207: Metoda končnih elementov - Poissonova enačba

Peter Rupnik

14. april 2021

1 Nalogi

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

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

Približno rešitev zapišemo kot linearno kombinacijo poskusnih funkcij

$$u = \sum_{i=1}^{N} a_i \phi_i$$

Zahteva, naj bo variacija funkcionala (1) enaka nič, vodi do sistema enačb za koeficiente a_i

$$\begin{array}{l} \sum_{j=1}^{N} A_{ij} a_j = b_i \\ A_{ij} = \langle \nabla \phi_i, \nabla \phi_j \rangle \,, \quad b_i = \langle f, \phi_i \rangle \end{array}$$

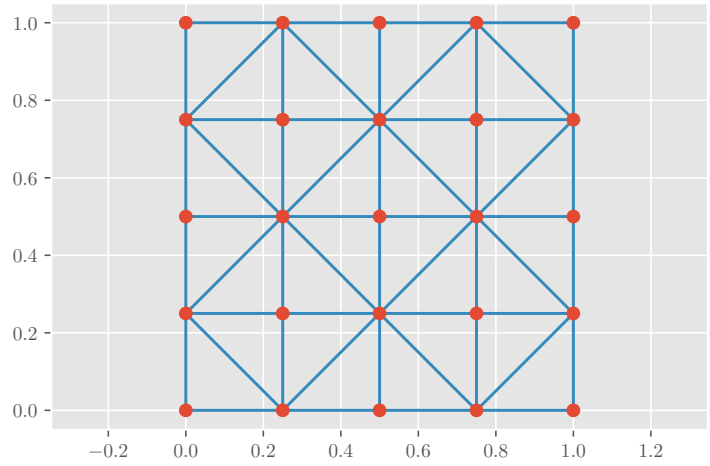
V metodi končnih elementov razdelimo definicijsko področje z mrežo točk in zveznic med njimi v same trikotnike. Za funkcije ϕ_i izberemo piramidalne funkcije, ki imajo v točki i vrednost $1,$ v vseh drugih točkah vrednost 0, med oglišči pa se spreminjajo linearno. Funkciji nesosednih točk sta vedno ortogonalni, 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. 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

$$S_r = \frac{1}{2} \mid x_i \left(y_j - y_k\right) + \text{ cikl. perm. } \mid$$

Velikost $\nabla \phi_i$ na r-tem trikotniku je $d_{j,k}/2S_r$, kjer je $d_{j,k}$ razdalja točk j in $k.A_{ij}$ je vsota skalarnih produktov gradientov $\left(\left(x_j - x_k\right)\left(x_k - x_i\right) + \left(y_j - y_k\right)\left(y_k - y_i\right)\right)/4S_r$, po trikotnikih ob zveznici $i - j$ Za diagonalne elemente A_{ii} se gornji izraz skrči na $d_{j,k}^2/4S_r$, seštet po vseh trikotnikih s točko iv oglišču. Podobno preprosto se izraža $\langle f, \phi_i \rangle$ v našem primeru, ko je $f = -1$ konstanta. Preostane integral linearne funkcije po trikotniku, ki da ravno $\frac{S_r}{3}$. Desne strani b_i v sistemu (3) so torej enake $-\frac{1}{3}$ vsote ploščin trikotnikov ob točki i. Natančnost se oceni z največjo stranico in s kotom, ki je najdlje od $\frac{\pi}{3}$. 1. Izračunaj pretok skozi polkrožno cev. 2. Za primerjavo z metodo SOR izračunaj pretok po cevi iz 5. naloge.

2 Prva naloga

Pričel sem s tipanjem problema za kvadratno cev. Za začetek sem si izbral klasično mrežo s 5 točkami v vsaki dimenziji, ki so bile razporejene ekvidistančno v x in y. Te točke sem uporabil kot input metode `scipy.spatial.Delaunay`, rezultat prikazujem spodaj:

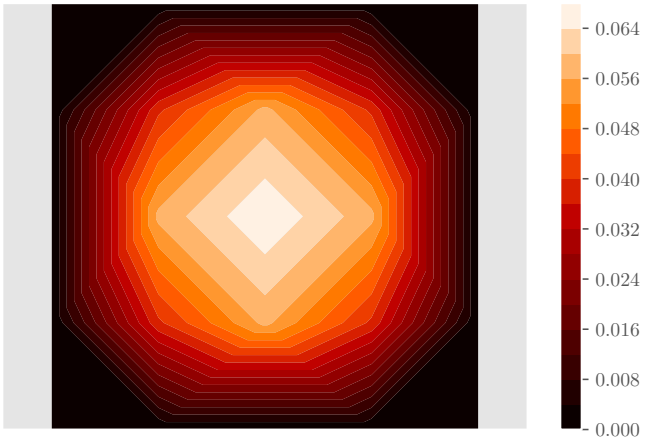


Na ta način dobim $5 \times 5 = 25$ točk in 32 elementov (*simplices*).

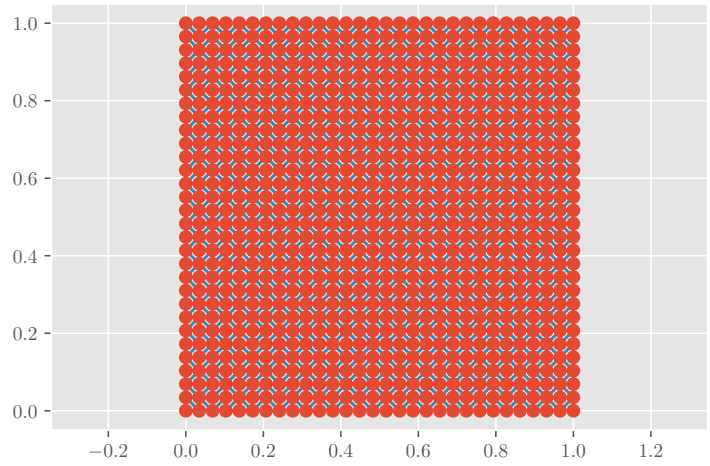
Pripravil sem si globalno togostno matriko S in vektor $\mathbf{g}$, elemente katerih sem sprva postavil na nič. Sledila je iteracija po elementih, za vsak element sem izračunal lokalne prispevke k S in $\mathbf{g}$ in jih prištel k njima. Ko sem pravilno našel robne točke in jih pripisal robne pogoje, sem lahko rešil sistem

$$S\mathbf{u} = \mathbf{g}.$$

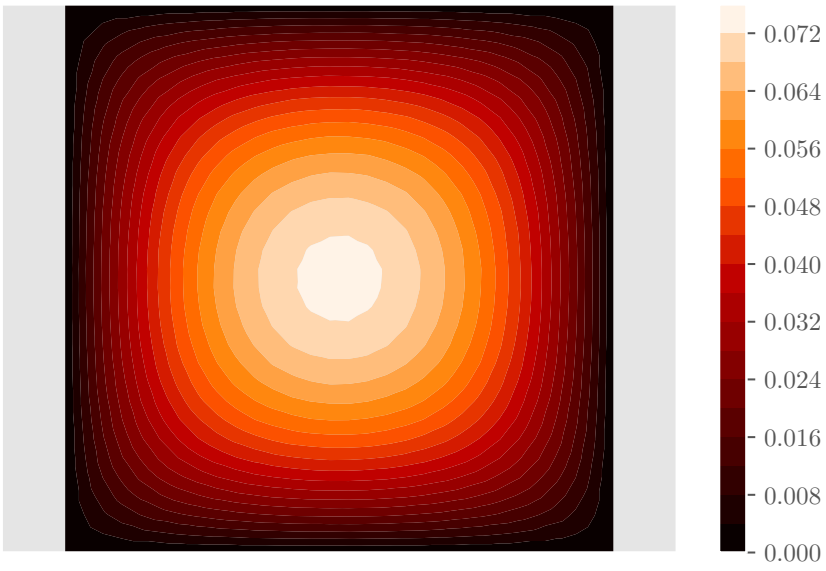
Rezultirajoče hitrostno polje v kvadratni cevi prikazujem spodaj.



Rezultat izgleda ravno nedoločen, da sem se odločil poskusiti z večjim številom točk. Ko ponovim isto operacijo s 40 točkami v vsaki dimenziji, dobim mrežo, ki je ravno še dovolj redka, da se vidijo posamezne točke:

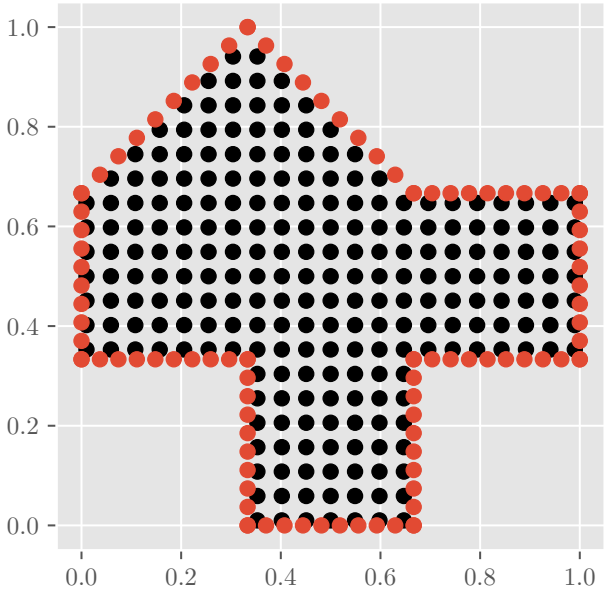


V tem primeru izgleda hitrostni profil v cevi bolj domače:

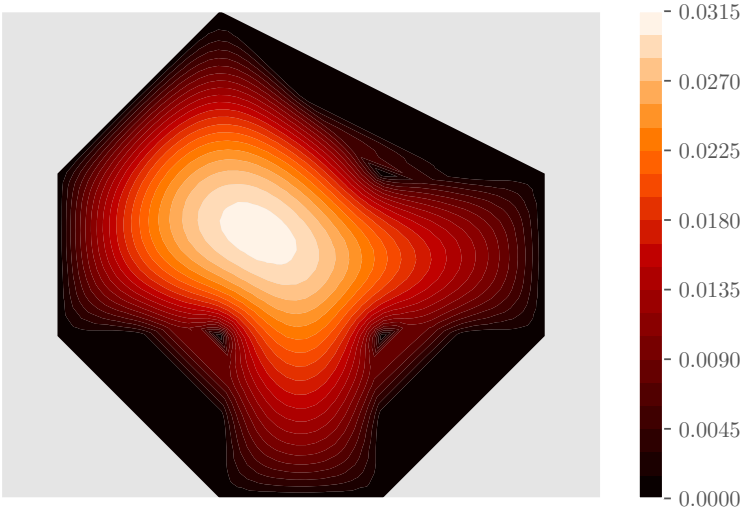


Uverjen, da 'delam napore' v pravo smer, sem se namenil obdelati hiškasto cev iz naloge 5. Pripravil sem si nekaj pomožnih funkcij, ki so mi naračunale točke na robu cevi, izračunale, ali je neka točka znotraj lika, in naračunale tudi ekvidistančno razporejene točke znotraj cevi.

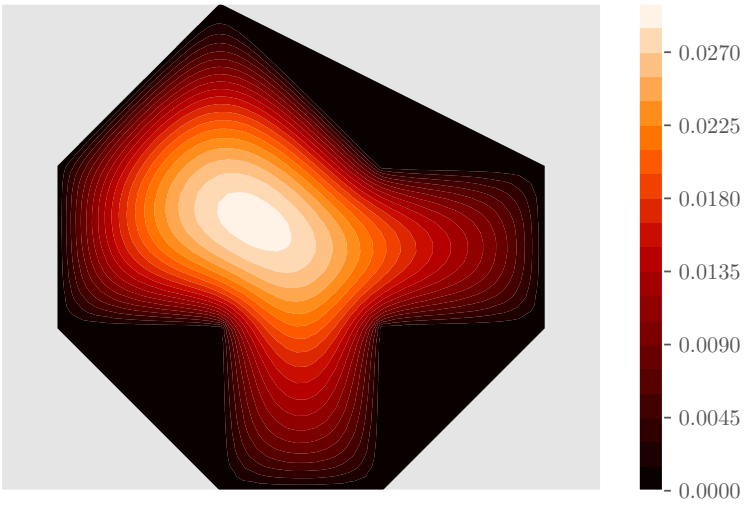
Vozlišča in elementi so prikazani na spodnji sliki. Čeprav izgleda, kot da se točke na robu prekrivajo z ekvidistanto pozicioniranimi notranjimi točkami, sem poskrbel, da temu ni tako, saj bi v nasprotnem primeru imel bistveno več težav pri vzpostavljanju robnih pogojev, tako pa sem lahko zaradi ločenih provenienc notranjih in robnih točk lažje kontroliral, katere točke so postavljene na rob.



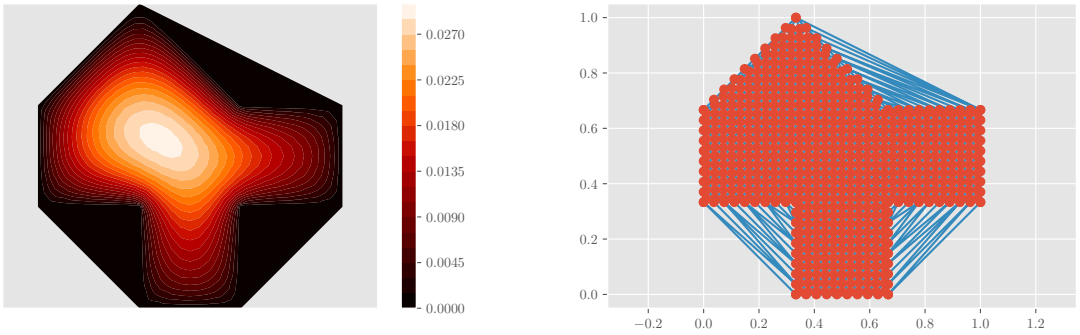
Nadaljevanje je precej podobno kot v preludiju zgoraj. Za razliko od kvadratne cevi je tu zaradi konkavne geometrije potrebno več pozornosti na ostrih robovih cevi. Za ilustracijo prikazujem skrajno asimetričen primer, kjer notranjost cevi popišem s 50 točkami v vsaki dimenziji, rob pa s samo tremi na vsakem ravnem segmentu. Tedaj opazimo 'tuneliranje' skozi robove, ki so preredko definirani:



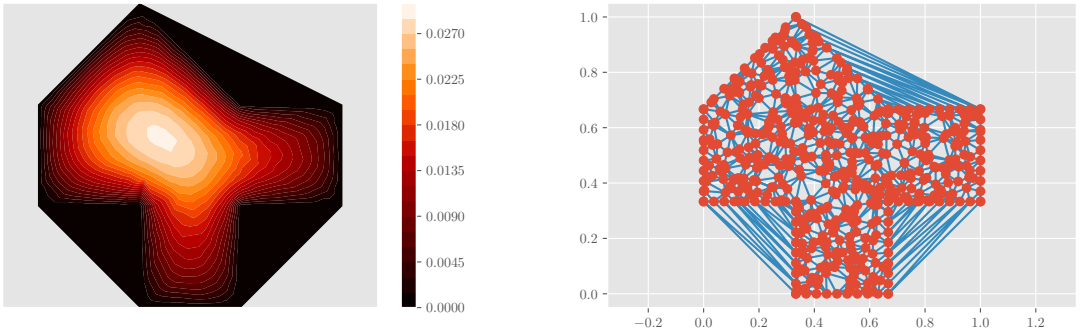
Če rob podajam vsaj tako pogosto kot notranjost, artefakt izgine.



Poanta končnih elementov je ravno v dejstvu, da ne potrebujemo ekvidistančnih točk, čeprav se pri meni neprestano pojavljajo. Iz akademske vedoželjnosti sem si pogledal še rezultate pri naključni razporeditvi notranjih točk, pri čemer držim celotno število točk konstantno:



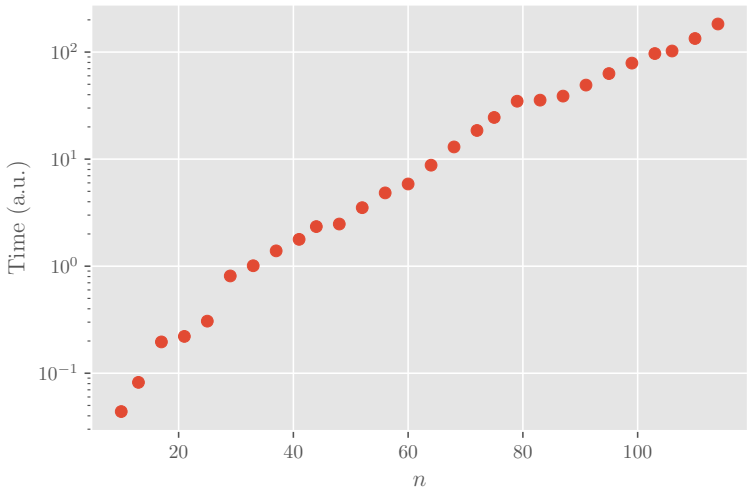
Hitrostno polje in točke pri ekvidistančnem razvrščanju notranjih točk. V liku je 484 notranjih točk.



Hitrostno polje in točke pri naključnem razvrščanju notranjih točk. V liku je 500 notranjih točk, a zaradi naključne razporeditve mestoma izgleda, kot da je skupno število točk manjše kot pri ekvidistanti razporeditvi.

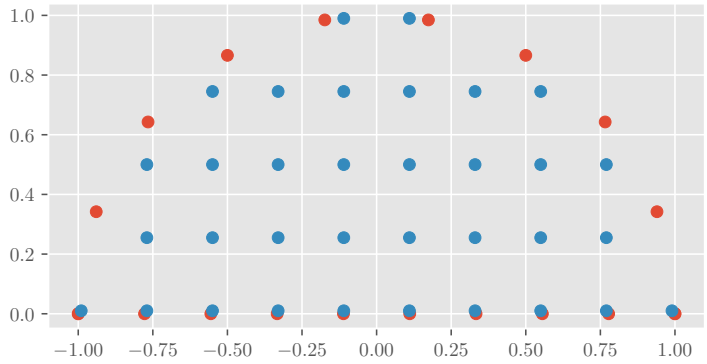
Po optični oceni rezultatov bi zaključil, da se vseeno splača obdržati ekvidistančno razporejanje točk. V primeru, da lokalno potrebujemo večjo gostoto točk, bi dodatne točke veljalo razporediti čim bolj enakomerno in ne le pometati naključno.

V pehanju za čim lepšimi sličicami sem opazil, da pri več točkah čas reševanja raste. Čas izvajanja sem izrisal za različne parametre n, kar pomeni, da je pri posameznem n število notranjih točk $\mathcal{O}(n^2)$, število robnih točk pa sem držal fiksirano:

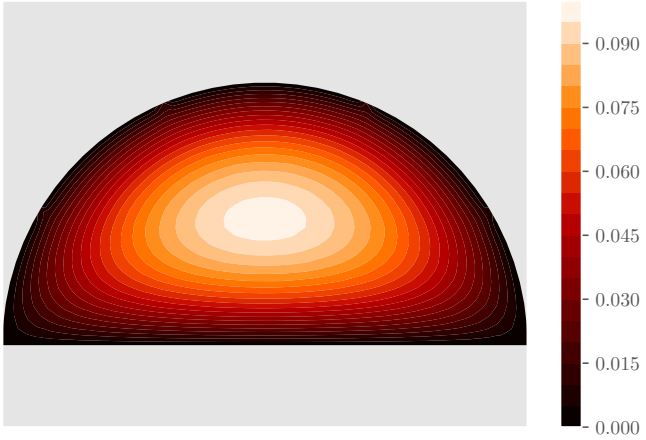


3 Druga naloga

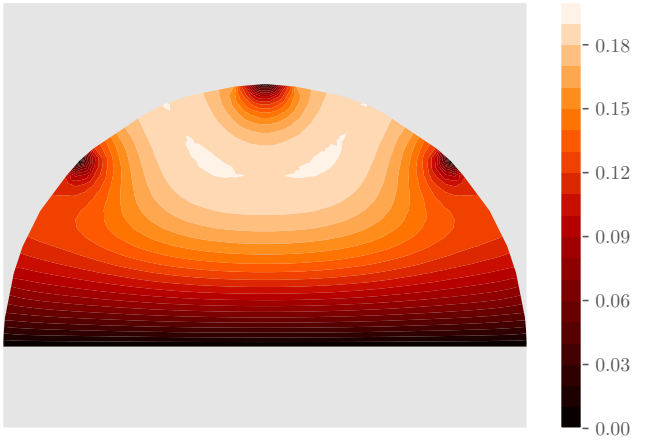
Pričel sem s pripravo funkcij za generacijo notranjih in robnih točk. Spet sem posegel po ekvidistančni mreži, primer katere podajam s spodnjo sliko.



Linearna gostota točk po spodnjem robu in po krožnem loku je enaka in nastavljiva, za notranje točke pa spet formiram ekvidistančne kartezične točke, ki jih filtriram s pogojem, da morajo biti znotraj cevi, kar je v tem primeru dosti bolj enostavno kot v prejšnjem primeru.



Tudi tukaj opazimo zanimiv primer, ko je uporabljenih premalo robnih točk. Za ilustracijo sem ponovno generiral grotesken primer, ki ima po robovih samo 10 točk, notranje točke pa so diskretizirane s 30 točkami v vsaki dimenziji:



Kot pri zgornji cevi tudi pri polkrožni očitno velja po robu postaviti dovolj točk, da jih notranje točke ‘vidijo’ kot rob in ne le kot izolirane točke, saj v tem primeru model ne popisuje več realnega stanja.