnega približka, pri čemer imamo v primeru hierarhičnih prostorov zlepkov bistveno manjšo prostorsko zahtevnost. To se je dobro videlo pri vseh metodah in obeh vrstah zlepkov, kjer smo v nekaj korakih lahko dosegli vnaprej predpisano toleranco napake numeričnega približka. Primer, kjer tega nismo jasno videli, je le Galerkinova metoda z uporabo hierarhičnih prostorov škatlastih zlepkov. Tukaj smo lahko le videli, da se napaka na vsakem koraku zmanjšuje, vendar bistveno počasneje kot pri ostalih aplikacijah, kar pa je bilo pričakovano po rezultatih določanja reda konvergence v primeru globalnega zgoščevanja. Izboljšano metodo reševanja Poissonove enačbe z uporabo hierarhičnih prostorov škatlastih zlepkov bralec najde v [10].

8 LITERATURA IN VIRI

- [1] E. ARGE, M. DÆHLEN in A. TVEITO, Box spline interpolation; a computational study. *J. Comput* 44 (1992) 303–329. (*Citirano na straneh 104 in 128.*)
- [2] J. COTTRELL, T. HUGHES in Y. BAZILEVS, *Isogeometric Analysis Toward Integration of CAD and FEA*. Wiley, 2009. (*Citirano na strani 1.*)
- [3] C. DE BOOR, *A practical guide to splines, Applied Mathematical Sciences, Vol. 27*, Springer-Verlag, Revised edition, 1994. (*Citirano na straneh 1, 97 in 128.*)
- [4] C. DE BOOR, in K. HÖLLIG, B-splines from parallelepipeds. *J. Anal. Math* 42 (1982) 99–115. (*Citirano na straneh 1 in 37.*)
- [5] C. DE BOOR, K. HÖLLIG in S. RIEMENSCHNEIDER, *Box Splines*. Springer Science+Business Media, New York, 1993. (*Citirano na straneh 1 in 23.*)
- [6] C. DE BOOR, *Spline Toolbox for Use with MATLAB: User's Guide, Version 3*, MathWorks, 2005. (*Citirano na straneh 3 in 64.*)
- [7] D. FORSEY, Hierarchical B-spline refinement. V *Proc. 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, 1988, 205–212. (*Citirano na strani 2.*)
- [8] C. GIANNELLI, B. JÜTTLER in H. SPELEERS, THB-splines: The truncated basis for hierarchical splines. *Comput. Aided Geom. Des.* 29 (2012) 485–498. (*Citirano na strani 2.*)
- [9] T. HUGHES, J. COTTRELL in Y. BAZILEVS, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Eng.* 194 (2005) 4135–4195. (*Citirano na strani 1.*)
- [10] T. KANDUČ, C. GIANNELLI, F. PELOSI in H. SPELEERS, Adaptive isogeometric analysis with hierarchical box splines. *Comput. Methods Appl. Mech. Eng.* 316 (2017) 817–838. (*Citirano na straneh 2, 46, 57, 59, 90, 110 in 129.*)
- [11] H. KANG, F. CHEN in J. DENG, Hierarchical Box splines. V *2015 14th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)*, 2015, 73–80. (*Citirano na straneh 2 in 104.*)

- [12] M. KAPL, in V. VITRIH, Space of C^2 -smooth geometrically continuous isogeometric functions on two-patch geometries. *Comput. Math. with Appl.* 73 (2017) 37–59. (*Citirano na strani 85.*)
- [13] M. KAPL, V. VITRIH, B. JÜTTLER in K. BIRNER, Isogeometric analysis with geometrically continuous functions on two-patch geometries. *Comput. Math. with Appl.* 73 (2015) 1518–1538. (*Ni citirano.*)
- [14] J. KOZAK, *Numerična analiza*, DMFA-založništvo, Ljubljana, 2008. (*Citirano na straneh 3, 86, 87, 88, 97 in 128.*)
- [15] M. LAI in L. SCHUMAKER, Bivariate Box Splines. V: M. Lai, L. Schumaker, *Spline Functions on Triangulations*, Cambridge University Press, 2007, 334–377. (*Ni citirano.*)
- [16] T. LYCHE, C. MANNI in H. SPELEERS, Foundations of Spline Theory: B-Splines, Spline Approximation, and Hierarchical Refinement. V: A. Kunoth et al., *Splines and PDEs: From Approximation Theory to Numerical Linear Algebra*, Springer, 2018, 1–76. (*Citirano na straneh 11, 17 in 71.*)
- [17] *Triangulation Representations*, MATLAB. (2021). Natick, Massachusetts: The MathWorks Inc.
<https://www.mathworks.com/help/matlab/math/triangulation-representations.html>. (Datum ogleda: 5. 4. 2021.) (*Citirano na straneh 3 in 64.*)
- [18] W.F. MITCHELL, A collection of 2D elliptic problems for testing adaptive grid refinement algorithms. *Appl. Math. Comput.* 220 (2013) 350–364. (*Ni citirano.*)
- [19] F. PELOSI, C. GIANNELLI, C. MANNI, B. JÜTTLER, M. SAMPOLI, in H. SPELEERS, Splines over regular triangulations in numerical simulation. *Comput. Aided Des.* 29 (2017) 100–111. (*Citirano na strani 2.*)
- [20] B. PLESTENJAK, *Razširjen uvod v numerične metode*, DMFA-založništvo, Ljubljana, 2015. (*Citirano na strani 3.*)
- [21] H. PRAUTZSCH in W. BOEHM, Box Splines. V: G. Farin, J. Hoschek, M. Kim, *Handbook of Computer Aided Geometric Design*, North-Holland, 2002, 255–282. (*Citirano na strani 1.*)
- [22] H. PRAUTZSCH, W. BOEHM in M. PALUSZNY, *Bézier and B-Spline Techniques*. Springer, Berlin, 2002. (*Citirano na strani 1.*)

- [23] J. RONG-QING, Linear independence of translates of a box spline. *J. Approx. Theory* 40 (1984) 158–160. (*Ni citirano.*)
- [24] C. SHENE, *CS3621 Introduction to Computing with Geometry Notes*, <https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/>. (Datum ogleda: 8. 3. 2021.) (*Ni citirano.*)
- [25] W. STRAUSS, *Partial Differential Equations: An Introduction*, John Wiley & Sons, Ltd, Second edition, 2007. (*Citirano na straneh 3 in 91.*)
- [26] A. VUONG, C. GIANNELLI, B. JÜTTLER in B. SIMEON, A Hierarchical Approach to Adaptive Local Refinement in Isogeometric Analysis. *Comput. Methods Appl. Mech. Eng.* 29 (2011) 3554–3567. (*Citirano na straneh 2 in 110.*)

Priloge

PRILOGA A Baza B-zlepkov

```
1 function B = baza(U,p)
2 % Naredi bazo B-zlepkov stopnje p nad
3 % vektorjem vozlov U.
4 m = length(U)-1;n = m - p - 1;
5 koef = zeros(1,n+1);
6 koef(1) = 1;
7 B = [spmak(U, koef)];
8 for i = 2:n+1
9     koef = zeros(1,n+1);
10    koef(i) = 1;
11    B = [B, spmak(U, koef)];
12 end
13 end
```

PRILOGA B Prirezana hierarhična baza tenzorskih produktov B-zlepkov

```
1 function T_out = THB_baza(U,V,p,q,Omega, PDE_SWITCH)
2 %THB_baza Konstruirana 2D prerezana hierarhična baza
3 % Končna baza je vektor tenzorskih produktov zlepkov skonstruiranih s
4 % funkcijo spmak(). U,V sta vektorja vozlov nad katerima skonstruiramo
5 % hierarhično bazo. Stopnji p,q sta na vseh nivojih hierarhične baze
6 % enaka. Omega je cell array območij, na katerih zelimo hierarhično
7 % bazo. Vsako območje omega_l je predstavljeno z matriko [a b ; c d],
8 % kjer je [a b] interval v eni smeri, [c d] pa interval v drugi. Območje
9 % omega_l je torej [a, b] x [c, d]. OPOMBA: omega_0 sovpada s prvim in
10 % zadnjim vozlov vektorja U oz. V.
11 B_U = baza(U,p); B_V = baza(V,q);
12 if PDE_SWITCH == 1
13     B_U = B_U(2:end-1); B_V = B_V(2:end-1);
14 end
15 T = {};
16 % Za delitev vedno predpostavimo dvakrat finejšo. Ta delta se na vsakem
17 % nivoju deli z 2, da res vedno dobimo dvakrat finejšo delitev.
18 delta = 0.5;
19 % Ves čas hranimo najfinejša vektorja vozlov, ki jih ustvarimo med alg.
20 U_max = U; V_max = V;
21 k = 1;
22 for i = 1:length(B_U)
23     for j = 1:length(B_V)
24         sp_U = B_U(i); sp_V = B_V(j);
25         [U_tmp, P_tmp] = fnbrk(sp_U, 'knots', 'coeffs');
26         [V_tmp, Q_tmp] = fnbrk(sp_V, 'knots', 'coeffs');
27         T{k} = spmak({U_tmp, V_tmp}, Q_tmp .* P_tmp);
28         k = k+1;
29     end
30 end
31 for l = 2:length(Omega)
32     omega = Omega{l};
33     a=omega(1,1);b=omega(1,2);c=omega(2,1);d=omega(2,2);
34     ind = 1:length(T); % Indeksi vseh baznih funkcij, ki jih imamo v bazi
35     ostate = []; izbrise = []; presek = [];
36     for n = 1:length(T)
37         bspi = T{n};
38         [knts, cofs] = fnbrk(bspi, 'knots', 'coefficients');
39         U_tmp = knts{1}; V_tmp = knts{2};
40         velikost = size(cofs);
41         if numel(velikost) == 2
42             A_cofs = cofs;
43         else
44             A_cofs = reshape(cofs, velikost(2), velikost(3));
45         end
46         % Dolocimo tiste bazne funkcije, katere kontrolne tocke niso 0.
47         [ind_i, ind_j] = find(A_cofs>0);
48         %prehodimo se cez vse bazne funkcije, da dolocimo nosilec
49         nosilci = zeros(length(ind_i), 4);
50         vel_supp = size(nosilci);
51         len = vel_supp(1);
52         for k = 1:length(ind_i)
53             i = ind_i(k); j = ind_j(k);
54             x0 = U_tmp(i); x1 = U_tmp(i+p+1); y0 = V_tmp(j); y1 = V_tmp(j+q+1);
55             nosilci(k, :) = [x0 x1 y0 y1];
56         end
57         if sum(nosilci(:,2) < a | nosilci(:,1) > b) == len || sum(nosilci(:,4) < c | nosilci(:,3) >
58             d) == len
59             ostate = [ostate, n];
60         elseif sum(nosilci(:,1)>=a & nosilci(:,2)<=b) == len && sum(nosilci(:,3)>=c & nosilci(:,4)<=
61             d) == len
62             izbrise = [izbrise, n];
63         else
64             presek = [presek, n];
65         end
66     end
67 end
```



```

63     end
64 end
65 for i=1:length(presek)
66     k = presek(i);
67     bsp = T{k};
68     bsp_trunc = trunc_2D(bsp, omega);
69     T{k} = bsp_trunc;
70 end
71 UV = fnbrk(T{k}, 'knots');
72 T(izbrise) = [];
73 tmp_P = zeros(1, length(U_max) - p - 1); tmp_P(1) = 1;
74 U_l = fnbrk(finejse(spmak(U_max,tmp_P)), 'knots');
75 tmp_Q = zeros(1, length(V_max) - q - 1); tmp_Q(1) = 1;
76 V_l = fnbrk(finejse(spmak(V_max,tmp_Q)), 'knots');
77 delta = delta/2;
78 T_BU = baza(U_l,p); T_BV = baza(V_l,q);
79 if PDESWITCH == 1
80     T_BU = T_BU(2:end-1); T_BV = T_BV(2:end-1);
81 end
82 k = length(T)+1;
83 for i = 1:length(T_BU)
84     spi = T_BU(i);
85     P = fnbrk(spi, 'coefficients');
86     ind_i = find(P>0);
87     supp_i = [U_l(min(ind_i)), U_l(min(ind_i)+p+1)];
88     if supp_i(1) >= a && supp_i(2) <= b
89         for j=1:length(T_BV)
90             spj = T_BV(j);
91             Q = fnbrk(spj, 'coefficients');
92             ind_j = find(Q>0);
93             supp_j = [V_l(min(ind_j)), V_l(min(ind_j)+q+1)];
94             if supp_j(1) >= c && supp_j(2) <= d
95                 T{k} = spmak({U_l, V_l}, Q .* P');
96                 k = k+1;
97             end
98         end
99     end
100 end
101 U_max = U_l; V_max = V_l;
102 end
103 T_out = T;
104 end

```

PRILOGA C Konstrukcija novega zleпка

```
1 function [bsp, bsp_f] = THB_zlepek(baza, kontrolne)
2 % THB_zlepek konstruira prirezan zlepek z kontrolnimi točkami.
3 % Predpostavljamo, da je baza dvo-dimenzionalna in prirezana. To pomeni, da
4 % so matrike koeficientov različnih dimenzij. Za rezultat vrnemo bsp, ki je
5 % v večini primerov cell array s kar se da malo zleпки konstruiranih z
6 % ukazom spmak, ce pa so vse matrike enake velikosti pa vrnemo en sam
7 % zlepek. Poleg tega lahko funkcija vrne tudi zlepek predstavljen z
8 % anonimno funkcijo, kar je v dveh dimezijah uporabno za risanje in
9 % racunanje.
10 velikosti = []; matrike = {}; v_vozlov = {};
11 for i=1:length(baza)
12     bspi = baza{i};
13     [vozli, koef, nmb] = fnbrk(bspi, 'knots', 'coeffs', 'number');
14     M = reshape(koef, nmb);
15     m = numel(M);
16     if ~ismember(m, velikosti)
17         velikosti = [velikosti, m];
18         k = find(velikosti == m);
19         matrike{k} = kontrolne(i).*M;
20     else
21         k = find(velikosti == m);
22         matrike{k} = matrike{k} + kontrolne(i).*M;
23     end
24     v_vozlov{k} = vozli;
25 end
26 if length(velikosti) == 1
27     bsp = spmak(v_vozlov{1}, matrike{1});
28     bsp_f = @(x,y) fnval(bsp, [x;y]);
29 else
30     bsp = {};
31     for i = 1:length(matrike)
32         bsp{i} = spmak(v_vozlov{i}, matrike{i});
33     end
34     bsp_f = @(x,y) fnval(bsp{1}, [x;y]);
35     for i=2:length(bsp)
36         bsp_fi = @(x,y) fnval(bsp{i}, [x;y]);
37         bsp_f = @(x,y) bsp_f(x,y) + bsp_fi(x,y);
38     end
39 end
40 end
```

PRILOGA D Hierarhična baza premika zlepka B_{222}

```
1 function [basis, triang, koordinate] = HBox_Baza_alt(Omega, m)
2 %HBOX_BAZA Vrne hierarhico bazo prostora premikov zlepka B222.
3 % Parameter Omega je cell array v katerem so shranjena obmocja
4 % hierarhije. Prvo obmocje je kvadratno obmocje predstavljeno z matriko
5 % [a b; c d], ki predstavlja obmocje [a,b]x[c,d]. Vsa ostala obmocja pa so
6 % dolocena s stirimi parametri [x0, y0, sh_right, sh_up], kjer sta x0, y0
7 % kartezični koordinati nekega skrajno levo-spodaj oglisca nosilca iz
8 % visjega prostora, sh_up,
9 % sh_down pa naravni stevili, ki povesta kolikokrat nosilec premaknemo
10 % desno in gor in vse kombinacije vmes. Na ta nacín ohranimo krepki
11 % pogoj, da so obmocja v hierarhiji unija nosilcev zlepkov iz visjega
12 % nivoja. Parameter m je
13 % zacetni faktor skaliranja. Na vsakem nivoju ga pomnozimo z 2.
14 % Funkcija vrne cell array basis, kjer so shranjeni bazni zlepki
15 % predstavljeni kot @funkcije in cell array triang, kjer shranimo nosilce
16 % baznih zlepkov predstavljenih s strukturo triang.
17 % Hard-codamo vse premike, da nosilec nekega zlepka skaliranega z m,
18 % pokrijemo z nosilci zlepkov, ki skalirani z 2*m.
19 shifts = [0 0; ...
20           1 0; ...
21           2 0; ...
22           0 1; ...
23           1 1; ...
24           2 1; ...
25           3 1; ...
26           0 2; ...
27           1 2; ...
28           2 2; ...
29           3 2; ...
30           4 2; ...
31           1 3; ...
32           2 3; ...
33           3 3; ...
34           4 3; ...
35           2 4; ...
36           3 4; ...
37           4 4];
38 % Dolocimo kvadratno obmocje Omega_0.
39 a=Omega{1}(1,1); b=Omega{1}(1,2); c=Omega{1}(2,1); d=Omega{1}(2,2);
40 % Diskretizacija kvadratnega obmocja za namen testiranja vsebovanosti
41 x = linspace(a+10.^-4, b-10.^-4); y = linspace(c+10.^-4, d-10.^-4);
42 [mx, my] = meshgrid(x, y); rx = mx(:); ry = my(:);
43 % Ustvarimo zacetno bazo, ki jo tekom funkcije spreminjamo.
44 [basis, triang] = Box_Baza(Omega{1}, m);
45 % Za hitro iskanje aktivnih elementov si prepisemo skrajno levo-spodaj
46 % oglisca vseh baznih zlepkov. Tudi to matriko tekom funkcije skrbno
47 % posodabljamó, da se bodo indeksi ujemali vse do konca.
48 nivo = 0;
49 koordinate = [];
50 for i=1:length(triang)
51     koordinate = [koordinate; triang{i}.Points(1,:), nivo];
52 end
53 % Sprehodimo se cez vse nivoje dodajamo/odstranjujemo bazne zlepke
54 for l = 2:length(Omega)
55     % Najprej dolocimo indekse skrajnih koordinat vseh baznih zlepkov, ki jih
56     % odstranimo.
57     premiki = Omega{l};
58     [vrst, stolp] = size(premiki);
59     ind_odstrani = [];
60     % Tukaj shranimo koordinate skrajnih oglisc, kjer bomo kasneje dodali nove zlepke
61     oglisca = [];
62     for i = 1:vrst
63         koor = [premiki(i,:), nivo];
64         enakost = koordinate == koor;
```

```

65     ind = enakost(:,1).*enakost(:,2);
66     ind_odstrani = [ind_odstrani; find(ind==1)];
67     oglisca = [oglisca; koor];
68     end
69     % Najprej odstranimo koordinate iz matrike koordinate, da se na
70     % naslednjem nivoju vec ne pojavijo.
71     koordinate(ind_odstrani,:) = [];
72     % Odstranimo ustrezni bazni zlepek iz hierarhicne baze
73     basis(ind_odstrani) = [];
74     % Odstranimo tudi nosilec iz triangulacije
75     triang(ind_odstrani) = [];
76     % Sedaj lahko dodamo nove finejse bazne zlepke na celotnem obmocju.
77     % Dolocimo zadnje mesto v bazi, kjer dodamo novega.
78     k = length(basis)+1;
79     % Skalirni faktor pomnozimo z 2, da dobimo finejsega
80     m = m*2;
81     nivo = nivo+1;
82     % Sprehodimo se cez vse koordinate, kjer bodo finejsi zlepki
83     vel = size(oglisca);
84     for i = 1:vel(1)
85         ogl = oglisca(i,1:2);
86         for j = 1:length(shifts)
87             shift = [ogl + [shifts(j,1) * 1/m, shifts(j,2) * 1/m], nivo];
88             vsebovan = double(koordinate == shift);
89             if sum(vsebovan(:,1).*vsebovan(:,2).*vsebovan(:,3)) == 0 % Se ne obstaja
90                 [BB, TR] = B222.fun(shift, m);
91                 % Preverimo, ali je nosilec finejsega sploh vsebovan v
92                 % kvadratnem obmocju [0, 1]^2
93                 ID = pointLocation(TR, [rx ry]);
94                 if ~isempty(find(~isnan(ID)))
95                     basis{k} = BB;
96                     triang{k} = TR;
97                     koordinate = [koordinate; shift];
98                     k = k+1;
99                 end
100             end
101         end
102     end
103 end
104 end
105 end

```

PRILOGA E Hierarhična baza premika zlepka B_{222} (Homogena)

```
1 function [basis, triang, koefs, koordinate] = HBox_Baza_PDE_alt(Omega, m)
2 %HBOX_BAZA_PDE Vrne hierarhično bazo prostora premikov zlepka B222 primerno za uporabo pri reševanju
   PDE.
3 % Parameter Omega je cell array v katerem so shranjena območja
4 % hierarhije. Prvo območje je kvadratno območje predstavljeno z matriko
5 % [a b; c d], ki predstavlja območje [a, b] x [c, d]. Vsa ostala območja pa so
6 % določena s štirimi parametri [x0, y0, sh-right, sh-up], kjer sta x0, y0
7 % kartezični koordinati nekega skrajno levo-spodaj oglisca nosilca iz
8 % visjega prostora, sh-up,
9 % sh-down pa naravni stevili, ki povesta kolikokrat nosilec premaknemo
10 % desno in gor in vse kombinacije vmes. Na ta način ohranimo krepki
11 % pogoj, da so območja v hierarhiji unija nosilcev zlepkov iz visjega
12 % nivoja. Parameter m je
13 % zacetni faktor skaliranja. Na vsakem nivoju ga pomnožimo z 2.
14 % Funkcija vrne cell array basis, kjer so shranjeni bazni zlepki
15 % predstavljeni kot @funkcije in cell array triang, kjer shranimo nosilce
16 % baznih zlepkov predstavljenih s strukturo triang. Funkcija ustvari
17 % robne funkcije, ki izpolnjujejo homogene pogoje na robu območja Omega.
18
19 % Hard-codamo vse premike, da nosilec nekega zlepka skaliranega z m,
20 % pokrijemo z nosilci zlepkov, ki skalirani z 2*m.
21 shifts = [0 0; ...
22           1 0; ...
23           2 0; ...
24           0 1; ...
25           1 1; ...
26           2 1; ...
27           3 1; ...
28           0 2; ...
29           1 2; ...
30           2 2; ...
31           3 2; ...
32           4 2; ...
33           1 3; ...
34           2 3; ...
35           3 3; ...
36           4 3; ...
37           2 4; ...
38           3 4; ...
39           4 4];
40 % Določimo kvadratno območje Omega_0.
41 a=Omega{1}(1,1); b=Omega{1}(1,2); c=Omega{1}(2,1); d=Omega{1}(2,2);
42 [basis_zacetna, triang_zacetni] = Box_Baza(Omega{1}, m);
43 basis = {}; triang = {};
44 % Zacetno bazo preuredimo, tako, da izluscimo take linearne kombinacije, da
45 % upostevajo homogene robne pogoje. Potem to bazo nadalje preoblikujemo!
46 % Iscemo linearne kombinacije baznih funkcij, ki bodo imele na robu
47 % vrednost 0. Te poiscemo s pomocjo MDS algoritma, kot jedro matrike.
48 tocke = [linspace(0,1,4*m+1)' zeros(4*m+1,1); ...
49          zeros(4*m+1,1) linspace(0,1,4*m+1)'; ...
50          linspace(0,1,4*m+1)' ones(4*m+1,1); ...
51          ones(4*m+1,1) linspace(0,1,4*m+1)'];
52 A = zeros(max(size(tocke)), length(basis_zacetna));
53 for i = 1:length(basis_zacetna)
54     A(:,i) = box_val(basis_zacetna{i}, triang_zacetni{i}, tocke);
55 end
56 ker = mds_jedro(A);
57 k = 1;
58 basis_homogen = {}; triang_homogen = {}; koefs = {};
59 for i = 1:min(size(ker))
60     koef_orig = 1/norm(ker(:,i)) .* ker(:,i); %koeficienti
61     ind = find(abs(koef_orig) >= 10^-3);
62     koef = koef_orig(ind);
63     fun = Box_LinCmb({basis_zacetna{ind}}, {triang_zacetni{ind}}, koef);
```

```

64     basis{k} = fun;
65     triang{i} = {triang_zacetni{ind}};
66     koefs{i} = koef;
67     k = k+1;
68 end
69 notranji_ind = find(cellfun(@length, triang) == 1); % tej so zagotovo v notranjosti
70
71 nivo = 0;
72 koordinate = []; % Samo te funkcije so lahko potencialni kandidati.
73 for i=1:length(notranji_ind)
74     ind = notranji_ind(i);
75     koordinate = [koordinate; [triang{ind}{1}.Points(1,:), nivo]];
76 end
77 for l = 2:length(Omega)
78     premiki = Omega{l};
79     [vrst, stolp] = size(premiki);
80     ind_odstrani = [];
81     oglisca = [];
82     for i = 1:vrst
83         koor = [premiki(i,:), nivo];
84         enakost = koordinate == koor;
85         ind = enakost(:,1).*enakost(:,2);
86         ind_odstrani = [ind_odstrani; find(ind==1)];
87         oglisca = [oglisca; koor];
88         koordinate(find(ind==1),:) = [];
89     end
90     basis(ind_odstrani) = []; triang(ind_odstrani) = []; koefs(ind_odstrani) = [];
91     % Sedaj lahko dodamo nove finejse bazne zlepke na celotnem območju.
92     % Dolocimo zadnje mesto v bazi, kjer dodamo novega.
93     k = length(basis)+1;
94     % Skalirni faktor pomnozimo z 2, da dobimo finejsega
95     m = m*2;
96     nivo = nivo+1;
97     % Sprehodimo se cez vse koordinate, kjer bodo finejsi zlepki
98     vel = size(oglisca);
99     for i = 1:vel(1)
100         ogl = oglisca(i,1:2);
101         for j = 1:length(shifts)
102             shift = [ogl + [shifts(j,1) * 1/m, shifts(j,2) * 1/m], nivo];
103             vsebovan = double(koordinate == shift);
104             if sum(vsebovan(:,1).*vsebovan(:,2).*vsebovan(:,3)) == 0 % Se ne obstaja
105                 [BB, TR] = B222_fun(shift, m);
106
107                 basis{k} = Box_LinCmb({BB}, {TR}, 1);
108                 triang{k} = {TR};
109                 koefs{k} = 1;
110                 koordinate = [koordinate; shift];
111                 k=k+1;
112             end
113         end
114     end
115 end
116 end
117 end

```

PRILOGA F Nova domena

```
1 function [G_fun,J,J_mat,kont] = kolobar_alt(1)
2 %KOLOBAR Vrne predpis in jakobijan ploskve, ki opise kolobar v ravnini.
3 % Funkcija sprejme kot opsijski parameter l, ki oznacuje sirino kolobarja
4 % med dvema kroznicama v prvem kvadrantu. Prva kroznica ima vedno polmer
5 % r=1, medtem ko ima druga kroznica parameter l+1. Ploskev je
6 % predstavljena s tenzorskim produktom B-zlepkov stopnje 3. V primeru, da
7 % parameter l ni podan se privzame, da je l=1.
8 if nargin==0
9     l=1;
10 end
11 U = [0 0 0 linspace(0,1,10) 1 1 1];
12 V = [0 0 0 linspace(0,1,10) 1 1 1];
13 p = 3;
14 q = 3;
15 B_U = baza(U,p);B_V = baza(V,q);
16 T0 = {};
17 k = 1;
18 for i = 1:length(B_U)
19     for j = 1:length(B_V)
20         sp_U = B_U(i);
21         sp_V = B_V(j);
22         [U_tmp, P_tmp] = fnbrk(sp_U, 'knots', 'coeffs');
23         [V_tmp, Q_tmp] = fnbrk(sp_V, 'knots', 'coeffs');
24         T0{k} = spmak({U_tmp, V_tmp}, Q_tmp .* P_tmp);
25         k = k+1;
26     end
27 end
28 length(T0);
29 st = length(U)-p-1;
30 tocke = [cos(linspace(0,st-1,st).*(pi/(2*(st-1))))];...
31         sin(linspace(0,st-1,st).*(pi/(2*(st-1))))];
32 tocke2 = (1+l)*[cos(linspace(0,st-1,st).*(pi/(2*(st-1))))];...
33             sin(linspace(0,st-1,st).*(pi/(2*(st-1))))];
34 n = 12;
35 tocke_mreza = [linspace(tocke(1,1), tocke2(1,1),n)];...
36               linspace(tocke(2,1), tocke2(2,1),n)];
37 for i =2:length(B_U)
38     tocke_mreza = [tocke_mreza [linspace(tocke(1,i), tocke2(1,i),n)];...
39                   linspace(tocke(2,i), tocke2(2,i),n)]];
40 end
41 kont = [tocke_mreza;zeros(1,144)];
42 bsp = spmak({U,V}, kont, [3 12 12]);
43 G_fun = @(x,y) fnval(bsp, [x;y]);
44 dxBsp = fnder(bsp, [1 0]);
45 dyBsp = fnder(bsp, [0 1]);
46 dxBspVal_fun = @(u) fnval(dxBsp,[u(:,1)';u(:,2)']);
47 dyBspVal_fun = @(u) fnval(dyBsp,[u(:,1)';u(:,2)']);
48 paren = @(x, varargin) x(varargin{:});
49 J_mat = @(u) [paren(dxBspVal_fun(u), 1,1)   paren(dxBspVal_fun(u), 2,1); ...
50               paren(dyBspVal_fun(u), 1,1)   paren(dyBspVal_fun(u), 2,1)];
51 J = @(x,y) paren(dxBspVal_fun([x,y]), 1,1:length([x,y])).* ...
52 paren(dyBspVal_fun([x,y]), 2,1:length([x,y])) - ...
53 paren(dyBspVal_fun([x,y]), 1,1:length([x,y])) .* ...
54 paren(dxBspVal_fun([x,y]), 2,1:length([x,y]));
55 end
```

PRILOGA G Minimalna določitvena množica

```
1 function MDS = md_set(T)
2 %md_set Poisce minimalno dolocitveno mnozico za matriko T
3 % Funkcijo nadalje uporabimo za racunanje jedra matrike T s cim vec
4 % nicelnimi koeficienti. Vhodni podatek je matrika T velikosti m*n.
5 MDS = []; %Minimalna Dolocitvena Mnozica
6 [rws_T, cls_T] = size(T);
7 I = linspace(1,cls_T,cls_T); % zacetni vektor vseh indeksov, ki ga spreminjamo
8 I_const = linspace(1,cls_T,cls_T); %zacetni vektor vseh indeksov, ki se ne spreminja
9 for i=1:cls_T
10     enka = [i]; %koeficient, ki mora biti 1
11     nule = [MDS, I(find(I~=i))]; %koeficienti, ki jih postavimo na 0
12     prosti = I_const(find(~ismember(I_const,[enka nule]))); %prosti koeficienti, ki jih dolocamo
13     %Prvi primer, ko nimamo nobenega prostega koeficienta in se resitev
14     %ponudi sama. Moramo preveriti ali je ustrezna.
15     if isempty(prosti)
16         v = zeros(cls_T,1); v(i) = 1;
17         %Preverimo ali je ta vektor v jedru matrike T
18         if T*v == zeros(rws_T,1)
19             v_abs = abs(v);
20             r = min(find(v_abs == max(v_abs)));
21             MDS = [MDS r];
22         end
23     else %Imamo kaksen prost koeficient
24         A0 = [T(:,enka), T(:,prosti)]; %Vsi prosti in tisti, ki mora biti ena
25         [rws, cls] = size(A0);
26         %Preverimo, ali resitev obstaja. Ce obstaja to pomeni, da je rang
27         %te matrike za eno manjsi kot stevilo stolpcev te matrike od koder
28         %sklepamo, da lahko en stolpec zapisemo s kombinacijo ostalih. Ta
29         %stolpec pa je ravno tisti, ki ima koeficient enak 1.
30         if rank(A0) == cls-1
31             A = T(:,prosti); b = -T(:,enka);
32             a = linsolve(A,b);
33             %Sestavimo ustrezen vektor
34             v = zeros(cls_T,1);
35             v(enka) = 1;
36             v(prosti) = a;
37             v_abs = abs(v);
38             r = min(find(v_abs == max(v_abs)));
39             MDS = [MDS, r];
40         end
41     end
42     I = I(find(I~=i));
43 end
44 end

1 function ker = mds_jedro(T)
2 %mds_jedro Doloci jedro matrike T z upostevanjem minimalne dolocitvene mnozice
3 % Matrika T je velikosti m*n, izhodna matrika ker pa je velikosti n*dim(ker(T)).
4 MDS = md_set(T); %dolocimo minimalno dolocitveno mnozico indeksov
5 [rws, cls] = size(T);
6 ker = zeros(cls, length(MDS)); % Ta bo postala jedro matrike T
7 ind = linspace(1,cls,cls);
8 for i=1:length(MDS)
9     %Na vsakem koraku postavimo eno iz dolocitvene mnozice na 1, ostale iz
10    %MDS na 0, indeksi ki ostanejo pa jih lahko prosto dolocimo
11    enka = [MDS(i)]; nule = MDS(find(MDS~=enka)); prosti = ind(find(~ismember(ind, [enka, nule])));
12    %Resimo podsystem
13    A = T(:,prosti); b = -T(:,enka); a = linsolve(A,b);
14    %sestavimo vektor iz jedra
15    v = zeros(cls,1); v(enka) = 1; v(prosti) = a; ker(:,i) = v;
16 end
17 end
```


PRILOGA H Gaussova integracijska pravila

```
1 function area = gauss_quad_1D(func,a,b)
2 %gauss_quad_1D Izracuna dolocen integral funkcije func na intervalu [a,b] s
3 %pomocjo Gaussove integracije s sestimi tockami in natančnostjo 70.
4 % Parameter func je @ funkcija, parametra a,b pa dolocata interval [a,b] nad katerim integriramo.
5
6 % Dolocimo interpolacijske tocke
7 z0=((0.5*(-a+b))*-(0.93246951420315202781230155449399460913476573771228982487254961652661350084422)
+((a+b)/2));
8 z1=((0.5*(-a+b))*-(0.66120938646626451366139959501990534700644856439517007081452670585218349660716)
+((a+b)/2));
9 z2=((0.5*(-a+b))*-(0.23861918608319690863050172168071193541861063014002135018139516457427493427565)
+((a+b)/2));
10 z3=((0.5*(-a+b))*(0.23861918608319690863050172168071193541861063014002135018139516457427493427563)
+((a+b)/2));
11 z4=((0.5*(-a+b))*(0.66120938646626451366139959501990534700644856439517007081452670585218349660715)
+((a+b)/2));
12 z5=((0.5*(-a+b))*(0.93246951420315202781230155449399460913476573771228982487254961652661350084421)
+((a+b)/2));
13 z = [z0;z1;z2;z3;z4;z5];
14 alpha = [...
15 (0.171324492379170345040296142172732893526822501484043982398642289863797262898); ...
16 (0.360761573048138607569833513837716111661521892746745482289738520362451949918); ...
17 (0.467913934572691047389870343989550994811655605769210535311619189773750787182); ...
18 (0.467913934572691047389870343989550994811655605769210535311619189773750787182); ...
19 (0.360761573048138607569833513837716111661521892746745482289738520362451949918); ...
20 (0.171324492379170345040296142172732893526822501484043982398642289863797262898)];
21 area = ((b-a)/2 .* sum(alpha.*(func(z))));
22 end

1 function [area, mx, my] = gauss_quad_2D(func,a,b,c,d)
2 %gauss_quad_2D Izracuna dolocen integral funkcije func na obmocju [a,b]x[c,d] s
3 %pomocjo Gaussove integracije s sestimi tockami in natančnostjo 70.
4 % Parameter func je @ funkcija, parameter Omega pa doloca obmocje [a,b]x[c,d] nad katerim
integriramo.
5
6 % Dolocimo interpolacijske tocke
7 z0_x=((0.5*(-a+b))
*-(0.93246951420315202781230155449399460913476573771228982487254961652661350084422)+((a+b)/2));
8 z1_x=((0.5*(-a+b))
*-(0.66120938646626451366139959501990534700644856439517007081452670585218349660716)+((a+b)/2));
9 z2_x=((0.5*(-a+b))
*-(0.23861918608319690863050172168071193541861063014002135018139516457427493427565)+((a+b)/2));
10 z3_x=((0.5*(-a+b))*(0.23861918608319690863050172168071193541861063014002135018139516457427493427563)
+((a+b)/2));
11 z4_x=((0.5*(-a+b))*(0.66120938646626451366139959501990534700644856439517007081452670585218349660715)
+((a+b)/2));
12 z5_x=((0.5*(-a+b))*(0.93246951420315202781230155449399460913476573771228982487254961652661350084421)
+((a+b)/2));
13 z0_y=((0.5*(-c+d))
*-(0.93246951420315202781230155449399460913476573771228982487254961652661350084422)+((c+d)/2));
14 z1_y=((0.5*(-c+d))
*-(0.66120938646626451366139959501990534700644856439517007081452670585218349660716)+((c+d)/2));
15 z2_y=((0.5*(-c+d))
*-(0.23861918608319690863050172168071193541861063014002135018139516457427493427565)+((c+d)/2));
16 z3_y=((0.5*(-c+d))*(0.23861918608319690863050172168071193541861063014002135018139516457427493427563)
+((c+d)/2));
17 z4_y=((0.5*(-c+d))*(0.66120938646626451366139959501990534700644856439517007081452670585218349660715)
+((c+d)/2));
18 z5_y=((0.5*(-c+d))*(0.93246951420315202781230155449399460913476573771228982487254961652661350084421)
+((c+d)/2));
19 z_x = [z0_x;z1_x;z2_x;z3_x;z4_x;z5_x]; z_y = [z0_y;z1_y;z2_y;z3_y;z4_y;z5_y];
20 alpha = [(0.171324492379170345040296142172732893526822501484043982398642289863797262898); ...
21 (0.360761573048138607569833513837716111661521892746745482289738520362451949918); ...
22 (0.467913934572691047389870343989550994811655605769210535311619189773750787182); ...
```

```

23 (0.467913934572691047389870343989550994811655605769210535311619189773750787182);...
24 (0.360761573048138607569833513837716111661521892746745482289738520362451949918);...
25 (0.171324492379170345040296142172732893526822501484043982398642289863797262898)];
26 % Naredimo kartezicni produkt vseh z-vrednosti, da dobimo mrezo. To mrezo potem
27 % podremo v en sam stolpec. Problem je, ce imamo vec intervalov nad
28 % katerimi bi zeleti racunati na enkrat. V tem primeru moramo za vsak
29 % interval narrediti mrezo in jo podreti v stolpec. Vse te stolpce potem
30 % shranimo v eno samo matriko, da lahko izracunamo vse na enkrat.
31 mx_ = repmat(z_x(:)', [6,1]); mx = reshape(mx_, [36, length(a)]);
32 my_ = repmat(z_y, [6 1]);
33 my_ = reshape(my_, [6, length(a).*6]); my = reshape(my_, [36, length(a)]);
34 [malphax_, malphay_] = meshgrid(alpha', alpha');
35 malphax_ = repmat(malphax_(:), [1, length(a)]); malphay_ = repmat(malphay_(:), [1, length(a)]);
36 vrednosti_funkcij = (func(double([mx(:)' ; my(:)'])));
37 vsota = sum(malphax_.*malphay_.*reshape(vrednosti_funkcij, [36, length(a)]));
38 area = (((d-c))./2) .*(((b-a))./2) .* vsota;
39 end

```

PRILOGA I Implementacija zlepka B_{222}

```
1 function [bb_form,TR] = B222_fun(shift, scale)
2 %B222_fun Skonstruira BB-obliko skatlatega zlepka B211
3 % Funkcija kot argument lahko sprejme dva neobvezna parametra. Brez njiju
4 % je zlepek postavljen v izhodišče in skaliran s faktorjem ena. Argument
5 % shift naj bo dvoelementni vektor s katerim opisemo premik v x in y
6 % smeri. Argument scale pa je realno stevilo s katerim ustrezno skaliramo
7 % oglisca triangulacije, da zlepek povečamo ali zmanjšamo. Zlepek skaliramo s
8 % faktorjem 1/scale! Rezultat
9 % funkcije je cell array bb_form, ki vsebuje predpise Bezierevih krp
10 % predstavljene z anonimnimi funkcijami nad vsakim delom triangulacije.
11 % Poleg BB oblike vrnemo tudi Triangulacijo kot Matlab strukturo.
12
13 switch nargin
14     case 2
15         mu = scale; shift_x = shift(1); shift_y = shift(2);
16     case 1
17         shift_x = shift(1); shift_y = shift(2); mu = 1;
18     case 0
19         shift_x = 0; shift_y = 0; mu = 1;
20     otherwise
21         error('Prevec vhodnih podatkov')
22 end
23 B_4 = @(u,v,i,j,k) (24 / (factorial(i)*factorial(j)*factorial(k))) .* u.^i .* v.^j .* (-u-v+1).^k;
24 P = [0 0 ; 1 0; 2 0 ; ...
25     0 1 ; 1 1 ; 2 1 ; 3 1 ;...
26     0 2 ; 1 2 ; 2 2 ; 3 2 ; 4 2 ;
27     1 3 ; 2 3 ; 3 3 ; 4 3 ;
28     2 4 ; 3 4 ; 4 4];
29 P = P./mu;
30 P(:,1) = P(:,1) + shift_x;
31 P(:,2) = P(:,2) + shift_y;
32 % Zapisimo vse trikotnike, na nacin, da je prva koordinata vedno pri pravem
33 % kotu, druga desno in tretja levo.
34 T = [4 1 5 ;... %1
35     2 5 1 ; ... %2
36     5 2 6 ; ... %3
37     3 6 2 ; ... %4
38     6 3 7 ; ... %5
39     8 4 9 ; ... %6
40     5 9 4 ; ... %7
41     9 5 10; ... %8
42     6 10 5; ... %9
43     10 6 11; ... %10
44     7 11 6; ... %11
45     11 7 12; ... %12
46     9 13 8; ... %13
47     13 9 14; ... %14
48     10 14 9; ... %15
49     14 10 15; ...%16
50     11 15 10; ...%17
51     15 11 16; ...%18
52     12 16 11; ...%19
53     14 17 13; ...%20
54     17 14 18; ...%21
55     15 18 14; ...%22
56     18 15 19; ...%23
57     16 19 15]; %24
58 TR = triangulation(T,P);
59 %Na silo shranimo BB-koeficiente v matriko.
60 koef = zeros(17);
61 koef(4, 5:8) = 1/24;
62 koef(5, 4:10) = [1 2 3 4 3 2 1]/24;
63 koef(6, 4:11) = [1 3 4 6 6 4 3 1]/24;
64 koef(7, 4:12) = [1 4 6 8 10 8 6 4 1]/24;
65 koef(8, 4:13) = [1 3 6 10 12 12 10 6 3 1]/24;
66 koef(9, 5:13) = [2 4 8 12 12 12 8 4 2]/24;
67 koef(10, 5:14) = [1 3 6 10 12 12 10 6 3 1]/24;
```

```

68 koef(11, 6:14) = [1 4 6 8 10 8 6 4 1]/24;
69 koef(12, 7:14) = [1 3 4 6 6 4 3 1]/24;
70 koef(13, 8:14) = [1 2 3 4 3 2 1]/24;
71 koef(14, 10:13) = 1/24;
72 %Povezava oglisc trikotnika z BB koeficienti
73 mapp = [1 1 ; ...
74         1 5 ; ...
75         1 9 ; ...
76         5 1 ; ...
77         5 5 ; ...
78         5 9 ; ...
79         5 13 ; ...
80         9 1 ; ...
81         9 5 ; ...
82         9 9 ; ...
83         9 13 ; ...
84         9 17 ; ...
85         13 5 ; ...
86         13 9 ; ...
87         13 13 ; ...
88         13 17 ; ...
89         17 9 ; ...
90         17 13 ; ...
91         17 17];
92 bb_form = cell(24);
93 for i=1:length(TR.ConnectivityList)
94     % Vzamemo ustrezen trikotnik
95     Ti = TR.ConnectivityList(i,:);
96     %Pridobimo indekse oglisc
97     Ai = Ti(2); Bi = Ti(1); Ci = Ti(3);
98     % A je vedno pri pravem kotu, B je desno in C levo.
99     %Kontrolna tocka pri pravem kotu
100    C400_ind = mapp(Ai, :);
101    C400 = [P(Ai,:) koef(C400_ind(1), C400_ind(2))];
102    C400_koor = P(Ai,:);
103    %Kontrolna tocka desno od pravega kota
104    C004_ind = mapp(Bi, :);
105    C004 = [P(Bi,:) koef(C004_ind(1), C004_ind(2))];
106    C004_koor = P(Bi,:);
107    %Kontrolna tocka levo od pravega kota
108    C040_ind = mapp(Ci, :);
109    C040 = [P(Ci,:) koef(C040_ind(1), C040_ind(2))];
110    C040_koor = P(Ci,:);
111    % Na vsaki daljici trikotnika moramo dolociti se tri vmesne kontrolne
112    % tocke kot prikazuje naslednja skica:
113    %
114    %
115    %
116    %
117    %
118    %
119    %
120    %
121    %
122    %
123    %
124    %
125    %
126    %
127    %
128    %
129    %
130    %
131    %
132    %
133    %
134    %
135    %
136    % Kontrolna tocka med C400 in C004 --> C202 (Pravi kot in desno)
137    C202 = [(C400_koor + C004_koor)./2, koef((C400_ind(1)+C004_ind(1))/2, (C400_ind(2) + C004_ind(2)
138    )/2)];
139    C202_koor = C202(1:2); C202_ind = [(C400_ind(1)+C004_ind(1))/2, (C400_ind(2) + C004_ind(2))/2];
140    % Kontrolna tocka med C040 in C004 --> C220 (Pravi kot in levo)
141    C220 = [(C400_koor + C040_koor)./2, koef((C400_ind(1)+C040_ind(1))/2, (C400_ind(2) + C040_ind(2)
142    )/2)];
143    C220_koor = C220(1:2); C220_ind = [(C400_ind(1)+C040_ind(1))/2, (C400_ind(2) + C040_ind(2))/2];
144    % Kontrolna tocka med C040 in C004 --> C022 (hipotenuza)
145    C022 = [(C040_koor + C004_koor)./2, koef((C004_ind(1)+C040_ind(1))/2, (C004_ind(2) + C040_ind(2)
146    )/2)];

```

```

144 C022_koor = C022(1:2); C022_ind = [(C004_ind(1)+C040_ind(1))/2, (C004_ind(2) + C040_ind(2))/2];
145 % KONTROLNE TOCKE NA CETRTINAH DALJIC
146 % Kontrolna tocka med C400 in C202
147 C301 = [(C400_koor+ C202_koor)./2, koef((C400_ind(1) + C202_ind(1))/2, (C400_ind(2) + C202_ind
(2))/2)];
148 %C301_koor = (C400_koor+ C202_koor)./2; C301_ind = [(C400_ind(1) + C202_ind(1))/2, (C400_ind(2)
+ C202_ind(2))/2];
149 %Kontrolna tocka med C202 in C004
150 C103 = [(C202_koor + C004_koor)./2, koef((C004_ind(1) + C202_ind(1))/2, (C004_ind(2) + C202_ind
(2))/2)];
151 %Kontrolna tocka med C040 in C022
152 C031 = [(C022_koor + C040_koor)./2, koef((C040_ind(1) + C022_ind(1))/2, (C040_ind(2) + C022_ind
(2))/2)];
153 %Kontrolna tocka med C004 in C022
154 C013 = [(C022_koor + C004_koor)./2, koef((C004_ind(1) + C022_ind(1))/2, (C004_ind(2) + C022_ind
(2))/2)];
155 %Kontrolna tocka med C040 in C220
156 C130 = [(C220_koor + C040_koor)./2, koef((C040_ind(1) + C220_ind(1))/2, (C040_ind(2) + C220_ind
(2))/2)];
157 %Kontrolna tocka med C220 in C400
158 C310 = [(C220_koor + C400_koor)./2, koef((C400_ind(1) + C220_ind(1))/2, (C400_ind(2) + C220_ind
(2))/2)];
159 C310_koor = (C220_koor + C400_koor)./2; C310_ind = [(C400_ind(1) + C220_ind(1))/2, (C400_ind(2)
+ C220_ind(2))/2];
160 %KONTROLNE TOCKE ZNOTRAJ TRIKOTNIKA
161 %Kontrolna tocka med C202 in C022
162 C112 = [(C202_koor + C022_koor)./2, koef((C202_ind(1) + C022_ind(1))./2, (C202_ind(2) + C022_ind
(2))./2)];
163 C112_koor = (C202_koor + C022_koor)./2; C112_ind = [(C202_ind(1) + C022_ind(1))./2, (C202_ind(2)
+ C022_ind(2))./2];
164 %Kontrolni tocki med C301 in C031
165 C211 = [(C310_koor + C112_koor)./2, koef((C310_ind(1) + C112_ind(1))./2, (C310_ind(2) + C112_ind
(2))./2)];
166 C121 = [(C220_koor+C022_koor)./2, koef((C220_ind(1)+C022_ind(1))./2, (C220_ind(2)+C022_ind(2)
./2)];
167 %Sestavimo zlepek
168 Bi = @(u,v) C400.*B_4(u,v,4,0,0) + C004.*B_4(u,v,0,0,4) + C040.*B_4(u,v,0,4,0) + ...
169 C202.*B_4(u,v,2,0,2) + C220.*B_4(u,v,2,2,0) + C022.*B_4(u,v,0,2,2) + ...
170 C301.*B_4(u,v,3,0,1) + C103.*B_4(u,v,1,0,3) + C031.*B_4(u,v,0,3,1) + ...
171 C013.*B_4(u,v,0,1,3) + C130.*B_4(u,v,1,3,0) + C310.*B_4(u,v,3,1,0) + ...
172 C112.*B_4(u,v,1,1,2) + C211.*B_4(u,v,2,1,1) + C121.*B_4(u,v,1,2,1);
173 bb_form{i}=Bi;
174 end
175 end

```

PRILOGA J Implementacija odvoda zlepka B_{222} po smeri e_1

```
1 function [bb_form,TR] = De1_B222_fun(shift , scale)
2 %De2_B222_fun Skonstruira BB-obliko skatlastega zlepka De2_B211
3 % Funkcija kot argument lahko sprejme dva neobvezna parametra. Brez njiju
4 % je zlepek postavljen v izhodišče in skaliran s faktorjem ena. Argument
5 % shift naj bo dvoelementni vektor s katerim opisemo premik v x in y
6 % smeri. Argument scale pa je realno stevilo s katerim ustrezno skaliramo
7 % oglisca triangulacije, da zlepek povečamo ali zmanjšamo. Zlepek skaliramo s
8 % faktorjem 1/scale! Rezultat
9 % funkcije je cell array bb_form, ki vsebuje predpise Bezierevih krp
10 % predstavljene z anonimnimi funkcijami nad vsakim delom triangulacije.
11 % Poleg BB oblike vrnemo tudi Triangulacijo kot Matlab strukturo.
12
13 switch nargin
14     case 2
15         mu = scale; shift_x = shift(1); shift_y = shift(2);
16     case 1
17         shift_x = shift(1); shift_y = shift(2); mu = 1;
18     case 0
19         shift_x = 0; shift_y = 0; mu = 1;
20     otherwise
21         error('Prevec vhodnih podatkov')
22 end
23 B_3 = @(u,v,i,j,k) (6 / (factorial(i)*factorial(j)*factorial(k))).* u.^i .* v.^j.*(-u-v+1).^k;
24 P = [0 0 ; 1 0; 2 0 ; ...
25      0 1 ; 1 1 ; 2 1 ; 3 1 ;...
26      0 2 ; 1 2 ; 2 2 ; 3 2 ; 4 2 ;
27      1 3 ; 2 3 ; 3 3 ; 4 3 ;
28      2 4 ; 3 4 ; 4 4];
29 P = P./mu; P(:,1) = P(:,1) + shift_x; P(:,2) = P(:,2) + shift_y;
30 % Zapisimo vse trikotnike, na nacin, da je prva koordinata vedno pri pravem
31 % kotu, druga desno in tretja levo.
32 T = [4 1 5 ;... %1
33      2 5 1 ; ... %2
34      5 2 6 ; ... %3
35      3 6 2 ; ... %4
36      6 3 7 ; ... %5
37      8 4 9 ; ... %6
38      5 9 4 ; ... %7
39      9 5 10; ... %8
40      6 10 5; ... %9
41      10 6 11; ... %10
42      7 11 6; ... %11
43      11 7 12; ... %12
44      9 13 8; ... %13
45      13 9 14; ... %14
46      10 14 9; ... %15
47      14 10 15; ...%16
48      11 15 10; ...%17
49      15 11 16; ...%18
50      12 16 11; ...%19
51      14 17 13; ...%20
52      17 14 18; ...%21
53      15 18 14; ...%22
54      18 15 19; ...%23
55      16 19 15]; %24
56 TR = triangulation(T,P);
57 %Na silo shranimo BB-koeficiente v matriko.
58 koef = zeros(17);
59 koef(4, 5:8) = 1/24;
60 koef(5, 4:10) = [1 2 3 4 3 2 1]/24;
61 koef(6, 4:11) = [1 3 4 6 6 4 3 1]/24;
62 koef(7, 4:12) = [1 4 6 8 10 8 6 4 1]/24;
63 koef(8, 4:13) = [1 3 6 10 12 12 10 6 3 1]/24;
64 koef(9, 5:13) = [2 4 8 12 12 12 8 4 2]/24;
```

```

65 koef(10, 5:14) = [1 3 6 10 12 12 10 6 3 1]/24;
66 koef(11, 6:14) = [1 4 6 8 10 8 6 4 1]/24;
67 koef(12, 7:14) = [1 3 4 6 6 4 3 1]/24;
68 koef(13, 8:14) = [1 2 3 4 3 2 1]/24;
69 koef(14, 10:13) = 1/24;
70 %Povezava oglisc trikotnika z BB koeficienti
71 mapp = [1 1 ; ...
72         1 5 ; ...
73         1 9 ; ...
74         5 1 ; ...
75         5 5 ; ...
76         5 9 ; ...
77         5 13 ; ...
78         9 1 ; ...
79         9 5 ; ...
80         9 9 ; ...
81         9 13 ; ...
82         9 17 ; ...
83         13 5 ; ...
84         13 9 ; ...
85         13 13 ; ...
86         13 17 ; ...
87         17 9 ; ...
88         17 13 ; ...
89         17 17];
90 bb_form = cell(24);
91 for i=1:length(TR.ConnectivityList)
92     % Vzamemo ustrezen trikotnik
93     Ti = TR.ConnectivityList(i,:);
94     %Pridobimo indekse oglisc
95     Ai = Ti(2); Bi = Ti(1); Ci = Ti(3);
96     % A je vedno pri pravem kotu, B je desno in C levo.
97     %Kontrolna tocka pri pravem kotu
98     C400_ind = mapp(Ai, :);
99     C400 = [P(Ai,:) koef(C400_ind(1), C400_ind(2))];
100    C400_koor = P(Ai,:);
101    %Kontrolna tocka desno od pravega kota
102    C004_ind = mapp(Bi, :);
103    C004 = [P(Bi,:) koef(C004_ind(1), C004_ind(2))];
104    C004_koor = P(Bi,:);
105    %Kontrolna tocka levo od pravega kota
106    C040_ind = mapp(Ci, :);
107    C040 = [P(Ci,:) koef(C040_ind(1), C040_ind(2))];
108    C040_koor = P(Ci,:);
109    % Na vsaki daljici trikotnika moramo dolociti se tri vmesne kontrolne
110    % tocke kot prikazuje naslednja skica:
111    %
112    %
113    %
114    %
115    %
116    %
117    %
118    %
119    %
120    %
121    %
122    %
123    %
124    %
125    %
126    %
127    %
128    %
129    %
130    %
131    %
132    % Kontrolna tocka med C400 in C004 --> C202 (Pravi kot in desno)
133    C202 = [(C400_koor + C004_koor)./2, koef((C400_ind(1)+C004_ind(1))/2, (C400_ind(2) + C004_ind(2)
134    )/2)];
134    C202_koor = C202(1:2); C202_ind = [(C400_ind(1)+C004_ind(1))/2, (C400_ind(2) + C004_ind(2))/2];
135    % Kontrolna tocka med C040 in C400 --> C220 (Pravi kot in levo)
136    C220 = [(C400_koor + C040_koor)./2, koef((C400_ind(1)+C040_ind(1))/2, (C400_ind(2) + C040_ind(2)
137    )/2)];
137    C220_koor = C220(1:2); C220_ind = [(C400_ind(1)+C040_ind(1))/2, (C400_ind(2) + C040_ind(2))/2];
138    % Kontrolna tocka med C040 in C004 --> C022 (hipotenuza)
139    C022 = [(C040_koor + C004_koor)./2, koef((C004_ind(1)+C040_ind(1))/2, (C004_ind(2) + C040_ind(2)
140    )/2)];
140    C022_koor = C022(1:2); C022_ind = [(C004_ind(1)+C040_ind(1))/2, (C004_ind(2) + C040_ind(2))/2];

```

```

141 % KONTROLNE TOCKE NA CETRINAH DALJIC
142 % Kontrolna tocka med C400 in C202
143 C301 = [(C400.koor+ C202.koor)./2, koef((C400.ind(1) + C202.ind(1))/2, (C400.ind(2) + C202.ind
144 (2))/2)];
145 %C301.koor = (C400.koor+ C202.koor)./2; C301.ind = [(C400.ind(1) + C202.ind(1))/2, (C400.ind(2)
146 + C202.ind(2))/2];
147 %Kontrolna tocka med C202 in C004
148 C103 = [(C202.koor + C004.koor)./2, koef((C004.ind(1) + C202.ind(1))/2, (C004.ind(2) + C202.ind
149 (2))/2)];
150 %Kontrolna tocka med C040 in C022
151 C031 = [(C022.koor + C040.koor)./2, koef((C040.ind(1) + C022.ind(1))/2, (C040.ind(2) + C022.ind
152 (2))/2)];
153 %Kontrolna tocka med C004 in C022
154 C013 = [(C022.koor + C004.koor)./2, koef((C004.ind(1) + C022.ind(1))/2, (C004.ind(2) + C022.ind
155 (2))/2)];
156 %Kontrolna tocka med C040 in C220
157 C130 = [(C220.koor + C040.koor)./2, koef((C040.ind(1) + C220.ind(1))/2, (C040.ind(2) + C220.ind
158 (2))/2)];
159 %Kontrolna tocka med C220 in C400
160 C310 = [(C220.koor + C400.koor)./2, koef((C400.ind(1) + C220.ind(1))/2, (C400.ind(2) + C220.ind
161 (2))/2)];
162 C310.koor = (C220.koor + C400.koor)./2; C310.ind = [(C400.ind(1) + C220.ind(1))/2, (C400.ind(2)
163 + C220.ind(2))/2];
164 %KONTROLNE TOCKE ZNOTRAJ TRIKOTNIKA
165 %Kontrolna tocka med C202 in C022
166 C112 = [(C202.koor + C022.koor)./2, koef((C202.ind(1) + C022.ind(1))./2, (C202.ind(2) + C022.ind
167 (2))./2)];
168 C112.koor = (C202.koor + C022.koor)./2; C112.ind = [(C202.ind(1) + C022.ind(1))./2, (C202.ind(2)
169 + C022.ind(2))./2];
170 %Kontrolni tocki med C301 in C031
171 C211 = [(C310.koor + C112.koor)./2, koef((C310.ind(1) + C112.ind(1))./2, (C310.ind(2) + C112.ind
172 (2))./2)];
173 C121 = [(C220.koor+C022.koor)./2, koef((C220.ind(1)+C022.ind(1))./2, (C220.ind(2)+C022.ind(2))
174 ./2)];
175 %Izracunamo kontrolne tocke odvoda po smeri e1 = [1 0]^T
176 sgn = sign(C040(1) - C400(1)); %Tako definiramo x-smer
177 C003 = C004 - C013; C003(3) = -sgn*C003(3);
178 C102 = C103 - C112; C102(3) = -sgn*C102(3);
179 C201 = C202 - C211; C201(3) = -sgn*C201(3);
180 C300 = C301 - C310; C300(3) = -sgn*C300(3);
181 C210 = C211 - C220; C210(3) = -sgn*C210(3);
182 C120 = C121 - C130; C120(3) = -sgn*C120(3);
183 C030 = C031 - C040; C030(3) = -sgn*C030(3);
184 C021 = C022 - C031; C021(3) = -sgn*C021(3);
185 C012 = C013 - C022; C012(3) = -sgn*C012(3);
186 C111 = C112 - C121; C111(3) = -sgn*C111(3);
187 %Sestavimo zlepek
188 Bi = @(u,v) 4*(C003.*B_3(u,v,0,0,3) + C102.*B_3(u,v,1,0,2)+ C201.*B_3(u,v,2,0,1)+...
189 C300.*B_3(u,v,3,0,0) + C210.*B_3(u,v,2,1,0) + C120.*B_3(u,v,1,2,0) + ...
190 C030.*B_3(u,v,0,3,0) + C021.*B_3(u,v,0,2,1) + C012.*B_3(u,v,0,1,2)+...
191 C111.*B_3(u,v,1,1,1));
192 bb_form{i}=Bi;
193 end
194 end

```


PRILOGA K Implementacija odvoda zlepka B_{222} po smeri e_2

```
1 function [bb_form,TR] = De2.B222_fun(shift , scale)
2 %Del_B222_fun Skonstruira BB-obliko skatlastega zlepka Del_B211
3 % Funkcija kot argument lahko sprejme dva neobvezna parametra. Brez njiju
4 % je zlepek postavljen v izhodišče in skaliran s faktorjem ena. Argument
5 % shift naj bo dvoelementni vektor s katerim opisemo premik v x in y
6 % smeri. Argument scale pa je realno stevilo s katerim ustrezno skaliramo
7 % oglisca triangulacije, da zlepek povečamo ali zmanjšamo. Zlepek skaliramo s
8 % faktorjem 1/scale! Rezultat
9 % funkcije je cell array bb_form, ki vsebuje predpise Bezierevih krp
10 % predstavljene z anonimnimi funkcijami nad vsakim delom triangulacije.
11 % Poleg BB oblike vrnemo tudi Triangulacijo kot Matlab strukturo.
12
13 switch nargin
14     case 2
15         mu = scale; shift_x = shift(1); shift_y = shift(2);
16     case 1
17         shift_x = shift(1); shift_y = shift(2); mu = 1;
18     case 0
19         shift_x = 0; shift_y = 0; mu = 1;
20     otherwise
21         error('Prevec vhodnih podatkov')
22 end
23 B_3 = @(u,v,i,j,k) (6 / (factorial(i)*factorial(j)*factorial(k))) .* u.^i .* v.^j .* (-u-v+1).^k;
24 P = [0 0 ; 1 0; 2 0 ; ...
25      0 1 ; 1 1 ; 2 1 ; 3 1 ;...
26      0 2 ; 1 2 ; 2 2 ; 3 2 ; 4 2 ;
27      1 3 ; 2 3 ; 3 3 ; 4 3 ;
28      2 4 ; 3 4 ; 4 4];
29 P = P./mu; P(:,1) = P(:,1) + shift_x; P(:,2) = P(:,2) + shift_y;
30 % Zapisimo vse trikotnike, na način, da je prva koordinata vedno pri pravem
31 % kotu, druga desno in tretja levo.
32 T = [4 1 5 ;... %1
33      2 5 1 ; ... %2
34      5 2 6 ; ... %3
35      3 6 2 ; ... %4
36      6 3 7 ; ... %5
37      8 4 9 ; ... %6
38      5 9 4 ; ... %7
39      9 5 10; ... %8
40      6 10 5; ... %9
41      10 6 11; ... %10
42      7 11 6; ... %11
43      11 7 12; ... %12
44      9 13 8; ... %13
45      13 9 14; ... %14
46      10 14 9; ... %15
47      14 10 15; ...%16
48      11 15 10; ...%17
49      15 11 16; ...%18
50      12 16 11; ...%19
51      14 17 13; ...%20
52      17 14 18; ...%21
53      15 18 14; ...%22
54      18 15 19; ...%23
55      16 19 15]; %24
56 TR = triangulation(T,P);
57 %Na silo shranimo BB-koeficiente v matriko.
58 koef = zeros(17);
59 koef(4, 5:8) = 1/24;
60 koef(5, 4:10) = [1 2 3 4 3 2 1]/24;
61 koef(6, 4:11) = [1 3 4 6 6 4 3 1]/24;
62 koef(7, 4:12) = [1 4 6 8 10 8 6 4 1]/24;
63 koef(8, 4:13) = [1 3 6 10 12 12 10 6 3 1]/24;
64 koef(9, 5:13) = [2 4 8 12 12 12 8 4 2]/24;
```

```

65 koef(10, 5:14) = [1 3 6 10 12 12 10 6 3 1]/24;
66 koef(11, 6:14) = [1 4 6 8 10 8 6 4 1]/24;
67 koef(12, 7:14) = [1 3 4 6 6 4 3 1]/24;
68 koef(13, 8:14) = [1 2 3 4 3 2 1]/24;
69 koef(14, 10:13) = 1/24;
70 %Povezava oglisc trikotnika z BB koeficienti
71 mapp = [1 1 ; ...
72         1 5 ; ...
73         1 9 ; ...
74         5 1 ; ...
75         5 5 ; ...
76         5 9 ; ...
77         5 13 ; ...
78         9 1 ; ...
79         9 5 ; ...
80         9 9 ; ...
81         9 13 ; ...
82         9 17 ; ...
83         13 5 ; ...
84         13 9 ; ...
85         13 13 ; ...
86         13 17 ; ...
87         17 9 ; ...
88         17 13 ; ...
89         17 17];
90 bb_form = cell(24);
91 for i=1:length(TR.ConnectivityList)
92     % Vzamemo ustrezen trikotnik
93     Ti = TR.ConnectivityList(i,:);
94     %Pridobimo indekse oglisc
95     Ai = Ti(2); Bi = Ti(1); Ci = Ti(3);
96     % A je vedno pri pravem kotu, B je desno in C levo.
97     %Kontrolna tocka pri pravem kotu
98     C400_ind = mapp(Ai, :);
99     C400 = [P(Ai,:) koef(C400_ind(1), C400_ind(2))];
100    C400_koor = P(Ai,:);
101    %Kontrolna tocka desno od pravega kota
102    C004_ind = mapp(Bi, :);
103    C004 = [P(Bi,:) koef(C004_ind(1), C004_ind(2))];
104    C004_koor = P(Bi,:);
105    %Kontrolna tocka levo od pravega kota
106    C040_ind = mapp(Ci, :);
107    C040 = [P(Ci,:) koef(C040_ind(1), C040_ind(2))];
108    C040_koor = P(Ci,:);
109    % Na vsaki daljici trikotnika moramo dolociti se tri vmesne kontrolne
110    % tocke kot prikazuje naslednja skica:
111    %
112    %
113    %
114    %
115    %
116    %
117    %
118    %
119    %
120    %
121    %
122    %
123    %
124    %
125    %
126    %
127    %
128    %
129    %
130    %
131    %
132    % Kontrolna tocka med C400 in C004 --> C202 (Pravi kot in desno)
133    C202 = [(C400_koor + C004_koor)/2, koef((C400_ind(1)+C004_ind(1))/2, (C400_ind(2) + C004_ind(2)
134    )/2)];
134    C202_koor = C202(1:2); C202_ind = [(C400_ind(1)+C004_ind(1))/2, (C400_ind(2) + C004_ind(2))/2];
135    % Kontrolna tocka med C040 in C400 --> C220 (Pravi kot in levo)
136    C220 = [(C400_koor + C040_koor)/2, koef((C400_ind(1)+C040_ind(1))/2, (C400_ind(2) + C040_ind(2)
137    )/2)];
137    C220_koor = C220(1:2); C220_ind = [(C400_ind(1)+C040_ind(1))/2, (C400_ind(2) + C040_ind(2))/2];
138    % Kontrolna tocka med C040 in C004 --> C022 (hipotenuza)
139    C022 = [(C040_koor + C004_koor)/2, koef((C004_ind(1)+C040_ind(1))/2, (C004_ind(2) + C040_ind(2)
140    )/2)];
140    C022_koor = C022(1:2); C022_ind = [(C004_ind(1)+C040_ind(1))/2, (C004_ind(2) + C040_ind(2))/2];

```

```

141 % KONTROLNE TOCKE NA CETRTINAH DALJIC
142 % Kontrolna tocka med C400 in C202
143 C301 = [(C400.koor+ C202.koor)./2, koef((C400.ind(1) + C202.ind(1))/2, (C400.ind(2) + C202.ind
144 (2))/2)];
145 %C301.koor = (C400.koor+ C202.koor)./2; C301.ind = [(C400.ind(1) + C202.ind(1))/2, (C400.ind(2)
146 + C202.ind(2))/2];
147 %Kontrolna tocka med C202 in C004
148 C103 = [(C202.koor + C004.koor)./2, koef((C004.ind(1) + C202.ind(1))/2, (C004.ind(2) + C202.ind
149 (2))/2)];
150 %Kontrolna tocka med C040 in C022
151 C031 = [(C022.koor + C040.koor)./2, koef((C040.ind(1) + C022.ind(1))/2, (C040.ind(2) + C022.ind
152 (2))/2)];
153 %Kontrolna tocka med C004 in C022
154 C013 = [(C022.koor + C004.koor)./2, koef((C004.ind(1) + C022.ind(1))/2, (C004.ind(2) + C022.ind
155 (2))/2)];
156 %Kontrolna tocka med C040 in C220
157 C130 = [(C220.koor + C040.koor)./2, koef((C040.ind(1) + C220.ind(1))/2, (C040.ind(2) + C220.ind
158 (2))/2)];
159 %Kontrolna tocka med C220 in C400
160 C310 = [(C220.koor + C400.koor)./2, koef((C400.ind(1) + C220.ind(1))/2, (C400.ind(2) + C220.ind
161 (2))/2)];
162 C310.koor = (C220.koor + C400.koor)./2; C310.ind = [(C400.ind(1) + C220.ind(1))/2, (C400.ind(2)
163 + C220.ind(2))/2];
164 %KONTROLNE TOCKE ZNOTRAJ TRIKOTNIKA
165 %Kontrolna tocka med C202 in C022
166 C112 = [(C202.koor + C022.koor)./2, koef((C202.ind(1) + C022.ind(1))./2, (C202.ind(2) + C022.ind
167 (2))./2)];
168 C112.koor = (C202.koor + C022.koor)./2; C112.ind = [(C202.ind(1) + C022.ind(1))./2, (C202.ind(2)
169 + C022.ind(2))./2];
170 %Kontrolni tocki med C301 in C031
171 C211 = [(C310.koor + C112.koor)./2, koef((C310.ind(1) + C112.ind(1))./2, (C310.ind(2) + C112.ind
172 (2))./2)];
173 C121 = [(C220.koor+C022.koor)./2, koef((C220.ind(1)+C022.ind(1))./2, (C220.ind(2)+C022.ind(2))
174 ./2)];
175 %Izracunamo kontrolne tocke odvoda po smeri e2 = [0 1]^T
176 %Dolocimo "smer" trikotnika
177 sgn = sign(C004(2) - C400(2)); %Tako definiramo y-smer
178 C003 = C004 - C103; C003(3) = sgn*C003(3);
179 C102 = C103 - C202; C102(3) = sgn*C102(3);
180 C201 = C202 - C301; C201(3) = sgn*C201(3);
181 C300 = C301 - C400; C300(3) = sgn*C300(3);
182 C210 = C211 - C310; C210(3) = sgn*C210(3);
183 C120 = C121 - C220; C120(3) = sgn*C120(3);
184 C030 = C031 - C130; C030(3) = sgn*C030(3);
185 C021 = C022 - C121; C021(3) = sgn*C021(3);
186 C012 = C013 - C112; C012(3) = sgn*C012(3);
187 C111 = C112 - C211; C111(3) = sgn*C111(3);
188 %Sestavimo zlepek
189 Bi = @(u,v) 4*(C003.*B_3(u,v,0,0,3) + C102.*B_3(u,v,1,0,2)+ C201.*B_3(u,v,2,0,1)+...
190 C300.*B_3(u,v,3,0,0) + C210.*B_3(u,v,2,1,0) + C120.*B_3(u,v,1,2,0) + ...
191 C030.*B_3(u,v,0,3,0) + C021.*B_3(u,v,0,2,1) + C012.*B_3(u,v,0,1,2)+...
192 C111.*B_3(u,v,1,1,1));
193 bb_form{i}=Bi;
194 end
195 end

```

PRILOGA L Aproksimacija po metodi najmanjših kvadratov (B-zlepki)

```
1 x = linspace(0,1,51); y = linspace(0,1,51); [msh_x, msh_y] = meshgrid(x,y);
2 F = @(x,y) ((x-y).^3).*double(x <= y) + (-((x-y).^3)).*double(x > y);
3 % F = @(x,y) 2./(3.*exp(sqrt((10.*(x-0.5)).^2 + (10.*(y-0.5)).^2)));
4
5 % f1 = @(x) -cos(3*pi/2 * x);
6 % f2 = @(x) -x.^3 .* sin(x-1/3 + pi) .* sin(x-2/3 + pi).*double(x > 1/3 & x < 2/3);
7 % f3 = @(x) cos(3*pi/4 * x);
8 % f = @(x) f1(x) + f2(x) + f3(x);
9 % F = @(x,y) f(x);
10 msh_z = F(msh_x, msh_y);
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Zacetni parametri %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12 a = 0; b = 1; c = 0; d = 1;
13 p = 4;
14 k = 0;
15 U = [a*ones(1,p) linspace(a,b,2+k), b*ones(1,p)]; V = [c*ones(1,p) linspace(c,d,2+k), d*ones(1,p)];
16 Omega = {[0 1; 0 1]};
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 tol = 10.^-4;
19 nadaljuj = true;
20 pogojno = false;
21 vektorji_vozlov = {{U,V}};
22 figure;
23 while nadaljuj == true
24     tic;
25     T = THB_baza(U,V,p,p,Omega,0);
26     G = zeros(length(T));
27     rx = reshape(msh_x,[1 numel(msh_x)]); ry = reshape(msh_y,[1 numel(msh_y)]);
28     z = reshape(msh_z,[1 numel(msh_z)]);
29     G_stolpci = cell(1,length(T));
30     for k=1:length(T)
31         stolpec = fval(T{k}, [rx;ry]); G_stolpci{k} = stolpec';
32     end
33     for i=1:length(T)
34         for j=i:length(T)
35             G(i,j) = sum(G_stolpci{i} .* G_stolpci{j});
36             G(j,i) = G(i,j);
37         end
38     end
39     b = zeros(length(T), 1);
40     for i=1:length(T)
41         bsp_i = T{i};
42         z_i = G_stolpci{i}; msh_zi = reshape(z_i, size(msh_z));
43         b(i) = sum(sum(msh_zi.*msh_z));
44     end
45     a = (G)\b;
46     [bsp_nov2, f_nov] = THB_zlepek(T, a);
47     z1 = f_nov(rx,ry); z_rez = reshape(z1, size(msh_z));
48     subplot(1,2,1)
49     surf(msh_x, msh_y, msh_z)
50     subplot(1,2,2)
51     surf(msh_x, msh_y, z_rez)
52     err = abs(msh_z - z_rez);
53     err_l2 = sqrt(1/16 .* (x(2)-x(1)).^2 .* sum(sum((msh_z-z_rez).^2)));
54     err_inf = max(max(err));
55     % Kvazi-Adaptivno lokalno zgoscevanje
56     U_tmp = T{end}.knots{1}; V_tmp = T{end}.knots{2};
57     x_min = min(msh_x(find(err >= tol)));
58     ind_x_min = find(abs(U_tmp-x_min) == min(abs(U_tmp-x_min)));
59     if isempty(x_min)
60         A = 0;
61     elseif U_tmp(ind_x_min(end)) <= x_min
62         A = U_tmp(ind_x_min(end));
63     else
64         A = U_tmp(ind_x_min(end)-1);
```

```

65     end
66     x_max = max(msh_x(find(err >= tol)));
67     ind_x_max = find(abs(U_tmp-x_max) == min(abs(U_tmp-x_max)));
68     if isempty(x_max)
69         B = 0;
70     elseif U_tmp(ind_x_max(end)) >= x_max
71         B = U_tmp(ind_x_max(end));
72     else
73         B = U_tmp(ind_x_max(end)+1);
74     end
75     y_min = min(msh_y(find(err >= tol)));
76     ind_y_min = find(abs(V_tmp-y_min) == min(abs(V_tmp-y_min)));
77     if isempty(y_min)
78         C = 0;
79     elseif V_tmp(ind_y_min(end)) <= y_min
80         C = V_tmp(ind_y_min(end));
81     else
82         C = V_tmp(ind_y_min(end)-1);
83     end
84     y_max = max(msh_y(find(err >= tol)));
85     ind_y_max = find(abs(V_tmp-y_max) == min(abs(V_tmp-y_max)));
86     if isempty(y_max)
87         D = 0;
88     elseif V_tmp(ind_y_max(end)) >= y_max
89         D = V_tmp(ind_y_max(end));
90     else
91         D = V_tmp(ind_y_max(end)+1);
92     end
93     vektorji_vozlov = [vektorji_vozlov, {{U_tmp, V_tmp}}];
94     if pogojno == true
95         nadaljuj = false;
96     end
97     if length([A B C D]) < 4 || ((A-B)*(C-D) == 0) || err_inf <= tol
98         nadaljuj = false;
99     end
100    Omega = [Omega, {[A B; C D]}];
101    toc;
102 end
103 figure;
104 hold on;
105 for i = 1:length(Omega)
106     pravokotnik = Omega{i};
107     if ~isempty(pravokotnik)
108         U_tmp = vektorji_vozlov{i}{1}; V_tmp = vektorji_vozlov{i}{2};
109         x_start = max(find(U_tmp == pravokotnik(1,1))); x_end = min(find(U_tmp == pravokotnik(1,2)))
110         ;
111         y_start = max(find(V_tmp == pravokotnik(2,1))); y_end = min(find(V_tmp == pravokotnik(2,2)))
112         ;
113         U_tmp = U_tmp(x_start:x_end); V_tmp = V_tmp(y_start:y_end);
114         for l = 1:length(U_tmp)
115             x1 = U_tmp(l);
116             plot3([x1 x1], [pravokotnik(2,1), pravokotnik(2,2)], [i-1, i-1], 'k');
117         end
118         for l = 1:length(V_tmp)
119             y1 = V_tmp(l);
120             plot3([pravokotnik(1,1), pravokotnik(1,2)], [y1, y1], [i-1, i-1], 'k');
121         end
122     end
123 end
124 hold off;

```

PRILOGA M Aproksimacija po metodi najmanjših kvadratov (Škatlasti zleпки)

```
1 m = 1; m_lok = m; % Lokalni faktor skaliranja!
2 Omega = {[0 1; 0 1]};
3 % F = @(x,y) -((x-y).^3).*double(x <= y) - (-(x-y).^3).*double(x > y);
4 % F = @(x,y) 2./(3.*exp(sqrt((10.*(x-0.5)).^2 + (10.*(y-0.5)).^2)));
5
6 % f1 = @(x) -cos(3*pi/2 * x); f2 = @(x) cos(3*pi/4 * x);
7 % f3 = @(x) -x.^3 .* sin(x-1/3 + pi) .* sin(x-2/3 + pi).*double(x > 1/3 & x < 2/3);
8 % F = @(x,y) f1(x) + f2(x) + f3(x);
9
10 % f = @(x) -10*(x-1/3).*(x-2/3).*(x-3/6).*double(x >= 1/3 & x <= 2/3);
11 % g = @(x) -10*(x-1/4).*(x-1/2).*(x-3/6).*double(x >= 1/4 & x <= 1/2);
12 % F = @(x,y) f(x).*g(y);
13 X = linspace(0,1,51); Y = linspace(0,1,51);
14 [msh_x, msh_y] = meshgrid(X,Y);
15 rx = reshape(msh_x, [numel(msh_x), 1]); ry = reshape(msh_y, [numel(msh_y), 1]);
16 msh_z = F(msh_x, msh_y); rz = reshape(msh_z, [numel(msh_z), 1]);
17 %APROKSIMACIJA
18 tol = 10.^-4; nadaljuj = true; l=7;
19 while nadaljuj
20     [basis, triang, koordinate] = HBox_Baza_alt(Omega, m);
21     g_stolpci = {};
22     for k = 1:length(basis)
23         g_stolpci{k} = box_val(basis{k}, triang{k}, [rx, ry]);
24     end
25     G = zeros(length(basis));
26     for i = 1:length(basis)
27         for j = i:length(basis)
28             G(i,j) = sum(g_stolpci{i}.*g_stolpci{j}); G(j,i) = G(i,j);
29         end
30     end
31     b = zeros(length(basis),1);
32     rz2 = reshape(msh_z, [numel(msh_z),1]);
33     for i = 1:length(basis)
34         b(i) = sum(g_stolpci{i}.*rz2);
35     end
36     a = G\b;
37     u_approx = Box_LinCmb(basis, triang, a);
38     msh_z_approx = u_approx({msh_x, msh_y});
39     err = abs(msh_z - msh_z_approx); r_err = err(:); err_inf = max(max(err));
40     err_l2 = sqrt((1/16 * 1/(m.^2))*sum(sum(abs(msh_z-msh_z_approx).^2)));
41     if (err_inf <= tol) | (l == 1)
42         nadaljuj = false;
43     else
44         omega = [];
45         for i=1:length(triang)
46             TR = triang{i}; prve = TR.Points(1,:); zadnje = TR.Points(end,:);
47             ID = pointLocation(TR,[rx ry]);
48             err_infi = max(r_err(find(~isnan(ID))));
49             if err_infi > tol
50                 omega = [omega; prve];
51             end
52         end
53         m_lok = m_lok*2; Omega = [Omega, {omega}];
54     end
55     l=l+1;
56 end
57 hold on;
58 for i=1:length(triang)
59     triplot(triang{i}, 'k')
60 end
61 rectangle('Position', [0 0 1 1], 'EdgeColor', [0.8500 0.3250 0.0980], 'LineWidth', 3)
62 hold off;
```

PRILOGA N Reparametrizirana aproksimacija po metodi najmanjših kvadratov (B-zlepki)

```
1 % Nova domena
2 [G_fun, J] = kolobar_alt(1);
3 x_omega = linspace(0,1,51);
4 y_omega = linspace(0,1,51);
5 [msh_x_omega, msh_y_omega] = meshgrid(x_omega, y_omega);
6 rx_omega = reshape(msh_x_omega, [1 numel(msh_x_omega)]);
7 ry_omega = reshape(msh_y_omega, [1 numel(msh_y_omega)]);
8 x = linspace(0,1,51);
9 y = linspace(0,1,51);
10 [msh_x, msh_y] = meshgrid(x,y);
11 F = @(x,y) 1./(x.^2 + y.^2);
12 msh_z = F(msh_x, msh_y);
13 % Zacetni parametri
14 a = 0; b = 1; c = 0; d = 1;
15 p = 4;
16 k = 0;
17 U = [a*ones(1,p) linspace(a,b,2+k), b*ones(1,p)];
18 V = [c*ones(1,p) linspace(c,d,2+k), d*ones(1,p)];
19 % Zacetno obmocje
20 Omega = {[0 1; 0 1]};
21 tol = 10.^-4; nadaljij = true; pogojno = false; vektorji_vozlov = {{U,V}};
22 while nadaljij == true
23     tic;
24     T = THB_baza(U,V,p,p,Omega,0);
25     G = zeros(length(T));
26     rx = reshape(msh_x,[1 numel(msh_x)]); ry = reshape(msh_y,[1 numel(msh_y)]);
27     z = reshape(msh_z,[1 numel(msh_z)]);
28     G_stolpci = cell(1,length(T));
29     for k=1:length(T)
30         stolpec = fnval(T{k}, [rx;ry]);
31         G_stolpci{k} = stolpec';
32     end
33     % Transponiraj, ko delas z alt verzijo kolobarja!!!!
34     J_val = abs(J(rx',ry'));
35     for i=1:length(T)
36         for j=i:length(T)
37             G(i,j) = sum(G_stolpci{i} .* G_stolpci{j}.*J_val');
38             G(j,i) = G(i,j);
39         end
40     end
41     b = zeros(length(T), 1);
42     msh_Jval = reshape(J_val, size(msh_x));
43     xy_reparam = G_fun(rx,ry); %Reparametrizacija
44     rz_rep = F(xy_reparam(1,:), xy_reparam(2,:));
45     msh_z_rep = reshape(rz_rep, size(msh_x));
46     for i=1:length(T)
47         bsp_i = T{i};
48         z_i = G_stolpci{i};
49         msh_zi = reshape(z_i, size(msh_z));
50         b(i) = sum(sum(msh_zi.*msh_z_rep.*abs(msh_Jval)));
51     end
52     a = (G)\b;
53     [bsp_nov2, f_nov] = THB_zlepek(T, a);
54     F_approx = @(x,y) f_nov(x,y);
55     z1 = F_approx(rx,ry);
56     z_rez = reshape(z1, size(msh_z));
57     subplot(1,2,1)
58     msh_x_rep = reshape(xy_reparam(1,:), size(msh_x));
59     msh_y_rep = reshape(xy_reparam(2,:), size(msh_x));
60     surf(msh_x_rep, msh_y_rep, F(msh_x_rep,msh_y_rep))
```

```

61 subplot(1,2,2)
62 surf(msh_x_rep, msh_y_rep, z_rez)
63 err = abs(F(msh_x_rep, msh_y_rep) - z_rez);
64 err_l2 = sqrt(1/16 * (x(2)-x(1)).^2 * sum(sum((F(msh_x_rep, msh_y_rep)-z_rez).^2)));
65 err_inf = max(max(err));
66 % Kvazi-Adaptivno lokalno zgoscevanje
67 U_tmp = T{end}.knots{1}; V_tmp = T{end}.knots{2};
68 x_min = min(msh_x(find(err >= tol)));
69 ind_x_min = find(abs(U_tmp-x_min) == min(abs(U_tmp-x_min)));
70 if isempty(x_min)
71     A = 0;
72 elseif U_tmp(ind_x_min(end)) <= x_min
73     A = U_tmp(ind_x_min(end));
74 else
75     A = U_tmp(ind_x_min(end)-1);
76 end
77 x_max = max(msh_x(find(err >= tol)));
78 ind_x_max = find(abs(U_tmp-x_max) == min(abs(U_tmp-x_max)));
79 if isempty(x_max)
80     B = 0;
81 elseif U_tmp(ind_x_max(end)) >= x_max
82     B = U_tmp(ind_x_max(end));
83 else
84     B = U_tmp(ind_x_max(end)+1);
85 end
86 y_min = min(msh_y(find(err >= tol)));
87 ind_y_min = find(abs(V_tmp-y_min) == min(abs(V_tmp-y_min)));
88 if isempty(y_min)
89     C = 0;
90 elseif V_tmp(ind_y_min(end)) <= y_min
91     C = V_tmp(ind_y_min(end));
92 else
93     C = V_tmp(ind_y_min(end)-1);
94 end
95 y_max = max(msh_y(find(err >= tol)));
96 ind_y_max = find(abs(V_tmp-y_max) == min(abs(V_tmp-y_max)));
97 if isempty(y_max)
98     D = 0;
99 elseif V_tmp(ind_y_max(end)) >= y_max
100     D = V_tmp(ind_y_max(end));
101 else
102     D = V_tmp(ind_y_max(end)+1);
103 end
104 vektorji_vozlov = [vektorji_vozlov, {{U_tmp, V_tmp}}];
105 if pogojno == true
106     nadaljaj = false;
107 end
108 if length([A B C D]) < 4 || ((A-B)*(C-D) == 0) || err_inf <= tol
109     nadaljaj = false;
110 end
111 % Globalno zgoscevanje
112 % if pogojno == true
113 %     nadaljaj = false;
114 % elseif (tol <= err_inf && err_inf <= tol*10 )
115 %     pogojno = true;
116 % else
117 %     A=0;B=1;C=0;D=1;
118 % end
119 [A B;C D]
120 Omega = [Omega, {[A B;C D]}];
121
122 toc;
123 end
124
125 jeZacetno = @(A) sum(sum(A == [0 1;0 1])) == 4;
126 start = max(find(cellfun(jeZacetno, Omega) == 1));
127 figure;
128 hold on;
129 for i=start:length(vektorji_vozlov)
130     pravokotnik = Omega{i};
131     U_tmp = vektorji_vozlov{i}{1};
132     V_tmp = vektorji_vozlov{i}{2};
133     x_start = max(find(U_tmp == pravokotnik(1,1)));
134     x_end = min(find(U_tmp == pravokotnik(1,2)));
135     y_start = max(find(V_tmp == pravokotnik(2,1)));
136     y_end = min(find(V_tmp == pravokotnik(2,2)));
137     U_tmp = U_tmp(x_start:x_end);
138     V_tmp = V_tmp(y_start:y_end);
139     [mu, mv] = meshgrid(U_tmp, V_tmp);

```



```
140     rep_mesh = G_fun(mu(:) ', mv(:) ');
141     rep_mu = reshape(rep_mesh(1,:), size(mu));
142     rep_mv = reshape(rep_mesh(2,:), size(mv));
143     mesh(rep_mu, rep_mv, zeros(size(mu)))
144 end
145 hold off
```

PRILOGA O Reparametrizirana aproksimacija po metodi najmanjših kvadratov (Škatlasti zleпки)

```
1 [G_fun, J] = kolobar(1);
2 x_omega = linspace(0,1,51); y_omega = linspace(0,1,51);
3 [msh_x_omega, msh_y_omega] = meshgrid(x_omega, y_omega);
4 rx_omega = reshape(msh_x_omega, [1 numel(msh_x_omega)]);
5 ry_omega = reshape(msh_y_omega, [1 numel(msh_y_omega)]);
6 m = 1 ;
7 m_lok = m; % Lokalni faktor skaliranja!
8 a=0;b=1;c=0;d=1; Omega = {[a b;c d]};
9 F = @(x,y) 1./(x.^2 + y.^2);
10 X = linspace(a,b,51); Y = linspace(c,d,51); [msh_x, msh_y] = meshgrid(X,Y);
11 rx = reshape(msh_x, [numel(msh_x), 1]); ry = reshape(msh_y, [numel(msh_y), 1]);
12 msh_z = F(msh_x, msh_y); rz = reshape(msh_z, [numel(msh_z), 1]);
13 %APROKSIMACIJA
14 tol = 10.^-4; nadaljuj = true; l=1;
15 while nadaljuj
16     [basis, triang] = HBox_Baza_alt(Omega, m);
17     g_stolpci = {};
18     for k = 1:length(basis)
19         g_stolpci{k} = box_val(basis{k}, triang{k}, [rx, ry]);
20     end
21     G = zeros(length(basis));
22     J_val = abs(J(rx', ry'));
23     for i = 1:length(basis)
24         for j = i:length(basis)
25             G(i, j) = sum(g_stolpci{i}.*g_stolpci{j}.*J_val'); G(j, i) = G(i, j);
26         end
27     end
28     b = zeros(length(basis),1);
29     msh_Jval = reshape(J_val, size(msh_x));
30     xy_reparam = G_fun(rx', ry'); %Reparametrizacija
31     rz_rep = F(xy_reparam(1,:), xy_reparam(2,:));
32     msh_z_rep = reshape(rz_rep, size(msh_x));
33     for i = 1:length(basis)
34         msh_zi = reshape(g_stolpci{i}, size(msh_x));
35         b(i) = sum(sum(msh_zi.*msh_z_rep.*abs(msh_Jval)));
36     end
37     a = G\b; u_approx = Box_LinCmb(basis, triang, a);
38     msh_z_approx = u_approx({msh_x, msh_y});
39     msh_x_rep = reshape(xy_reparam(1,:), size(msh_x)); msh_y_rep = reshape(xy_reparam(2,:), size(msh_x));
40     err = abs(F(msh_x_rep, msh_y_rep) - msh_z_approx); r_err = err(:);
41     err_inf = max(max(err));
42     if (err_inf <= tol) | (l == 7)
43         nadaljuj = false;
44     else
45         omega = [];
46         for i=1:length(triang)
47             TR = triang{i}; prve = TR.Points(1,:); zadnje = TR.Points(end,:);
48             ID = pointLocation(TR, [rx ry]);
49             err_infi = max(r_err(find(~isnan(ID))));
50             if err_infi > tol
51                 omega = [omega; prve];
52             end
53         end
54         m_lok = m_lok*2;
55         Omega = [Omega, {omega}];
56     end
57     length(basis)
58     l=l+1;
59 end
```

```
60 hold on;
61 for i=1:length(triang)
62     TR = triang{i};
63     rep_Points = G_fun(TR.Points(:,1)', TR.Points(:,2)');
64     rep_Points = rep_Points(1:2,:);
65     rep_TR = triangulation(TR.ConnectivityList, rep_Points);
66     triplot(rep_TR, 'k')
67 end
68 sample = linspace(0,pi/2);
69 krog = [cos(sample);sin(sample)];
70 pgon = polyshape([0 krog(1,:)],[1 krog(2,:)]);
71 plot(pgon, 'FaceColor', 'w')
72 hold off;
```

PRILOGA P Reševanje homogene Poissonove enačbe (B-zlepki)

```
1 x = linspace(0,1,70);
2 y = linspace(0,1,70);
3 [msh_x, msh_y] = meshgrid(x,y);
4 F = @(x,y) x.*y.*(x-1).*(y-1) .* exp(1000*((x-0.5).^2 + (y-0.5).^2)).^(-1);
5 msh_z = F(msh_x, msh_y);
6 % Zacetni parametri
7 p = 2; q = 2; k = 0;
8 U = [0*ones(1,p) linspace(0,1,2+k), 1*ones(1,p)];
9 V = [0*ones(1,p) linspace(0,1,2+k), 1*ones(1,p)];
10 Omega = {[0 1; 0 1]};
11 tol = 10.^-3; nadaljuj = true; pogojno = false; vektorji_vozlov = {{U,V}};
12 while nadaljuj == true
13     tic
14     T = THB_baza(U,V,p,q,Omega,1);
15     G = zeros(length(T));
16     rx = reshape(msh_x,[1 numel(msh_x)]);
17     ry = reshape(msh_y,[1 numel(msh_y)]);
18     rz = reshape(msh_z,[1 numel(msh_z)]);
19     gradienti = {};
20     for i=1:length(T)
21         grd_x = fnder(T{i}, [1 0]);
22         grd_y = fnder(T{i}, [0 1]);
23         grd_x_val = fnval(grd_x, [rx;ry]);
24         grd_y_val = fnval(grd_y, [rx;ry]);
25         gradienti{i} = {grd_x_val,grd_y_val};
26     end
27     for i=1:length(T)
28         for j=i:length(T)
29             produkt = gradienti{i}{1}.*gradienti{j}{1} + gradienti{i}{2}.*gradienti{j}{2};
30             G(i,j) = sum(produkt);
31             G(j,i) = G(i,j);
32         end
33     end
34     G_stolpci = {};
35     for k=1:length(T)
36         stolpec = fnval(T{k}, [rx;ry]);
37         G_stolpci{k} = stolpec';
38     end
39     b = zeros(length(T), 1);
40     %%%%%%%%% Diskretni laplace %%%%%%%%%
41     h = x(2)-x(1);
42     rf = -4*reshape(del2(F(msh_x,msh_y),x,y), size(rx));
43     %%%%%%%%%
44     for i=1:length(T)
45         stolpec = G_stolpci{i};
46         b(i) = sum(stolpec.*rf');
47     end
48     a = G\b;
49     [bsp_nov2, f_nov] = THB_zlepki(T, a);
50     z1 = f_nov(rx,ry);
51     z_rez = reshape(z1, size(msh_z));
52     subplot(1,2,1)
53     mesh(msh_x, msh_y, msh_z)
54     subplot(1,2,2)
55     mesh(msh_x, msh_y, z_rez)
56     err = abs(msh_z-z_rez);
57     err_l2 = sqrt(1/16 .* (x(2)-x(1)).^2 .* sum(sum((msh_z-z_rez).^2)));
58     err_inf = max(max(err));
59     % Kvazi-Adaptivno lokalno zgoscevanje
60     U_tmp = T{end}.knots{1}; V_tmp = T{end}.knots{2};
61     x_min = min(msh_x(find(err >= tol)));
62     ind_x_min = find(abs(U_tmp-x_min) == min(abs(U_tmp-x_min)));
63     if isempty(x_min)
64         A = 0;
```

```

65     elseif U_tmp(ind_x_min(end)) <= x_min
66         A = U_tmp(ind_x_min(end));
67     else
68         A = U_tmp(ind_x_min(end)-1);
69     end
70     x_max = max(msh_x(find(err >= tol)));
71     ind_x_max = find(abs(U_tmp-x_max) == min(abs(U_tmp-x_max)));
72     if isempty(x_max)
73         B = 0;
74     elseif U_tmp(ind_x_max(end)) >= x_max
75         B = U_tmp(ind_x_max(end));
76     else
77         B = U_tmp(ind_x_max(end)+1);
78     end
79     y_min = min(msh_y(find(err >= tol)));
80     ind_y_min = find(abs(V_tmp-y_min) == min(abs(V_tmp-y_min)));
81     if isempty(y_min)
82         C = 0;
83     elseif V_tmp(ind_y_min(end)) <= y_min
84         C = V_tmp(ind_y_min(end));
85     else
86         C = V_tmp(ind_y_min(end)-1);
87     end
88     y_max = max(msh_y(find(err >= tol)));
89     ind_y_max = find(abs(V_tmp-y_max) == min(abs(V_tmp-y_max)));
90     if isempty(y_max)
91         D = 0;
92     elseif V_tmp(ind_y_max(end)) >= y_max
93         D = V_tmp(ind_y_max(end));
94     else
95         D = V_tmp(ind_y_max(end)+1);
96     end
97     vektorji_vozlov = [vektorji_vozlov, {{U_tmp, V_tmp}}];
98     if pogojno == true
99         nadaljuj = false;
100    end
101    if length([A B C D]) < 4 || ((A-B)*(C-D) == 0) || err_inf <= tol
102        nadaljuj = false;
103    end
104    Omega = [Omega, {[A B; C D]}];
105    toc;
106 end
107 figure;
108 hold on;
109 for i = 1:length(Omega)
110     pravokotnik = Omega{i};
111     if ~isempty(pravokotnik)
112         U_tmp = vektorji_vozlov{i}{1};
113         V_tmp = vektorji_vozlov{i}{2};
114         x_start = max(find(U_tmp == pravokotnik(1,1)));
115         x_end = min(find(U_tmp == pravokotnik(1,2)));
116         y_start = max(find(V_tmp == pravokotnik(2,1)));
117         y_end = min(find(V_tmp == pravokotnik(2,2)));
118         U_tmp = U_tmp(x_start:x_end);
119         V_tmp = V_tmp(y_start:y_end);
120         for l = 1:length(U_tmp)
121             x1 = U_tmp(l);
122             plot3([x1 x1], [pravokotnik(2,1), pravokotnik(2,2)], [i-1, i-1], 'k');
123         end
124         for l = 1:length(V_tmp)
125             y1 = V_tmp(l);
126             plot3([pravokotnik(1,1), pravokotnik(1,2)], [y1, y1], [i-1, i-1], 'k');
127         end
128     end
129 end
130 hold off;

```

PRILOGA Q Reševanje homogene Poissonove enačbe (Škatlasti zleпки)

```
1 u_tocna = @(x,y) x.*y.*(x-1).*(y-1) .* exp(1000*((x-0.5).^2 + (y-0.5).^2)).^(-1);
2 X = linspace(0,1,100);
3 Y = linspace(0,1,100);
4 [msh_x, msh_y] = meshgrid(X,Y);
5 rx = reshape(msh_x, [numel(msh_x), 1]);
6 ry = reshape(msh_y, [numel(msh_y), 1]);
7 m = 16 ;
8 a=0;b=1;
9 c=0;d=1;
10 Omega = {[a b;c d]};
11 nadaljuij = true;
12 tol = 10.^-3;
13 l = 1;
14 while nadaljuij == true
15     tic;
16     [basis_pde, triang_pde, koefs_pde, koordinate] = ...
17         HBox_Baza_PDE_alt(Omega, m);
18     k=1;
19     gradienti_val = {};
20     for i = 1:length(triang_pde)
21         kl = 1;
22         gradienti_x={};gradienti_y={};
23         triang_x={};triang_y={};
24         for j=1:length(triang_pde{i})
25             tocke = triang_pde{i}{j}.Points;
26             prve = tocke(1,:);
27             druge = tocke(2,:);
28             % Ker delamo s hierarhicno bazo so nekatere funkcije finejse in ker
29             % a priori ne poznamo vrstnega reda v bazi ponovno izluscimo faktor
30             % skaliranja!
31             m2 = inv(abs(prve(1)-druge(1)));
32             [grad_x, tr_x] = De1_B222_fun(prve,m2);
33             [grad_y, tr_y] = De2_B222_fun(prve,m2);
34             gradienti_x{kl} = grad_x;gradienti_y{kl} = grad_y;
35             triang_x{kl} = tr_x;triang_y{kl} = tr_y;
36             kl = kl+1;
37         end
38         grad_fun = {Box_LinCmb(gradienti_x, triang_x, koefs_pde{i});...
39             Box_LinCmb(gradienti_y, triang_y, koefs_pde{i})};
40         grd_x = grad_fun{1};
41         grd_y = grad_fun{2};
42         gradienti_val{k} = {m2.*grd_x([rx ry]), m2.*grd_y([rx ry])};
43         k=k+1;
44     end
45     end
46     G = zeros(length(basis_pde));
47     for i=1:length(basis_pde)
48         for j=1:length(basis_pde)
49             G(i,j) = sum(gradienti_val{i}{1}.*gradienti_val{j}{1} + gradienti_val{i}{2}.*
50                 gradienti_val{j}{2});
51         end
52     end
53     g_stolpci = {};
54     for k = 1:length(basis_pde)
55         g_stolpci{k} = basis_pde{k}([rx ry]);
56     end
57     b = zeros(length(basis_pde),1);
58     %%%%%%%%% Diskretni laplace %%%%%%%%%%%%%%%
59     h = X(2)-X(1);
60     rf = -4*reshape(del2(u_tocna(msh_x,msh_y),X,Y), size(rx));
61     %%%%%%%%%%%%%%%
62     for i = 1:length(basis_pde)
63         b(i) = sum(g_stolpci{i}.*rf);
64     end
65 end
```

```

64 a = G\b;
65 u_approx_z = zeros(size(msh_x));
66 for k=1:length(basis_pde)
67     u_approx_z = u_approx_z + a(k).*basis_pde{k}({msh_x, msh_y});
68 end
69 u_h = u_approx_z;
70 surf(msh_x,msh_y,u_h)
71 err = abs(u_tocna(msh_x,msh_y)-u_h);
72 r_err = err(:);
73 err_inf = max(max(err));
74 if (err_inf <= tol) | (l == 4)
75     nadaljuj = false;
76 else
77     omega = [];
78     for i=1:length(triang_pde)
79         TR_cell = triang_pde{i};
80         if length(TR_cell) == 1
81             TR = TR_cell{1};
82             prve = TR.Points(1,:);
83             zadnje = TR.Points(end,:);
84             ID = pointLocation(TR,[rx ry]);
85             err_infi = max(r_err(find(~isnan(ID))));
86             if err_infi > tol
87                 omega = [omega; prve];
88             end
89         end
90     end
91 end
92 Omega = [Omega, {omega}];
93 l=l+1;
94 toc;
95 end
96 figure;
97 hold on;
98 for i=1:length(triang_pde)-1
99     for j=1:length(triang_pde{i})
100         TR = triang_pde{i}{j};
101         prve = TR.Points(1,:);
102         druge = TR.Points(2,:);
103         if abs(druge(1)-prve(1)) ~= 1/128
104             triplot(TR, 'k')
105         end
106     end
107 end
108 rectangle('Position', [0 0 1 1], 'EdgeColor', [0.8500 0.3250 0.0980],...
109 'LineWidth', 3)
110 xlim([0 1])
111 ylim([0 1])
112 hold off;

```

PRILOGA R Reševanje nehomogene Poissonove enačbe (B-zlepki)

```
1 x = linspace(0,1,70); y = linspace(0,1,70); [msh_x, msh_y] = meshgrid(x,y);
2 % Zacetni parametri
3 p = 2; q = p; k = 63; Omega = {[0 1; 0 1]};
4 U = [0*ones(1,p) linspace(0,1,2+k), 1*ones(1,p)]; V = [0*ones(1,p) linspace(0,1,2+k), 1*ones(1,p)];
5 tol = 10.^-3; nadaljij = true; pogojno = false; vektorji_vozlov = {{U,V}}; err_inf_rob = inf;
6 while nadaljij == true
7     T = THB_baza(U,V,p,q,Omega,1);
8     if err_inf_rob > tol
9         T_Rob = THB_baza(U,V,p,q,Omega,0);
10        %Najprej z L2 aproksimacijo aproksimiramo rob.
11        f_Omega = @(x,y) 0.01.*sin(x) + 1; msh_z = f_Omega(msh_x, msh_y);
12        G = zeros(length(T_Rob));
13        rx = reshape(msh_x,[1 numel(msh_x)]); ry = reshape(msh_y,[1 numel(msh_y)]);
14        z = reshape(msh_z,[1 numel(msh_z)]);
15        G_stolpci = {};
16        for k=1:length(T_Rob)
17            stolpec = fnval(T_Rob{k}, [rx;ry]);
18            G_stolpci{k} = stolpec';
19        end
20        for i = 1:length(T_Rob)
21            for j = i:length(T_Rob)
22                G(i,j) = sum(G_stolpci{i}.*G_stolpci{j}); G(j,i) = G(i,j);
23            end
24        end
25        b = zeros(length(T_Rob), 1);
26        for i=1:length(T_Rob)
27            bsp_i = T_Rob{i}; z_i = G_stolpci{i}; msh_zi = reshape(z_i, size(msh_z));
28            b(i) = sum(sum(msh_zi.*msh_z));
29        end
30        a = G\b;
31        [bsp_nov2, f_nov] = THB_zlepek(T_Rob, a);
32        z1 = f_nov(rx,ry); z_rez_Omega = reshape(z1, size(msh_z));
33        err_inf_rob = max(max(abs(msh_z-z_rez_Omega)));
34        if iscell(bsp_nov2)
35            grd_tmp = bsp_nov2{1};
36            grad_rob = @(x,y) [fnval(fnder(grd_tmp,[1 0]), [x;y]);...
37                            fnval(fnder(grd_tmp,[0 1]), [x;y])];
38            for i=2:length(bsp_nov2)
39                grd_tmp = bsp_nov2{i};
40                grad_rob = @(x,y) grad_rob(x,y) + ...
41                            [fnval(fnder(grd_tmp,[1 0]), [x;y]);...
42                             fnval(fnder(grd_tmp,[0 1]), [x;y])];
43            end
44        else
45            grad_rob = @(x,y) [fnval(fnder(bsp_nov2,[1 0]), [x;y]);...
46                             fnval(fnder(bsp_nov2,[0 1]), [x;y])];
47        end
48    end
49    %Resevanje Homogenega dela
50    F_orig = @(x,y) x.*y.*(x-1).*(y-1) .* exp(1000*((x-0.5).^2 + (y-0.5).^2).^(-1));
51    msh_z = F_orig(msh_x, msh_y);
52    G = zeros(length(T));
53    rx = reshape(msh_x,[1 numel(msh_x)]); ry = reshape(msh_y,[1 numel(msh_y)]);
54    rz = reshape(msh_z,[1 numel(msh_z)]);
55    gradienti = {};
56    for i=1:length(T)
57        grd_x = fnder(T{i}, [1 0]); grd_y = fnder(T{i}, [0 1]);
58        grd_x_val = fnval(grd_x, [rx;ry]); grd_y_val = fnval(grd_y, [rx;ry]);
59        gradienti{i} = {grd_x_val, grd_y_val};
60    end
61    for i=1:length(T)
62        for j=1:length(T)
63            produkt = gradienti{i}{1}.*gradienti{j}{1} + gradienti{i}{2}.*gradienti{j}{2};
64            G(i,j) = sum(produkt);
```



```

65     end
66 end
67 G_stolpci = {};
68 for k=1:length(T)
69     stolpec = fnval(T{k}, [rx;ry]);
70     G_stolpci{k} = stolpec';
71 end
72 b = zeros(length(T), 1);
73 grad_rob_val = grad_rob(rx,ry);
74 %%%%%% Diskretni laplace %%%%%%%%%%%%%%
75 h = x(2)-x(1); rf = -4*reshape(del2(F_orig(msh_x,msh_y),x,y), size(rx));
76 for i=1:length(T)
77     stolpec = G_stolpci{i}; grad = gradient{i};
78     b(i) = sum(stolpec.*rf')-sum(grad_rob_val(1,:).*grad{1} + grad_rob_val(2,:).*grad{2});
79 end
80 a = G\b; [bsp_nov3, f_nov] = THB_zlepek(T, a); z1 = f_nov(rx,ry);
81 z_rez = reshape(z1, size(msh_z)); err = abs(msh_z - z_rez); err_inf = max(max(err));
82 % Kvazi-Adaptivno lokalno zgoscevanje
83 U_tmp = T{end}.knots{1}; V_tmp = T{end}.knots{2};
84 x_min = min(msh_x(find(err >= tol))); ind_x_min = find(abs(U_tmp-x_min) == min(abs(U_tmp-x_min)
85 ));
86 if isempty(x_min)
87     A = 0;
88 elseif U_tmp(ind_x_min(end)) <= x_min
89     A = U_tmp(ind_x_min(end));
90 else
91     A = U_tmp(ind_x_min(end)-1);
92 end
93 x_max = max(msh_x(find(err >= tol))); ind_x_max = find(abs(U_tmp-x_max) == min(abs(U_tmp-x_max)
94 ));
95 if isempty(x_max)
96     B = 0;
97 elseif U_tmp(ind_x_max(end)) >= x_max
98     B = U_tmp(ind_x_max(end));
99 else
100    B = U_tmp(ind_x_max(end)+1);
101 end
102 y_min = min(msh_y(find(err >= tol))); ind_y_min = find(abs(V_tmp-y_min) == min(abs(V_tmp-y_min)
103 ));
104 if isempty(y_min)
105     C = 0;
106 elseif V_tmp(ind_y_min(end)) <= y_min
107     C = V_tmp(ind_y_min(end));
108 else
109     C = V_tmp(ind_y_min(end)-1);
110 end
111 y_max = max(msh_y(find(err >= tol))); ind_y_max = find(abs(V_tmp-y_max) == min(abs(V_tmp-y_max)
112 ));
113 if isempty(y_max)
114     D = 0;
115 elseif V_tmp(ind_y_max(end)) >= y_max
116     D = V_tmp(ind_y_max(end));
117 else
118     D = V_tmp(ind_y_max(end)+1);
119 end
120 vektorji_vozlov = [vektorji_vozlov, {{U_tmp, V_tmp}}];
121 if pogojno == true
122     nadaljuj = false;
123 end
124 if length([A B C D]) < 4 || ((A-B)*(C-D) == 0) || err_inf <= tol
125     nadaljuj = false;
126 end
127 Omega = [Omega, {[A B;C D]};
128 nadaljuj = false;
129 end
130 hold on;
131 for i = 1:length(Omega)
132     pravokotnik = Omega{i};
133     if ~isempty(pravokotnik)
134         U_tmp = vektorji_vozlov{i}{1}; V_tmp = vektorji_vozlov{i}{2};
135         x_start = max(find(U_tmp == pravokotnik(1,1))); x_end = min(find(U_tmp == pravokotnik(1,2)))
136         ;
137         y_start = max(find(V_tmp == pravokotnik(2,1))); y_end = min(find(V_tmp == pravokotnik(2,2)))
138         ;
139         U_tmp = U_tmp(x_start:x_end); V_tmp = V_tmp(y_start:y_end);
140         for l = 1:length(U_tmp)
141             x1 = U_tmp(l); plot3([x1 x1], [pravokotnik(2,1), pravokotnik(2,2)], [i-1, i-1], 'k');
142         end
143     end
144     for l = 1:length(V_tmp)

```

```
138         y1 = V_tmp(1); plot3([pravokotnik(1,1), pravokotnik(1,2)], [y1,y1], [i-1, i-1], 'k')
139     end
140 end
141 end
142 hold off;
```

PRILOGA S Reševanje nehomogene Poissonove enačbe (Škatlasti zleпки)

```
1 u_tocna = @(x,y) x.*y.*(x-1).*(y-1) .* exp(1000*((x-0.5).^2 + (y-0.5).^2)).^(-1);
2 f_Omega = @(x,y) 0.01.*x; %Funkcija na robu
3 X = linspace(0,1,70); Y = linspace(0,1,70);
4 [msh_x, msh_y] = meshgrid(X,Y);
5 rx = reshape(msh_x, [numel(msh_x), 1]);
6 ry = reshape(msh_y, [numel(msh_y), 1]);
7 m = 16 ;
8 a=0; b=1; c=0; d=1;
9 Omega = {[a b; c d]};
10 tol = 10^-3; nadaljuj = true;
11 l=1;
12 while nadaljuj == true
13     [basis, triang] = HBox_Baza_alt(Omega, m);
14     [basis_pde, triang_pde, koefs_pde] = HBox_Baza_PDE_alt(Omega, m);
15     %Najprej z vsemi baznimi funkcijami aproksimiramo rob
16     g_stolpci = {};
17     for k = 1:length(basis)
18         g_stolpci{k} = box_val(basis{k},triang{k}, [rx ry]);
19     end
20     G = zeros(length(basis));
21     for i = 1:length(basis)
22         for j = i:length(basis)
23             G(i,j) = sum(g_stolpci{i}.*g_stolpci{j});
24             G(j,i) = G(i,j);
25         end
26     end
27     b = zeros(length(basis),1);
28     rz2 = f_Omega(rx,ry);
29     for i = 1:length(basis)
30         b(i) = sum(g_stolpci{i}.*rz2);
31     end
32     a_rob = G\b;
33     u_rob_approx = Box_LinCmb(basis, triang, a_rob);
34     u_rob_val = u_rob_approx([rx ry]);
35     grad_rob_val = {zeros(size(rx)), zeros(size(ry))};
36     gradienti_vsi_val = {};
37     for i = 1:length(basis)
38         tocke = triang{i}.Points;
39         prve = tocke(1,:);
40         [grd_x, tr_x] = De1_B222_fun(prve,m);
41         [grd_y, tr_y] = De2_B222_fun(prve,m);
42         gradienti_vsi_val{i} = {box_val(grd_x, tr_x, [rx ry]), box_val(grd_y, tr_y, [rx ry])};
43         grad_rob_val{1} = grad_rob_val{1} + a_rob(i)*gradienti_vsi_val{i}{1};
44         grad_rob_val{2} = grad_rob_val{2} + a_rob(i)*gradienti_vsi_val{i}{2};
45     end
46     k=1;
47     gradienti_val = {};
48     for i = 1:length(triang_pde)
49         k1 = 1;
50         gradienti_x={};gradienti_y={};
51         triang_x={};triang_y={};
52         for j=1:length(triang_pde{i})
53             tocke = triang_pde{i}{j}.Points;
54             prve = tocke(1,:);
55             druge = tocke(2,:);
56             % Ker delamo s hierarhicno bazo so nekatere funkcije finejse in ker
57             % a priori ne poznamo vrstnega reda v bazi ponovno izluscimo faktor
58             % skaliranja!
59             m2 = inv(abs(prve(1)-druge(1)));
60             [grad_x, tr_x] = De1_B222_fun(prve,m2);
61             [grad_y, tr_y] = De2_B222_fun(prve,m2);
62             gradienti_x{k1} = grad_x;gradienti_y{k1} = grad_y;
63             triang_x{k1} = tr_x;triang_y{k1} = tr_y;
64             k1 = k1+1;
```

```

65     end
66     grad_fun = {Box_LinCmb(gradienti_x, triang_x, koefs_pde{i});...
67                 Box_LinCmb(gradienti_y, triang_y, koefs_pde{i})};
68     grd_x = grad_fun{1};
69     grd_y = grad_fun{2};
70     gradienti_val{k} = {m2.*grd_x([rx ry]), m2.*grd_y([rx ry])};
71     k=k+1;
72
73     end
74     G = zeros(length(basis_pde));
75     for i=1:length(basis_pde)
76         for j=i:length(basis_pde)
77             G(i,j) = sum(gradienti_val{i}{1}.*gradienti_val{j}{1} + ...
78                         gradienti_val{i}{2}.*gradienti_val{j}{2});
79             G(j,i) = G(i,j);
80         end
81     end
82     g_stolpci = {};
83     for k = 1:length(basis_pde)
84         g_stolpci{k} = basis_pde{k}([rx ry]);
85     end
86     b = zeros(length(basis_pde),1);
87     %%%%%%%%% Diskretni laplace %%%%%%%%%%%%%%%
88     h = X(2)-X(1);
89     rf = -4*reshape(del2(u_tocna(msh_x,msh_y),X,Y), size(rx));
90     %%%%%%%%%%%%%%%
91     for i = 1:length(basis_pde)
92         b(i) = sum(g_stolpci{i}.*rf)-sum(gradienti_val{i}{1} .* ...
93             grad_rob_val{1} + gradienti_val{i}{2} .* grad_rob_val{2});
94     end
95     a = G\b;
96     u_approx_z = zeros(size(msh_x));
97     for k=1:length(basis_pde)
98         u_approx_z = u_approx_z + a(k).*basis_pde{k}([msh_x, msh_y]);
99     end
100    u_rob = u_rob_approx({msh_x, msh_y});
101    u_h = u_approx_z + u_rob;
102    mesh(msh_x, msh_y, u_h)
103    err = abs(u_tocna(msh_x, msh_y)+f_Omega(msh_x, msh_y)-u_h);
104    r_err = err(:);
105    err_inf = max(max(err))
106    length(basis_pde)
107    if (err_inf <= tol) | (l == 2)
108        nadaljuj = false;
109    else
110        omega = [];
111        for i=1:length(triang_pde)
112            TR_cell = triang_pde{i};
113            if length(TR_cell) == 1
114                TR = TR_cell{1};
115                prve = TR.Points(1,:);
116                zadnje = TR.Points(end,:);
117                ID = pointLocation(TR,[rx ry]);
118                err_infi = max(r_err(find(~isnan(ID))));
119                if err_infi > tol
120                    omega = [omega; prve];
121                end
122            end
123        end
124    end
125    Omega = [Omega, {omega}];
126    l=l+1;
127    toc;
128 end

```

PRILOGA T Reševanje nehomogene Poissonove enačbe z reparametrizacijo (B-zlepki)

```
1 U0 = [0 0 1/3 2/3 1 1]; V0 = [0 0 1/3 2/3 1 1];
2 p0 = 1; q0 = 1;
3 B_U0 = baza(U0,p0); B_V0 = baza(V0,q0);
4 T0 = {};
5 k = 1;
6 for i = 1:length(B_U0)
7     for j = 1:length(B_V0)
8         sp_U = B_U0(i);
9         sp_V = B_V0(j);
10        [U_tmp, P_tmp] = fnbrk(sp_U, 'knots', 'coeffs');
11        [V_tmp, Q_tmp] = fnbrk(sp_V, 'knots', 'coeffs');
12
13        T0{k} = spmak({U_tmp, V_tmp}, Q_tmp .* P_tmp');
14        k = k+1;
15    end
16 end
17 kont = [zeros(1,4) 2/3 .* ones(1,4) 4/3 .* ones(1,4) 2*ones(1,4); ...
18         linspace(0,3,4) linspace(0, 7/3, 4) linspace(0,5/3,4) linspace(0,1,4);
19         zeros(1,16)];
20 m = 100; %Koraki delitev v diskretizaciji
21 x_omega = linspace(0,1,m);
22 y_omega = linspace(0,1,m);
23 [msh_x_omega, msh_y_omega] = meshgrid(x_omega, y_omega);
24 rx_omega = reshape(msh_x_omega, [1 numel(msh_x_omega)]);
25 ry_omega = reshape(msh_y_omega, [1 numel(msh_y_omega)]);
26 bsp = spmak({U0,V0}, kont, [3 4 4]);
27 G_fun = @(x,y) fnval(bsp, [x;y]);
28 dxBsp = fnder(bsp, [1 0]);
29 dyBsp = fnder(bsp, [0 1]);
30 dxBspVal_fun = @(u) fnval(dxBsp, [u(:,1)';u(:,2)']);
31 dyBspVal_fun = @(u) fnval(dyBsp, [u(:,1)';u(:,2)']);
32 paren = @(x, varargin) x(varargin{:});
33 J = @(u) [paren(dxBspVal_fun(u), 1,1) paren(dxBspVal_fun(u), 2,1); ...
34           paren(dyBspVal_fun(u), 1,1) paren(dyBspVal_fun(u), 2,1)];
35 detJ = @(x,y) paren(dxBspVal_fun([x,y]), 1,1:length([x,y])) .* ...
36 paren(dyBspVal_fun([x,y]), 2,1:length([x,y])) - ...
37 paren(dyBspVal_fun([x,y]), 1,1:length([x,y])) .* ...
38 paren(dxBspVal_fun([x,y]), 2,1:length([x,y]));
39 x = linspace(0,1,m);
40 y = linspace(0,1,m);
41 [msh_x, msh_y] = meshgrid(x,y);
42 F_orig = @(x,y) x.*y.*(x-2).*(y+x-3).*cos(3.*x).*sin(3.*y);
43
44 f = @(x,y) -( -6.*(x-2).*.x.*y.*sin(3.*x).*sin(3.*y) - ...
45              6.*(x-2).*y.*(x+y-3).*sin(3.*x).*sin(3.*y) - ...
46              6.*x.*y.*(x+y-3).*sin(3.*x).*sin(3.*y) + ...
47              6.*(x-2).*.x.*y.*cos(3.*x).*cos(3.*y) + ...
48              6.*(x-2).*.x.*(x+y-3).*cos(3.*x).*cos(3.*y) + ...
49              2.*(x-2).*.x.*cos(3.*x).*sin(3.*y) + ...
50              2.*(x-2).*y.*cos(3.*x).*sin(3.*y) + ...
51              2.*x.*y.*cos(3.*x).*sin(3.*y) - ...
52              18.*(x-2).*.x.*y.*(x+y-3).*cos(3.*x).*sin(3.*y) + ...
53              2.*y.*(x+y-3).*cos(3.*x).*sin(3.*y));
54 msh_z = f(msh_x, msh_y);
55 rx = reshape(msh_x, [1 numel(msh_x)]);
56 ry = reshape(msh_y, [1 numel(msh_y)]);
57 rz = reshape(msh_z, [1 numel(msh_z)]);
58 rxy = [rx;ry];
59 J_val_cell_inv = {};
60 for i = 1:max(size(rxy))
```

```

61     u = rxy(:,i)';
62     Ji = J(u);
63     J_val_cell_inv{i} = inv(Ji);
64     end
65 p = 3; q = 3; n = 0;
66 U = [zeros(1,p) linspace(0,1,2+n) ones(1,p)];
67 V = [zeros(1,q) linspace(0,1,2+n) ones(1,q)];
68 Omega = {[0 1; 0 1]};
69 tol = 0.02; nadaljuj = true; pogojno = false; vektorji_vozlov = {{U,V}};
70 while nadaljuj == true
71     tic;
72     T = THB.baza(U,V,p,q,Omega,1);
73     G = zeros(length(T));
74     gradienti = {};
75     for i=1:length(T)
76         grd_x = fnder(T{i}, [1 0]); grd_y = fnder(T{i}, [0 1]);
77         grd_x_val = fnval(grd_x, [rx;ry]); grd_y_val = fnval(grd_y, [rx;ry]);
78         %Popravi gradiente z inverzom J-ja
79         grad_cell = mat2cell([grd_x_val;grd_y_val], 2,...
80                             ones(1, max(size([grd_x_val;grd_y_val]))));
81         popravljen_grad_cell = cellfun(@mtimes, J_val_cell_inv, grad_cell, 'UniformOutput', false);
82         popravljen_grad = cell2mat(popravljen_grad_cell);
83         gradienti{i} = {popravljen_grad(1,:), popravljen_grad(2,:)};
84     end
85     detJ_val = abs(detJ(rx',ry'));
86     for i=1:length(T)
87         for j=i:length(T)
88             produkt = (gradienti{i}{1}.*gradienti{j}{1} + gradienti{i}{2}.*gradienti{j}{2}) .*
89                     detJ_val;
90             G(i,j) = sum(produkt);
91             G(j,i) = G(i,j);
92         end
93     end
94     G_stolpci = {};
95     for k=1:length(T)
96         stolpec = fnval(T{k}, [rx;ry]);
97         G_stolpci{k} = stolpec';
98     end
99     b = zeros(length(T), 1);
100     xy_reparam = G_fun(rx,ry); %Reparametrizacija
101     rz_rep = f(xy_reparam(1,:), xy_reparam(2,:));
102     msh_x_rep = reshape(xy_reparam(1,:), size(msh_x));
103     msh_y_rep = reshape(xy_reparam(2,:), size(msh_y));
104     msh_z_rep = reshape(rz_rep, size(msh_x));
105     for i=1:length(T)
106         z_i = G_stolpci{i};
107         msh_zi = reshape(z_i, size(msh_x));
108         msh_detJval = reshape(detJ_val, size(msh_x));
109         b(i) = sum(sum(msh_zi.*msh_z_rep.*abs(msh_detJval)));
110     end
111     a = G\b;
112     [bsp_nov2, f_nov] = THB.zlepek(T, a);
113     z1 = f_nov(rx,ry);
114     z_rez = reshape(z1, size(msh_x_rep));
115     err = abs(F_orig(msh_x_rep, msh_y_rep) - z_rez);
116     err2 = sqrt(sum(sum((F_orig(msh_x_rep, msh_y_rep) - z_rez).^2))/sum(sum(F_orig(msh_x_rep,
117     msh_y_rep).^2)));
118     err_inf = max(max(err));
119     % Kvazi-Adaptivno lokalno zgoscevanje
120     U_tmp = T{end}.knots{1}; V_tmp = T{end}.knots{2};
121     x_min = min(msh_x(find(err >= tol)));
122     ind_x_min = find(abs(U_tmp-x_min) == min(abs(U_tmp-x_min)));
123     if isempty(x_min)
124         A = 0;
125     elseif U_tmp(ind_x_min(end)) <= x_min
126         A = U_tmp(ind_x_min(end));
127     else
128         A = U_tmp(ind_x_min(end)-1);
129     end
130     x_max = max(msh_x(find(err >= tol)));
131     ind_x_max = find(abs(U_tmp-x_max) == min(abs(U_tmp-x_max)));
132     if isempty(x_max)
133         B = 0;
134     elseif U_tmp(ind_x_max(end)) >= x_max
135         B = U_tmp(ind_x_max(end));
136     else
137         B = U_tmp(ind_x_max(end)+1);
138     end
139     y_min = min(msh_y(find(err >= tol)));

```

```

138 ind_y_min = find(abs(V_tmp-y_min) == min(abs(V_tmp-y_min)));
139 if isempty(y_min)
140     C = 0;
141 elseif V_tmp(ind_y_min(end)) <= y_min
142     C = V_tmp(ind_y_min(end));
143 else
144     C = V_tmp(ind_y_min(end)-1);
145 end
146 y_max = max(msh_y(find(err >= tol)));
147 ind_y_max = find(abs(V_tmp-y_max) == min(abs(V_tmp-y_max)));
148 if isempty(y_max)
149     D = 0;
150 elseif V_tmp(ind_y_max(end)) >= y_max
151     D = V_tmp(ind_y_max(end));
152 else
153     D = V_tmp(ind_y_max(end)+1);
154 end
155 vektorji_vozlov = [vektorji_vozlov, {{U_tmp, V_tmp}}];
156 if pogojno == true
157     nadaljuj = false;
158 end
159 if length([A B C D]) < 4 || ((A-B)*(C-D) == 0) || err_inf <= tol
160     nadaljuj = false;
161 end
162 [A B;C D]
163 Omega = [Omega, {[A B;C D]}];
164 end
165 jeZacetno = @(A) sum(sum(A == [0 1;0 1])) == 4;
166 start = max(find(cellfun(jeZacetno, Omega) == 1));
167 figure;
168 hold on;
169 for i=start:length(vektorji_vozlov)
170     pravokotnik = Omega{i};
171     U_tmp = vektorji_vozlov{i}{1};
172     V_tmp = vektorji_vozlov{i}{2};
173     x_start = max(find(U_tmp == pravokotnik(1,1)));
174     x_end = min(find(U_tmp == pravokotnik(1,2)));
175     y_start = max(find(V_tmp == pravokotnik(2,1)));
176     y_end = min(find(V_tmp == pravokotnik(2,2)));
177     U_tmp = U_tmp(x_start:x_end);
178     V_tmp = V_tmp(y_start:y_end);
179     [mu, mv] = meshgrid(U_tmp, V_tmp);
180     rep_mesh = G_fun(mu(:) ', mv(:) ');
181     rep_mu = reshape(rep_mesh(1,:), size(mu));
182     rep_mv = reshape(rep_mesh(2,:), size(mv));
183     mesh(rep_mu, rep_mv, zeros(size(mu)))
184 end
185 hold off

```