

UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – 2. stopnja

Jure Slak

TBA

Magistrsko delo

Mentor: doc. dr. George Mejak

Somentor: dr. Gregor Kosec

Ljubljana, 2017

Kazalo

1	Uvod	1
2	Teorija linearne elastičnosti	2
2.1	Osnove gibanja	2
2.1.1	Aksiomi gibanja	4
2.2	Napetostni tenzor	4
2.3	Enačbe gibanja	7
2.4	Konstitutivne enačbe	8
2.4.1	Mera deformacije	9
2.4.2	Hookov zakon	11
2.5	Navierova enačba	11
2.6	Obstoj in enoličnost rešitve	13
2.7	Priprava na numerično reševanje	13
2.7.1	Poenostavitev na dve dimenziji	13
2.7.2	Robni pogoji	14
3	Numerična metoda	14
3.1	Izpeljava	15
3.1.1	Ideja in motivacija	15
3.1.2	Splošna izpeljava	17
3.2	Posebni primeri	20
3.3	Algoritem	23
3.3.1	Diskretizacija	23
3.3.2	Iskanje najbližjih sosedov	27
3.3.3	Reševanje razpršenega sistema	28
3.3.4	Časovna zahtevnost	28
3.3.5	Prostorska zahtevnost	29
3.4	Pogoste vrednosti parametrov	30
4	Implementacija	32
5	Osnovni numerični zgledi	32
5.1	Enodimenzionalni robni problem	32
5.2	Poissonova enačba	34
6	Raba materialov	35
6.1	Zaključek	35

TBA

POVZETEK

TBA

TBA

ABSTRACT

TBA

Math. Subj. Class. (2010):

Ključne besede: ??

Keywords: ??

1 Uvod

Uporabili bomo [1].

2 Teorija linearne elastičnosti

Teorija linearne elastičnosti se ukvarja ...

2.1 Osnove gibanja

Definicija 2.1. *Materialno telo* \mathcal{B} je odprta povezana podmnožica v \mathbb{R}^3 z odsekoma glatkim robom skupaj z družino bijekcij

$$\chi = \{\chi: \mathcal{B} \rightarrow \chi(\mathcal{B}) \subseteq \mathbb{R}^3\},$$

da je za vsaki $\chi_1, \chi_2 \in \chi$ preslikava $\chi_2 \circ \chi_1^{-1}: \chi_1(\mathcal{B}) \rightarrow \chi_2(\mathcal{B})$ difeomorfizem.

Preslikavam $\chi \in \chi$ pravimo *konfiguracije* telesa. Odlikovani konfiguraciji χ_R pravimo *referenčna konfiguracija* in telo v tej konfiguraciji označevali z B .

Definicija 2.2. Gibanje telesa \mathcal{B} je gladka družina konfiguracij

$$\{\chi_t: \mathcal{B} \rightarrow \chi_t(\mathcal{B}) \subseteq \mathbb{R}^3, t \in \mathbb{R}\}.$$

Tem konfiguracijam pravimo *prostorske konfiguracije*. Telo v prostorski konfiguraciji označujemo z B_t .

Koordinatam telesa v referenčni konfiguraciji pravimo *referenčne koordinate* in jih pišemo z velikimi tiskanimi črkami, npr. $X \in \chi_R(\mathcal{B})$. Koordinatam telesa v prostorski konfiguraciji pravimo *prostorske koordinate* in jih pišemo z malimi črkami, npr. $x \in \chi_t(\mathcal{B})$.

Gladkost gibanja glede na t iz definicije 2.2 pomeni, da je preslikava

$$\begin{aligned} \tilde{x}: \mathbb{R} \times \chi_R(\mathcal{B}) &\subset \mathbb{R} \times \mathbb{R}^3 \rightarrow \mathbb{R}^3 \\ (t, X) &\mapsto \chi_t(\chi_R^{-1}(X)) \end{aligned}$$

gladka kot funkcija iz \mathbb{R}^4 v \mathbb{R}^3 . Preslikava \tilde{x} nam podaja zvezo med prostorskimi in referenčnimi koordinatami

$$x = \tilde{x}(t, X).$$

Pogosto opustimo strog zapis s preslikavami in pišemo kar $x = x(t, X)$. Ker je za vsak t preslikava $X \mapsto x(t, X)$ difeomorfizem, lahko funkcijo obrnemo in izrazimo tudi referenčne koordinate kot funkcijo prostorskih $X = X(t, x)$.

Pogosto za referenčno konfiguracijo vzamemo kar konfiguracijo na začetku gibanja, $\chi_R = \chi_{t=t_0}$, ni pa nujno temu tako.

Primer 2.3. gibanje, nek konkreten x ...

Vsako količino φ definirano na telesu \mathcal{B} lahko zapišemo na dva načina, v prostorskih ali v referenčnih koordinatah. Za funkcijo

$$\begin{aligned} \tilde{\varphi}: B &\rightarrow \varphi(\mathcal{B}) \\ X &\mapsto \varphi(\chi_R^{-1}(X)), \end{aligned}$$

pravimo, da je zapisana v prostorskih koordinatah in pišemo $\tilde{\varphi} = \varphi(X)$. Podobno za funkcijo

$$\begin{aligned}\hat{\varphi}: B_t &\rightarrow \varphi(\mathcal{B}) \\ x &\mapsto \varphi(\chi_R^{-1}(x)),\end{aligned}$$

pravimo, da je zapisana v prostorskih koordinatah in pišemo $\hat{\varphi} = \varphi(x)$. Med obema zapisoma lahko prehajamo s pomočjo izražave $\varphi(x) = \varphi(x(t, X))$ ali $\varphi(X) = \varphi(X(x, t))$.

Podoben zapis bomo uporabljali tudi pri diferencialnih operatorjih. Pri odvodih moramo povedati, ali na količino gledamo v prostorskih ali v referenčnih koordinatah. Z velikimi črkami bomo pisali operatorje, kjer mislimo da so referenčne koordinate konstantne, z malimi pa tiste, kjer mislimo, da so konstantne prostorske. Tako velja na primer

$$\frac{D}{Dt}\varphi(t, x) = \frac{D}{Dt}\varphi(t, x(t, X)) = \frac{\partial \varphi}{\partial t}(t, x) + \frac{\partial \varphi}{\partial x}(t, x) \frac{\partial x}{\partial t}.$$

Primer 2.4. Če imamo dano količino ...

Gibanje, kot opisano sedaj, res modelira makroskopsko gibanje kot ga poznamo iz resničnega sveta. Pogoji gladkosti zagotavljajo, da se telesa ne morejo kar izginiti in se pojaviti drugje, ter da se ne morejo sploščiti ali sekati samih sebe. Sedaj definirajmo še pojma pomika, hitrosti in pospeška.

Definicija 2.5. Količino

$$u(X) = x(t, X) - X$$

imenujemo *pomik*. Količino

$$\vec{v}(X) = \frac{\partial x}{\partial t}(t, X)$$

imenujemo *hitrost*. Količino

$$\vec{a}(X) = \frac{\partial^2 x}{\partial t^2}(t, X)$$

imenujemo *pospešek*.

Opomba 2.6. Z uporabo zgornje definicije lahko odvod po času v referenčnem koordinatnem sistemu zapišemo kot

$$\frac{D\varphi}{Dt} = \frac{\partial \varphi}{\partial t} + (\text{grad } \varphi) \vec{v}$$

Poleg že naštetih količin pa imajo telesa tudi druge fizikalne lastnosti, kot so masa in volumen, ki vplivajo na gibanje. Za modeliranje teh so pomagamo z merami.

Definicija 2.7. Predpostavimo, da imamo na \mathcal{B} definirani dve Radonovi meri, m in V , ki nam predstavljata maso in volumen. Predpostavimo še, da je $m \ll V$, torej, če je volumen nekega podtelesa nič, je tudi njegova masa nič. Od tod po Radon-Nikodymovem izreku sledi, da obstaja merljiva funkcija $\rho = \frac{dm}{dV}$, da velja

$$m(A) = \int_A \rho dV$$

za vsako merljivo podmnožico $A \subseteq \mathcal{B}$. Funkciji $\rho: \mathcal{B} \rightarrow [0, \infty)$ pravimo *gostota*.

Opomba 2.8. Ponavadi za V vzamemo kar Lebesgueovo mero na \mathbb{R}^3 , mero m pa podamo tako, da podamo gostoto telesa ρ . Meri m in V s potiskom prek konfiguracij razširimo na referenčni in prostorski položaj.

2.1.1 Aksiomi gibanja

Iz mehanike točke in iz resničnega sveta vemo, da gibanje zadošča nekim zakonom, ki jih bomo za nadaljnje izpeljave privzeli kot aksiome.

Aksiom 1 (Zakon o ohranitvi mase). Za vsako telo \mathcal{B} in vsako njegovo gibanje

$$\frac{D}{Dt}m(B_t) = 0. \quad (1)$$

Masa se med gibanjem niti ne izgublja niti ne nastaja in je vseskozi konstantna.

Druga dva aksioma bosta poleg mase imela opraviti s silami. Predpostavili bomo, da lahko sile na telo delujejo na dva načina: kot telesne sile ali pa kot površinske sile.

Definicija 2.9. *Telesna sila* \vec{f} je zvezna funkcija $f: \mathcal{B} \rightarrow \mathbb{R}^3$. *Površinska sila* \vec{t} je zvezna funkcija $t: \partial\mathcal{B} \rightarrow \mathbb{R}^3$.

Aksiom 2 (Zakon o ohranitvi gibalne količine). Za vsako telo \mathcal{B} velja

$$\frac{D}{Dt} \int_{\mathcal{B}} \dot{x} dm = \int_{\mathcal{B}} \vec{f} dV + \int_{\partial\mathcal{B}} \vec{t} dS. \quad (2)$$

Sprememba gibalne količine je enaka vsoti vseh sil, ki delujejo na telo.

Aksiom 3 (Zakon o ohranitvi vrtilne količine). Za vsako telo \mathcal{B} velja

$$\frac{D}{Dt} \int_{\mathcal{B}} \dot{x} \times x dm = \int_{\mathcal{B}} \vec{f} \times x dV + \int_{\partial\mathcal{B}} \vec{t} \times x dS. \quad (3)$$

Sprememba vrtilne količine je enaka vsoti vseh zunanjih navorov, ki delujejo na telo.

Opomba 2.10. Pri aksiomu 3 smo predpostavili, da na kontinuum ne delujejo notranji, ampak da so vsi navori posledica delovanja zunanjih sil. Drugače povedano, predpostavili smo *nepolarnost* kontinuuma.

2.2 Napetostni tenzor

Predstavljajmo si, da na telo deluje neka površinska sila \vec{t} . Ta sila deluje na zunanjo površino telesa, nato pa se prenaša skozi telo od točke do točke. Če skozi telo potegnemo namišljeno ravnino, ki ga razdeli na dva dela. Na tej površini en del kontinuuma deluje na drug del, toda ekvivalentno bi bilo, če bi enega izmed teh delov odstranili in na površino preostanka delovali z enakimi silami, kot je prej deloval drugi del.

Ta razmislek nam ponuja naslednji opis notranjih in površinskih sil telesa. V nekem trenutku t v času se postavimo v neko točko x v telesu in skozi izbrano točko potegnemo ravnino z normalo \vec{n} . Tedaj obstaja vektor \vec{t} , ki prestavlja silo, s katero

kontinuum z ene strani ravnine deluje v tej točki na tistega na drugi strani. Poleg tega, če potegnemo katero koli drugi ploskev skozi točko x z enako normalo \vec{n} , bo vektor \vec{t} , še vedno enak, saj bo kontinuum deloval z enako silo. Temu razmisleku se reče *Cauchyeva hipoteza*.

Aksiom 4 (Cauchyeva hipoteza). V telesu obstaja vektorsko polje \vec{t} inducirano s površinskimi silami, ki je odvisno samo od položaja, časa in normale na površino (resnično ali namišljeno), kjer ga opazujemo. S simboli lahko zapišemo

$$\vec{t} = \vec{t}(t, x, \vec{n}).$$

Iz zgornjega razmisleka se zdi, da bi morala biti sila, s katero prvi del kontinuuma deluje na drugega nasprotno enaka sili, s katero drugi deluje na prvega. Temu se reče tudi tretji Newtonov zakon za mehaniko kontinuuma in ga ni treba privzeti kot aksiom, ampak lahko dokažemo iz do sedaj privzetih aksiomov. Dokaza tega in naslednjega izreka bosta povzeta po dokazih iz [2, str. 104–107].

Trditev 2.11 (Cauchyeva recipročna relacija).

$$\vec{t}(t, x, -\vec{n}) = -\vec{t}(t, x, \vec{n})$$

Dokaz. Izberimo trenutek v času t in točko x . Krajše pišimo $\vec{t}_{\vec{n}} = \vec{t}(t, x, \vec{n})$. S časom t in točko x je določena tudi točka X v referenčni konfiguraciji. Oglejmo si podtelo v obliki krožnega valja z višino h^2 in radijem h , ki ima v središču točko X in je \vec{n} normala na eno izmed osnovnic. Tako podtelo zaradi odprtosti \mathcal{B} gotovo obstaja za dovolj majhen h . Označimo osnovnico z normalo \vec{n} z B^+ , nasprotno z B^- in plašč s P . Površina vsake osnovnice je πh^2 , površina plašča pa $2\pi h^3$. Volumen valja je πh^4 . Primer valja je prikazan na sliki 1.

Slika 1: asdfasdf

Uporabimo za ta valj aksiom o gibalni količini (2):

$$\frac{D}{Dt} \int_B \dot{x} dm = \int_B \vec{f} dV + \int_{\partial B} \vec{t} dS.$$

Zaradi gladkosti \dot{x} in ohranitve mase lahko zamenjamo odvod in integral in rob valja razpišemo po ploskvah. Vektor \vec{m} naj označuje normalo na plašč valja.

$$\int_B \frac{D}{Dt} (\dot{x} \rho) dV = \int_B \vec{f} dV + \int_{B^+} \vec{t}_{\vec{n}} dS + \int_{B^-} \vec{t}_{-\vec{n}} dS + \int_P \vec{t}_{\vec{m}} dS$$

Sedaj na vsakem od integralov uporabimo izrek o povprečni vrednosti

$$V(B) \frac{D}{Dt} (\dot{x} \rho)(X_1) = V(B) \vec{f}(X_2) + S(B^+) \vec{t}_{\vec{n}}(X_3) + S(B^-) \vec{t}_{-\vec{n}}(X_4) + S(P) \vec{t}_{\vec{m}}(X_5),$$

kjer so X_1, X_2, X_3, X_4, X_5 neke vmesne točke v ali na površini valja. Če vstavimo notri izraze za volumen in površine dobimo

$$\pi h^4 \frac{D}{Dt} (\dot{x} \rho)(X_1) = \pi h^4 \vec{f}(X_2) + \pi h^2 \vec{t}_{\vec{n}}(X_3) + \pi h^2 \vec{t}_{-\vec{n}}(X_4) + 2\pi h^3 \vec{t}_{\vec{m}}(X_5).$$

Če celoten izraz delimo z πh^2 in pogledamo limito $h \rightarrow 0$ gredo vse točke X_i proti X in dobimo

$$0 = \vec{t}_{\vec{n}}(X) + \vec{t}_{-\vec{n}}(X),$$

kar je po prenosu nazaj v prostorsko konfiguracijo točno to, kar smo želeli pokazati. \square

Izkaže se, da velja še več: vektor \vec{t} je linearno odvisen od normale. S pomočjo tega.

Izrek 2.12 (Cauchyev izrek o napetosti). *Obstaja tenzor σ , tako da velja*

$$\vec{t}(t, x, \vec{n}) = \sigma(t, x)\vec{n}.$$

Tenzor σ se v prostorski konfiguraciji imenuje Cauchyev napetostni tenzor, v referenčni pa Piola–Kirchhoffov napetostni tenzor in se običajno označuje drugače, npr. s τ .

Dokaz. Podobno kot pri Cauchyevi recipročni relaciji si izberimo majhno telo okoli točke $X = X(x, t)$. Tokrat naj bo to tetraeder z tremi stranicami vzporednimi koordinatnim osem in normalo \vec{n} na ploskev, ki jo razpenjajo preostale tri stranice, kot prikazano na sliki 2.

Slika 2: Tetraeder, uporabljen za dokaz Cauchyvega izreka o napetosti.

Označimo oglišča tetraedra z $A_1, A_2, A_3, A_4 = X$ in ploskev nasproti izbranega oglišča z malimi črkami S_1, S_2, S_3, S_4 . Ploščine teh ploskev označimo z a_1, a_2, a_3, a_4 . Naj bodo pravokotne stranice tetraedra dolge $\varepsilon_1 h$, $\varepsilon_2 h$ in $\varepsilon_3 h$, kjer razmerje $\varepsilon_1 : \varepsilon_2 : \varepsilon_3$ določimo tako, da je normala na S_4 enaka \vec{n} . Celotna situacija je prikazana na sliki 2. Sedaj lahko izračunamo ploščine ploskev $a_1 = \frac{1}{2}\varepsilon_2\varepsilon_3h^2$, $a_2 = \frac{1}{2}\varepsilon_1\varepsilon_3h^2$ in $a_3 = \frac{1}{2}\varepsilon_1\varepsilon_2h^2$. Volumen tetraedra je $V = \frac{1}{6}\varepsilon_1\varepsilon_2\varepsilon_3h^3$. Vektorje v smeri koordinatnih osi standardno označimo z $\vec{e}_1, \vec{e}_2, \vec{e}_3$. Zunanja normala na ploskev S_1 je $-\vec{e}_1$, na ploskev S_2 je $-\vec{e}_2$ in na ploskev S_3 je $-\vec{e}_3$. Normalo na preostalo ploskev, ki bo dolga ravno toliko, kot je ploščina te ploskve, dobimo s pomočjo vektorskega produkta stranic, ki jo oklepata.

$$\begin{aligned} a_4\vec{n} &= \frac{1}{2}(\varepsilon_1 h\vec{e}_1 - \varepsilon_2 h\vec{e}_2) \times (\varepsilon_3 h\vec{e}_3 - \varepsilon_2 h\vec{e}_2) = \\ &= -\frac{1}{2}\varepsilon_2\varepsilon_3h^2\vec{e}_1 - \frac{1}{2}\varepsilon_1\varepsilon_3h^2\vec{e}_2 - \frac{1}{2}\varepsilon_1\varepsilon_2h^2\vec{e}_3 = \\ &= a_1\vec{e}_1 + a_2\vec{e}_2 + a_3\vec{e}_3. \end{aligned}$$

Če to pomnožimo z i -tim baznim vektorjem, dobimo $a_i = (\vec{n} \cdot \vec{e}_i)a_4$. Zopet uporabimo aksiom 2 o ohranitvi gibalne količine in izrek o povprečni vrednosti

$$\begin{aligned} \frac{D}{Dt} \int_B \dot{x} dm &= \int_B \vec{f} dV + \int_{\partial B} \vec{t} dS \\ \int_B \frac{D}{Dt} (\dot{x} \rho) dV &= \int_B \vec{f} dV + \int_{S_1} \vec{t}_{-\vec{e}_1} dS + \int_{S_2} \vec{t}_{-\vec{e}_2} dS + \int_{S_3} \vec{t}_{-\vec{e}_3} dS + \int_{S_4} \vec{t}_{\vec{n}} dS \\ V \frac{D}{Dt} (\dot{x} \rho)(X_1) &= V \vec{f}(X_2) + a_1 \vec{t}_{-\vec{e}_1}(X_3) + a_2 \vec{t}_{-\vec{e}_2}(X_4) + a_3 \vec{t}_{-\vec{e}_3}(X_5) + a_4 \vec{t}_{\vec{n}}(X_6), \end{aligned}$$

pri čemer so X_i kot v prejšnjem dokazu neke točke v notranjosti ali na površini tetraedra. Sedaj enačbo delimo s a_4 in pošljemo limito $h \rightarrow 0$. Vse točke X_i limitirajo proti X , člena, ki vsebujeta volumen tetraedra se približujeta 0. Z upoštevanjem

$$\lim_{h \rightarrow 0} \frac{a_i}{a_4} = \lim_{h \rightarrow 0} \frac{(\vec{n} \cdot \vec{e}_i)a_4}{a_4} = \vec{n} \cdot \vec{e}_i$$

dobimo

$$0 = \vec{t}_{\vec{n}} + \sum_{i=1}^3 (\vec{n} \cdot \vec{e}_i) \vec{t}_{-\vec{e}_i}.$$

Z upoštevanjem Cauchyve recipročne relacije 2.11 dobimo

$$\vec{t}_{\vec{n}} = \sum_{i=1}^3 (\vec{n} \cdot \vec{e}_i) \vec{t}_{\vec{e}_i}.$$

Ta zveza nam pove, kako je napetost na poljubno ploskev povezana z napetostmi na koordinatnih ploskvah. To nam dovoljuje definicijo *Piola-Kirchhoffovega napetostnega tenzorja* τ

$$\tau = \sum_{i=1}^3 \vec{t}_{\vec{e}_i} \otimes \vec{e}_i.$$

Zanj res velja

$$\tau \vec{n} = \sum_{i=1}^3 (\vec{t}_{\vec{e}_i} \otimes \vec{e}_i)(\vec{n}) = \sum_{i=1}^3 (\vec{n} \cdot \vec{e}_i) \vec{t}_{\vec{e}_i} = \vec{t}_{\vec{n}}.$$

Če ta tenzor prenesemo v prostorske koordinate, dobimo *Cauchyev napetostni tenzor*. □

Do sedaj še nismo uporabili aksioma 3 o vrtilni količini. Z njegovo pomočjo bomo dokazali, da je σ simetričen.

Trditev 2.13. *Cauchyev napetostni tenzor je simetričen:*

$$\sigma^T = \sigma.$$

Dokaz. TODO □

2.3 Enačbe gibanja

V tem razdelku bomo iz aksiomov izpeljali lokalne enačbe gibanja. V teh enačbah bo nastopala divergenca tenzorja drugega reda, zato jo definirajmo.

Definicija 2.14. Divergenca tenzorja t drugega reda je vektorsko polje, za katerega za vsak vektor \vec{a} velja

$$\text{div}(t) \cdot \vec{a} = \text{div}(t^T \vec{a}).$$

Zakaj je ravno to prava definicija, nam pove naslednji izrek

Izrek 2.15 (Gauss). *Naj bo Ω odprta množica z odsekoma gladkim robom, \vec{n} zunanja enotska normala na $\partial\Omega$ in t tenzorsko polje na Ω . Potem velja*

$$\int_{\partial\Omega} t \vec{n} dS = \int_{\Omega} \text{div } t dV.$$

Dokaz. Naj bo \vec{a} poljuben konstanten vektor. Izračunajmo

$$\begin{aligned}\vec{a} \cdot \left(\int_{\partial\Omega} t \vec{n} dS \right) &= \int_{\partial\Omega} \vec{a} \cdot t \vec{n} dS = \int_{\partial\Omega} t^\top \vec{a} \cdot \vec{n} dS = \int_{\Omega} \operatorname{div}(t^\top \vec{a}) dV = \\ &= \int_{\Omega} \operatorname{div}(t) \cdot \vec{a} dV = \left(\int_{\Omega} \operatorname{div} t dV \right) \cdot \vec{a}.\end{aligned}$$

Pri računu smo uporabili definicijo t^\top , definicijo divergence in Gaussov izrek za vektorska polja. Ker enakost velja za vsak vektor \vec{a} , velja tudi enakost vektorskih polj v izreku. \square

Prevedimo aksiom 2 o ohranitvi gibalne količine v lokalno obliko.

Izrek 2.16 (Cauchyeva momentna enačba). *Za gibanje kontinuuma velja Cauchyeva momentna enačba*

$$\frac{D}{Dt}(\rho \vec{v}) = \vec{f} + \operatorname{Div} \tau.$$

Dokaz. Za vsako podtelo B velja

$$\begin{aligned}\frac{D}{Dt} \int_B \dot{x} dm &= \int_B \vec{f} dV + \int_{\partial B} \vec{t} dS \\ \frac{D}{Dt} \int_B \vec{v} \rho dV &= \int_B \vec{f} dV + \int_{\partial B} \tau \vec{n} dS \\ \int_B \frac{D}{Dt}(\vec{v} \rho) dV &= \int_B \vec{f} dV + \int_B \operatorname{Div} \tau dV \\ 0 &= \int_B \left(\frac{D}{Dt}(\vec{v} \rho) - \vec{f} - \operatorname{Div} \tau \right) dV.\end{aligned}$$

V računu smo uporabili Gaussov izrek 2.15 za tenzorje drugega reda. Ker to velja za poljuben B , mora biti integrand enak nič in izrek je s tem dokazan. \square

BLAH BLAH zanemarimo en kup stvari.

$$\rho \ddot{u} = \vec{f} + \operatorname{div} \sigma \quad (4)$$

2.4 Konstitutivne enačbe

Do sedaj izpeljane enačbe veljajo za poljuben kontinuum, naj bo to tekočina, plastična ali elastična trdnina. V tem razdelku si bomo ogledali enačbe, ki definirajo obnašanje našega kontinuuma kot elastične trdnine preko posplošitve Hookovega zakona, ki povezuje napetosti in deformacijo. Naučili smo se že, kako izražamo napetost, sedaj pa si pogledajmo, kako merimo deformacijo teles. Tukaj bomo tudi privzeli, da je referenčna konfiguracija kar prostorska konfiguracija na začetku gibanja.

2.4.1 Mera deformacije

Definicija 2.17 (Gradient deformacije). Količino $F = \frac{\partial x}{\partial X}(t, X) = \text{Grad } x$ imenujemo *gradient deformacije*.

Tenzor F je drugega reda in nam v vsaki točki predstavlja lokalno deformacijo telesa. Privzeli smo že, da je x difeomorfizem, torej je F neizrojev, dodatno pa bomo privzeli še, da gibanje x ohranja orientacijo, saj so gibanja realnih teles taka. Od tod sledi, da je $\det F > 0$. Fizikalno interpretacijo F dobimo z naslednjim Taylorjevim razvojem:

$$dx := x(t, X + dX) - x(t, X) = FdX + O(dX^2).$$

Tenzor F do prvega reda natančno opiše kako se vektor dX iz referenčne konfiguracije deformira v vektor dx v prostorski konfiguraciji.

Vendar, ni vsa deformacija, ki jo opiše tenzor F taka, da bi povzročala napetosti. Hookov zakon v eni dimenziji pravi, da je sila sorazmerna raztežku. Če imamo opravka s togim premikom $X \mapsto QX + a$, ta premik ne bo povzročil nobene napetosti, saj se bo telo samo premaknilo, ne pa raztegnilo.

Oglejmo si najprej primer v eni dimenziji.

Primer 2.18. Naj bo dana tanka palica kot na sliki 3, ki jo raztegnemo vzdolž njene nosilke. Premik v točki X je $u(X) = x - X$, v točki $X + dX$ pa $u(X + dX)$.

Slika 3: Razteg tanke palice vzdolž njene osi.

Dejanski relativni raztezek delčka palice dolžine dX pa je

$$\begin{aligned} \frac{dx - dX}{dX} &= \frac{x(X + dX) - x(X) - (X + dX - X)}{dX} = \\ &= \frac{u(X + dX) - u(X)}{dX} = u'(X) + O(dX). \end{aligned}$$

Do prvega reda je relativni raztezek v točki X enak kar $u'(X)$.

Preverimo še, da ta mera raztezka zadošča nekaterih intuitivnim zahtevam. Za togi premik $X \mapsto X + c$ je $u(X) = c$ in $u'(X) \equiv 0$, kot pričakovano.

Za enakomerni razteg $X \mapsto aX$ je $u(X) = aX - X$ in $u'(X) = a - 1$. Tudi to ustreza pričakovanjem, saj za $a = 1$ ni raztezka, za $a < 1$ je to skrčitev in je raztezek negativen, za $a > 1$ pa dobimo pozitivno število.

Analogna mera deformacije med točkama X in $X + dX$ v višji dimenziji bi bila

$$\varepsilon_1 = \frac{\|dx\| - \|dX\|}{\|dX\|}.$$

Toda veliko lažje je računati s kvadrati norm kot z normami vektorjev, zato je bolj primerna mera

$$\varepsilon_2 = \frac{\|dx\|^2 - \|dX\|^2}{\|dX\|^2}.$$

Ker velja $\varepsilon_2 = (1 + \varepsilon_1)^2 - 1 = 2\varepsilon_1 + \varepsilon_1^2$ je za majhne pomike $\varepsilon_2 \approx 2\varepsilon_1$ in sta meri ekvalentni.

Izračunajmo infinitezimalno aproksimacijo mere ε_2 .

$$\begin{aligned}\varepsilon_2 &= \frac{\|dx\|^2 - \|dX\|^2}{\|dX\|^2} = \frac{\|FdX + O(dX^2)\|^2}{\|dX\|^2} - 1 = \\ &= \frac{dX^\top F^\top F dX + O(dX^3)}{\|dX\|^2} - 1 = \frac{dX}{\|dX\|} \underbrace{(F^\top F - I)}_{2E} \frac{dX}{\|dX\|} + O(dX)\end{aligned}$$

V limiti $dX \rightarrow 0$ lahko mero ε_2 predstavimo s tenzorjem $F^\top F - I$, mero ε_1 pa s tenzorjem E .

Definicija 2.19 (deformacijski tenzor). Količina $E = \frac{1}{2}(F^\top F - I)$ se imenuje (Cauchy-Greenov) deformacijski tenzor.

Trditev 2.20. Deformacijski tenzor E lahko izrazimo z gradientom pomika kot

$$E = \frac{1}{2} (\text{Grad } u + \text{Grad } u^\top + \text{Grad } u^\top \text{Grad } u).$$

Dokaz. Trditev pokaže preprost račun. Spomnimo se, da je pomik definiran kot $u(X) = x(X) - X$ in je gradient pomika enak

$$\text{Grad } u(X) = \frac{\partial u}{\partial X} = \frac{\partial x}{\partial X} - I = F - I.$$

Izračunamo:

$$\begin{aligned}\frac{1}{2} (\text{Grad } u + \text{Grad } u^\top + \text{Grad } u^\top \text{Grad } u) &= \\ &= \frac{1}{2} (F - I + F^\top - I + (F - I)^\top (F - I)) = \\ &= \frac{1}{2} (F + F^\top - 2I + F^\top F - F - F^\top + I) = \\ &= \frac{1}{2} (F^\top F - I) = \\ &= E.\end{aligned}$$

□

Deformacijo F lahko z pomočjo polarne razcepa zapišemo kot $F = RU$, kjer je R ortogonalna in U pozitivno definitna matrika. Tenzor R predstavlja rotacijo, U pa razteg in strig. Predstavljamo si lahko, da je deformacijo F izvedemo tako, da najprej telo raztegnemo in strižno deformiramo, nato pa zavrtimo v končno lego.

Rotacija R ne vpliva na rabo materiala, saj je deformacija toga in ne povzroča notranjih napetosti. Prava mera deformacije, ki vpliva na napetost v materialu torej ne vsebuje nobenih rotacij. Za mero deformacije bi lahko vzeli kar to, koliko se U razlikuje od identitete, toda polarni razcep matrike je težko izračunati. Poglejmo si kakšno zvezo ima zgoraj definirani deformacijski tenzor E z U :

$$E = \frac{1}{2}(F^\top F - I) = \frac{1}{2}(U^\top R^\top RU - I) = \frac{1}{2}(U^2 - I).$$

Vidimo, da E meri, kako se U^2 razlikuje od identitete, ki predstavlja gibanje brez deformacij.

Kot pri enodimenzionalnem primeru si oglejmo, kako se E obnaša pri enostavnih deformacijah. Za toge premike imamo želeno obnašanje, kot pokazano v naslednji trditvi.

Primer 2.21. Kot prej si oglejmo, da je za toge deformacije $E = 0$. Naj bo deformacija toga, torej oblike $X \mapsto QX + c$ z ortogonalno konstantno matriko Q in konstantnim c . Tedaj je

$$E = \frac{1}{2}(F^T F - I) = \frac{1}{2}(Q^T Q - I) = \frac{1}{2}(I - I) = 0.$$

Oglejmo si še enostavni razteg v smeri osi, dan kot $X \mapsto \text{diag}(\lambda_1, \lambda_2, \lambda_3)X$. V tem primeru velja $F = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ in

$$E = \text{diag}\left(\frac{\lambda_1^2 - 1}{2}, \frac{\lambda_2^2 - 1}{2}, \frac{\lambda_3^2 - 1}{2}\right).$$

Če je $\lambda_i = 1$ je E res 0, sicer pa so v njegovih diagonalnih komponentah zapisani raztezki vzdolž posameznih osi.

V teoriji linearne elastičnosti se bomo ukvarjali z majhnimi pomiki in majhnimi gradienti pomikov. Zato poenostavimo deformacijski tenzor z geometrijsko linearizacijo: zanemarimo člen $\text{Grad } u^T \text{Grad } u$.

Definicija 2.22 (infinitesimalni deformacijski tenzor). Količino

$$\varepsilon = \frac{1}{2}(\text{Grad } u + \text{Grad } u^T) \quad (5)$$

imenujemo *infinitesimalni deformacijski tenzor*, ki je geometrijska linearizacija deformacijskega tenzorja.

2.4.2 Hookov zakon

Sedaj potrebujemo relacijo, ki povezuje premike z napetostjo. Ta bo posplošitev običajnega Hookovega zakona.

Aksiom 5 (Hookov zakon). Napetost je linearno odvisna od deformacije, preko tenzorja četrtega reda C :

$$\sigma = C\varepsilon.$$

Tenzor C se imenuje ?? (*angl.* compliance tensor) in ima v splošnem $3^4 = 81$ prostih parametrov.

Opomba 2.23. Aksiom 5 se po komponentah glasi $\sigma_{ij} = C_{ijkl}\varepsilon_{kl}$

Toda, iz trditve 2.13 o simetričnosti σ sledi, da lahko prosto zamenjamo indeksa i in j in velja $C_{ijkl} = C_{jikl}$. Podobno iz simetričnosti ε sledi, da je $C_{ijkl} = C_{ijlk}$. Še ena simetrija in potem izotropičnost.

2.5 Navierova enačba

blah blah

Definicija 2.24. Laplacev operator je na vektorskem polju \vec{v} definiran kot

$$\text{lap } \vec{v} = \text{div grad } \vec{v}.$$

V izotropičnem materialu velja med napetostjo in deformacijo relacija

$$\sigma = \lambda \operatorname{tr}(\varepsilon)I + 2\mu\varepsilon.$$

Sedaj vstavimo definicijo ε (5) v to zvezo, to pa vstavimo v poenostavljeno gibalno enačbo (4) in dobimo končen rezultat.

Izrek 2.25 (Navierova enačba). *Če so pomiki in njihovi gradienti v linearno elastičnem izotropičnem homogenem nepolarnem mediju majhni, potem za njih velja Navierova enačba*

$$\rho\ddot{u} = \vec{f} + (\lambda + \mu) \operatorname{grad} \operatorname{div} u + \mu \operatorname{lap} u. \quad (6)$$

Dokaz. Od enačbe (4)

$$\rho\ddot{u} = \vec{f} + \operatorname{div} \sigma$$

do Navierove enačbe nas loči le še izračun $\operatorname{div} \sigma$. Za primerjavo naredimo izračun koordinatno in nekoordinatno.

$$\begin{aligned} \varepsilon_{ij} &= \frac{1}{2}(\operatorname{grad} u)_{ij} + \frac{1}{2}(\operatorname{grad} u^\top)_{ij} = \frac{1}{2}u_{i,j} + \frac{1}{2}u_{j,i} \\ \sigma_{ij} &= \lambda\varepsilon_{kk}\delta_{ij} + 2\mu\varepsilon_{ij} = \lambda\frac{1}{2}(u_{k,k} + u_{k,k})\delta_{ij} + 2\mu(\frac{1}{2}u_{i,j} + \frac{1}{2}u_{j,i}) = \\ &= \lambda u_{k,k}\delta_{ij} + \mu(u_{i,j} + u_{j,i}) \\ (\operatorname{div} \sigma)_i &= \sigma_{ij,j} = \lambda u_{k,kj}\delta_{ij} + \mu(u_{i,jj} + u_{j,ij}) = \lambda u_{k,ki} + \mu(u_{i,kk} + u_{k,ki}) = \\ &= (\lambda + \mu)u_{k,ki} + \mu u_{i,kk} = (\lambda + \mu)(\operatorname{grad}(u_{k,k}))_i + \mu(\operatorname{lap} u)_i = \\ &= ((\lambda + \mu) \operatorname{grad} \operatorname{div} u + \mu \operatorname{lap} u)_i \end{aligned}$$

Nekoordinatni dokaz pa potrebuje nekaj dodatnih relacij, ki jih je tudi najhitreje dokazati koordinatno.

Lema 2.26.

$$\operatorname{tr} \operatorname{grad} \vec{v} = \operatorname{div} \vec{v}$$

Dokaz.

$$\operatorname{tr} \operatorname{grad} \vec{v} = (\operatorname{grad} \vec{v})_{ii} = \vec{v}_{i,i} = \operatorname{div} \vec{v} \quad \square$$

Lema 2.27.

$$\operatorname{div}(\operatorname{grad} \vec{v}^\top) = \operatorname{grad} \operatorname{div} \vec{v}$$

Dokaz.

$$\operatorname{div}(\operatorname{grad} \vec{v}^\top)_i = (\operatorname{grad} \vec{v}^\top)_{ij,j} = (\operatorname{grad} \vec{v})_{ji,j} = \vec{v}_{j,ij} = (\operatorname{grad}(\vec{v}_{j,j}))_i = (\operatorname{grad} \operatorname{div} \vec{v})_i \quad \square$$

Lema 2.28.

$$\operatorname{div}(\varphi I) = \operatorname{grad} \varphi$$

Dokaz.

$$(\operatorname{div}(\varphi I))_i = (\varphi I)_{ij,j} = (\varphi \delta_{ij})_{,j} = \varphi_{,i} = (\operatorname{grad} \varphi)_i \quad \square$$

Sedaj lahko izračunamo $\operatorname{div} \sigma$.

$$\begin{aligned}
\operatorname{div} \sigma &= \lambda \operatorname{div} \operatorname{tr}(\varepsilon) I + 2\mu \operatorname{div} \varepsilon = \\
&= \lambda \frac{1}{2} (\operatorname{div}((\operatorname{tr} \operatorname{grad} u) I) + \operatorname{div} \operatorname{tr} \operatorname{grad} u^T) + \mu \operatorname{div}(\operatorname{grad} u + \operatorname{grad} u^T) = \\
&= \frac{\lambda}{2} (\operatorname{div}((\operatorname{div} u) I) + \operatorname{div}((\operatorname{div} u) I)) + \mu \operatorname{lap} u + \mu \operatorname{grad} \operatorname{div} u = \\
&= \lambda \operatorname{grad} \operatorname{div} u + \mu \operatorname{grad} \operatorname{div} u + \operatorname{lap} u
\end{aligned}$$

Izračun prinese enak rezultat kot prej in izrek je še enkrat dokazan. \square

2.6 Obstoje in enoličnost rešitve

Izrek povzet po [1, izrek 3.15.1, str. 226] Prevedemo na Riezsov izrek.

2.7 Priprava na numerično reševanje

Za numerično reševanje se postavimo v kartezični koordinatni sistem. Komponente tenzorja σ razpišimo:

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{bmatrix},$$

pri čemer smo že upoštevali simetrijo. Reševati tridimenzionalne enačbe je težje kot dvodimenzionalne, zato si oglejmo dve poenostavitvi.

2.7.1 Poenostavitev na dve dimenziji

Tridimenzionalne probleme se zavoljo lažje formulacije, manjše računske zahtevnosti in lažje implementacije pogosto s pomočjo neke simetrije prevede na nižjedimenzionalne. Spodaj sta opisani dve najbolj pogosti poenostavitvi.

Ravninska napetost: Pri tej poenostavitvi predpostavimo, da napetost nima ničelnih komponent v smeri ene izmed koordinatnih osi, torej, po primerni rotaciji koordinatnega sistema, da so komponente σ_{xz} , σ_{yz} in σ_{zz} enake 0. Ta poenostavitev je primerna za telesa, ki so “tanke plošče”, torej, imajo eno dimenzijo veliko manjšo od ostalih dveh. Poleg tega morajo biti vse obremenitve vzdolž plošče.

Ravninska deformacija: Pri tej poenostavitvi predpostavimo, da deformacijski tenzor ε nima komponent v eni izmed koordinatnih smeri, torej, podobno kot pri ravninski napetosti lahko predpostavimo, da so ε_{xz} , ε_{yz} in ε_{zz} enaki 0. Ta poenostavitev je primerna za telesa, ki imajo eno dimenzijo mnogo večjo od drugih iz se vzdolž nje ne spreminjajo, torej za (ne nujno krožne) valje.

Pri obeh poenostavitvah se iz pogojev izpelje, da v Navierovi enačbi ne nastopa tretja komponenta u . Še več poenostavitvi sta celo ekvivalentni, saj se razlikujeta le za konstantni člen pred enim izmed faktorjev. V dveh dimenzijah pišemo prvo komponento pomika z u in drugo z v , koordinati pa označujemo z x in y . Če si

razpišemo izraze za deformacijski tenzor in napetostni tenzor za primer ravninske napetosti dobimo:

$$\varepsilon = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) \\ \frac{1}{2}(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \frac{\partial v}{\partial y} \end{bmatrix}$$

$$\sigma = \begin{bmatrix} \lambda \frac{\partial v}{\partial y} + (\lambda + 2\mu) \frac{\partial u}{\partial x} & \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) \\ \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \lambda \frac{\partial u}{\partial x} + (\lambda + 2\mu) \frac{\partial v}{\partial y} \end{bmatrix}.$$

2.7.2 Robni pogoji

Pri reševanju robnih problemov bomo imeli tri vrste robnih pogojev. Dirichletove robne pogoje $u|_{\partial\Omega} = u_0$ bomo uporabili, ko bomo poznali pomike na robu. Najpogostejše bo pogoj oblike $u = 0$, ko nek konec držimo fiksen. Pri drugi vrsti robnih pogojev poznamo napetosti na robovih. Najpogostejše bomo podali kar napetost na robu $\vec{t} = \vec{t}_0$, kjer je \vec{t}_0 neka znana vrednost, npr. 0, če je ta rob prost. Tretja vrsta pogojev bodo simetrijski pogoji, ko bomo problem reducirali preko neke osi simetrije, na osi pa bomo zahtevali kompatibilnostne pogoje.

Oglejmo si bolj natančno pogoje, podane z napetostjo. Napetost na robu izračunamo preko napetostnega tenzorja, $\vec{t} = \sigma \vec{n}$, kjer ne \vec{n} zunanja normala na rob. Za primer desnega roba, na katerega enakomerno deluje neka gostota sile p v normalni smeri dobimo dva pogoja:

$$\vec{t} = \sigma \vec{n} = \begin{bmatrix} \lambda \frac{\partial v}{\partial y} + (\lambda + 2\mu) \frac{\partial u}{\partial x} & \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) \\ \mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) & \lambda \frac{\partial u}{\partial x} + (\lambda + 2\mu) \frac{\partial v}{\partial y} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} p \\ 0 \end{bmatrix}.$$

3 Numerična metoda

V 20. stoletju je skupaj z razvojem računalnikov začel svojo pot razvoj numeričnih metod za reševanje parcialnih diferencialnih enačb. Do danes je bilo razvitih veliko metod za numerično reševanje parcialnih diferencialnih enačb. Dva pomembna razreda metod se ločita glede na obliko v kateri rešujemo parcialno diferencialno enačbo: šibki (*angl.* weak form) ali močni (*angl.* strong form). Najznamenitejši in zelo uspešen predstavnik prve skupine je metoda končnih elementov (MKE) (*angl.* finite element method (FEM)), kjer problem najprej prevedemo v šibko obliko, nato pa rešitev poiščemo kot linearno kombinacijo baznih funkcij iz izbranega prostora. Najbolj poznan, predstavnik druge pa je metoda končnih diferenc (MKD) (*angl.* finite difference method (FDM)), pri kateri direktno diskretiziramo operator, ki nastopa v enačbi.

Poleg tega se metode delijo tudi, glede na tip diskretizacije domene, ki ga potrebujejo. Metoda končnih elementov potrebuje *mrežo*, nad katero deluje, tj. triangulacijo notranjosti domene, ki inducira tudi mrežo na robu. Metoda robnih elementov potrebuje samo mrežo na robu domene. Metoda končnih diferenc je običajno formulirana na pravokotni mreži. Obstajajo pa tudi metode, ki mreže ne potrebujejo, imenujemo jih *brezmrežne metode* (*angl.* meshfree methods). Predstavljena metoda v tem razdelku, bo reševala enačbo v močni obliki in bo brezmrežna.

3.1 Izpeljava

Izpeljavo začnimo z osvežitvijo spomina na metodo končnih diferenc, ki nam bo služila kot motivacija.

3.1.1 Ideja in motivacija

Primer 3.1. Rešujemo enodimenzionalno Poissonovo enačbo. Izpeljava metode končnih diferenc ne bo povsem običajna in tudi ne najkrajša možna, ampak bo narejena tako, da jo bomo lahko splošili v brez mrežno metodo.

Rešujemo problem z mešanimi robnimi pogoji

$$\begin{aligned} u''(x) &= f(x) & \text{na } (a, b) \\ u(a) &= A \\ u'(b) &= B, \end{aligned} \tag{7}$$

katerega rešitev poznamo v kvadraturah

$$u(x) = \int_a^x \left(\int_b^\eta f(\xi) d\xi \right) d\eta + B(x - a) + A.$$

Numeričnega reševanja se lotimo tako, da interval $[a, b]$ diskretiziramo na N enakih delov dolžine $h = \frac{b-a}{N}$ z delilnimi točkami $x_i = a + ih$, za $i = 0, \dots, N$. Za vsako od teh točk uvedemo neznanko u_i , ki predstavlja neznano funkcijsko vrednost v točki x_i . S pomočjo vrednosti u_{i-1} , u_i in u_{i+1} želimo sedaj aproksimirati $u''(x_i)$. To nam bo dalo zvezo med spremenljivkami in ko jo uporabimo za vse notranje točke ter upoštevamo še robne pogoje, bomo dobili sistem linearnih enačb, katerega rešitev nam bo dala dobro aproksimacijo funkcije u .

Funkcijo u v okolici x_i aproksimiramo z interpolacijskim polinomom, njene odvode pa z odvodi interpolacijskega polinoma. Da najdemo interpolacijski polinom \hat{u} , zapišimo

$$\hat{u}(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \mathbf{b}(x)^\top \boldsymbol{\alpha}$$

in poiščimo koeficiente $\boldsymbol{\alpha}$, da bo veljalo

$$\begin{aligned} \hat{u}(x_{i-1}) &= u_{i-1} \\ \hat{u}(x_i) &= u_i \\ \hat{u}(x_{i+1}) &= u_{i+1}. \end{aligned}$$

Če sistem enačb razpišemo, dobimo

$$\begin{aligned} \alpha_0 + \alpha_1(x_i - h) + \alpha_2(x_i - h)^2 &= u_{i-1} \\ \alpha_0 + \alpha_1 x_i + \alpha_2 x_i^2 &= u_i \\ \alpha_0 + \alpha_1(x_i + h) + \alpha_2(x_i + h)^2 &= u_{i+1} \end{aligned}$$

oziroma v matrični obliki

$$\begin{bmatrix} 1 & x_i - h & (x_i - h)^2 \\ 1 & x_i & x_i^2 \\ 1 & x_i + h & (x_i + h)^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} u_{i-1} \\ u_i \\ u_{i+1} \end{bmatrix}.$$

Krajše ga zapišemo kar kot $B\boldsymbol{\alpha} = \mathbf{u}$. Sistem rešimo in dobimo

$$\begin{aligned} \alpha_0 &= \frac{2h^2 u_i + h(u_{i-1} - u_{i+1})x_i + (u_{i-1} - 2u_i + u_{i+1})x_i^2}{2h^2} \\ \alpha_1 &= \frac{h(u_{i+1} - u_{i-1}) - 2(u_{i-1} - 2u_i + u_{i+1})x_i}{2h^2} \\ \alpha_2 &= \frac{u_{i-1} - 2u_i + u_{i+1}}{2h^2}. \end{aligned}$$

Interpolacijski polinom skozi točke (x_i, u_i) lahko sedaj zapišemo kot

$$\begin{aligned} \hat{u}(x) &= \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \\ &= u_i + \frac{u_{i+1} - u_{i-1}}{2h}(x - x_i) + \frac{u_{i-1} - 2u_i + u_{i+1}}{2h^2}(x - x_i)^2 \end{aligned}$$

Toda, ker u_j nastopajo linearno, lahko napišemo tudi v obliki

$$\hat{u}(x) = \begin{bmatrix} \frac{(x_i - x)(h + x_i - x)}{2h^2} & \frac{(h + x - x_i)(h + x_i - x)}{h^2} & \frac{(x - x_i)(h + x - x_i)}{2h^2} \end{bmatrix} \begin{bmatrix} u_{i-1} \\ u_i \\ u_{i+1} \end{bmatrix} = \boldsymbol{\varphi}(x)^\top \mathbf{u}.$$

S tem smo ločili podatke, ki se nanašajo na vrednost funkcije, od podatkov, ki se nanašajo na pozicije točk. Če na primer vemo, da bomo večkrat potrebovali vrednost interpolacijskega polinoma v neki točki x^* za različne nabore funkcijskih vrednosti (vendar še vedno izmerjene v istih točkah) \mathbf{u} , potem se nam splača poračunati $\boldsymbol{\varphi}(x^*)$ vnaprej in vrednosti interpolacijskega polinoma dobimo vsakič znova le s skalarnim produktom $\hat{u}(x^*) = \boldsymbol{\varphi}(x^*)^\top \mathbf{u}$.

Za aproksimacijo u' in u'' bomo vzeli kar odvode \hat{u} . Izračunajmo jih v točki x_i in dobimo znane formule

$$\begin{aligned} \hat{u}'(x_i) &= \boldsymbol{\varphi}'(x_i)^\top \mathbf{u} = \begin{bmatrix} -\frac{1}{2h} & 0 & \frac{1}{2h} \end{bmatrix} \begin{bmatrix} u_{i-1} \\ u_i \\ u_{i+1} \end{bmatrix} \\ \hat{u}''(x_i) &= \boldsymbol{\varphi}''(x_i)^\top \mathbf{u} = \begin{bmatrix} \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \end{bmatrix} \begin{bmatrix} u_{i-1} \\ u_i \\ u_{i+1} \end{bmatrix}. \end{aligned}$$

To lahko uporabimo za reševanje našega problema (7). Namesto enakosti

$$u''(x_i) = f(x_i)$$

za vsako točko x_i v notranjosti naredimo zapišemo podobno enakost

$$\begin{bmatrix} \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \end{bmatrix} \begin{bmatrix} u_{i-1} \\ u_i \\ u_{i+1} \end{bmatrix} = f(x_i).$$

Dirichletov pogoj na levem robu zapišemo preprosto kot $u_0 = A$, za Neumannovega na desnem robu pa lahko uporabimo npr. enostransko diferenco na treh točkah

$$\begin{bmatrix} \frac{1}{2h} & \frac{-2}{h} & \frac{3}{2h} \end{bmatrix} \begin{bmatrix} u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix} = B,$$

ki bi jo izpeljali na enak način.

Vse te enakosti zložimo sistem enačb in ga zapišimo v matrični obliki

$$\begin{bmatrix} 1 & & & & & \\ -1 & 2 & 1 & & & \\ & -1 & 2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & 1 \\ & & & 1/2h & -2/h & 3/2h \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix}.$$

Rešitev tega sistema nam dobro aproksimira neznano funkcijo u v izbranih točkah x_i .

3.1.2 Splošna izpeljava

Postavimo se sedaj v splošnejši okvir. Rešujemo parcialno diferencialno enačbo

$$\begin{aligned} \mathcal{L}u &= f \text{ na } \Omega, \\ \mathcal{R}u &= g \text{ na } \partial\Omega, \end{aligned} \tag{8}$$

kjer je $\Omega \subseteq \mathbb{R}^d$ omejena domena, torej, omejena povezana odprta množica z odsekom gladkim robom, $u \in C^r(\mathbb{R}^d)$ funkcija, $\mathcal{L}: C^r(\mathbb{R}^d) \rightarrow C(\mathbb{R})$ linearen parcialni diferencialni operator reda r in $\mathcal{R}u$ robni pogoji, pri katerih je problem enolično rešljiv.

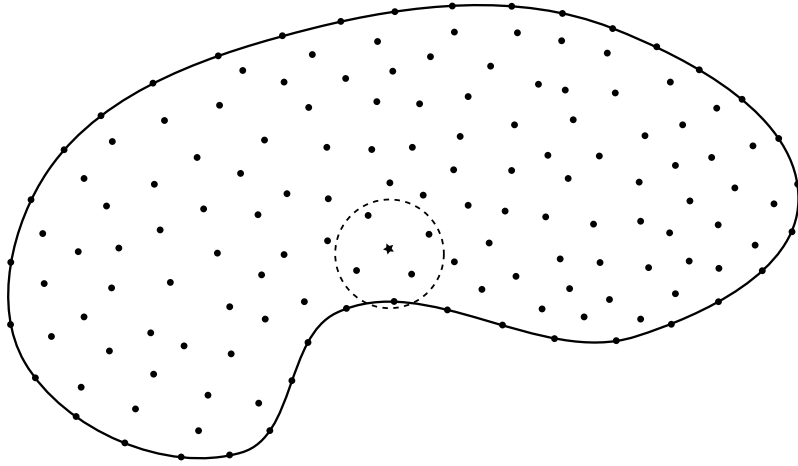
Domeno in njen rob sedaj diskretiziramo, tako da izberemo N točk v zaprtju domene, $x_1, \dots, x_N \in \overline{\Omega}$. Podobno kot pri končnih diferencah bomo v teh točkah aproksimirali vrednost funkcije u . Izberimo fiksno točko $p \in \overline{\Omega}$ in n izmed točk $\{x_1, \dots, x_N\}$, ki bodo sestavljali *sosesčino* (*angl.* support) točke p . Število n imenujemo velikost sosesčine. Označimo z $\mathcal{N}(p)$ sosesčino točke p in z $\mathcal{I}(p) = \{i_1, \dots, i_n\}$ množico indeksov, za katere so izbrani x_{i_j} v sosesčini p . Velja torej

$$\mathcal{N}(p) = \bigcup_{i \in \mathcal{I}(p)} x_i.$$

Običajno bo $n \ll N$, npr. $n = 9$ in $N = 10^6$. Primer domene Ω , točke p in njene sosesčine je prikazan na sliki 4.

V okolici točke p aproksimirajmo u z elementi iz nekega končno dimenzionalnega prostora funkcij $\mathcal{B} = \text{Lin}\{b_1, \dots, b_m\}$. Funkcijam $b_i: \mathbb{R}^d \rightarrow \mathbb{R}$ pravimo *bazne funkcije*, številu m pa moč baze. Aproksimacijo za \hat{u} za u lahko torej zapišemo kot

$$u \approx \hat{u} = \sum_{i=1}^m \alpha_i b_i = \mathbf{b}^\top \boldsymbol{\alpha},$$



Slika 4: Primer domene z diskretizirano notranjostjo in robom skupaj z izbrano točko in njeno sosesčino.

pri čemer smo z $\boldsymbol{\alpha} = (\alpha_i)_{i=1}^m$ označili vektor neznanih koeficientov in $\mathbf{b} = (b_i)_{i=1}^m$ vektor baznih funkcij.

Če bi poznali vrednosti $u(x_i)$ za $i \in I(p)$, potem bi lahko aproksimirano \hat{u} izračunali po metodi najmanjših kvadratov. Ker pa teh vrednosti ne poznamo, uvedimo spremenljivke u_i za vsako točko v domeni, ki nam bodo predstavljale neznane prave vrednosti in nadaljujmo s simbolnim računanjem. Za funkcijo \hat{u} zahtevamo, da aproksimira u v smislu utežene diskretne 2-norme, torej da minimizira

$$\|u - \hat{u}\|_{2,N(p),w} = \sum_{i \in I(p)} w(p - x_i)(u_i - \hat{u}(x_i))^2$$

kar je utežena vsota kvadratov odstopanj od pravih vrednosti. Pri tem je $w: \mathbb{R}^d \rightarrow \mathbb{R}$ nenegativna funkcija, ki jo imenujemo *utež*, \mathbf{w} pa je vektor sestavljen iz vrednosti te funkcije v točkah v sosesčini.

Matrično lahko zapišemo sistem enačb, ki po vrsticah postavlja našo zahtevo $\hat{u}(x_j) = u_j$, za vsak $j \in I(p)$:

$$\begin{bmatrix} b_1(x_{i_1}) & \cdots & b_m(x_{i_1}) \\ \vdots & \ddots & \vdots \\ b_1(x_{i_n}) & \cdots & b_m(x_{i_n}) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix} = \begin{bmatrix} u_{i_1} \\ \vdots \\ u_{i_n} \end{bmatrix}. \quad (9)$$

Na krajše sistem zapišemo kot $B\boldsymbol{\alpha} = \tilde{\mathbf{u}}$. Odvisno od n , m , b_i in uteži w je ta sistem lahko poddoločen, predoločen ali običajen. V vsakem primeru lahko minimiziranje utežene napake prevedemo na minimiziranje diskretne 2-norme $\|WB\boldsymbol{\alpha} - W\tilde{\mathbf{u}}\|_{2,N(p)}$, kjer je W diagonalna matrika iz korenov uteži za posamezne točke, $W = \text{diag}(\sqrt{w(x_{i_1} - p)}, \dots, \sqrt{w(x_{i_n} - p)})$. Tak sistem pa lahko ne glede na njegovo določenost v rešimo s pomočjo Moore-Penroseovega psevdoinverza, ki ga lahko izračunamo s pomočjo singularnega razcepa matrike WB . Tako lahko izrazimo

$$\boldsymbol{\alpha} = (WB)^+ W\tilde{\mathbf{u}},$$

kjer $+$ označuje Moore-Penroseov psevdoinverz.

To lahko vstavimo nazaj v izraz za \hat{u} in dobimo

$$\hat{u} = \mathbf{b}^\top \boldsymbol{\alpha} = \mathbf{b}^\top (WB)^+ W \tilde{\mathbf{u}}.$$

Sedaj lahko za izbrano točko p izračunamo

$$\hat{u}(p) = \underbrace{\mathbf{b}(p)^\top (WB)^+ W}_{\boldsymbol{\varphi}_p} \tilde{\mathbf{u}}.$$

Izračunljivi kos $\boldsymbol{\varphi}_p$ je v praksi vrstica velikosti n , matematično pa je linearen funkcional $\boldsymbol{\varphi}_p \in (\mathbb{R}^n)^*$, ki naboru funkcijskih vrednosti v soseščini $N(p)$ priredi aproksimacijo za funkcijsko vrednost v točki p .

Podobno kot pri deljenih diferencah odvede funkcije u aproksimiramo z odvodi aproksimacijskega polinoma skozi točke v soseščini, bomo tudi v našem primeru aproksimirali odvode funkcije u z odvodi \hat{u} ,

$$(\mathcal{L}u)(p) \approx (\mathcal{L}\hat{u})(p) = (\mathcal{L}\mathbf{b})(p)^\top (WB)^+ W \tilde{\mathbf{u}}$$

od koder kot prej definiramo

$$\boldsymbol{\varphi}_{\mathcal{L},p} = (\mathcal{L}\mathbf{b})(p)^\top (WB)^+ W \quad (10)$$

Funkcional $\boldsymbol{\varphi}_{\mathcal{L},p}$ je aproksimacija operatorja \mathcal{L} v točki p . Pogosto se ga imenuje tudi *funkcija oblike* (*angl.* shape function), saj v sebi nosi podatke o lokalni obliki domene in izboru okoliških točk, ter seveda o obnašanju \mathcal{L} v tej okolici. Tudi če funkcijskih vrednosti $\tilde{\mathbf{u}}$ v okolici p ne poznamo, lahko $\boldsymbol{\varphi}_{\mathcal{L},p}$ izračunamo in kasneje samo s skalar-nim produktom dobimo aproksimacijo za $(\mathcal{L}u)(p)$. Lahko pa to izkoristimo za zapis linearne enačbe

$$\boldsymbol{\varphi}_{\mathcal{L},p} \cdot \tilde{\mathbf{u}} = f(p),$$

ki je direktna aproksimacija diferencialne enačbe (8) v točki p ,

$$(\mathcal{L}u)(p) = f(p).$$

To lahko sedaj storimo za vsako diskretizacijsko točko x_i v domeni in dobimo sistem enačb

$$\begin{aligned} \boldsymbol{\varphi}_{\mathcal{L},x_i} \cdot \tilde{\mathbf{u}} &= f(x_i), \text{ za vsak } i, \text{ tak da je } x_i \in \Omega \\ \boldsymbol{\varphi}_{\mathcal{R},x_i} \cdot \tilde{\mathbf{u}} &= g(x_i), \text{ za vsak } i, \text{ tak da je } x_i \in \partial\Omega. \end{aligned}$$

Te enačbe lahko zapišemo v matrični sistem

$$A\mathbf{u} = \mathbf{f}, \quad (11)$$

kjer ima matrika A v vrsticah zapisane funkcionalne $\boldsymbol{\varphi}_{\mathcal{L},x_i}$, tako da so neničelni elementi na tistih mestih, ki se pomnožijo z neznankami, ki ustrezajo sosedom x_i . Natančneje, elementi matrike A so

$$\begin{aligned} A(k, i_j) &= \boldsymbol{\varphi}_{\mathcal{L},p}(j), \text{ za vsak } k, \text{ tak da je } x_k \in \Omega \text{ in za vsak } i_j \in \mathcal{I}(x_k), \\ A(k, i_j) &= \boldsymbol{\varphi}_{\mathcal{R},p}(j), \text{ za vsak } k, \text{ tak da je } x_k \in \partial\Omega \text{ in za vsak } i_j \in \mathcal{I}(x_k). \end{aligned}$$

Razumljivejša je morda kar Matlab-ova notacija

$$A(k, I(x_k)) = \begin{cases} \varphi_{\mathcal{L},p} & x_k \in \Omega \\ \varphi_{\mathcal{R},p} & x_k \in \partial\Omega \end{cases}, \text{ za } k = 1, \dots, N.$$

Vektor $\mathbf{u} = (u_i)_{i=1}^N$ je vektor neznanih funkcijskih vrednosti, ki ga iščemo, v vektorju \mathbf{f} pa so zapisani robni pogoji

$$\mathbf{f}(k) = \begin{cases} f(x_k) & x_k \in \Omega \\ g(x_k) & x_k \in \partial\Omega \end{cases}, \text{ za } k = 1, \dots, N.$$

Vidimo, da je matrika A razpršena. Sama je dimenzij $N \times N$, v vsaki vrstici pa ima največ n neničelnih elementov, torej je skupno število neničelnih elementov

$$\text{nnz}(A) \leq nN.$$

Enakost je lahko dosežena, lahko pa je tudi stroga, saj so kakšni koeficienti v $\varphi_{\mathcal{L},x_i}$ lahko tudi 0, kot se to zgodi pri Dirichletovih robnih pogojih.

Sistem (11) nato rešimo in za aproksimacijo $u(x_i)$ vzamemo $u(x_i) \approx u_i$.

3.2 Posebni primeri

Metoda iz razdelka 3.1.2 je formulirana precej splošno, in v posebnih primerih lahko prepoznamo druge znane metode. Za začetek pokažimo enostavno trditev, ki bo metodo v določenih primerih poenostavila.

Trditev 3.2. Če je $m = n$ in je matrika B iz sistema (9) obrnljiva, potem je izbira uteži nepomembna.

Dokaz. Spomnimo se, da je

$$\varphi = (\mathcal{L}\mathbf{b})(p)^\top (WB)^+ W.$$

Matrika W je diagonalna s sami pozitivnimi števili na diagonali in je torej obrnljiva. Po predpostavki je obrnljiva tudi matrika B , torej je obrnljiv tudi produkt WB in velja

$$(WB)^+ = (WB)^{-1} = B^{-1}W^{-1}.$$

Od tod sledi, da je

$$\varphi = (\mathcal{L}\mathbf{b})(p)^\top B^{-1}W^{-1}W = (\mathcal{L}\mathbf{b})(p)^\top B^{-1},$$

kar je neodvisno od W . □

Opomba 3.3. Čeprav trditev 3.2 velja v eksaktni aritmetiki v praksi ne velja nujno. Če so izbrane uteži zelo majhne ali zelo različnih velikosti lahko to povzroči nepotrebne numerične nestabilnosti.

V praksi trditev kljub temu enostavno pomeni, da če izberemo toliko baznih funkcij kot imamo sosedov, je izbira uteži nepomembna.

Že v primeru 3.1 smo videli, da se za Laplaceev robni problem na intervalu naša metoda ujema z metodo končnih diferenc. Pokažimo to tudi v primer dvodimenzionalne Poissonove enačbe.

Trditev 3.4. *Metoda iz razdelka 3.1.2 se za reševanje Poissonove enačbe na enakomerni pravokotni mreži z razmakom h ujema z metodo končnih diferenc, če vzamemo $\mathbf{b} = \{1, x, y, x^2, y^2\}$, $w \equiv 1$ in $n = 5$.*

Dokaz. Kot ponavadi označimo točke na mreži s koordinatami (x_i, y_j) , tako da je $x_{i+1} = x_i + h$ in $y_{j+1} = y_j + h$. Pokazati moramo, da se funkcija oblike v vsaki točki x ujema z aproksimacijo končnih diferenc, ki pravi

$$(\Delta u)(x_i, y_j) = \frac{u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{i,j}}{h^2},$$

pri čemer spremenljivka $u_{i,j}$ pripada koordinati (x_i, y_j) . Če se aproksimaciji Laplaceovega operatorja ujemata, potem se ujemati tudi aproksimaciji rešitve, saj sistem pri obeh metodah gradimo na enak način. Ker je operator krajevno neodvisen so tudi funkcije oblike odvisne samo od medsebojne lege točk in torej lahko brez škode za splošnost predpostavimo, da računamo funkcijo oblike za točko $p = (0, 0)$. Najbližjih n sosedov je tako $N(p) = \{(0, 0), (0, h), (0, -h), (h, 0), (-h, 0)\}$. Matrika $B = [b_j(x_i)]$ iz sistema (9), ki ima po stolpic evalvirane bazne funkcije v vseh sosedih je

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -h & 0 & h^2 \\ 1 & 0 & h & 0 & h^2 \\ 1 & h & 0 & h^2 & 0 \\ 1 & -h & 0 & h^2 & 0 \end{bmatrix}$$

in njen psevdoinverz je

$$B^+ = B^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2h} & -\frac{1}{2h} \\ 0 & -\frac{1}{2h} & \frac{1}{2h} & 0 & 0 \\ -\frac{1}{h^2} & 0 & 0 & \frac{1}{2h^2} & \frac{1}{2h^2} \\ -\frac{1}{h^2} & \frac{1}{2h^2} & \frac{1}{2h^2} & 0 & 0 \end{bmatrix}.$$

Pri tem smo že upoštevali $w \equiv 1$. Vektor vrednosti operatorja na baznih funkcijah je

$$(\Delta \mathbf{b})(p) = [0 \ 0 \ 0 \ 2 \ 2]^\top.$$

Od tod dobimo po zvezi (10)

$$\boldsymbol{\varphi} = (\Delta \mathbf{b})(p)^\top B^+ = \left[-\frac{4}{h^2} \quad \frac{1}{h^2} \quad \frac{1}{h^2} \quad \frac{1}{h^2} \quad \frac{1}{h^2} \right],$$

kar je ravno iskana aproksimacija. □

Opomba 3.5. Metodo za izračun funkcije oblike je zelo elegantno implementirati v Mathematici, kar olajša simbolno raziskovanje takih aproksimacij. Zgornje matrike so bile izračunane s programa 1.

```

1 (* podatki *)
2 p = {0, 0};
3 sosed = {p, {0, -h}, {0, h}, {h, 0}, {-h, 0}};
4 bazne = {{1 &}, {#1 &}, {#2 &}, {#1^2 &}, {#2^2 &}};
5 operator = Function[f, Function[{x, y}, Derivative[2, 0][f][x, y] + Derivative[0, 2][f][x, y]]];
6 (* izračun *)
7 B = Table[Table[b @@ s, {b, bazne}], {s, sosed}];
8 Lb = Table[operator[b] @@ p, {b, bazne}];
9 phi = Lb.PseudoInverse[B] // Simplify

```

Program 1: Računanje funkcij oblike na pravokotni mreži.

Opomba 3.6. Tudi če izberemo $n = 9$ in $\mathbf{b} = \{1, x, x^2, y, xy, xy^2, y^2, y^2x, y^2x^2\}$, dobimo enako aproksimacijo kot v trditvi 3.4, le da uporabimo več sosedov. Če jih uredimo po oddaljenosti od $(0, 0)$ dobimo aproksimacijo

$$\varphi = \begin{bmatrix} -\frac{4}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} & \frac{1}{h^2} & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Če pa bi izbrali $n = 25$ točk in bazne funkcije $\mathbf{b} = \{x^i y^j, 0 \leq i, j \leq 4\}$, potem bi dobili aproksimacijo četrtega reda.

Primer 3.7. Naredimo še en primer, kjer za bazne funkcije vzamemo monome. Vzemimo tokrat $n = 9$ in za bazne funkcije vse monome skupne stopnje manj kot 3, torej $\mathbf{b} = \{1, x, y, x^2, y^2, xy\}$. V tem primeru je matrika B velikosti 9×6 in s pomočjo programa 1 dobimo

$$\varphi = \begin{bmatrix} -\frac{4}{3h^2} & -\frac{1}{3h^2} & -\frac{1}{3h^2} & -\frac{1}{3h^2} & -\frac{1}{3h^2} & \frac{2}{3h^2} & \frac{2}{3h^2} & \frac{2}{3h^2} & \frac{2}{3h^2} \end{bmatrix}.$$

Vidimo, da tokrat v aproksimaciji upoštevamo vseh 9 sosedov, z razliko od aproksimacije v opombi 3.6. Naravno sledi vprašanje, ali je ta aproksimacija kaj slabša, morda drugačnega reda? Izkaže se da ne, saj z razvojem aproksimacij v Taylorjevo vrsto dobimo

$$\begin{aligned}
& \frac{-4u(0, 0) + u(0, h) + u(0, -h) + u(h, 0) + u(-h, 0)}{h^2} = \\
& = \Delta u + \frac{h^2}{12} \left(\frac{\partial^4 u}{\partial x^4}(0, 0) + \frac{\partial^4 u}{\partial y^4}(0, 0) \right) + O(h^4) \\
& \frac{-4u(0, 0) - u(0, h) - u(0, -h) - u(h, 0) - u(-h, 0) + 2u(h, h) + 2u(h, -h) + 2u(-h, h) + 2u(-h, -h)}{3h^2} = \\
& = \Delta u + \frac{h^2}{12} \left(\frac{\partial^4 u}{\partial x^4}(0, 0) + \frac{\partial^4 u}{\partial x^2 \partial y^2}(0, 0) + \frac{\partial^4 u}{\partial y^4}(0, 0) \right) + O(h^4).
\end{aligned}$$

Obe aproksimaciji sta torej drugega reda, razlikujeta se le pri izražavi napake.

Primer 3.8. Poglejmo še, kaj se zgodi, če za bazne funkcije vzamemo radialne bazne funkcije. V tem primeru je smiselno vzeti $n = m$, torej postavimo po eno radialno funkcijo na vsakega izmed sosedov. Opisana metoda tako postane kolokacijska metoda z lokalnimi radialnimi baznimi funkcijami (*angl.* local radial basis function collocation method).

Če za $n = 9$ sosedov točke $(0, 0)$ izberemo točke

$$N(p) = \{(0, 0), (0, h), (0, -h), (h, 0), (-h, 0), (h, h), (h, -h), (-h, h), (-h, -h)\}$$

in za \mathbf{b} vzamemo

$$\mathbf{b} = \{x \mapsto \exp((x - c)^2 / \sigma^2); c \in N(p)\},$$

potem zopet s pomočjo programa 1 izračunamo aproksimacijo

$$\varphi = \frac{4}{(e^{\frac{2h^2}{\sigma^2}} - 1)^2 \sigma^4} \begin{bmatrix} (e^{\frac{2h^2}{\sigma^2}} - 1)^2 - 4h^2 e^{\frac{2h^2}{\sigma^2}} & h^2 e^{\frac{h^2}{\sigma^2}} & h^2 e^{\frac{h^2}{\sigma^2}} & h^2 e^{\frac{h^2}{\sigma^2}} & h^2 e^{\frac{h^2}{\sigma^2}} & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Tokrat vidimo, da se precej razlikuje od aproksimacije s končnimi diferencami. Toda, še vedno je drugega reda, saj za napako velja

$$\varphi \cdot \tilde{u} - \Delta u(0, 0) = h^2 \left(\frac{2u(0, 0)}{\sigma^2} - \frac{\Delta u(0, 0)}{\sigma^2} + \frac{1}{12} \left(\frac{\partial^4 u}{\partial x^4}(0, 0) + \frac{\partial^4 u}{\partial y^4}(0, 0) \right) \right) + O(h^4).$$

3.3 Algoritem

V izpeljavi metode v razdelku 3.1.2 je veliko detajlov ostalo neizdelanih, kot na primer, kako diskretiziramo domeno ali kako poiščemo sosede. Ti detajli so opisani v naslednjih podrazdelkih, celotno metodo pa podajamo v psevdokodi kot algoritem 1.

3.3.1 Diskretizacija

Splošno d -dimenzionalno domeno Ω je težko dobro diskretizirati. Želimo si, da bi bile točke v domeni čim bolj enakomerno razporejene, saj to pomeni, da smo dobro popisali celotno področje in upamo, da s tem tudi obnašanje funkcije u , hrakti pa si zaradi numerične stabilnosti ne želimo, da bi bile točke preveč skupaj. Uvedimo dve količini, ki nam merita ti dve lastnosti. Za dano domeno Ω in množico točk X definirajmo

$$h_\Omega(X) = \max_{p \in \Omega} \min_{x \in X} \|p - x\| \quad (12)$$

$$S(X) = \min_{\substack{x, y \in \Omega \\ x \neq y}} \|x - y\|$$

Količina h pove, da ne glede na to, kje v domeni smo, imamo na razdalji manj ali enako h vsaj eno diskretizacijsko točko. Količina S pa pove, kako blizu so si diskretizacijske točke med seboj. Dobra diskretizacija želi maksimizirati S in minimizirati h .

Algoritmi za diskretizacijo odvisni od načina, kako podamo domeno. V našem se izkaže, da je za trenutne potrebe dovolj, da podpiramo kvadre in krogle do vključno treh dimenzij, kot tudi unije in razlike osnovnih oblik.

Tako lahko za diskretizacijo kvadra uporabimo kar enakomerno diskretizacijo. Prav tako ni težko ugotoviti, kdaj so točke na robu. Za čimbolj enakomerno diskretizacijo notranjosti kroga ali površine sfere lahko uporabimo npr. Fibonaccijevo mrežo, kot predlagano v [3] ali v [4]. Pri razlikah domen preprosto izbrišemo točke, ki so padle izven domene, pri unijah pa naredimo tudi unijo diskretizacij. Pri tem lahko potem naredimo še en korak in pobrišemo ven kakšno izmed točk, ki so si preblizu skupaj. Primeri domen in njihovih diskretizacij dobljenih na ta način so prikazani na sliki 5.

Pri enakomerni diskretizaciji smo močno uporabili naše znanje o domeni in njeni obliki. Lahko pa, da tega nimamo na voljo in želimo splošnejši algoritem, ki potrebuje npr. samo karakteristično funkcijo domene in njene meje, torej, kakšen kvader,

Algoritem 1 Brezmrežna metoda za reševanje PDE iz razdelka 3.1.2.

Vhod: Parcialna diferencialna enačba, kot opisana v (8). Parametri metode:

N ... celotno število diskretizacijskih točk
 Q ... število diskretizacijskih točk v notranjosti Ω
 n ... število sosedov, ki jih ima vsaka točka
 m ... število baznih funkcij
 b ... seznam baznih funkcij dolžine m
 w ... utež

Izhod: Skalarno polje u , ki aproksimira rešitev enačbe (8).

```
1: function REŠI( $\Omega, \mathcal{L}, f, \mathcal{R}, g, N, Q, n, m, b, w$ )
2:    $x \leftarrow \text{DISKRETIZIRAJ}(\Omega, N, Q)$   $\triangleright x$  postane seznam  $N$  točk, brez škode za
   splošnost naj leži prvih  $Q$  točk v  $\Omega$  in preostalih  $N - Q$  na  $\partial\Omega$ .
3:    $s \leftarrow \text{SOSEDI}(x, n)$   $\triangleright s$  je seznam dolžine  $N$ , pri čemer je  $s[i]$  seznam
   indeksov elementov v  $x$ , ki so sosedu  $x[i]$ , vključno z  $i$ .
4:    $\varphi \leftarrow$  prazen seznam dolžine  $N$ .
5:   for  $i \leftarrow 1$  to  $Q$  do  $\triangleright$  Izračunamo funkcije oblik v notranjosti.
6:      $\varphi[i] \leftarrow \text{FUNKCIJA OBLIKE}(\mathcal{L}, x[i], x, s[i], n, m, b, w)$   $\triangleright$  Glej algoritem 2.
7:   end for
8:   for  $i \leftarrow Q + 1$  to  $N$  do  $\triangleright$  Izračunamo funkcije oblik na robu.
9:      $\varphi[i] \leftarrow \text{FUNKCIJA OBLIKE}(\mathcal{R}, x[i], x, s[i], n, m, b, w)$   $\triangleright$  Glej algoritem 2.
10:  end for
11:   $A \leftarrow$  prazna razpršena  $N \times N$  matrika
12:  for  $i \leftarrow 1$  to  $N$  do  $\triangleright$  Aproksimiramo enačbo.
13:    for  $j \leftarrow 1$  to  $n$  do
14:       $A[i, s[j]] \leftarrow \varphi[i][j]$ 
15:    end for
16:  end for
17:   $r \leftarrow$  prazen vektor dolžine  $N$ 
18:  for  $i \leftarrow 1$  to  $Q$  do  $\triangleright$  Izračunamo desno stran v notranjosti.
19:     $r[i] \leftarrow f(x[i])$ 
20:  end for
21:  for  $i \leftarrow Q + 1$  to  $N$  do  $\triangleright$  Izračunamo robne pogoje.
22:     $r[i] \leftarrow g(x[i])$ 
23:  end for
24:   $u \leftarrow \text{REŠIRAZPRŠENSISTEM}(A, r)$ 
25:  return  $u$ 
26: end function
```

ki našo domeno vsebuje. V tem primeru lahko vzamemo enakomerno diskretizacijo kvadra in odstranimo vse točke, ki niso v domeni, vendar je tu težko kontrolirati število točk. Druga metoda je, da naključno izbiramo točke v kvadru, in sprejmemo tiste, ki pristanejo v notranjosti. Še boljšo diskretizacijo dobimo, če namesto psevdonaključnih števil uporabimo kvazinaključna števila, ki imajo manjšo diskrepanco. Več o tem si bralec lahko prebere v [5].

Kljub njihovi splošnosti so naključne diskretizacije precej slabe, kot na primer naključna diskretizacija kroga na sliki 6 (levo). Pri tem, da bi točke v domeni poraz-

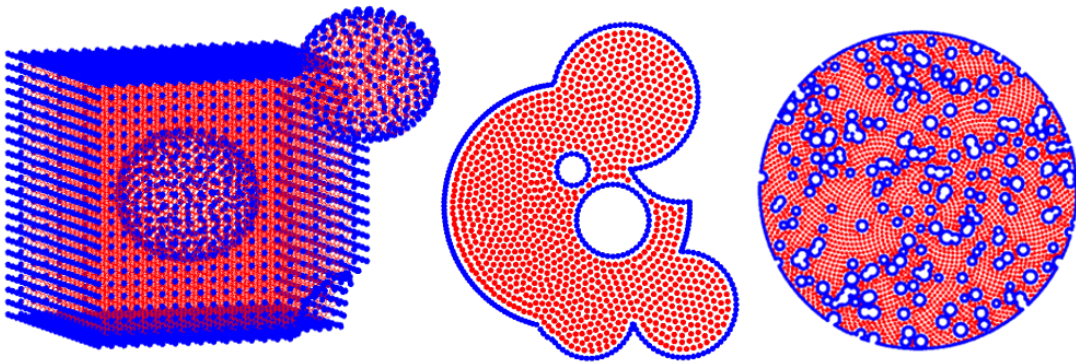
Algoritem 2 Izračun funkcije oblike.

Vhod: Parametri metode:

\mathcal{L} ... parcialen diferencialen operator
 p ... točka, v kateri aproksimiramo operator
 x ... seznam diskretizacijskih točk
 I ... množica indeksov točk v soseščini p
 n ... število sosedov, ki jih ima vsaka točka
 m ... število baznih funkcij
 b ... seznam baznih funkcij dolžine m
 w ... utež

Izhod: Funkcional, ki aproksimira operator \mathcal{L} v točki p .

```
1: function FUNKCIJA OBLIKE( $\mathcal{L}, p, x, I, n, m, b, w$ )
2:    $W \leftarrow$  prazen vektor dolžine  $n$ 
3:   for  $i \leftarrow 1$  to  $n$  do
4:      $W[i] \leftarrow \sqrt{w(x[I[i]] - p)}$ 
5:   end for
6:    $B \leftarrow$  prazna matrika velikosti  $n \times m$ 
7:   for  $i \leftarrow 1$  to  $n$  do
8:     for  $j \leftarrow 1$  to  $m$  do
9:        $B[i, j] \leftarrow W[i] \cdot b[j](x[I[i]])$ 
10:    end for
11:  end for
12:   $\ell \leftarrow$  prazen vektor dolžine  $m$ 
13:  for  $j \leftarrow 1$  to  $m$  do
14:     $\ell[j] \leftarrow (\mathcal{L}(b[j]))(p)$ 
15:  end for
16:   $\varphi \leftarrow (\ell \cdot \text{PINV}(B)) \odot W$        $\triangleright$  Direktna analogija enačbe (10),  $\odot$  označuje
17:  return  $\varphi$                                  $\triangleright$  Hadamardov produkt.
18: end function
```



Slika 5: Primeri domen in njihovih diskretizacij.

delili čimbolj enakomerno, si lahko pomagamo s preprostim iterativnim postopkom, za katerega se izkaže, da v praksi dobro deluje. Navdih jemljemo iz fizike in si zamislimo, da je vsaka točka naboj, ki se odbija od sosednjih točk in pustimo fiziki, da opravi svoje delo. Na vsako točko tako deluje neka sila, ki jo sili stran od dru-

gih točk, v prazen prostor. Točko lahko malo premaknemo v smeri sile, ponovno izračunamo medsebojne sile in postopek ponavljamo. Pri tem točkam na robu ne dovolimo premikanja in če kakšna točka iz notranjosti zleze iz domene, jo postavimo na naključno mesto nazaj v domeno. Zapis zgornjega postopka je v psevdokodi je podan kot algoritem 3. Postopku se kdaj reče tudi *sprostitev* (*angl.* relaxation) domene, kajti na začetku so med točkami napetosti, ki jih s premikanjem poskušajo minimizirati. Rezultati na primeru izboljšanja naključne diskretizacije kroga so podani na sliki 6.

Algoritem 3 Algoritem za izboljšanje kvalitete diskretizacije domene.

Vhod: Parametri metode:

Ω ... domena
 N ... število diskretizacijskih točk
 Q ... število diskretizacijskih točk v notranjosti Ω
 X ... seznam diskretizacijskih točk, prvih Q je v notranjosti.
 I ... število iteracij
 s ... število sosedov, ki jih upoštevamo pri delovanju sile
 F_0 ... delež sile, ki vpliva na premik
 α ... eksponent v sili

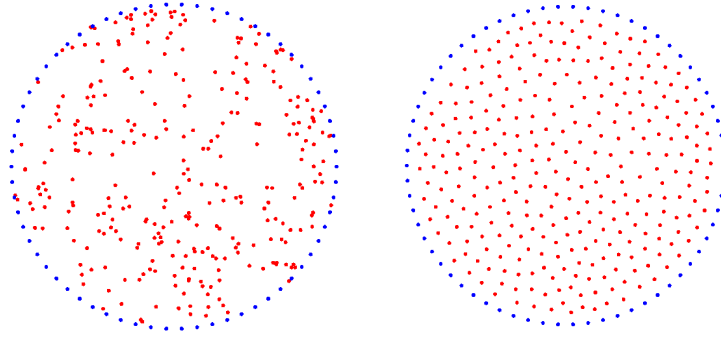
Izhod: Nov seznam diskretizacijskih točk.

```

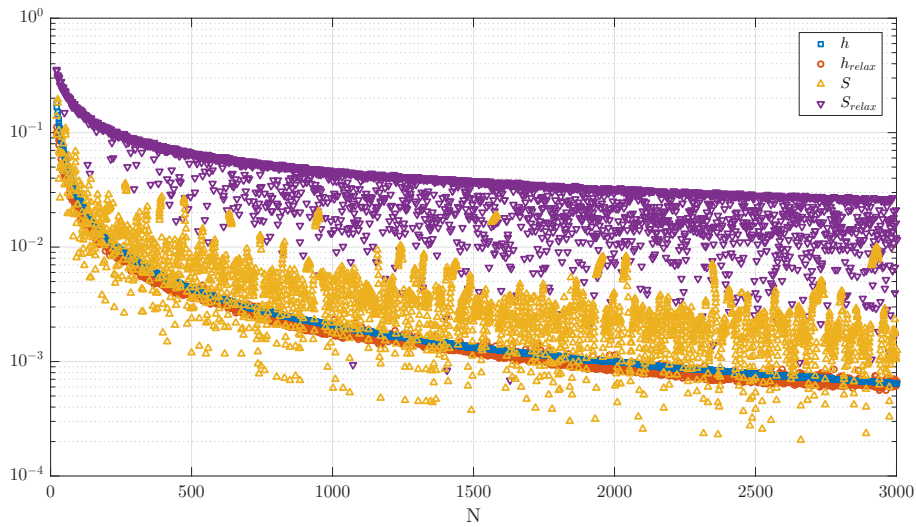
1: function IZBOLJŠAJ( $\Omega, N, Q, X, I, s, F_0, \alpha$ )
2:    $r_\chi \leftarrow \left( \frac{\text{VOLUMEN}(\Omega)}{N} \right)^{\frac{1}{\text{DIMENZIJA}(\Omega)}}$   $\triangleright$  Približek povprečne razdalje med točkami.
3:   for  $i \leftarrow 1$  to  $I$  do
4:     for  $j \leftarrow 1$  to  $Q$  do
5:        $\vec{F} \leftarrow \vec{0}$ 
6:       for each  $y$  in {najbližjih  $s$  sosedov  $X[i]$ } do
7:          $\vec{r} \leftarrow \frac{X[i]-y}{r_\chi}$   $\triangleright$  Brezdimenzijski vektor razdalje.
8:          $\vec{F} \leftarrow \vec{F} + \frac{\vec{r}}{\|\vec{r}\|^\alpha}$ 
9:       end for
10:       $X[i] \leftarrow X[i] + F_0 \vec{F}$ .
11:      if  $X[i] \notin \Omega$  then
12:         $X[i] \leftarrow$  naključna pozicija znotraj  $\Omega$ 
13:      end if
14:    end for
15:  end for
16:  return  $X$ 
17: end function

```

Da algoritem izboljša kvaliteto aproksimacije se vidi tudi, če izračunamo parametra h in S . Pričakujemo, da se bo S povečal, saj se točke, ki so bolj skupaj, bolj odbijajo. Na sliki 7 vidimo primerjavo med kvaliteto diskretizacije s Fibonaccijevo mrežo in njeno 10-kratno izboljšavo. Vidimo da se h ni bistveno spremenil, le varianca se mu je malce povečala, S pa se je v povprečju precej izboljšal in izkazuje celo lepše obnašanje kot prej.



Slika 6: Primerjava domene z naključno diskretizacijo (levo) in domene po izvedbi 10 iteracij algoritma 3 s parametri $F_0 = 10^{-2}$, $s = 6$, $\alpha = 3$ (desno).



Slika 7: Sprememba h in S po 10 iteracijah algoritma 3 s parametri $F_0 = 10^{-2}$, $s = 6$, $\alpha = 3$ v enotskem dvodimenzionalnem krogu z začetno Fibonaccijevo mrežo v odvisnosti od N z razmerjem robnih in notranjih točk $x : \frac{12}{\pi} \sqrt{x}$.

3.3.2 Iskanje najbližjih sosedov

Tako v algoritmu 1 v vrstici 3 kot v algoritmu 3 v vrstici 6 potrebujemo najti nekaj najbližjih sosedov dane točke. Navadno za okolico izberemo kar n najbližjih sosedov v evklidski metriki, vendar bi lahko soseščino definirali tudi drugače, na primer kot množico vozlišč, ki so oddaljeni manj kot neka fiksna razdalja R . Pri tem pristopu bi imeli manjšo kontrolo nad velikostjo soseščine, zato izberemo prvega.

Problem iskanja najbližjih sosedov (*angl.* nearest neighbour search) je znana in dobro raziskana tema z razvitimi veliko podatkovnimi strukturami, ki podpirajo grajenje, iskanje, vstavljanje in brisanje v logaritemskem času. Večina jih temelji na delitvi prostora na različne hierarhično urejene podprostore, ki jih potem hranimo v drevesni strukturi, kar nam omogoča logaritemski dostop. Med bolj znanimi strukturami so k -d tree [6], ball tree [7] in cover tree [8]. V članku [9] je narejena empirična primerjava med zgornjimi tremi strukturami, ki pokaže, da so v primeru nizkih dimenzij (kar gotovo vsebuje $d \leq 3$) najboljše obnese k -d tree. Poleg tega k -d tree najboljše deluje tudi, ko imajo točke, v katerih bomo iskali sosede podobno

porazdelitev kot točke, iz katerih smo naredili drevo, kar v našem primeru drži. Če vključimo še dejstvo, da je *k-d tree* tudi najpopularnejša rešitev za problem najbližjih sosedov in ima na voljo veliko prosto dostopnih implementacij, je to dovolj argumentov, da jo uporabimo tudi mi. Specifična uporabljena implementacija je predstavljena v [10], ki za shrambo N točk porabi $O(N)$ prostora in odgovarja na poizvedbe o n najbližjih sosedih v $O(n \log N)$ časa. S pomočjo tega postane implementacija funkcije SOSEDI iz algoritma 1 trivialna, pri algoritmu 3 pa si na vsaki iteraciji na novo zgradimo drevo in ga uporabimo za iskanje s najbližjih sosedov.

3.3.3 Reševanje razpršenega sistema

Za reševanje razpršenih sistemov poznamo veliko različnih metod, ki se v grobem delijo na direktne (obsežno opisane v [11]) in iterativne (obsežno opisane v [12]). Pri običajnih, polnih matrikah bi za reševanje splošnega sistema linearnih enačb uporabili LU razcep in razcepili matriko A kot $A = LU$. Toda, tudi če je matrika A razpršena, v splošnem L in U nista nujno in sta lahko celo polni, kar povzroči, da nam zmanjka pomnilnika. Direktne metode, kot na primer SuperLU [13] poskušajo z preureditvijo stolpcev v razpršeni matriki A minimizirati število novih neničenih elementov v matrikah L in U . Po drugi strani iterativne metode aproksimirajo pravilno rešitev sistema $Ax = b$ z zaporedjem približkov $\{x^{(r)}\}$, ki naj bi konvergirali k x . Te metode običajno ne zahtevajo veliko pomnilnika, je pa natančnost približka odvisna od števila porabljenih iteracij, tako da moramo za višjo natančnost porabiti večje število računskih operacij.

Za metode iz numerične linearne algebre bomo uporabljali knjižnico Eigen [14], ki nudi elegantno in hitro delo z matrikami v C++. Vgrajenih ima veliko direktnih in iterativnih metod za reševanje sistemov linearnih enačb.¹ Pri sistemih tipa (11) se je v praksi najbolje obnesel BiCGSTAB [15] z ILUT predpogojevanjem [16]. BiCGSTAB v primeru uporabe pravilne shrambe matrike v pomnilniku podpira tudi paralelizacijo, poleg tega pa je konvergenca v praksi dovolj hitra. Predpogojevanje z uporabo nepopolnega LU razcepa z dvojnim pragom (*angl.* dual threshold incomplete LU factorization (ILUT)) omogoča natančnejšo kontrolo nad porabljenim spominom in hitrostjo konvergence z uporabo dveh parametrov p in τ . Med LU faktorizacijo izpustimo vsak element, ki je manjši kot $\tau \cdot e$, kjer je e povprečna absolutna vrednost elementov v trenutni vrstici. Nato obdržimo le največjih f elementov v matrikah L in U , kjer je p uporabljen kot razmerje med f in začetnim številom neničenih elementov. V grobem nam tako parameter p kontrolira kolikokrat več spomina dovolimo za hranjenje predogojenke, parameter τ pa, kako natančna bo LU faktorizacija.

3.3.4 Časovna zahtevnost

V tem razdelku analiziramo časovno zahtevnost metode. Predpostavili bomo, da so vse evalvacije funkcij f , g , w in b_j , $\mathcal{L}b_j$, $\mathcal{R}b_j$ izvedene v $O(1)$. Prav tako bomo predpostavili, da je dimenzija problema majhna in konstantna in ne bo nastopala v analizah.

¹https://eigen.tuxfamily.org/dox/group__TopicSparseSystems.html
[obiskano 17. 6. 2017]

Trditev 3.9. *Pričakovana časovna zahtevnost algoritma 1 je*

$$O(INs \log^2 N + (N + n) \log N + m^2 nN) + T,$$

kjer je T čas, porabljen za reševanje razpršenega sistema enačb.

Dokaz. Pri funkciji DISKRETIZIRAJ predpostavimo, da uporabljamo enostavno diskretizacijo, ki deluje v $O(N)$ časa, skupaj z I iteracijami izboljšave na s sosedih, ki traja $O(INs \log^2 N)$ časa, saj na vsaki iteraciji konstruiramo drevo in iščemo s sosedov vsake točke.

Izvajanje funkcije SOSEDI traja $O((N + n) \log N)$ časa za konstrukcijo drevesa in iskanje sosedov.

Izvajanje funkcije FUNKCIJA OBLIKE traja po $O(n + nm + m + m^2 n + nm + n) = O(mn^2)$, kjer je prevladujoči faktor $O(m^2 n)$ rezultat računanja SVD razcepa z dvostranskim Jacobi SVD razcepom iz knjižnice Eigen.² Funkcija oblike se izračuna za vsako točko v domeni, torej da del algoritma traja $O(m^2 nN)$ časa.

Pri sestavljanju razpršene matrike lahko v naprej rezerviramo prostor za n elementov v vsaki vrstici, tako da nas sestavljanje matrike stane $O(nN)$ časa, sestavljanje robnih pogojev pa $O(N)$ časa. Na koncu še rešimo razpršen sistem linearnih enačb, kjer pa je čas zelo odvisen od problema, iteracijske metode in predpogojenke.

Skupna časovna zahtevnost je tako $O(INs \log^2 N + (N + n) \log N + m^2 nN) + T$, kjer je T čas, porabljen za reševanje razpršenega sistema enačb. \square

3.3.5 Prostorska zahtevnost

V tem razdelku analiziramo časovno zahtevnost metode. Podobno kot pri časovni zahtevnosti bomo predpostavili, da so vse evalvacije funkcij f , g , w in b_j , $\mathcal{L}b_j$, $\mathcal{R}b_j$ izvedene v $O(1)$ prostora. Prav tako bomo predpostavili, da je dimenzija problema majhna in konstantna in ne bo nastopala v analizah.

Trditev 3.10. *Prostorska zahtevnost algoritma 1 je $O(nN) + P$, kjer je P prostor, ki ga potrebujemo, za reševanje sistema linearnih enačb.*

Dokaz. Pri funkciji DISKRETIZIRAJ potrebujemo $O(N)$ spomina za shrambo N diskretizacijskih točk. Če izvajamo še dodatne izboljšave diskretizacije, nas to stane $O(n)$ dodatnega prostora.

Izvajanje funkcije SOSEDI nas stane $O(N)$ prosta za hranjenje drevesa in $O(nN)$ prosta za hranjenje indeksov n sosedov.

Izvajanje funkcije FUNKCIJA OBLIKE nas stane $O(n + nm + m + n) = O(nm)$ prostora za vsak klic, za hrambo N funkcij oblike pa potrebujemo $O(nN)$ prostora.

Razpršena matrika prav tako potrebuje $O(nN)$ prostora za neničelne elemente. Nato moramo rešiti le še sistem linearnih enačb, kar po predpostavki stane P prostora. Ker je $m = O(N)$ je prevladujoči faktor $O(nN)$ in skupna prostorska zahtevnost je $O(nN) + P$. \square

²https://eigen.tuxfamily.org/dox/classEigen_1_1JacobiSVD.html [obiskano 16. 6. 2017]

3.4 Pogoste vrednosti parametrov

Metoda je bila do sedaj formulirana zelo splošno, v praksi pa se uporablja nekaj ustaljenih kombinacij. Kot bomo videli razdalje pogosto merimo v večkratnikih r_χ , kot na vrstici 2 v algoritmu 3.

Definicija 3.11. Količino r_χ pripisano diskretizaciji X domene Ω , izračunano z

$$r_\chi(\Omega, X) = \left(\frac{\text{VOLUMEN}(\Omega)}{|X|} \right)^{\frac{1}{\text{DIMENZIJA}(\Omega)}}$$

imenujemo *karakteristična razdalja*.

Ideja definicije je, da r_χ predstavlja približno povprečno razdaljo med diskretizacijskimi točkami, če bi bile razporejene enakomerno. Celoten volumen domene razdelimo na $N = |X|$ delov, tako da vsaki točki pripada enak kos volumna $v = \frac{\text{VOLUMEN}(\Omega)}{N}$. Če bi bil ta kos neka hiperkocka v d dimenzijah, potem je $\sqrt[d]{v}$ dolžina njene stranice in to je ocena za medsebojno razdaljo med točkami. V eni dimenziji r_χ do faktorja $\frac{N}{N-1}$ natančno ustreza razdalji med enakomerno razporejenimi točkami. Drugo pogosto merilo za lokalno razdaljo bo kar razdalja do najbližjega (različnega) soseda, ki jo imenujmo r_c . To je težje izračunati, če nimamo že zgrajenega drevesa, zato se ponavadi na enostavnejših primerih poslužujemo kar r_χ . Kadar pa to ni dobra aproksimacija razdalje med vozlišči za celo domeno, kot na primer ob goščenju mreže, se poslužimo r_c .

Pogoste vrednosti za število sosedov so $n = 3, 5, 7$ v eni dimenziji, $n = 5, 9, 13, 25$ v dveh dimenzijah in $n = 7$ ali 27 v treh dimenzijah. Za regularne domene to simetrično izbire vozlišča pri neregularnih pa izbira n ni tako pomembna, dokler je dovolj visoka, da dobro popiše operator, ki ga aproksimiramo. Višji n ponavadi pomeni večjo stabilnost, morda višji red in počasnejše izvajanje. V praksi izberemo najmanjši n , ki daje zadovoljive rezultate.

Za utež se pogosto uporablja konstanta $w \equiv 1$, če ne želimo uteženih najmanjših kvadratov, sicer pa pogosto vzamemo za utež Gaussovo funkcijo

$$w(x) = \exp \left(- \left(\frac{x}{\sigma_w} \right)^2 \right),$$

kjer σ_w imenujemo parameter oblike (*angl.* shape parameter) uteži. Velja opozoriti, da je ta funkcija praktično nič, če smo več kot $3\sigma_w$ stran od središča ($w(3\sigma_w) \approx 0.0001234$). Če na primer izberemo $\sigma_w = r_\chi$ se nam vsi sosedi, ki so dlje kot r_χ stran v aproksimaciji ne upoštevajo, ne glede na to kako velik n smo izbrali. Ponavadi se za σ_w začne izbirati vrednosti okoli r_χ , je pa potrebno optimalno vrednost določiti za vsak posamezen problem posebej.

Pogosta izbira baznih funkcij so prostori \mathbb{P}_ℓ monomov skupne stopnje manj ali enako ℓ . Pri tem moramo seveda paziti, da izberemo dovolj visok n . Druga pogosta izbira je, da vzamemo $m = n$ in za bazne funkcije izberemo radialne bazne funkcije s centri v sosedih.

Definicija 3.12. Funkcija ψ_c se imenuje radialna bazna funkcija s centrom v točki c , če je odvisna samo od razdalje do centra, torej

$$\psi_c(x) = \tilde{\psi}(\|x - c\|),$$

za vsak x za neko funkcijo ψ . Kasneje kar identificiramo ψ_c in $\tilde{\psi}$ in pišemo $\psi_c = \psi_c(r)$.

V tabeli 1 so našteje najpogostejše uporabljene radialne bazne funkcije, povzete po [17, str. 5].

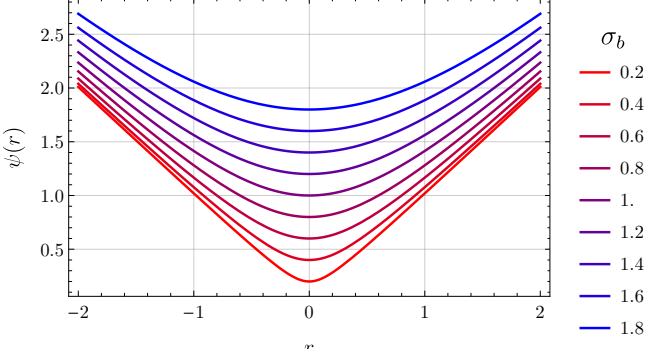
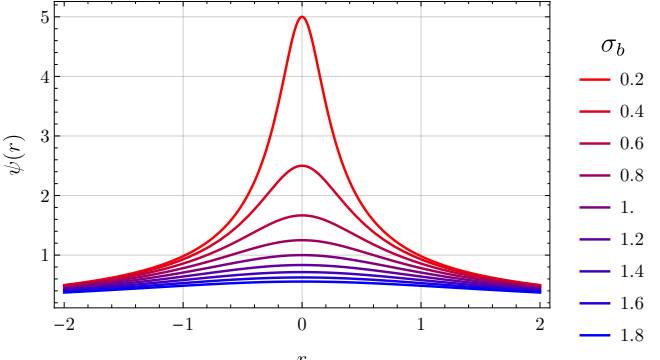
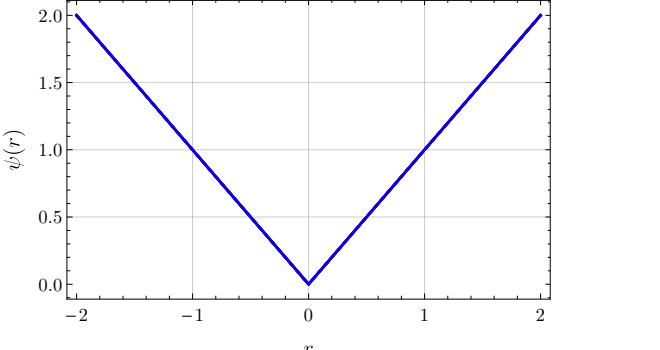
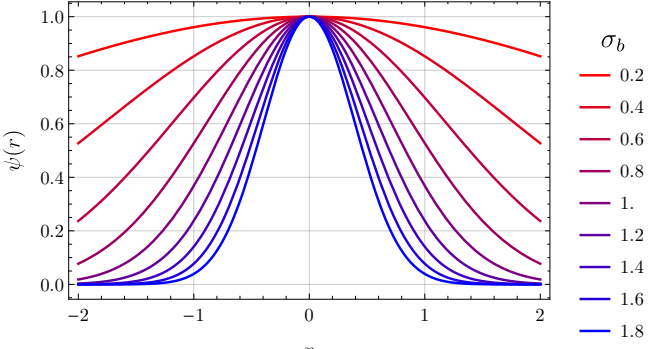
<p>Multikvadratične (MQ) (<i>angl.</i> multiquadric)</p> $\psi(r) = \sqrt{r^2 + \sigma_b^2}$	
<p>Inverzne multikvadratične (IMQ) (<i>angl.</i> Inverse multiquadrics)</p> $\psi(r) = 1/\sqrt{r^2 + \sigma_b^2}$	
<p>Linearna razdalja (L) (<i>angl.</i> Linear distance)</p> $\psi(r) = r$	
<p>Gaussove funkcije (G) (<i>angl.</i> Gaussians)</p> $\psi(r) = \exp(-(r/\sigma_b)^2)$	

Tabela 1: Najpogostejše uporabljene radialne bazne funkcije s parametrom oblike σ_b .

Ko si izberemo neko bazno funkcijo ψ , potem za bazne funkcije okoli točke p vzamemo

$$\mathbf{b} = \{\psi_s; s \in N(p)\}.$$

Pri izbiri parametra oblike σ_b moramo tudi biti do neke mere pazljivi. Izbrati ga moramo dovolj velikega, da se funkcije “prekrivajo”, torej tako, da nima funkcija vrednosti blizu nič pri vseh sosedih razen pri sebi. Hkrati mora biti parameter dovolj majhen, da sistem (9) ni preveč nestabilen.

Pogosto se k bazi radialnih baznih funkcij doda še monome nizkih stopenj, za boljšo aproksimacijo funkcij, ki so blizu konstantam. Več o kvaliteti, stabilnosti in redu konvergence aproksimacije z radialnimi baznimi funkcijami si lahko bralec prebere v [18].

4 Implementacija

blah blah [19].

5 Osnovni numerični zgledi

V tem razdelku bomo pogledali obnašanje metode na osnovnih numeričnih zgledih, da si zgradimo intuicijo o njenem delovanju. V nadaljnjem besedilu in na vseh grafih bo metoda, opisana v razdelku 3 označena s kratico MLSM (*angl.* Meshless Least Squares Method). Če ni drugače navedeno, so vse časovne meritve narejene na prenosnem računalniku s procesorjem Intel(R) Core(TM) i7-4700MQ CPU @2.40GHz in 16 GB DDR3 pomnilnika. Vsi programi so bili prevedeni s prevajalnikom g++ verzije 7.1.7 in stikali `-std=c++14 -O3 -DNDEBUG` na operacijskem sistemu Linux. Pri paralelizaciji je bila uporabljena knjižnica OpenMP [20] za paralelizacijo z deljenim pomnilnikom.

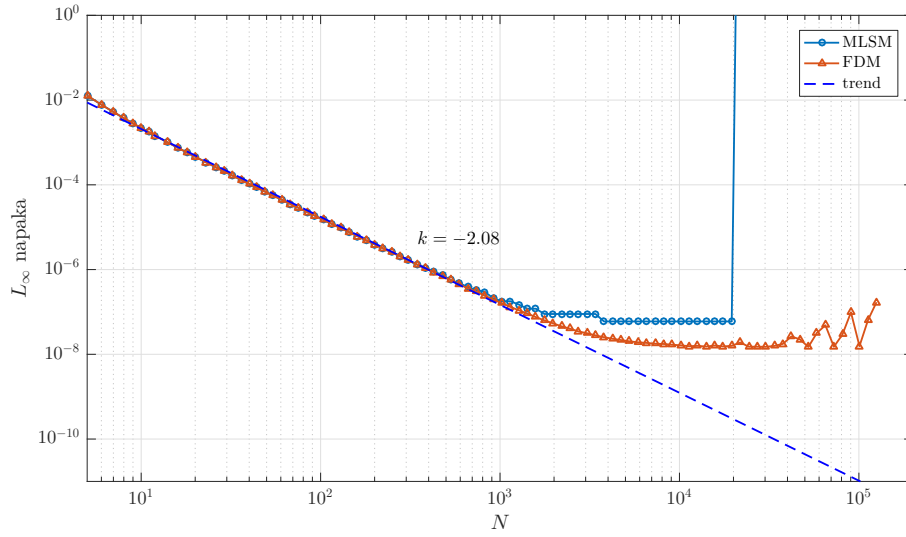
5.1 Enodimenzionalni robni problem

Oglejmo si najprej preprost primer enodimenzionalne Possonove enačbe, ki smo ga uporabili že za motivacijo izpeljave numerične metode. Rešujemo problem

$$\begin{aligned} u''(x) &= \sin(x), & x \in (0, 1) \\ u(0) &= 0, \\ u'(1) &= 0, \end{aligned}$$

ki ima analitično rešitev $u(x) = \cos(1)x - \sin(x)$. Na sliki 8 je prikazana konvergenca metode končnih diferenc in MLSM. Metoda MLSM je bila uporabljena s parametri, pri katerih velja trditev 3.4, torej $n = m = 3$, $w = 1$, $\mathbf{b} = \{1, x, x^2\}$. V obeh primerih smo uporabili direktno metodo SuperLU za reševanje sistema enačb.

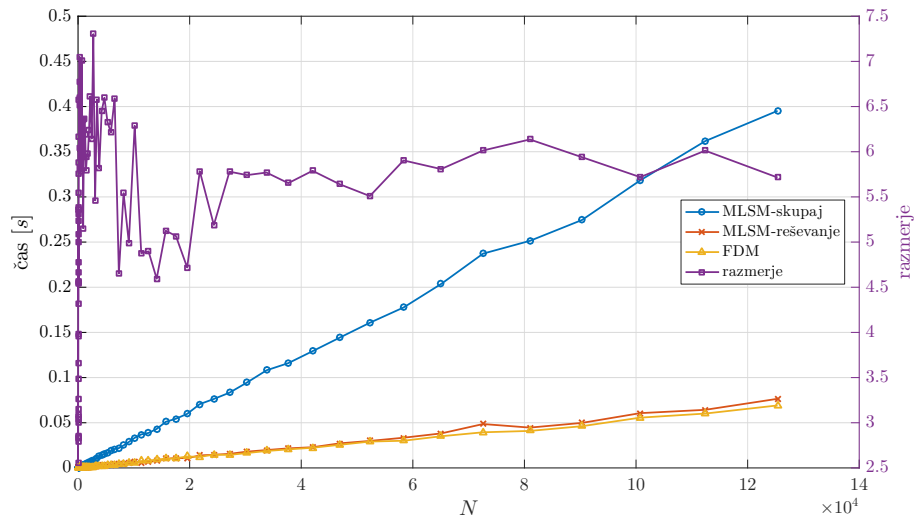
Na grafu vidimo, da metodi tudi v praksi sovpadata do približno $N = 1000$, nato pa natančnost MLSM postane konstantna, dokler ne pridemo v območje numerične nestabilnosti. Metoda končnih diferenc pride v območje, ko ne konvergira več z redom 2 ob približno enakem N kot MLSM, nestabilnosti pa se začnejo kasneje,



Slika 8: Napaka FDM in MLSM metode v primerjavi s pravilno rešitvijo.

med $N = 10^4$ in $N = 10^5$. Hitrejšo nestabilnost MLSM lahko pripišemo numeričnim napakam pri računanju aproksimacije drugih odvodov, pri FDM namreč v matriko direktno zapišemo koeficient $-\frac{2}{h^2}$, pri MLSM pa se ta numerično izračuna. Iz naklona premice vidimo, da sta obe metodi tudi v praksi reda 2.

Časovna primerjava obeh metod je prikazana na sliki 9. MLSM je približno konstantno 6-krat počasnejši kot FDM, kot prikazano z razmerjem, merjenim na desni osi. To je tudi pričakovano, saj MLSM išče sosede in računa aproksimacije, medtem je to pri FDM vgrajeno v metodo. Če pri MLSM posebej merimo samo čas, ki ga porabimo za sestavljanje matrike in reševanje sistema, vidimo, da se skoraj ujema z časom, porabljenim pri FDM. Minimalno razliko gre pripisati razliki pri načinu grajenja matrike, saj pri FDM natanko vemo pozicije in število neničelnih elementov vnaprej, pri MLSM pa v splošnem ne.



Slika 9: Primerjava časa izvajanja MLSM in FDM metod.

5.2 Poissonova enačba

Oglejmo si sedaj obnašanje metode na klasični Poissonovi enačbi na kvadratu:

$$\begin{aligned}\Delta u &= 1, & (x, y) &\in (0, 1) \times (0, 1) \\ u(x, 0) &= u(x, 1) = u(0, y) = u(1, y) = 0.\end{aligned}\tag{13}$$

Analitično rešitev lahko dobimo s pomočjo separacije spremenljivk in se glasi

$$u(x, y) = -8 \sum_{\substack{k=1 \\ k \text{ lih}}}^{\infty} \frac{\sin(k\pi x) \sinh \frac{k\pi(1-y)}{2} \sinh \frac{k\pi y}{2}}{\cosh(\frac{k\pi}{2}) k^3 \pi^3}.\tag{14}$$

Za primerjavo z numeričnimi rešitvami bomo uporabili končno vsoto, zato ocenimo ostanek.

Trditev 5.1. *Za ostanek vrste (14) velja*

$$\left| -8 \sum_{\substack{k=\ell \\ k \text{ lih}}}^{\infty} \frac{\sin(k\pi x) \sinh \frac{k\pi(1-y)}{2} \sinh \frac{k\pi y}{2}}{\cosh(\frac{k\pi}{2}) k^3 \pi^3} \right| < -\frac{\psi''(\ell/2)}{4\pi^3},$$

kjer je ψ digama funkcija.

Dokaz. Ocenjujmo vsak člen vrste posebej. Funkcijo \sin ocenimo z 1, funkcija $\sinh \frac{k\pi(1-y)}{2} \sinh \frac{k\pi y}{2}$ ima maksimum na sredini intervala in jo lahko ocenimo z njeno vrednostjo v $y = \frac{1}{2}$. Tako nam ostane za oceniti številska vrsta

$$-8 \sum_{\substack{k=\ell \\ k \text{ lih}}}^{\infty} \frac{(\sinh \frac{k\pi}{4})^2}{\cosh(\frac{k\pi}{2}) k^3 \pi^3}.$$

Uporabimo še neenakost $\frac{(\sinh \frac{k\pi}{4})^2}{\cosh(\frac{k\pi}{2})} < \frac{1}{2}$ in dobimo oceno

$$\left| -8 \sum_{\substack{k=\ell \\ k \text{ lih}}}^{\infty} \frac{\sin(k\pi x) \sinh \frac{k\pi(1-y)}{2} \sinh \frac{k\pi y}{2}}{\cosh(\frac{k\pi}{2}) k^3 \pi^3} \right| < \frac{4}{\pi^3} \sum_{\substack{k=\ell \\ k \text{ lih}}}^{\infty} \frac{1}{k^3} = -\frac{\psi''(\ell/2)}{4\pi^3},$$

kjer je ψ digama funkcija (in posledično ψ'' poligama funkcija). □

Za primer $\ell = 1$ v prejšnji trditvi dobimo oceno

$$|u(x, y)| \leq \frac{7\zeta(3)}{2\pi^2} \approx 0.1356,$$

kar drži, v resnici je najbolj ekstremna vrednost $u(1/2, 1/2) \approx -0.7367$. Kot posledico zgornje trditve lahko izračunamo, da je rep vrste (14) za $\ell = 59$ manjši kot 10^{-5} , za $\ell = 181$ pa manjši kot 10^{-6} .

Opomba 5.2. Računanje izraza

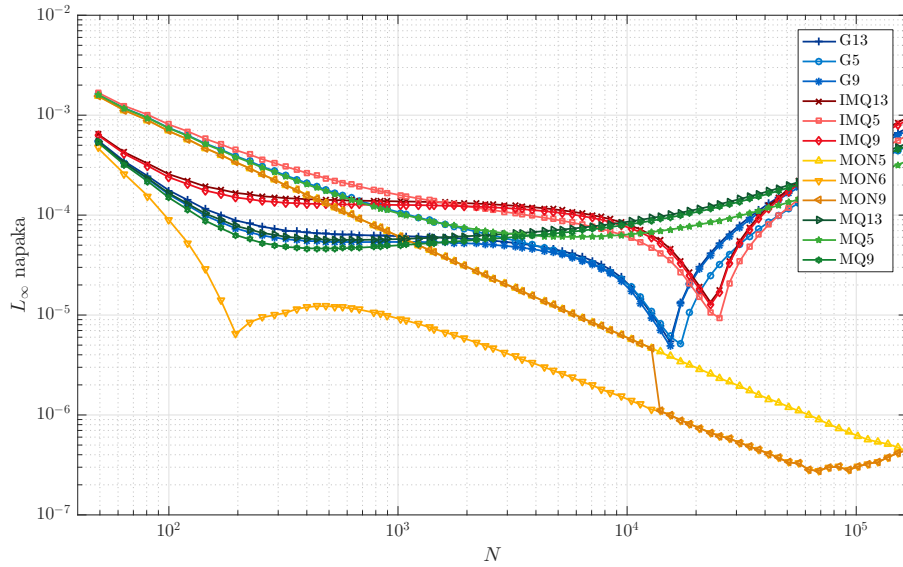
$$\frac{\sinh \frac{k\pi(1-y)}{2} \sinh \frac{k\pi y}{2}}{\cosh(\frac{k\pi}{2})}$$

ni numerično stabilno, ko k raste, kajti tako števec kot imenovalec se približujeta neskončno, kvocient pa ima končno limito. Ko za želimo izračunati numerično je bolje uporabiti stabilnejšo obliko

$$\frac{1}{2} \left(1 - \frac{\exp(-k\pi y) + \exp(-k\pi(1-y))}{1 + \exp(-k\pi)} \right),$$

kjer vedno nastopa negativen eksponent.

Rešimo problem (13) numerično, z štirimi različnimi nabori baznih funkcij: monomi, Gaussovimi funkcijami, multikvadratičnimi in inverznimi multikvadratičnimi. Za monome vzemimo $n = m = 5$, $m = 6$ in $n = 9$ in $m = n = 9$, kar sovпада s primeri iz razdelka 3.2. Za radialne bazne funkcije vzemimo vedno $n = m$ in sicer vrednosti 5, 9 in 13 za parameter oblike pa $\sigma_b = 40r_\chi$. Za utež vzemimo Gaussovo utež z $\sigma_w = r_\chi$. V vseh primerih je bila za reševanje sistema enačb uporabljena direktna metoda SuperLU. Iterativni BiCGSTAB algoritem je dal enake rezultate. Napaka metod je prikazana na sliki 10.

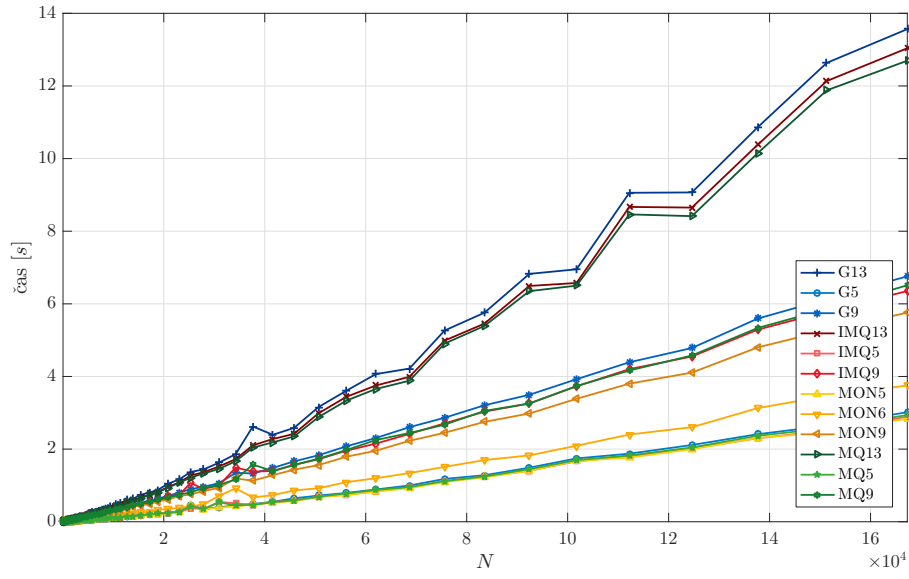


Slika 10: Konvergenca MLSM za različne parametre pri reševanju (13).

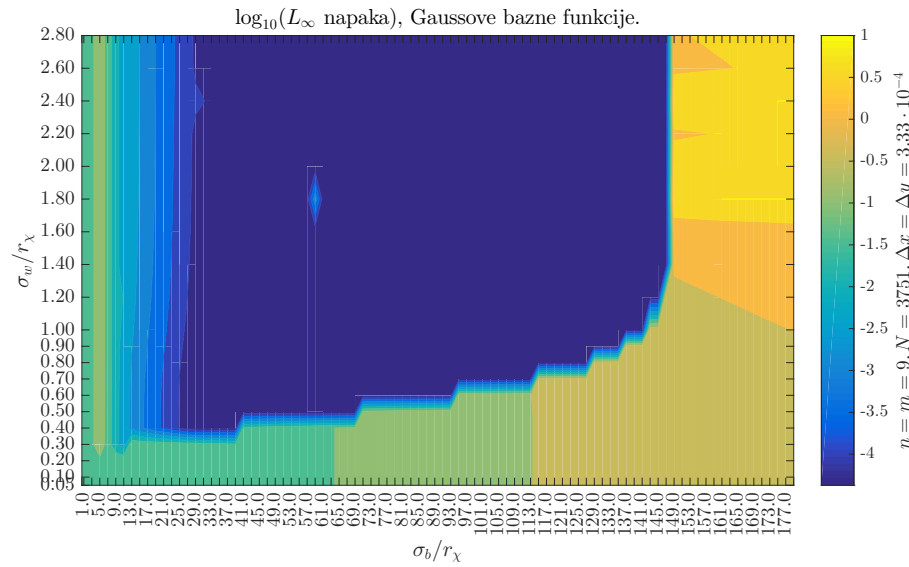
6 Raba materialov

blah blah

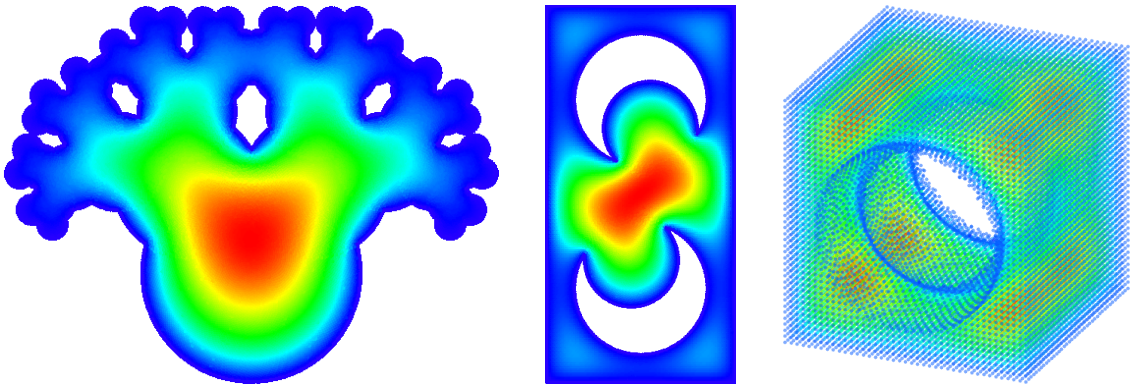
6.1 Zaključek



Slika 11: Čas izvajanja za različne izbore parametrov pri reševanju problema (13).



Slika 12: Napaka v odvisnosti od σ_b in σ_w za Gaussove funkcije.



Slika 13: Reševanje Poissonove enačbe $\Delta u = 1$ s homogenimi robnimi pogoji na zanimivejših domenah.

Literatura

- [1] L.P. Lebedev and M.J. Cloud. *Introduction to Mathematical Elasticity*. World Scientific, 2009.
- [2] Keith D Hjelmstad. *Fundamentals of structural mechanics*. Springer Science & Business Media, 2007.
- [3] JH Hannay and JF Nye. Fibonacci numerical integration on a sphere. *Journal of Physics A: Mathematical and General*, 37(48):11591, 2004.
- [4] Álvaro González. Measurement of areas on a sphere using Fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, 42(1):49–64, 2010.
- [5] William J Morokoff and Russel E Caflisch. Quasi-random sequences and their discrepancies. *SIAM Journal on Scientific Computing*, 15(6):1251–1279, 1994.
- [6] Andrew W Moore. An introductory tutorial on kd-trees. 1991.
- [7] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.
- [8] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM, 2006.
- [9] Ashraf M Kibriya and Eibe Frank. An empirical comparison of exact nearest neighbour algorithms. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 140–151. Springer, 2007.
- [10] David M Mount and Sunil Arya. ANN: library for approximate nearest neighbour searching. 1998. <https://www.cs.umd.edu/~mount/ANN/> [obiskano 16. 6. 2017].
- [11] Timothy A Davis. *Direct methods for sparse linear systems*. SIAM, 2006.
- [12] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
- [13] Xiaoye S Li. An overview of SuperLU: Algorithms, implementation, and user interface. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):302–325, 2005.
- [14] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3, 2010. <http://eigen.tuxfamily.org> [obiskano 16. 6. 2017].
- [15] Henk A Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644, 1992.
- [16] Yousef Saad. ILUT: A dual threshold incomplete LU factorization. *Numerical linear algebra with applications*, 1(4):387–402, 1994.

- [17] Robert Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264, 1995.
- [18] Martin D Buhmann. Radial basis functions. *Acta Numerica 2000*, 9:1–38, 2000.
- [19] Bjarne Stroustrup. *The C++ programming language*. Pearson Education India, 1995.
- [20] Leonardo Dagum and Ramesh Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998. <http://www.openmp.org/> [obiskano 21. 06. 2017].