Reševanje Poissonove enačbe z metodo kubičnih končnih elementov

14. julij 2023

1 Problem

Dano imamo poligonsko domeno $\Theta \subset \mathbb{R}^2$ ter funkciji $F \in L^2(\Theta)$, $G \in L^2(\partial\Theta)$. Iščemo rešitev $U \in H^2(\Theta)$, da bo v domeni Θ veljalo.

$$-\Delta U = F \tag{1}$$

ter hkrati $U = G$ na robu domene $\partial\Theta$.

2 Ideja rešitve

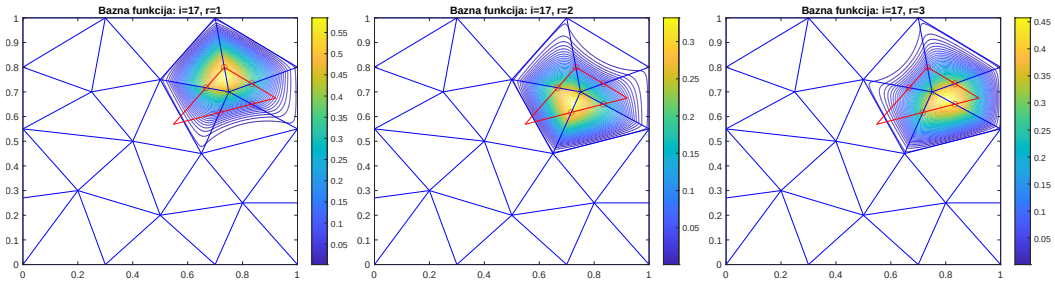
Rešitev U bomo nastavili kot linearno kombinacijo baznih funkcij zlepkov na triangulaciji S_i kot $U = \sum_i c_i S_i$. Koeficiente c_i bomo dobili z reševanjem primernih linearnih sistemov, kot bo opisano v nadaljevanju.

3 Priprava domene

Kot prvi korak domeno trianguliramo. To storimo z vgrajeno matlab funkcijo delaunay, ki za dano domeno vrne njeno Delaunayevo triangulacijo.

4 Priprava baznih funkcij

Baznih funkcij S_i bo $3n_V$, kjer je n_V število vozlišč triangulacije. Vsako vozlišče v_i porodi tri bazne funkcije na naslednji način: Najprej določimo trikotnik $[q_0, q_1, q_2]$ s čim manjšo ploščino, ki vsebuje točko v_i ter za vsakega njenega soseda v_j tudi točko $2/3v_i + 1/3v_j$. Bazno funkcijo $B_{i,r}$ potem dobimo kot zlepek, ki interpolira določene vrednosti in gradiente na vozliščih. Podrobnosti smo predelali na predavanjih in jih tu ne bomo ponavaljali. Povemo samo, da dobimo $3n_V$ baznih funkcij $B_{i,r}, i = 1, \ldots, n_V, r = 1, 2, 3$. Primere treh baznih funkcij na istem vozlišču so prikazani na sliki:



Primeri baznih funkcij $B_{i,r}$ za $i = 17, r = 1, 2, 3$. Prikazan je C^0 zlepek s kvadratično preciznostjo. Z rdečo je prikazan uporabljen trikotnik $[q_0, q_1, q_2]$. Nosilec so v tem primeru trikotniki, ki imajo $i = 17$ za eno izmed oglišč.

5 Kvaziinterpolacija robnega pogoja

Preuredimo $3n_V$ baznih funkcij S_i v takšen vrstni red, da jih je prvih n ničelnih na robu domene $\partial\Theta$. Nastavek za rešitev potem prepišemo kot:

$$U = \sum_{i=1}^{n} c_i S_i + \sum_{i=n+1}^{N} c_i S_i = \sum_{i=1}^{n} c_i S_i + S_0 \tag{2}$$

Ker velja robni pogoj $U = G$, ter hkrati $U = S_0$ na robu, bomo koeficiente c_i za $i > n$ določili tako, da bo zlepek S_0 čim bolje opisal G.

Ker imamo na robu podano le vrednost funkcije G in ne tudi gradienta (torej, imamo Dirichletov robni pogoj), ne moremo uporabiti prej omenjenih interpolacijskih metod in se zatečemo h kvaziinterpolaciji. Pri kvaziinterpolaciji iščemo zlepek S_0, ki minimizira $\sum_{i=1}^{m}(G(p_i) - S_0(p_i))^2$, za m izbranih točk na robu p_i. Za izbrane točke bomo izbrali vozlišča triangulacije na robu, ter nekaj ekvidistančnih točk na robovih.

Na least squares minimizacijo lahko gledamo tudi kot tisto rešitev naslednjega predoločenega sistema z minimalno normo:

$$\begin{pmatrix} S_{n+1}(p_1) & \dots & S_N(p_1) \\ \vdots & \ddots & \vdots \\ S_{n+1}(p_m) & \dots & S_N(p_m) \end{pmatrix} \begin{pmatrix} c_{n+1} \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} G(p_1) \\ \vdots \\ G(p_m) \end{pmatrix} \tag{3}$$

To lahko prevedemo na reševanje normalnega sistema:

$$\begin{pmatrix} \langle S_{n+1}, S_{n+1} \rangle & \dots & \langle S_N, S_{n+1} \rangle \\ \vdots & \ddots & \vdots \\ \langle S_{n+1}, S_N \rangle & \dots & \langle S_N, S_N \rangle \end{pmatrix} \begin{pmatrix} c_{n+1} \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} \langle G, S_{n+1} \rangle \\ \vdots \\ \langle G, S_N \rangle \end{pmatrix} \tag{4}$$

kjer je $\langle F, H \rangle = \sum_{i=1}^{m} F(p_i)H(p_i)$.

6 Metoda Končnih Elementov

Končno lahko predstavimo metodo končnih elementov. V ta namen najprej zapišimo Poissonovo enačbo v šibki obliki. Grobo rečeno celotno enačbo pomnožimo z t.i. testno funkcijo V in pointegriramo po domeni:

$$-\int_\Theta \nabla^2 UV \mathrm{d}\Theta = \int_\Theta FV \mathrm{d}\Theta. \tag{5}$$

Da znižamo red odvodov uporabimo Greenovo identiteto, da dobimo

$$\int_\Theta \nabla U \nabla V \mathrm{d}\Theta = \int_\Theta FV \mathrm{d}\Theta. \tag{6}$$

Po Bubnov-Galerkinu to zahtevamo za vsako testno funkcijo V iz istega prostora zlepkov, torej mora enačba veljati, če za V vzamemo katerokoli bazno funkcijo S_i:

$$\int_\Theta \nabla U \nabla S_i \mathrm{d}\Theta = \int_\Theta FS_i \mathrm{d}\Theta \tag{7}$$

Če spet razpišemo U po baznih funkcijah zlepkov, dobimo naslednji sistem enačb:

$$\begin{pmatrix} \int_\Theta \nabla S_1 \nabla S_1 \mathrm{d}\Theta & \dots & \int_\Theta \nabla S_n \nabla S_1 \mathrm{d}\Theta \\ \vdots & \ddots & \vdots \\ \int_\Theta \nabla S_1 \nabla S_n \mathrm{d}\Theta & \dots & \int_\Theta \nabla S_n \nabla S_n \mathrm{d}\Theta \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \int_\Theta FS_1 \mathrm{d}\Theta - \int_\Theta \nabla S_0 \cdot \nabla S_1 \mathrm{d}\Theta \\ \vdots \\ \int_\Theta FS_n \mathrm{d}\Theta - \int_\Theta \nabla S_0 \cdot \nabla S_n \mathrm{d}\Theta \end{pmatrix} \tag{8}$$

Zaradi lokalnih nosilcev baznih funkcij zlepkov se integrali prevedejo na integrale po trikotnikih, hkrati pa je sistem razpršen in bi ga lahko učinkovito reševali s sparse solverjem.

7 Penalizacija robnega pogoja

Rešitev poskusimo dobiti še z minimizacijo naslednje preslikave:

$$(c_1, \dots c_N) \to \sum_{i=1}^{N} \left(\int_\Theta (\nabla^2 S + F) \cdot S_i \mathrm{d}\Theta \right)^2 + \lambda \sum_{i=1}^{m} (S(p_i) - G(p_i))^2 \tag{9}$$

Intuicija je v dejstvu, da je vrednost te preslikave vedno nenegativna, nič pa je enaka natanko tedaj, ko je tako robnemu pogoju v danih točkah in tudi diferencialni enačbi zadoščeno. Parameter $\lambda > 0$ določa kako močno želimo upoštevati robni pogoj v primerjavi z zadovoljivostjo diferencialne enačbe. Pri $\lambda = 0$ robnega pogoja sploh ne upoštevamo, pri $\lambda \gg 1$ pa ga upoštevamo veliko močneje kot samo enačbo.

Izkaže se (izpeljava je v dodatku), da lahko tokrat koeficiente zlepka S dobimo z reševanjem sistema $Mc = B$, kjer $M = H^T H + \lambda K^T K$, $B = -H^T F + \lambda K^T G$

in: $H_{ij} = \int_\Theta \nabla^2 S_j \cdot S_i \mathrm{d}\Theta$, $K_{ij} = S_j(p_i)$

$c^T = (c_1, \dots c_N)$. $F^T = (\int_\Theta F \cdot S_1 \mathrm{d}\Theta, \dots, \int_\Theta F \cdot S_N \mathrm{d}\Theta)$ $G^T = (G(p_1), \dots, G(p_m))$.

8 Opis kode

V kodi zaradi enostavnosti najprej uvedemo dva razreda: Izbrana bazna funkcija $B_{i,r}$ je predstavljana kot objekt tipa *BasisFunction* in hkrati nosi informacijo o nosilcu funkcije ter koeficiente njenih odvodov. Ker je nosilec razmeroma majhen so koeficienti kompaktno podani kot slovar. Množica vseh baznih funkcij sestavlja objekt tipa *SplineSpace* , ki vsebuje tudi podatek o triangulaciji. Glavne funkcije:

- `solve_poisson(domain, f, g, options)`
- `solve_poisson_penalized(space, P, f, g, lambdas)`

Pri tem je:

- `domain`: seznam točk, ki na opisuje domeno (mora biti konveksna).
- `f, g` funkciji F in G.
- `P`: točke na robu domene Θ za namen lažje uporabe v "immersed penalized"metodi.
- `options`: Dodatne nastavitve metode, kot so `EdgeInterpolationPoints`, `SpaceType`, `TriangulationDivisions`.

Obe zgornji funkciji kot rezultat vrneta seznam koeficientov, ki pripadajo baznim funkcijam za aproksimacijo rešitve problema. Nato lahko s pomočjo metode `[X, Y] =DomainMesh(domain, size)` dobimo množico točk za evalvacijo ter eveluiramo našo rešitev z funkcijo `Z = bpolyval_spline(space, coefficients, basis, X, Y)`. S pomočjo tega lahko nato rešitev vizualiziramo ali pa analiziramo njeno napako.

8.1 Primer uporabe

```
% Square domain [-1 1]^2
domain = [-1 -1;-1 1;1 1;1 -1    ];

% Then define the functions f and g
g = @(x,y) zeros(size(x, 1), 1);
f = @(x,y) ones(size(x, 1), 1);

% Solve the equation using the solver
[ss, coeff] = solve_poisson(domain, f, g, TriangulationDivisions=2,
    SplineType=2);

% Visualize results
[X, Y] = DomainMesh(domain, 200);
nv = size(ss.tri.Points, 1);
Z = bpolyval_spline(ss, coeff', 1:3*nv', X, Y);
surf(X, Y, Z);
```

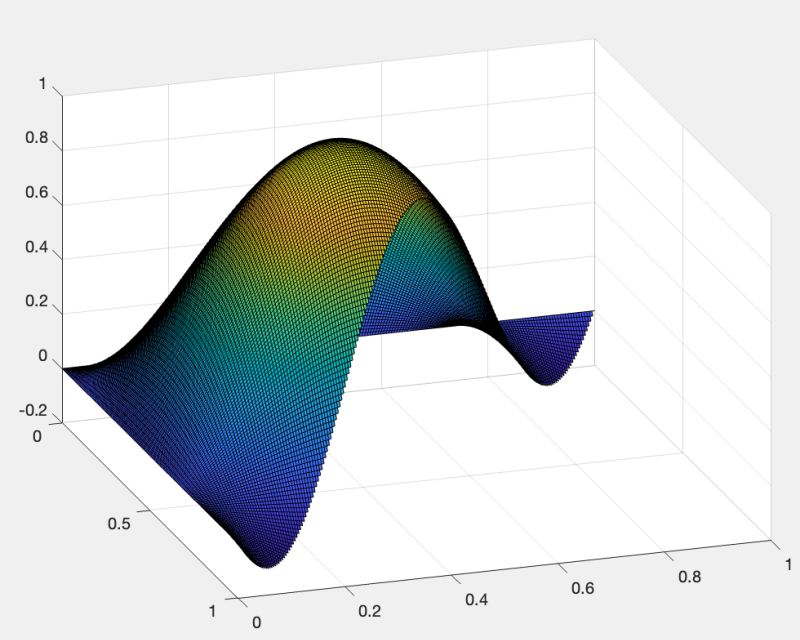
9 Rezulati

Zgornjo metodo smo testirali za različne domene ter rešitvene funkcije. Za vsak testni primer smo opazovali:

- Velikost prostora zlepkov ($3 \cdot n_v$, kjer je n_v št. točk tirangulacije)
- Napako rešitve (merili smo ℓ_{inf} napako).
- Pogojnostno število *cond* sistema za izračun koeficientov.

Za namen testiranja smo si izbrali rešitveno funkcijo $U(x, y) = \sin(4x)\sin(4y)$. Tako lahko problem formuliramo kot:

$$\begin{aligned} -\Delta U &= 32\sin(4x)\sin(4y) \\ U &= \sin(4x)\sin(4y) \text{ na } \partial\Theta \\ \Theta &= \{(x, y) \mid x \in [0, 1], 0 \leq y \leq 1 - x\} \end{aligned}$$



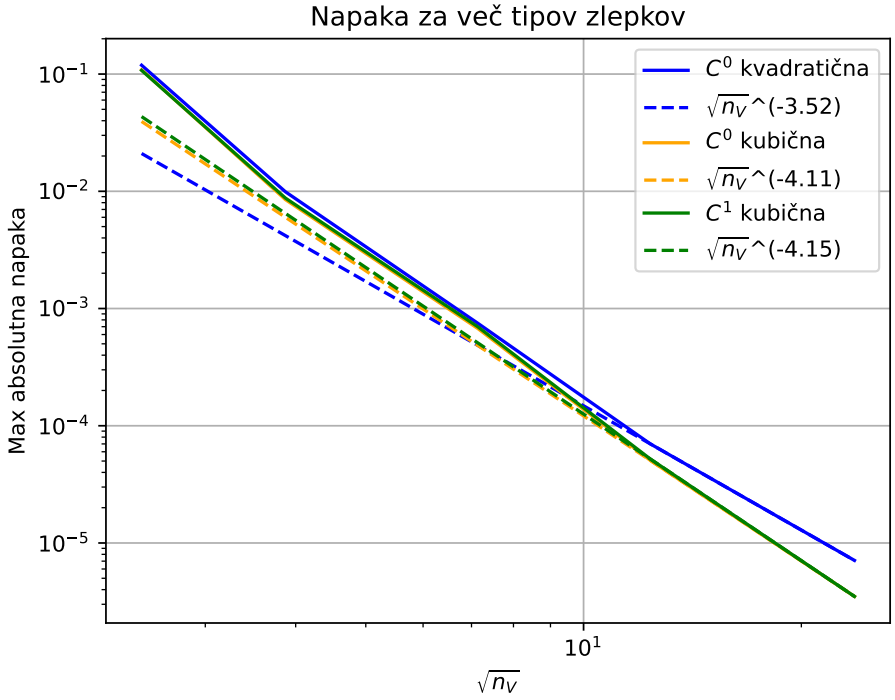
9.1 Rezultati z kvaziinterpolacijo robnega pogoja

Rezultate predstavimo z tabelo **??**. Primerjali smo različne prostore zlepkov ter različno fine triangulacije enotskega trikotnika (kar vpliva na velikost baze prostora). Opazimo lahko, da zlepki, ki lahko dosežejo kubično natančnost dosegajo boljše rezultate. To se zdi smiselno, saj za dobro aproksimacijo sinusnih funkcij potrebujejo kubične polinome. Po drugi strani pa opazimo, da bistvenih razlik med C0 in C1 ni, tudi pogojenost matrik se zdi zelo šibko odvisna od izbire zlepkov.

	C^0 kvadratični		C^0 kubični		C^1 kubični	
Velikost baze	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}
18	2.2	0.1190386	1.6	0.1087517	1.6	0.1075299
45	10	0.0098904	9.1	0.0084758	8.9	0.0087381
153	14	0.0007491	15	0.0006755	15	0.0006990
459	47	0.0000704	50	0.0000506	50	0.0000521
1683	185	0.0000071	199	0.0000035	199	0.0000035

Tabela 1: Primerjava rezultatov z kvaziinterpolacijo robnega pogoja.

Za lepšo preglednost si konvegenco napak tudi narišimo. Risali smo odvisnost napake od količine $\sqrt{n_V}$, saj je grobo rečeno tipična razdalja med točkama triangulacije v naši domeni $\propto 1/\sqrt{n_V}$. Pričakovano napaka kubičnih zlepkov pada s četrto potenco, kvadratičnega pa malce hitreje kot s pričakovano tretjo potenco.



Napaka v odvisnosti od $\sqrt{n_V}$ za več tipov zlepkov. Črtkane črte predstavljajo fit na zadnji dve točki.

9.2 Rezultati z penalizacijo robnega pogoja

Preverimo še obnašanje metode s penalizacijo. Takoj omenimo, da je prednost te metode ta, da ni potrebno baznih funkcij ločiti na zunanje in notranje - s prejšnjo metodo smo imeli težave, če

je bila katera izmed baznih funkcij skoraj ničelna na robu, ker to povzroči nestabilnost (sistem, ki ga pri kvaziinterpolaciji rešujemo postane skoraj singularen). Tako smo se bili prisiljeni odločiti za neko tolerance - majhno število pod katerim rečemo, da je zlepek ničelen na robu.

Pri penalizaciji te delitve ni, se je pa treba odločiti za vrednost parametra λ. Poglejmo kakšen vpliv ima, kjer spet rešujemo isti Poissonov problem kot pri prejšnjih analizah.

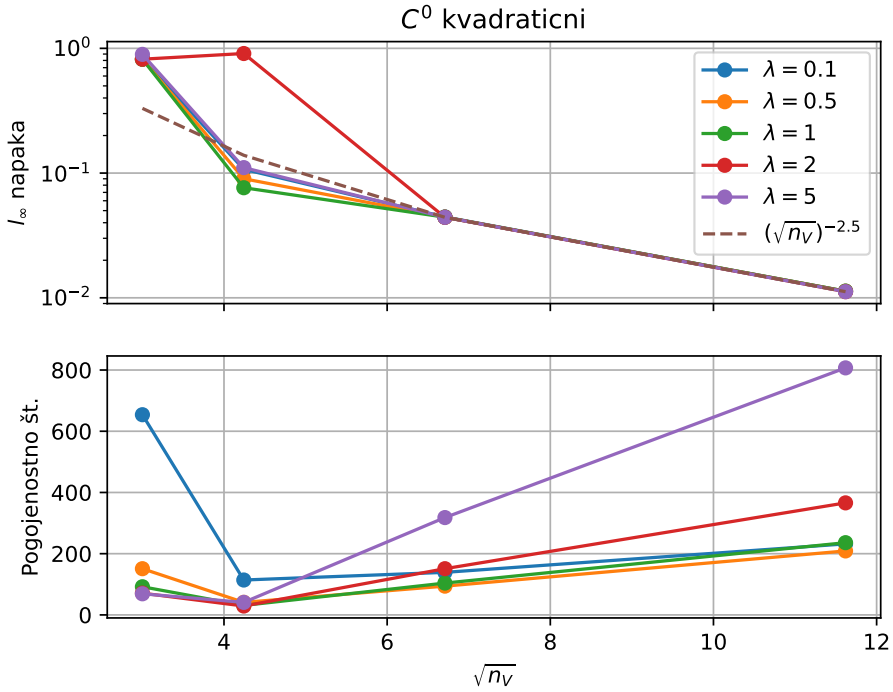
Rezultate si poglejmo najprej za C^0 zlepke z kvadratično preciznostjo:

	$\lambda = 0.1$		$\lambda = 0.5$		$\lambda = 1$		$\lambda = 2$		$\lambda = 5$	
Velikost baze	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*
9	0.8187	654	0.8187	151	0.8187	92	0.8187	71	0.8946	69
18	0.1072	114	0.0901	41	0.0763	31	0.909	29	0.1106	41
45	0.0445	139	0.0443	94	0.0443	104	.0442	151	0.0442	318
135	0.0113	232	0.0113	209	0.0113	236	0.0112	366	0.0112	807

Tabela 2: Penalizacija za C^0 zlepke s kvadratično preciznostjo.

Presenetljivo opazimo, da pri gostitvi triangulacije rešitev za več vrednosti parametra λ dobi podobno napako, po drugi strani pa vrednost parametra λ močneje vpliva na pogojenost matrike in sicer se zdi, da je najbolje vzeti $\lambda \approx 1$, vendar je to v splošnem morda odvisno od primera.

Podatki iz zgornje tabele so prikazani tudi grafično na spodnji sliki. Tu se nazorneje vidi, da je napaka pri finejši delitvi triangulacije manj odvisna od parametra λ, konvergira pa s potenco 2.5, kar je nekoliko manjše od pričakovane 3 za kvadratičen zlepek, verjetno kot posledica dejstva, da so zlepki zgolj zvezni, medtem ko za izračun matrik potrebujemo druge odvode.



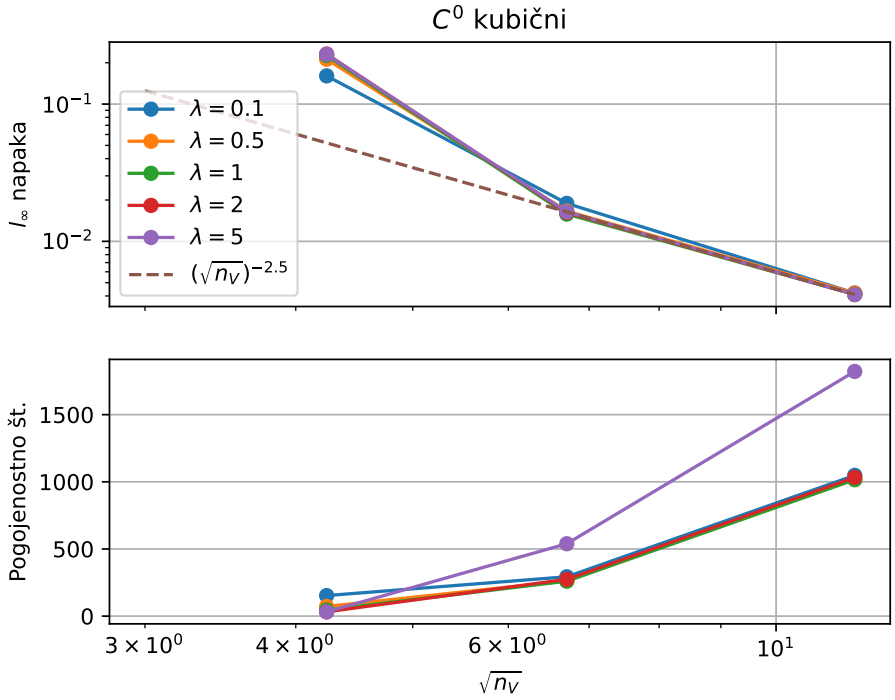
Na zgornjem grafu je prikazana konvergenca metode za več λ na spodnjem pa Pogojenostno število matrike M.

Ponovimo analizo še za C^0 zlepek s kubično preciznostjo:

	$\lambda = 0.1$		$\lambda = 0.5$		$\lambda = 1$		$\lambda = 2$		$\lambda = 5$	
Velikost baze	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*
9	/	/	/	/	/	/	/	/	/	/
18	0.1605	153	0.2121	72	0.2256	46	0.2313	32	0.2302	32
45	0.0189	293	0.0166	263	0.0158	260	0.0162	274	0.0164	539
135	0.0042	1048	0.0042	1018	0.0041	1016	0.0041	1033	0.0041	1822

Tabela 3: Penalizacija za C^0 zlepke s kubično preciznostjo.

Primer, ko je velikost baze enaka 9 sploh ni deloval, saj imamo v triangulaciji takrat le en trikotnik. Spet si lahko tabelo grafično pogledamo zgornji rezultat. Presenetljivo izgleda, da je red metode v tem primeru enak kot pri zlepkih s kvadratično preciznostjo.



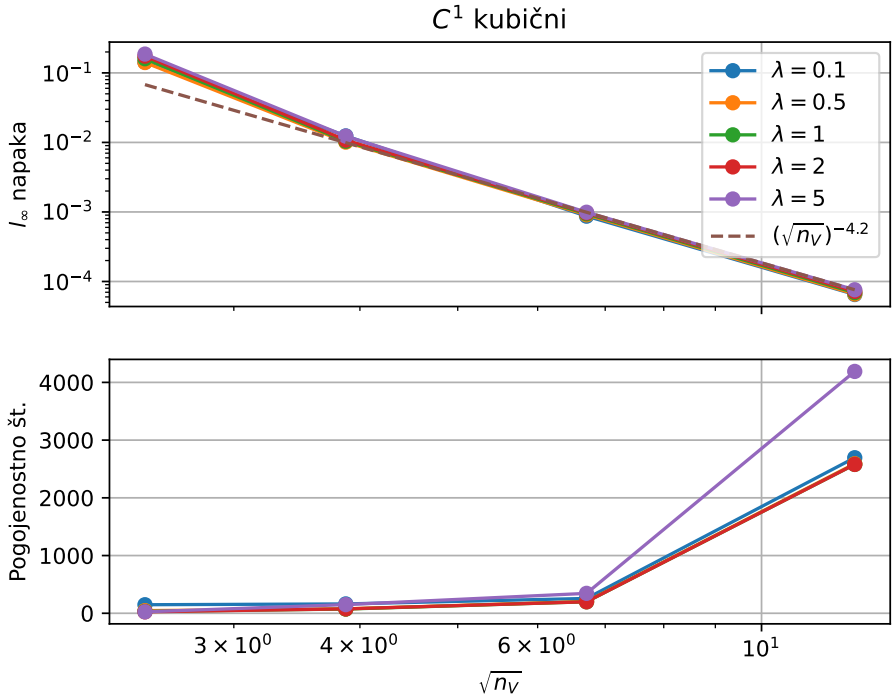
Na zgornjem grafu je prikazana konvergenca metode za več λ na spodnjem pa Pogojenostno število matrike M.

Poglejmo si še zvezne zlepke s kubično preciznostjo. Pričakujemo, da bo obnašanje bolje:

	$\lambda = 0.1$		$\lambda = 0.5$		$\lambda = 1$		$\lambda = 2$		$\lambda = 5$	
Velikost baze	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*
18	0.1428	148	0.141	46	0.1598	32	0.1742	25	0.1864	25
45	0.0124	162	0.0101	76	0.0104	73	0.0108	76	0.0123	150
135	8.710^{-4}	257	9.110^{-4}	204	9.410^{-4}	199	9.710^{-4}	199	9.910^{-4}	345
459	6.510^{-5}	2695	6.610^{-5}	2591	6.910^{-5}	2579	7.210^{-5}	2579	7.610^{-5}	4190

Tabela 4: Penalizacija za C^1 zlepke s kubično preciznostjo.

Če si rezultate spet prikažemo na grafu, vidimo, da je tokrat red metode 4, kar je pričakovano za zlepek s kubično preciznostjo.



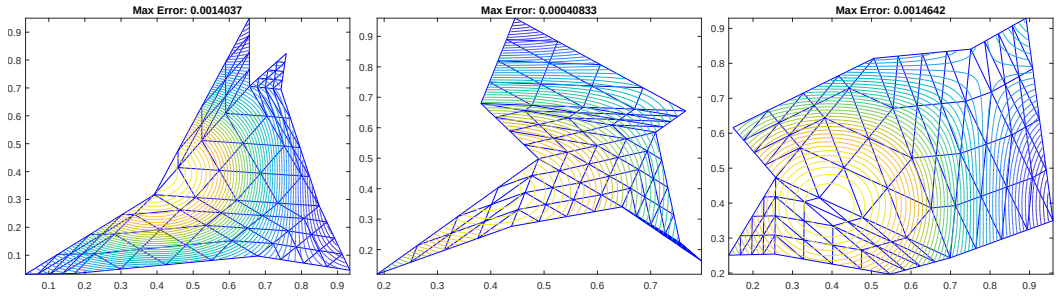
Na zgornjem grafu je prikazana konvergenca metode za več λ na spodnjem pa Pogojenostno število matrike M.

9.3 Kompleksnejše domene in drugi primeri

V tem razdelku še nadaljno raziščemo lastnosti naših metod. Spet bomo reševali $-\Delta u = 32 \sin(4x) \sin(4y)$.

Najprej preverimo, da metoda deluje tudi na kompleksnejših domenah, saj je geometrijska fleksibilnost ena izmed glavnih prednosti metode končnih elementov.

Ker v tem primeru domene niso nujno konveksne, nismo morali uporabiti Delaunayeve triangulacije in uporabimo naivno triangulacijo, ki zgolj poveže oglišča poligona. To triangulacijo potem enkrat podvojimo. Metodo smo preverili na nekaj naključno generiranih poligonih:



Rešitve danega testnega primera na čudnejših domenah in slabših triangulacijah z C^0 kvadratnimi zlepki. Max napaka je kljub temu majhna.

Pri kvaziinterpolaciji se zdi, da je lahko smiselno na robu vzet več točk kot le točke triangulacije, da bolje opišemo naš robni pogoj. V naši kodi to določa parameter "EdgeInterpolationPoints", ki pove koliko dodatnih točk upoštevamo na vsaki zunanji stranici triangulacije. Te dodatne točke so postavljene ekvidistančno. Opazimo, da je vpliv na pogojnostno število matrike (v tem primeru

	C^0 kvadratični		C^0 kubični		C^1 kubični	
Št. dodatnih točk na stranico	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*	ℓ_{inf}	*cond*
0	0.3354	∞	0.3354	∞	0.3354	∞
1	0.119	$5.6 * 10^{16}$	0.1104	$5.6 * 10^{16}$	0.1087	$5.6 * 10^{16}$
2	0.1183	30	0.1095	30	0.1087	30
3	0.1185	26	0.1097	26	0.1074	26
4	0.1188	26	0.1093	26	0.1075	26
5	0.119	27	0.1088	27	0.1075	27

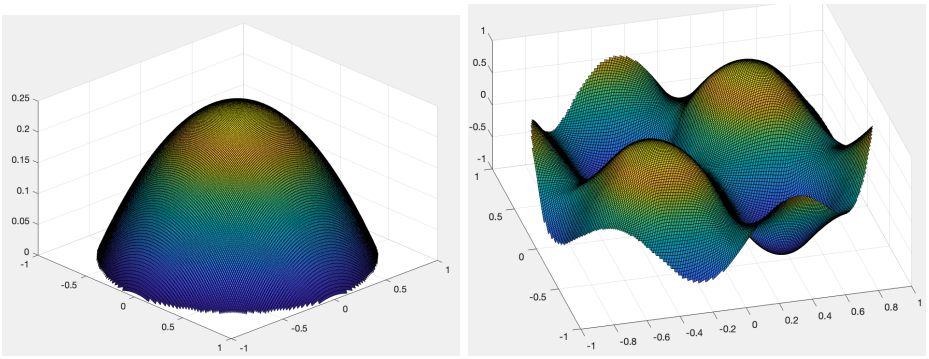
Tabela 5: Vpliv števila interpolacijskih točk na robu.

je to matrika, ki se pojavi pri kvaziinterpolaciji) ogromno če imamo manj kot dve dodatni točki na stranico. (Sistem smo rešili v least squares smislu), napaka pa se ne spreminja preveč, dokler imamo na vsaki stranici vsaj eno dodatno točko. Do sedaj smo delali z izbiro 5 dodatnih točk na stranico, kar se zdi varna izbira.

9.4 Ukrivljena domena

Metodo z penalizacijo robnega pogoja lahko uporabimo tudi za ukrivljene domene. Ideja je, da ukrivljeno domeno zaobjamemo z neko lepšo poligonsko domeno, katero lahko trianguliramo. Za vrednosti na robu pa si izberemo točke na robu ukrivljene domene in te točke podamo kot vhod funkciji za reševanje. Več informacij lahko bralec najde v [**?**]. Oglejmo si primer ko je naša domena Θ kar enotski krog. Pri tem za triangulacijsko območje izberemo kvadrat $[-1,1] \times [-1,1]$.Vse kar moramo storiti je to, da funkciji `solve_poisson_penalized(space, P, f, g, lambdas)` damo točke `P` iz roba ukrivljene domene.

Enaka metoda bi delovala tudi za druge ukrivljene domene. Glavni problem pri teh je to domeno lepo opisati ter pridobiti testne točke iz roba te domene.



Zlepek nad ukrivljeno domeno za različne začetne in robne pogoje.

10 Dodatek 1 - Izpeljava sistema za penalizacijo robnega pogoja

Dana preslikava je konveksna: Prvi člen je konveksen (integrand je linearen v c_i in torej konveksen, medtem ko integral, kvadriranje in vsota ohranjajo konveksnost), drugi pa prav tako. Nenegativna vsota (saj je $\lambda > 0$) konveksnih funkcij pa je spet konveksna.

Iz teorije optimizacije vemo, da če globalni minimum obstaja, ga dobimo s tem, da gradient preslikave enačimo z nič.

Če odvajamo po j-tem koeficientu c_j in enačimo z nič, dobimo naslednjo enačbo:

$$\sum_{i=1}^{N} 2 \int_\Theta (\nabla^2 S + F) \cdot S_i \mathrm{d}\Theta \int_\Theta \nabla^2 S_j \cdot S_i \mathrm{d}\Theta + \lambda \sum_{i=1}^{m} 2(S(p_i) - G(p_i)) S_j(p_i) = 0. \tag{10}$$

Sedaj damo člene z neznankami na levo stran:

$$\sum_{i=1}^{N} \int_\Theta \nabla^2 S \cdot S_i \mathrm{d}\Theta \int_\Theta \nabla^2 S_j \cdot S_i \mathrm{d}\Theta + \lambda \sum_{i=1}^{m} S(p_i) S_j(p_i) = \tag{11}$$

$$= -\sum_{i=1}^{N} \int_\Theta F \cdot S_i \mathrm{d}\Theta \int_\Theta \nabla^2 S_j \cdot S_i \mathrm{d}\theta + \lambda \sum_{i=1}^{m} G(p_i) S_j(p_i) \tag{12}$$

Razpišemo zlepek S (prej U).

$$\sum_{i=1}^{N} \int_\Theta \sum_{k=1}^{N} (c_k \nabla^2 S_k) \cdot S_i \mathrm{d}\Theta \int_\Theta \nabla^2 S_j \cdot S_i \mathrm{d}\Theta + \lambda \sum_{i=1}^{m} \sum_{k=1}^{N} (c_k S_k(p_i)) S_j(p_i) = \tag{13}$$

$$= -\sum_{i=1}^{N} \int_\Theta F \cdot S_i \mathrm{d}\Theta \int_\Theta \nabla^2 S_j \cdot S_i \mathrm{d}\theta + \lambda \sum_{i=1}^{m} G(p_i) S_j(p_i) \tag{14}$$

Za konec pa prejšnjo enačbo zapišemo še na nekoliko drugačen način:

$$\sum_{k=1}^{N}\left(\sum_{i=1}^{N}\int_{\Theta}\nabla^2 S_k\cdot S_i \mathrm{d}\Theta\int_{\Theta}\nabla^2 S_j\cdot S_i \mathrm{d}\Theta\right)c_k+\lambda\sum_{k=1}^{N}\left(\sum_{i=1}^{m}S_k(p_i)S_j(p_i)\right)c_k= \tag{15}$$

$$=-\sum_{i=1}^{N}\int_{\Theta}F\cdot S_i \mathrm{d}\Theta\int_{\Theta}\nabla^2 S_j\cdot S_i \mathrm{d}\theta+\lambda\sum_{i=1}^{m}G(p_i)S_j(p_i) \tag{16}$$

Če upoštevamo še, da to velja za vsak $j=1,\ldots,N$, dobimo ravno iskan sistem enačb $Mc=B$.

Literatura

[1] L. L. Schumaker, Solving Elliptic PDE's on Domains with Curved Boundaries with an Immersed Penalized Boundary Method, Journal of Scientific Computing (2019).