University *of Ljubljana*
Faculty *of Mathematics and Physics*

Modelska Analiza 2

7. naloga - Metoda končnih elementov: Poissonova enačba

Avtor: Matic Lubej
Asistent: dr. Simon Čopar
Predavatelj: prof. dr. Alojz Kodre

Ljubljana, 8.4.2014

Naloga:

Pri tej nalogi smo se spoznali z metodo končnih elementov (FEM). Območje smo razdelili na mrežo trikotnikov, s katerimi smo nato definirali piramidalne funkcije. Problem sem reševal po principu SOR, kjer sem moral paziti, da sem upošteval vse sosede. V prvem delu naloge smo reševali splošen primer za cev s polkrožnim presekom, v drugi nalogi pa smo imeli isti problem za cev s presekom iz naloge 5.

Del I

Cev s polkrožnim presekom

1 Naloga

V metodi končnih elementov razdelimo definicijsko območje z mrežo točk in zveznic med njimi v same trikotnike. Približno rešitev zapišemo kot linearno kombinacijo poskusnih funkcij:

$$u = \sum_{i=1}^{N} a_i w_i. \tag{1}$$

Za funkcije w_i izberemo piramidalne funkcije, ki imajo v točki i vrednost 1, v vseh drugih točkah vrednost 0, med oglišči pa se spreminjajo linearno. Funkciji nesosednjih točk sta vedno ortogonalni, za sosede pa je matrični element od nič različen samo na dveh trikotnikih ob zveznici. Vsaka linearna kombinacija takih funkcij je zvezna vendar odsekoma ravna funkcija. Koeficienti a_i te kombinacije so vrednosti funkcije v mrežnih točkah. Z metodo končnih elementov izračunaj pretok skozi polkrožno cev in cev s presekom iz 5. naloge.

2 Uvod

Splošno obliko variacijskega principa za reševanje Poissonove enačbe $\nabla^2 u = -f$ dobimo tako, da enačbo zapišemo v zahtevo po ekstremu funkcionala:

$$\mathcal{S}(u) = \frac{1}{2}\langle \nabla u,\, \nabla u\rangle - \langle f,\, u\rangle. \tag{2}$$

Zahteva, da bo variacija tega funkcionala enaka 0, vodi do sistema enačb za koeficiente a_i:

$$\begin{aligned} \sum_{j=1}^{N} A_{ij} a_j \quad &= \quad b_i, \\ A_{ij} = \quad &\langle \nabla w_i,\, \nabla w_j\rangle, \\ b_i = \quad &\langle f,\, w_i\rangle. \end{aligned} \tag{3}$$

Delitev na točke podamo s koordinatami $(x_i,\, y_i)$, $i \in [1,\, N]$. Točke povežemo v trikotnike $(i,\, j,\, k)_r$, $r \in [1,\, R]$. Njihove ploščine so:

$$S_r = \frac{1}{2}|x_i(y_j - y_k) + \text{cikl. prem.}|. \tag{4}$$

Velikost ∇w_i na r-tem trikotniku je $d_{j,k}/\left(2S_r\right)$, kjer je $d_{j,k}$ razdalje med točko j in k. A_{ij} je vsota skalarnih produktov gradientov $\left((x_j - x_k)(x_k - x_i) + (y_j - y_k)(y_k - y_i)\right)/\left(4S_r\right)$ po trikotnikih ob zveznici $i-j$. Za diagonalne elemente A_{ii} se gornji izraz poenostavi na $d_{j,k}^2/\left(4S_r\right)$, seštet po vseh trikotnikih s točko i v oglišču. Podobno preprosto se v našem primeru izraža $\langle f,\, w_i\rangle$, kjer je $f = -1$ konstanta. Preostane integral linearne funkcije po trikotniku, ki da ravno $S_r/3$. Koeficienti b_i so torej enake $-1/3$ vsote ploščin trikotnikov ob točki i.

3 Reševanje

Problem sem zasnoval v matematičnem orodju Mathematica, potem pa sem ga zaradi hitrosti prepisal v Matlab. Najprej sem moral ustvariti mrežo trikotnikov. To sem storil tako, da sem ustvaril točke po notranjosti polkrožne cevi, ki so ustrezno odmaknjene od roba. Sode in lihe vrstice so pri tem bile zamaknjene za 1/2, tako da sem ustvarih mrežo enakokrakih trikotnikov. Po tem sem ločeno zgeneriral še točke po robu. Na vseh točkah sem uporabil funkcijo DelaunayTriangulation[] iz paketa ComputationalGeometry. Ta funkcija vrne relacije med povezanimi točki v zelo uporabnem formatu naslednje oblike:

$$\vdots \tag{5}$$

$$\{i, \{j_1, j_2, \ldots j_{N_r}\}\}, \tag{6}$$

$$\vdots \tag{7}$$

kjer indeks i predstavlja indeks trenutne točke, indeksi $j_1, \ldots, j_{N_r}$ pa predstavljajo indekse vseh točk, ki so povezani s točko i v obratni smeri urinega kazalca. Takšen formt je zelo prikladen, saj opisuje sosednje trikotnike za poljubno točko. Pri prepisovanju v Matlab sem imel s tem delom malce težav, saj izhodni format iste funkcije v Matlabu nima omenjene oblike. Ko sem vse funkcije definiral, sem se lotil metode končnih elementov kar preko metode SOR. Naredil sem naslednje korake:

$$\sum_{j=1}^{N} A_{ij} a_j \quad = \quad A_{ii} a_i + \sum_{\circlearrowleft} A_{ij} a_j = b_i, \tag{8}$$

$$a_i \quad = \frac{1}{A_{ii}} \quad \left(b_i - \sum_{\circlearrowleft} A_{ij} a_j \right), \tag{9}$$

$$a_i^{(k+1)} \quad = \quad a_i^{(k)} + \frac{1}{A_{ii}} \left(b_i - \sum_{\circlearrowleft} A_{ij} a_j^{(k)} - A_{ii} a_i^{(k)} \right), \tag{10}$$

$$a_i^{(k+1)} \quad = \quad a_i^{(k)} + \frac{1}{A_{ii}} \xi \longrightarrow a_i^{(k)} + \frac{\omega}{A_{ii}} \xi, \tag{11}$$

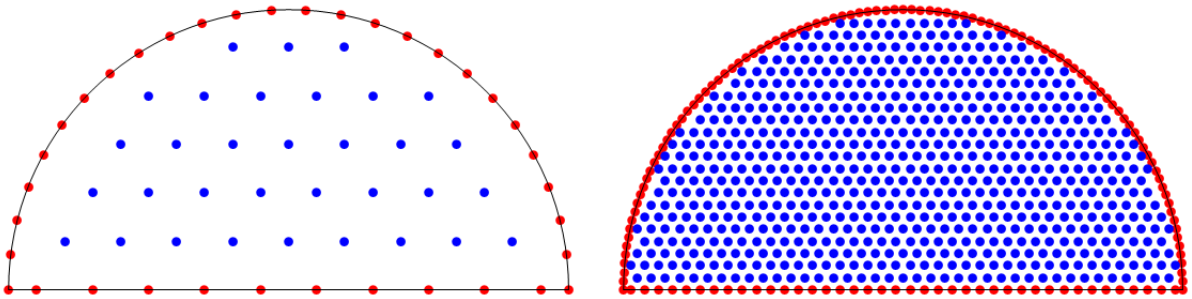
kjer smo v zadnjem koraku še vključili pospešeno relaksacijo, ki smo jo spoznali v 5. nalogi, za katero velja, da konvergira na območju $0 < \omega < 2$. Na tak način sem določil vse potrebne koeficiente a_i, ki sem jih določil iz zanke, ki je tekla le po točkah iz sredine območja, točke na robu pa sem ignoriral in na tak način enostavno izpolnil robni pogoj. S temi koeficienti sem nato sestavil funkcijo, postopek pa potem iteriral dokler maksimalni zaporedni popravek ni bil manjši od željene natančnosti.

3.1 Matlab solver

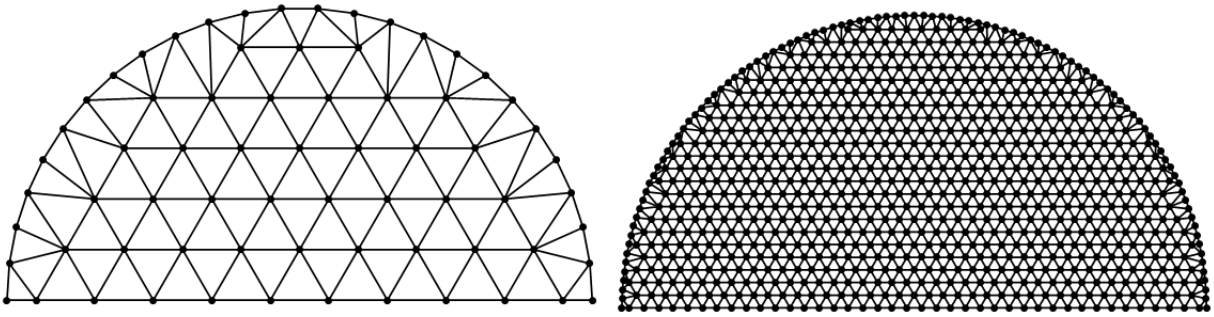
V matematičnem orodju Matlab sem odkril tudi že vgrajen način reševanja z metodo končnih elementov. Z ukazom „pdetool“ prikličemo grafični vmesnik, kjer lahko definamo željeno obliko območja. Matlab nato sam najde najbolj optimalno postavitev mreže, ki pa jo lahko naredimo tudi finejšo. Nato lahko izberemo željeno obliko parcialne diferencialne enačbe in nastavimo željene koeficiente. Naše rešitve sem primerjal še s tem orodjem.

4 Rezultati

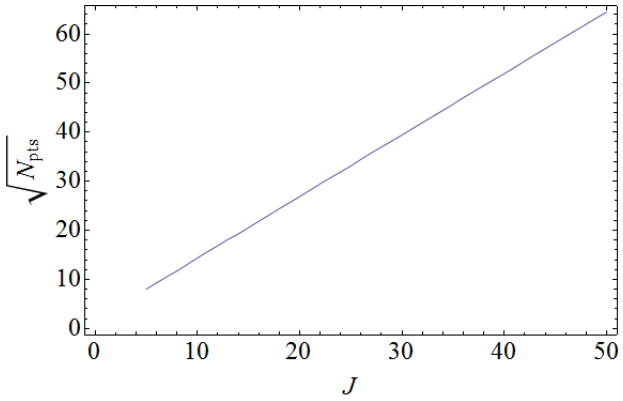
Najprej si oglejmo razdelitev mreže, kjer sem pozicije točk izbral sam:



Z rdečo so označene točke na robu, z modro pa točke iz notranjosti. Vidimo, da je sredinsko območje enakomerno posejano, pri robovih pa se pravilnost trikotnikov poruši z zahtevo po opisu roba. Delaunayeva triangulacija nam da naslednjo trikotno mrežo:

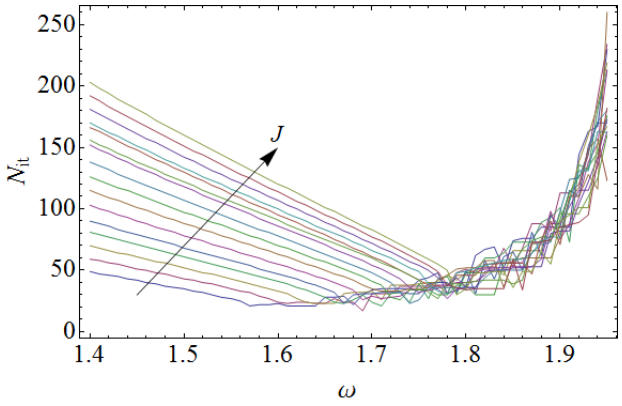


Dobimo trikotnike v takšnem formatu, ki sem ga opisal zgoraj. Prvi smiselni korak sedaj je, tako kot pri 5. nalogi, iskanje najbolj optimalnega parametra ω za naš problem. To nam je v interesu, ker lahko na tak način zelo znižamo število iteracij pri iskanju končne rešitve. V mojem primeru N pomeni razdelitev mreže, torej ni povsem enaka številu vseh točk, po drugi strani pa tudi ni enaka korenu števil vseh točk, kakor bi veljalo za kvadratno mrežo. Oglejmo si kako se število vseh točk spreminja z razdelitvijo N :



Vidimo, da naša razdelitev N res predstavlja neko proporcionalnost dimenziji mreže.

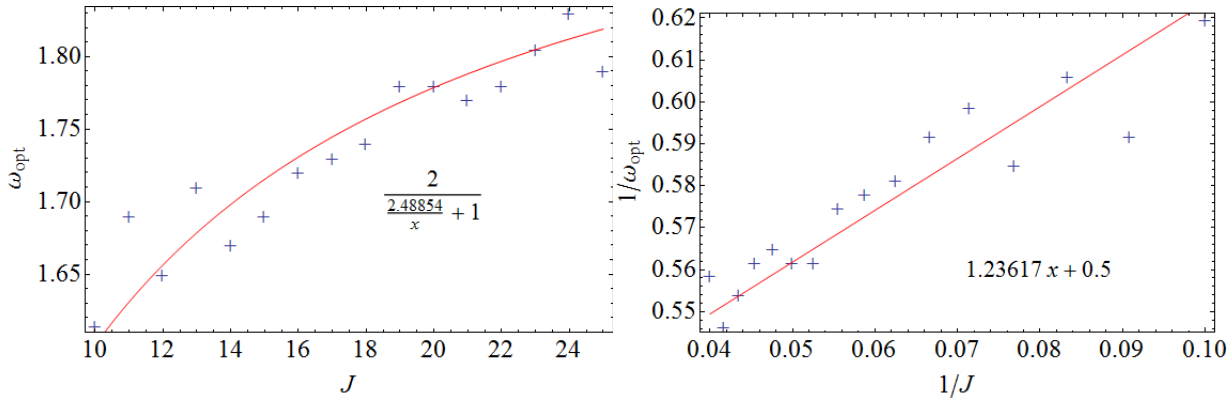
Oglejmo si število potrebnih iteracij, da dosežemo neko natančnost, v odvisnosti od pospeška ω, za nekaj različnih vrednosti števila točk N :



Seveda opazimo, da se optimalni ω_{opt} spreminja za različna števila točk, kar smo opazili tudi pri 5. nalogi. Potrebujemo torej neko obliko ω, ki je neodvisna od števila točk, tako da lahko najdemo optimalni pospešek za vse N. Temu primerno sem optimalnim pospeškom priredil funkcijo oblike:

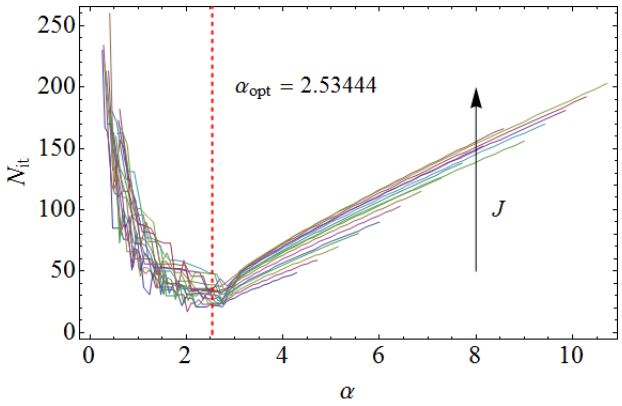
$$\omega = \frac{2}{1 + \alpha/N},$$

kjer sem iskal optimalno vrednost parametra α. Spodaj sta prikazana fita v dveh sistemih:

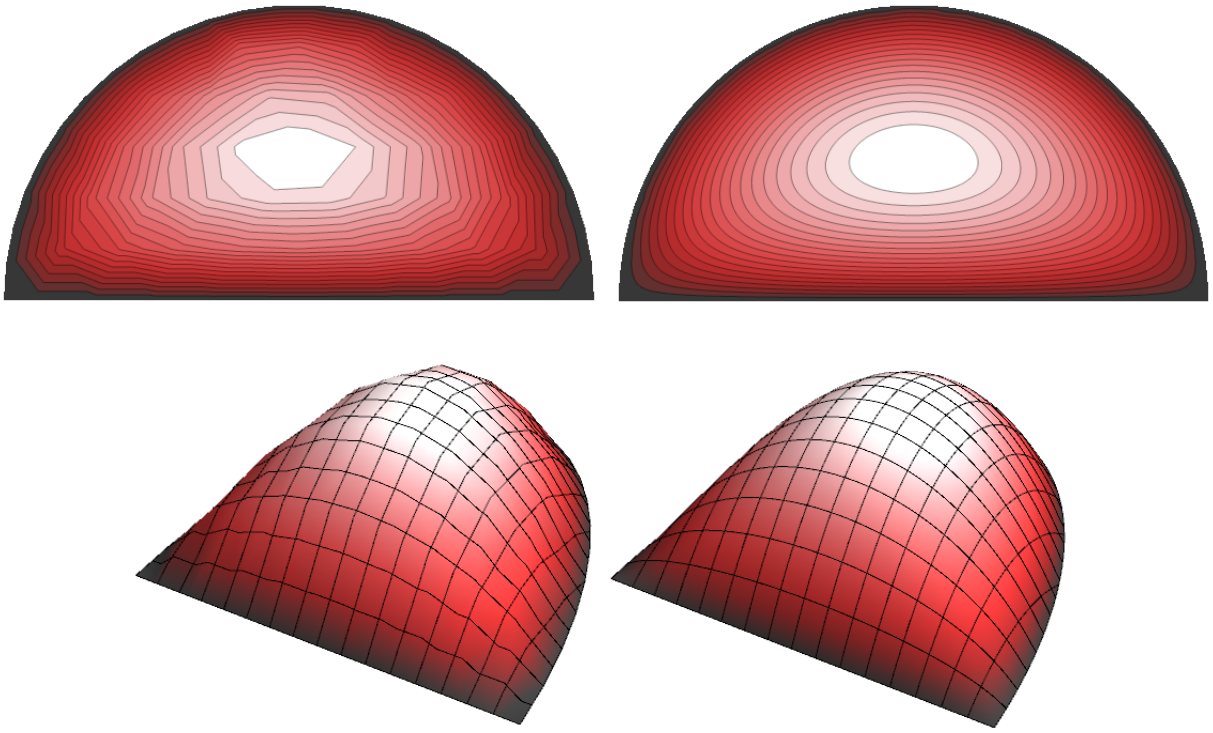


Na levi sliki je vrednost parametra očitna, desna slika pa prikazuje iste vrednosti v inverzni skali, kjer pričakujemo premico. Koeficient pred x na desni ssliki je enak $k = \alpha/2$.

Oglejmo si transformiran pospešek, ki naj bi bil neodvisen od N :



Dobimo vrednost $\alpha_{opt} \approx 2.5$, s katerim pospešimo naš sistem. Končno si lahko ogledamo rešitve. Spodaj sta prikazna rezultata za grobo ($N_p \approx 60$) in fino ($N_p \approx 4000$) razdelitev. Mejna natančnost je bila 10^{-11}:



Rešitve so smiselne. Opazimo, da potrebujemo večje število točk, da dosežemo gladke rešive.

4.1 Poisseuillov koeficient

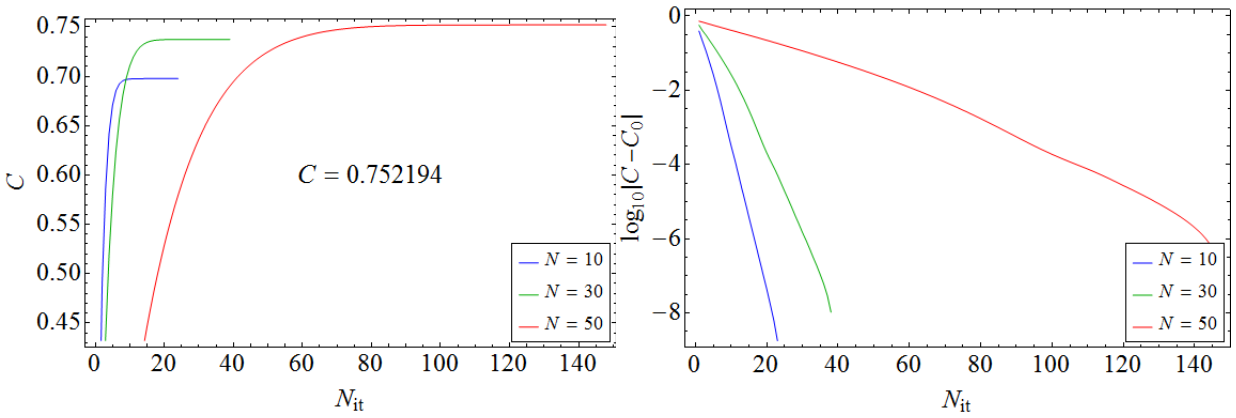
Tudi na takšni mreži lahko izračunamo vrednost Poisseuillovega koeficienta, ki je enak:

$$C = \frac{8\pi}{S_0^2} \int u \mathrm{d}S = \frac{32}{\pi} \int u \mathrm{d}S.$$

Površina je v tem primeru enaka $S_0 = \frac{1}{2}\pi|_{r=1}$. Ker v tem primeru nimamo kvadratne mreže, ne moremo tako enostavno določiti koeficienta. Po kratkem razmisleku vidimo, da se nam ponuja enostavna rešitev; koeficienti b_i so namreč ravno vsota površin trikotnikov po celotnem območju, vsebujejo pa še faktor 1/3, saj s štetjem površin piramide po celem območju vsak trikotnik v povprečju štejemo 3 krat. Zapišemo lahko torej:

$$C = \frac{32}{\pi} \sum_{i=1}^{N_p} a_i b_i.$$

Koeficient sem izvrednotil v vsaki iteraciji, oglejmo si njegovo konvergenco pri nekaj različnih vrednostih N:

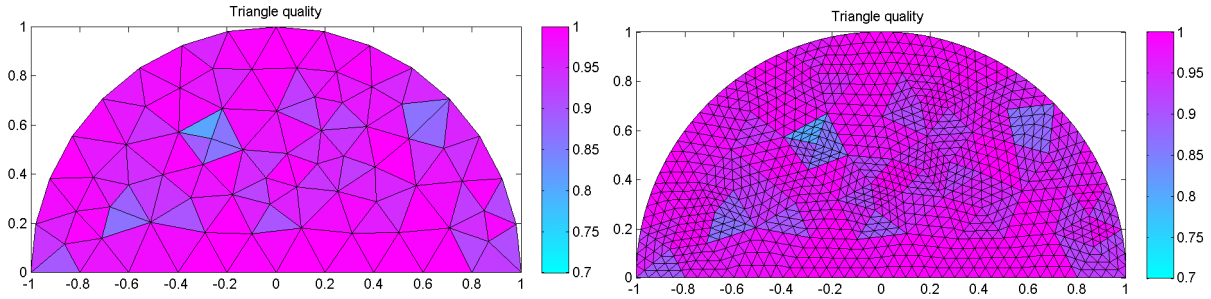


Leva slika prikazuje konvergenco Poiseuillovega koeficienta, desna pa njegovo natančnost v primerjavi z zadnjo vrednostjo. Meja konvergence je bila natančnost 10^{-11}. Opazimo, da se končna vrednost koeficienta spreminja za različne vrednosti razdelitve. Sklepamo torej, da moramo za pravilno oceno koeficienta vzeti finejšo razdelitev mreže. Vrednost Poiseuillovega koeficienta ocenimo na:

$$C \approx 0.752. \tag{12}$$

5 Matlabova rešitev

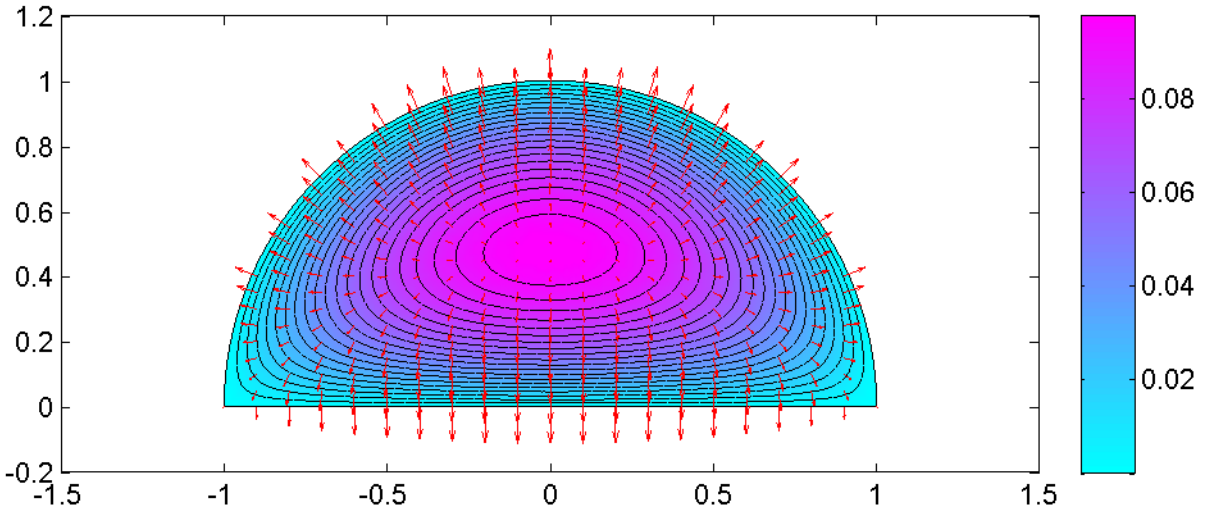
V tem razdelku si oglejmo še rešitve, ki nam jih matematično orodje Matlab ponudi samo. Najprej si oglejmo avtomatsko zgenerirane mreže:



Vidimo, da so v tem primeru vse finejše razdelitve osnovane na začetni. Vidimo lahko tudi barven prikaz kvalitete trikotnikov, kjer je najbolj kvaliteten trikotnik enakostraničen trikotnik. Ocena se izračuna preko formule:

$$q = \frac{4a\sqrt{3}}{h_1^2 + h_2^2 + h_3^2},$$

kjer je a površina trikotnika, h_1, h_2 in h_3 pa dolžine stranic. Oglejmo si še končno rešitev, kjer so prikazane tudi puščice v smeri gradienta:

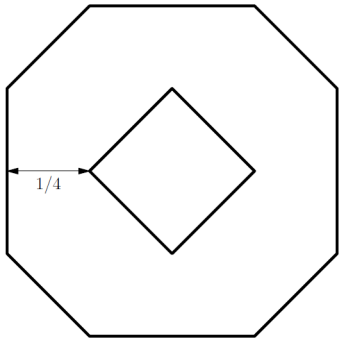


Vidimo, da je gradient smiselno usmerjen po celem območju, rešitev pa se sklada z našo.

Del II

Cev z izbranim presekom

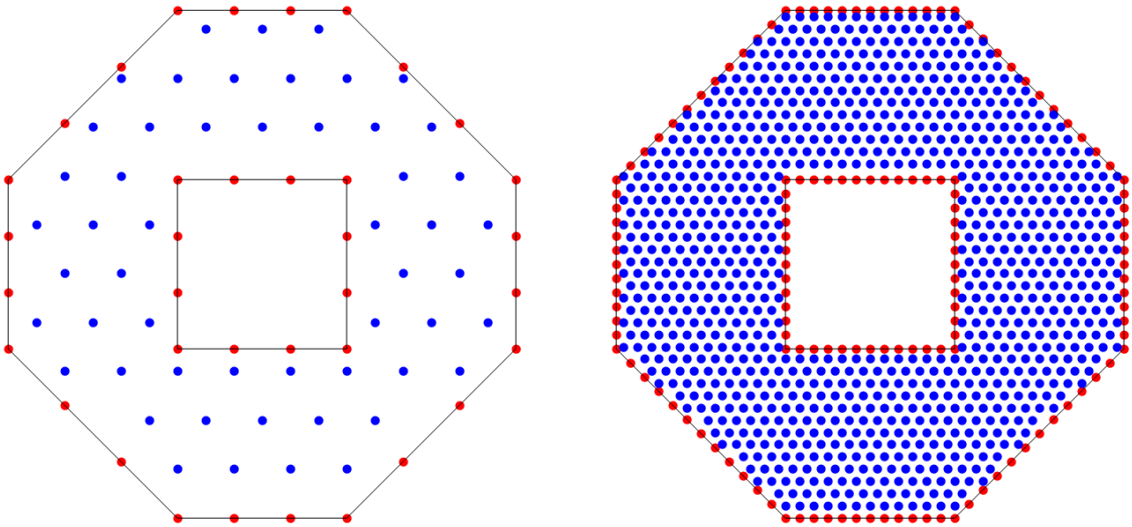
Z metodo končnih elementov smo morali izračunati še pretok skozi cev s presekom iz 5. naloge, ki je bila naslednje oblike:



Reševanje v tem primeru ni potekalo nič drugače kot prej, zato skočimo k rezultatom.

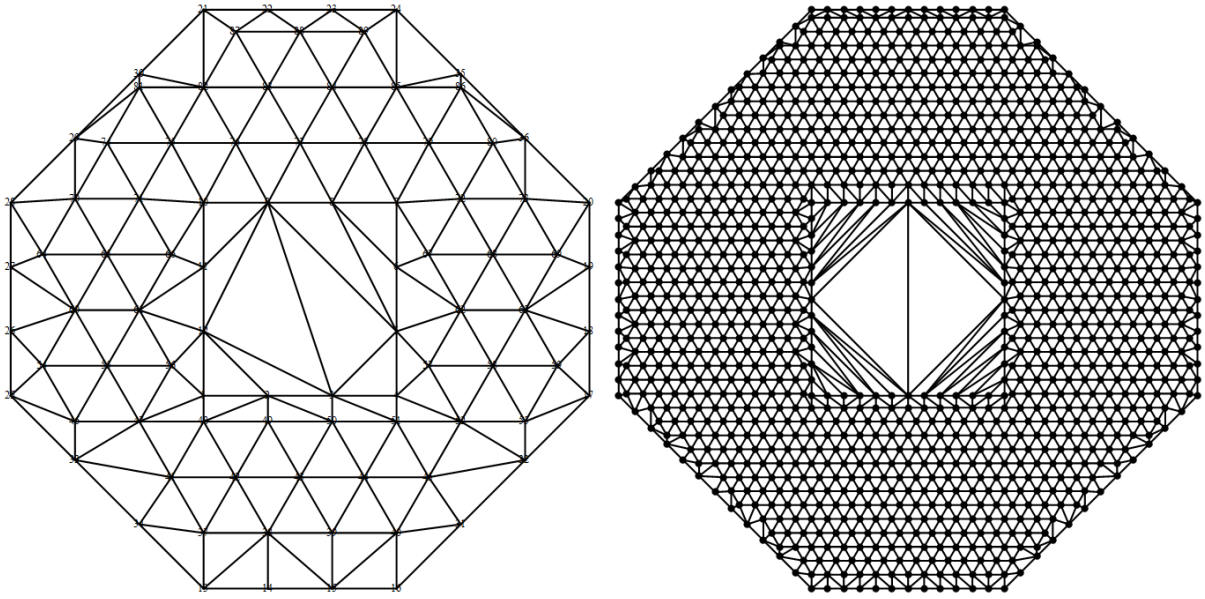
6 Rezultati

Tudi tu si oglejmo razdelitev mreže, kjer sem pozicije točk izbral sam:

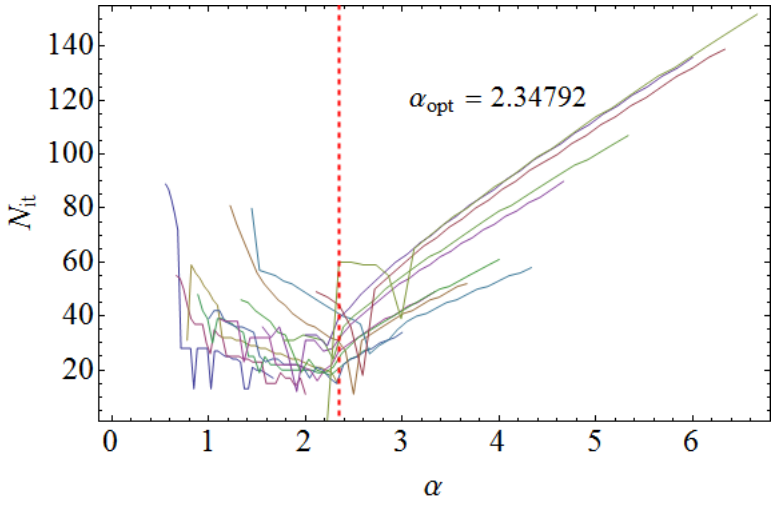


Z rdečo so označene točke na robu, z modro pa točke iz notranjosti.

Tudi tu imamo enakostranične trikotnike po območju, razen v sredini, kjer zahtevamo luknjo. Delaunayeva triangulacija nam da naslednjo trikotno mrežo:

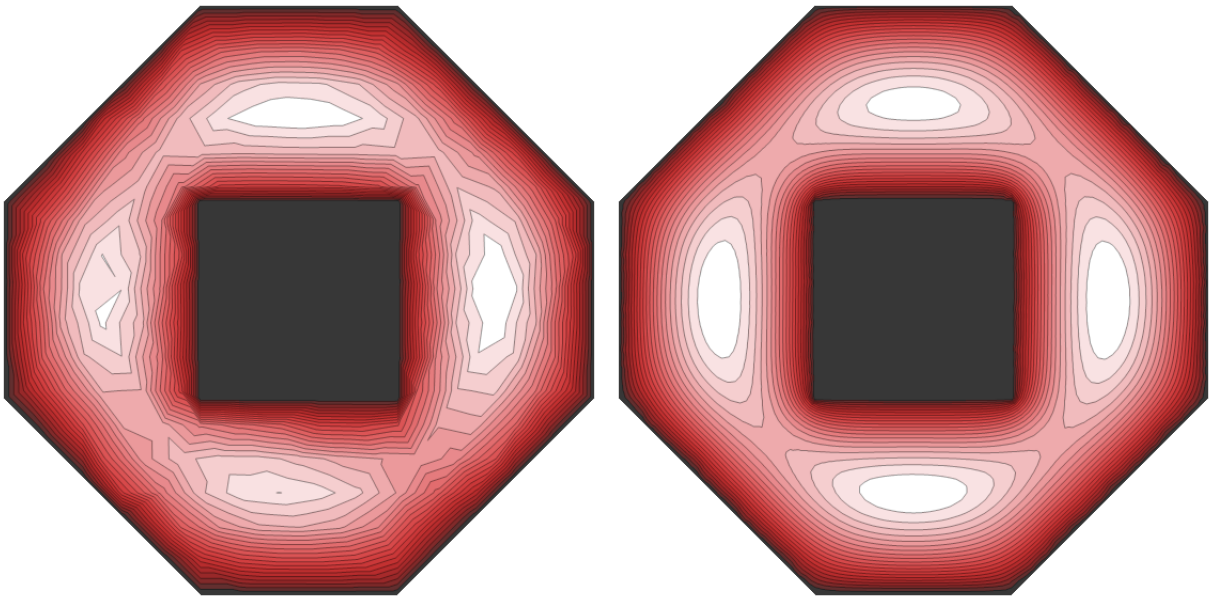


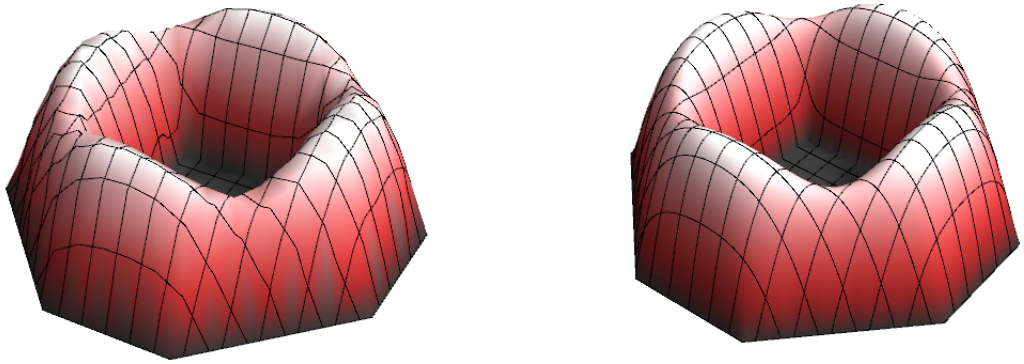
Opazimo, da imamo v tem primeru dodaten problem, saj se povezujejo trikotniki, ki so zunaj območja. Izkaže se, da če naša zanka teče le po točkah iz notranjosti, ta problem ne pride do izraza, zato lahko rešujemo na enak način kot prej. Zopet se najprej lotimo optimizacije, t.j. iskanja optimalnega pospeška α_{opt}. Postopek je enak kot prej, zato prilagam le končni rezultat:



Vidimo, da je v tem primeru vrednost tega parametra manj enostavno določljiva, vendar se bomo zadovoljili z grobo oceno $\alpha_{opt} \approx 2.35$.

Tako kot prej si sedaj lahko ogledamo rešitve za primer grobe ($N_p \approx 200$) in fine ($N_p \approx 3000$) razdelitve mreže. Mejna natančnost je bila 10^{-11}:

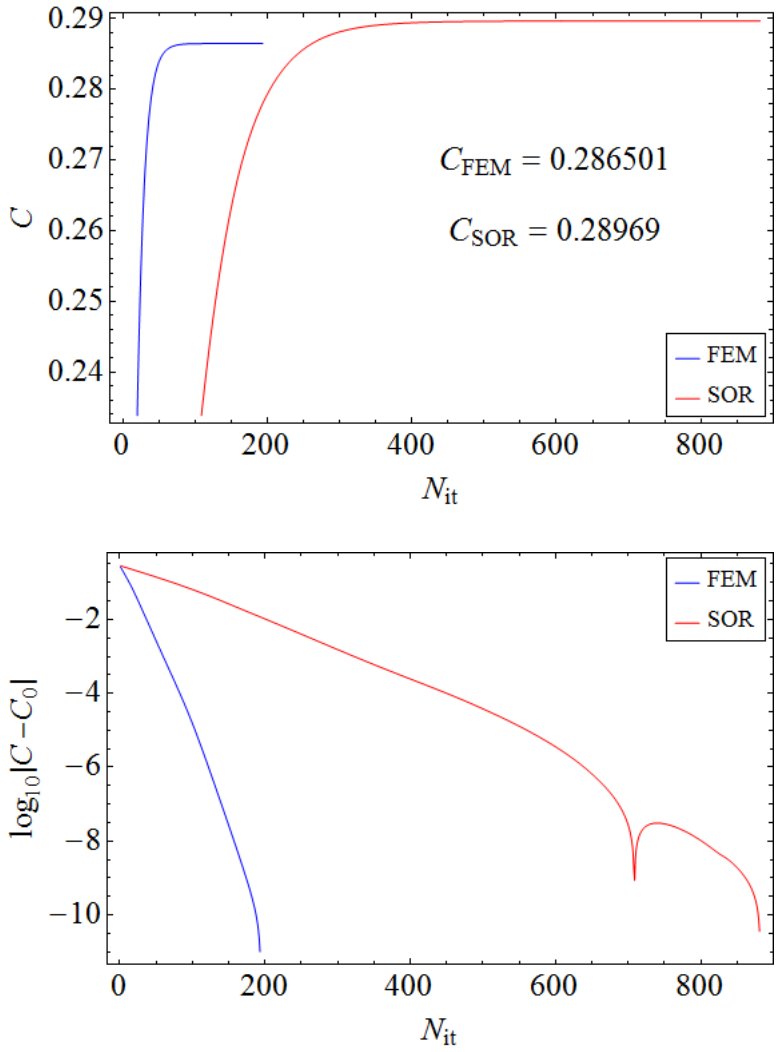




Tudi v tem primeru imamo smiselne rešitve, takšne kakršne smo imeli v 5. nalogi.

6.1 Poisseuillov koeficient

Na isti način izrčaunajmo še Poiseuillov koeficient, katerega vrednsot lahko primerjamo z vrednostjo iz 5. naloge. Površina je v tem primeru za kvadrat velikosti $[0,\,1]\times[0,\,1]$ enaka $S = 1 - \left(\frac{1}{3}\right)^2 - 4 + \frac{1}{2}\left(\frac{1}{3}\right)^2 = \frac{2}{3}$. Oglejmo si konvergenco Poiseuillovega koeficienta v primerjavi z isto količino, izračunano v 5. nalogi:



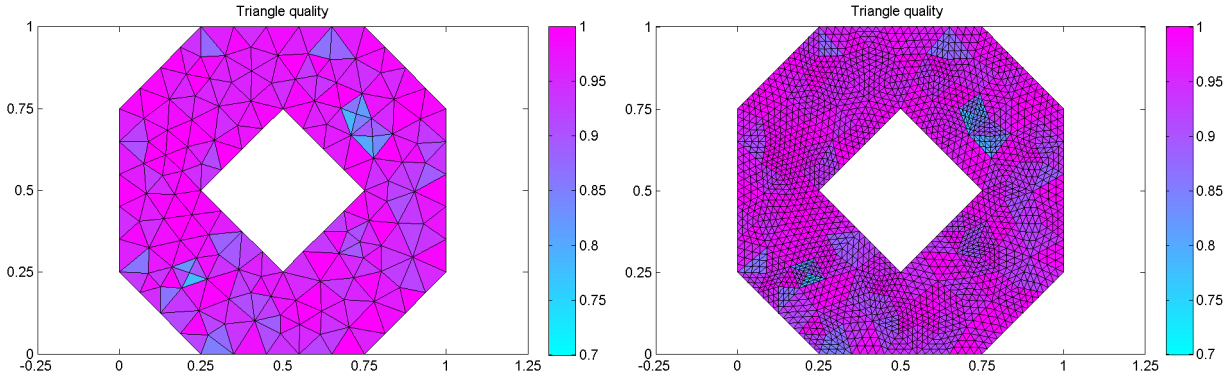
Opazimo, da je med njima razlika v vrednosti, saj je vrednost, dobljena z metodo FEM nekoliko manjša. Vrednost za SOR je bila izračunana na mreži 200×200, medtem ko je bilo število točk pri metodi FEM za namen Poiseuillovega koeficienta enako $N_p \approx 6500$. Ne smemo biti preveč pesimistični, saj sta obe metodi ujeli vrednost koeficienta na 2 decimalni mesti. Vrednost koeficienta ocenimo na:

$$C \approx 0.285. \tag{13}$$

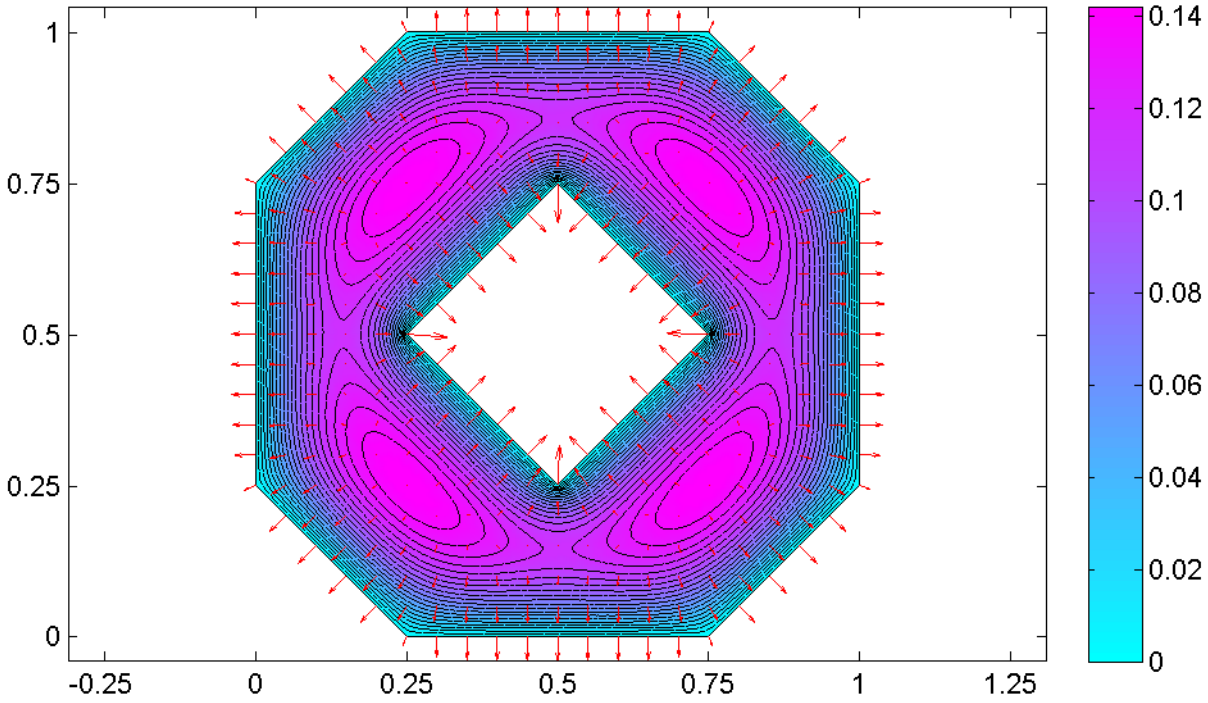
Vidimo tudi, da je metoda FEM opazno hitreje skonvergirala do poljubne natančnosti, vendar nisem prepričan, katera od njih se bolj približa pravi vrednosti.

7 Matlabova rešitev

Matlab nam lahko reši tudi tak problem. Najprej si oglejmo avtomatsko zgenerirane mreže:



Vidimo, da so tudi tu vse finejše razdelitve osnovane na začetni. Prikazana je tudi kvaliteta trikotnikov. Oglejmo si še končno rešitev, kjer so prikazane tudi puščice v smeri gradienta:



Tudi ta rešitev se sklada z našo.