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

Modelska Analiza 2

8. naloga - Metoda končnih elementov: lastne rešitve

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

Ljubljana, 15.4.2014

Naloga:

Pri tej nalogi smo ponovno koristili metodo končnih elementov (FEM). Tokrat smo iskali lastne rešitve polkrožne opne. V prvem delu naloge smo območje ponovno prekrili s trikotniki, kjer smo vsaki točki priredili piramidalno funkcijo. Končni sistem enačb smo prevedli na matrično obliko in poiskali lastne vrednosti in lastne nihajne načine. V drugem delu naloge smo za poskusne funkcije uporabili Galerkinov nastavek, s kombinacijo katerih lahko sestavimo lastne nihajne rešitve. Izkaže se, da je drugi način precej hitrejši in predvsem precej natančnejši.

Del I
Cev s polkrožnim presekom

1 Naloga

Z metodo končnih elementov poišči nekaj najnižjih lastnih vrednosti polkrožne opne. Z ekstrapolacijo rešitev za različne delitve radija oceni natančnost metode. Lastne vrednosti so seveda kar kvadrati ničel Besselovih funkcij.

2 Uvod

Variacijski funkcional za reševanje problema lastnih vrednosti $\nabla^2 u + k^2 u = 0$, zapišemo v obliki

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

Približno rešitev tako kot v prejšnji nalogi nastavimo kot

$$u = \sum_{i=1}^{N} a_i w_i, \tag{2}$$

kjer so w_i piramidalne funkcije s točko i v središču . Stacionaren pogoj funkcionala vodi do homogenega sistema enačb za koeficiente a_i :

$$\sum_{j=1}^{N} \left(A_{ij} - \lambda B_{ij}\right) a_j = 0, \tag{3}$$

$$A_{ij} = \langle \nabla w_i,\, \nabla w_j\rangle \qquad B_{ij} = \langle w_i,\, w_j\rangle. \tag{4}$$

Koeficiente matrike A_{ij} smo določili že pri prejšnji nalogi. Za diagonalne elemente prekrivalne matrike $\mathbf{B}$ dobimo $B_{ii} = \sum_r S_r/6$, seštet po vseh trikotnikih s točko i v oglišču. Za izvendiagonalne elemente dobimo $B_{ij} = \sum_{i\sim j} S_r/12$, seštet po vsakem od trikotnikov ob zveznici $i - j$. Matrika B je pozitivno definitna. Matriko $\mathbf{B}$ lahko na več načinov razstavimo, da s čim manj dela dobimo matriko $\mathbf{C} = \mathbf{B^{-1}A}$, v simetrični obliki, katere lastne vrednosti in vektorje iščemo.

3 Reševanje

Nalogo sem tokrat reševal v matematičnem orodju Mathematica. Mreže trikotnikov, ki smo jih uporabljali v prejšnji nalogi, lahko uporabimo tudi tu. Na vseh točkah sem uporabil funkcijo DelaunayTriangulation[] iz paketa ComputationalGeometry. Ta funkcija vrne relacije med povezanimi točki v zelo uporabnem formatu naslednje oblike:

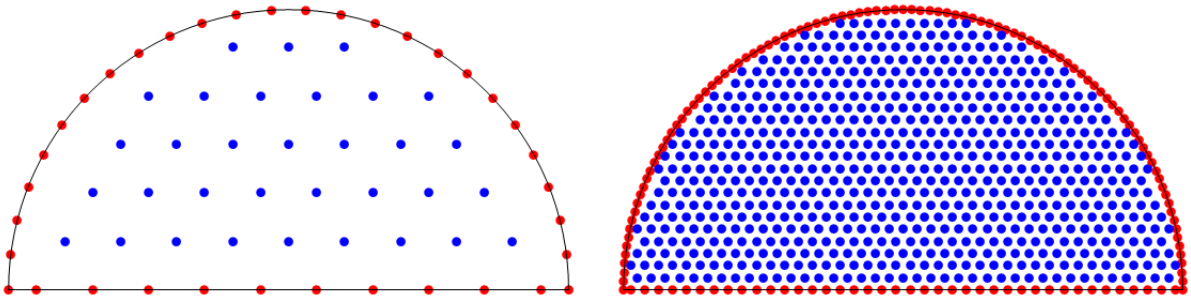
$$\begin{gathered}\vdots\\ \{i,\, \{j_1,\, j_2,\, \ldots j_{N_r}\}\},\\ \vdots\end{gathered} \tag{5}$$

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. Prav tako pa je bil v večini narejen že algoritem za iskanje vrednosti a_j, le prevesti sem moral sistem na matrike, saj sem v prejšnji nalogi reševal po principu SOR.

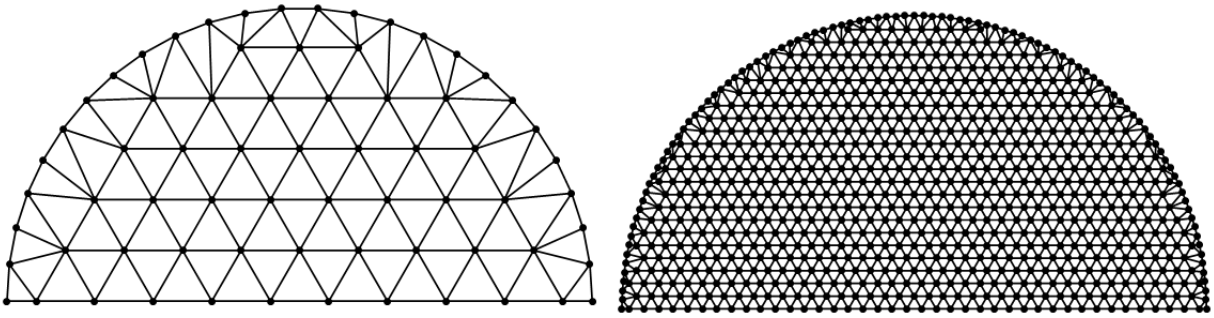
Ko sem imel enkrat mreže ustvarjene in kodo napisano, sem hitro lahko izračunal lastne vrednosti in vektorje ter jih predstavil spodaj. Z razcepom Choleskega ali diagonalizacijo matrike $\mathbf{B}$ se nisem ukvarjal, saj je moja matrika vsebovala le aktivne točke, torej brez robnih točk. Sklepam, da za to tak postopek tudi ni deloval, kot sem najprej poskusil. Rešitve sem lahko primerjal z analitično funkcijo, ki se glasi

4 Rezultati

Za začetek si oglejmo mreže, ki sem jih uporabil tako v tem kot prejšnjem primeru. Spodaj je primer generiranih točk in triangulacija točk za primere majhnega in velikega števila točk:

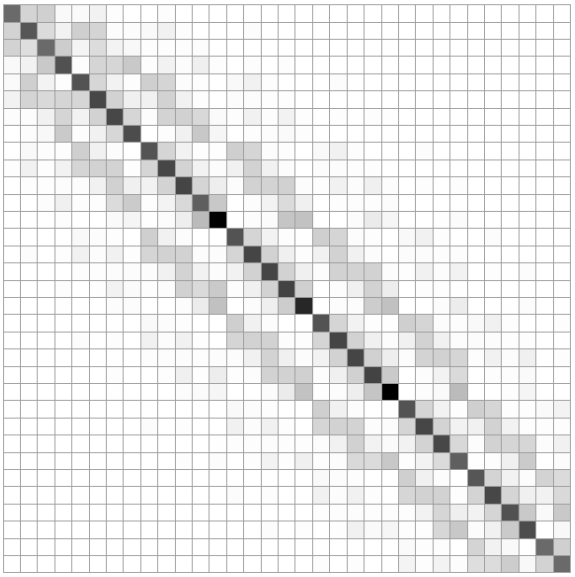


Z rdečo so označene točke na robu, z modro pa točke iz notranjosti. Vidimo, da je sredinsko območje enakomerno posejano, pri robovih pa se pravilnost trikotnikov poruši z zahtevo po opisu roba. Delaunayeva triangulacija nam da naslednjo trikotno mrežo:

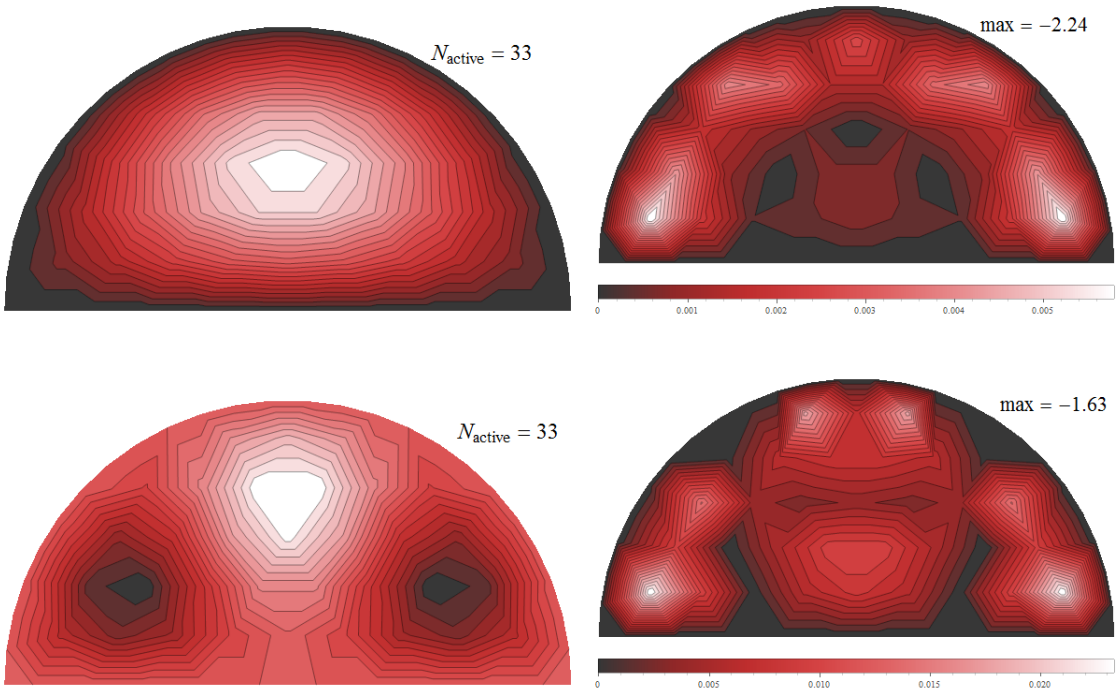


Dobimo trikotnike v takšnem formatu, ki sem ga opisal zgoraj.

Za občutek strukture matrik si oglejmo strukturo matrike $\mathbf{C}$ pri nekem danem številu točk:

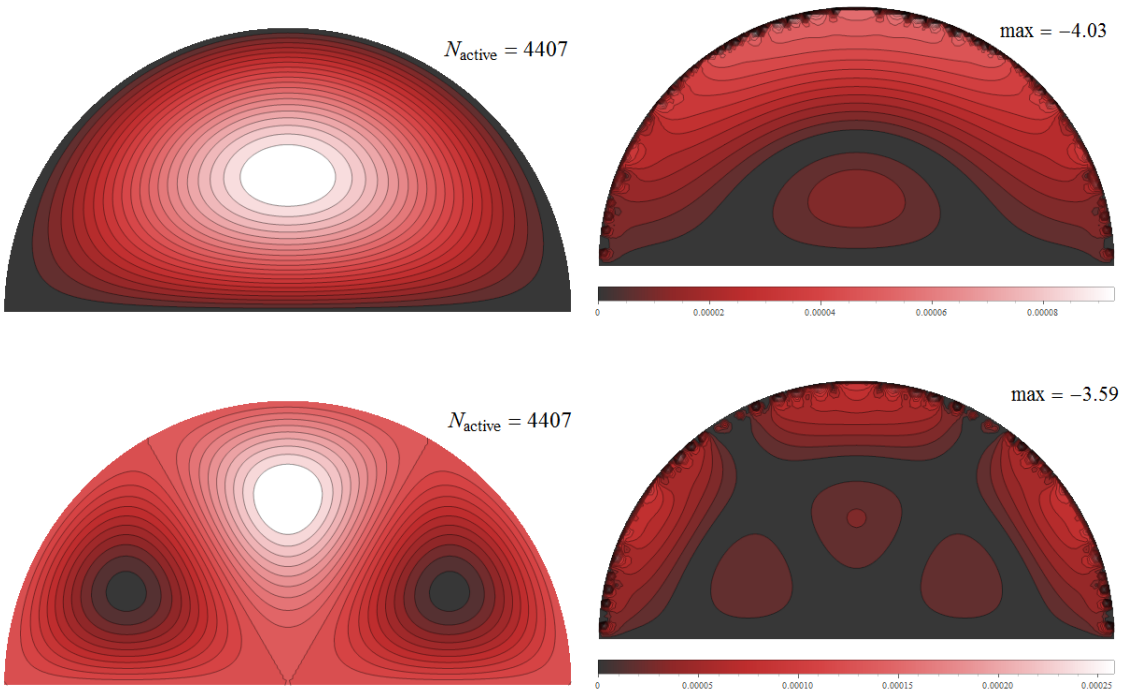


V tem primeru vidimo, da poleg tega, da so rešitve elegantnejše oblike, so t
Oglejmo si lastne načine nihanja, ki jih dobimo na taki mreži. Če vzamemo majhno število točk, bo lastna rešitev seveda popačena, vseeno pa lahko v definiranih točkah preverimo natančnost dobljene rešitve, tako da primerjamo z analitično funkcijo:

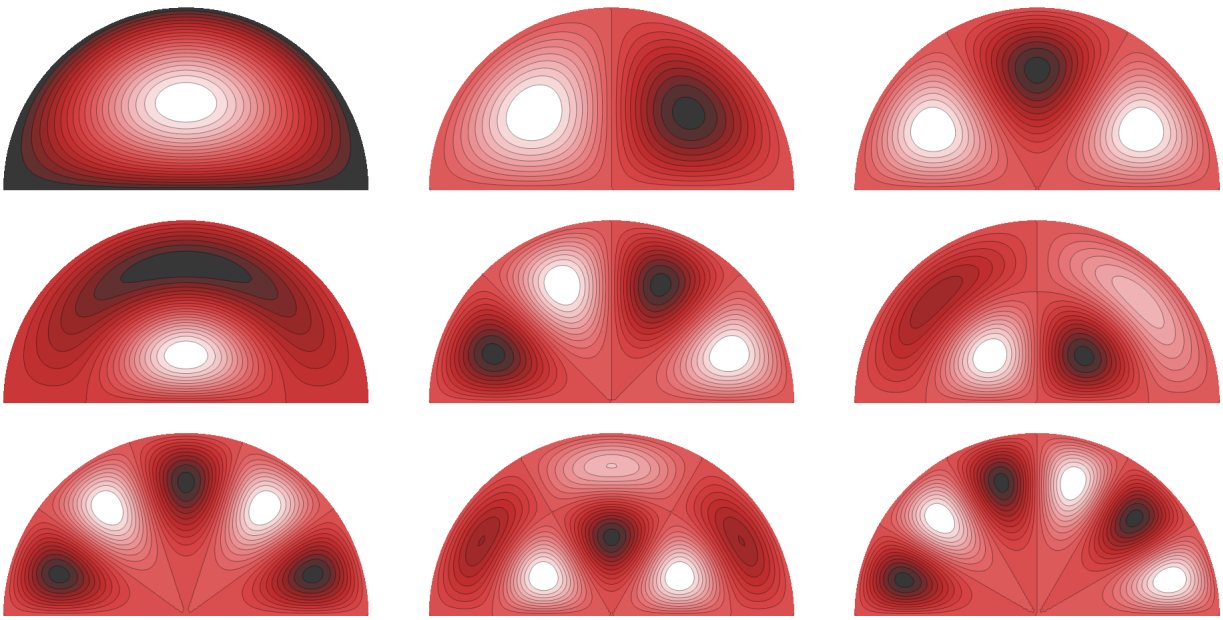


Levi sliki prikazujeta dve poljubni lastni funkciji, izračunani pri številu aktivnih trikotnikov $N = 33$, desni sliki pa prikazujeta napako po celem območju. Vidimo, da je tudi pri malem številu točk natančnost nekako zadovoljiva.

Oglejmo si sedaj še rešitve za primer večjega števila aktivnih trikotnikov $N = 4407$:

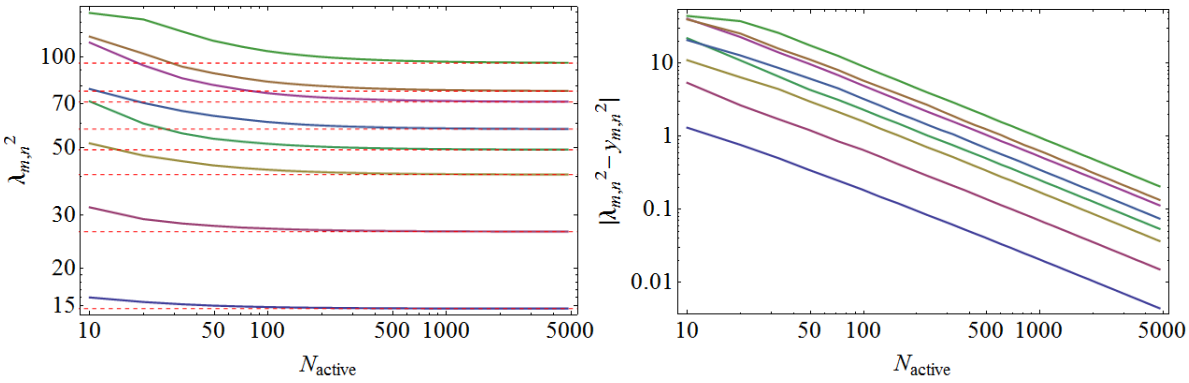


V tem primeru vidimo, da poleg tega, da so rešitve elegantnejše oblike, so tudi natančneje določene. Oglejmo si še večjo zbirko lastnih rešitev:



Res prepoznamo oblike, ki so nam že precej znane.

Končno si lahko ogledamo še lastne vrednosti naših rešitev. Vemo, da morajo biti lastne vrednosti enake kvadratom Besselovih ničel. Oglejmo si kako dobro to velja, ko spreminjamo število aktivnih točk na naši mreži:

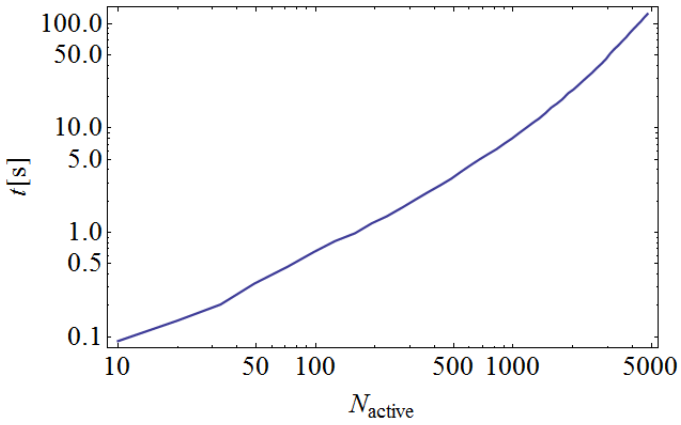


Opazimo, da se lastne vrednosti pri majhnem številu aktivnih točk precej razlikujejo od limitnih vrednosti. Zaradi pregrobe razdelitve mreže kaj drugega seveda ni pričakovati. Pri raznih številih točk si lahko ogledamo natančnost izračunane lastne vrednosti, vendar pa lahko gremo še korak dlje. Konvergenco lahko ekstrapoliramo, da dobimo natančnejše končne vrednosti. Izkazalo se je, da je funkcija, ki opisuje konvergenco, zelo čudna, saj sem moral podatke $\left(N,\, \lambda_{m,s}^2\right)$ logaritmirati, nato pa na te podatke prilagotiti eksponentno kriviljo. Za to, da je bil fit uspešen in natančen, sem vzel le podatke za $N > 3000$, saj so na začetku prisotni še neželeni prehodni pojavi. Mogoče to ni prava izbira, vendar druge rešitve nisem našel, ta pa se je obnašala zadovoljivo. Spodaj je prikazana tabela desetiških logaritmov odstopanj od pravih vrednosti kvadratov Besslovih ničel za nekaj različnih števil aktivnih točk in za primer ekstrapolacije:

$(m,\ n)$	$N = 229$	$N = 824$	$N = 1774$	$N = 3098$	$N = 4782$	$N \to \infty$	$y_{m,n}^2$
(1, 1)	-1.08	-1.61	-1.93	-2.17	-2.35	-4.2169	14.682
(2, 1)	-0.54	-1.07	-1.4	-1.63	-1.82	-4.17581	26.3746
(3, 1)	-0.15	-0.684	-1.01	-1.25	-1.43	-5.04841	40.7065
(1, 2)	0.0125	-0.518	-0.844	-1.08	-1.27	-3.05307	49.2185
(4, 1)	0.16	-0.374	-0.703	-0.94	-1.13	-3.27422	57.5829
(2, 2)	0.338	-0.196	-0.522	-0.759	-0.945	-2.99716	70.85
(5, 1)	0.431	-0.124	-0.443	-0.685	-0.871	-2.44082	76.9389
(3, 2)	0.601	0.0648	-0.262	-0.499	-0.685	-2.7497	95.2776

Vidimo, da smo z ekstrapolacijo v povprečju pridobili kar dve decimalni mesti in tako precej natančno določili lastne vrednosti.

Za konec si oglejmo še časovno potratnost programa za različne vrednosti števila aktivnih točk na mreži:



Čeprav imam precej zmogljiv računalnik in precej optimizirano kodo, je pri večjih vrednostih N porabljen čas še vedno velik. Te rezultate si je zanimivo zapomniti in jih primerjati s časovno potratnostjo naslednje metode

Del II
Galerkinov nastavek

1 Naloga

Oceni lastne frekvence opne z Galerkinovim nastavkom potenčnih funkcij

$$g_{m,n}(r,\varphi) = r^{m+n}\,(1-r)\,e^{im\varphi}, \qquad 1 \le m \le m_{max}, \qquad 0 \le n \le n_{max}. \tag{6}$$

2 Reševanje

Kot smo videli že v prejšnji nalogi, za poskusne funkcije ni nujno, da so vse med seboj ortogonalne, vendar pa morajo biti vsaj takšne, da zadoščajo robnim pogojem. Ponovno se lahko poslužimo zgornjega poteka reševanja, le da se tokrat matrični elementi izračunajo drugače. Velja:

$$A_{ij} = (\nabla\phi_i,\, \nabla\phi_j) = \left(\nabla^2\phi_i,\, \phi_j\right), \tag{7}$$

$$B_{ij} = -\left(\phi_i,\, \phi_j\right), \tag{8}$$

$$(p,\, q) = \int_0^1 \int_0^\pi p(r,\varphi) q(r,\varphi) \mathrm{d}\varphi r \mathrm{d}r, \tag{9}$$

kjer smo galerkinovo funkcijo $\phi_{m,n}$ prevedli v $i-$to poskusno funkcijo ϕ_i, tako da je matrika elementov $A_{i,j}$ v resnici bločna matrika $A_{(m,n),(m',n')}$.

Formuli za izračun matrik $\mathbf{A}$ in $\mathbf{B}$ sta nasledji:

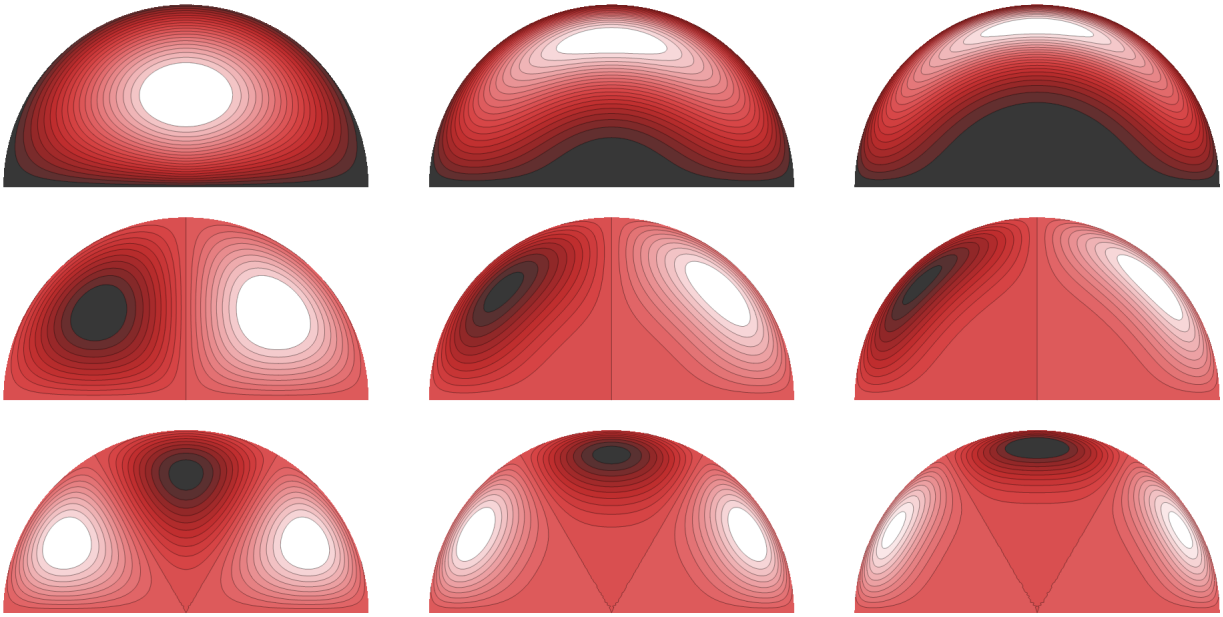
$$\begin{aligned} A_{mn,m'n'} &= \frac{-2m\pi\left(n+2m(1+2m+n)+n'+2(m+n)n'\right)}{4m(2m+n+n')(1+2m+n+n')(2+2m+n+n')}\delta_{m,m'}, \\ B_{mn,m'n'} &= \frac{-2m\pi}{2m(2+2m+n+n')(3+2m+n+n')(4+2m+n+n')}\delta_{m,m'}. \end{aligned} \tag{10}$$

Ko izračunamo te bločne matrike matrike, se poslužimo istega postopka kot prej in izračunamo lastne vrednosti in lastne vektorje. Lastni vektorji v tem primeru predstavljajo koeficiente razvoja, ki stojijo pred galerkinovimi funkcijami, tako da velja:

$$u = \sum_{m,n} a_{m,n} g_{m,n}(r,\varphi). \tag{11}$$

3 Rezultati

V tem primeru mreže seveda nismo potrebovali, saj so naše območje opisovale funkcije, ne točke. Za začetek si oglejmo strukturo Galerkinovih funkcij:



Res je očitno, da se razlikujejo od lastnih funkcij polkroga, vidimo pa seveda nekaj podobnosti, tako da lahko s pravo kombinacijo sestavimo lastne funkcije polkroga.

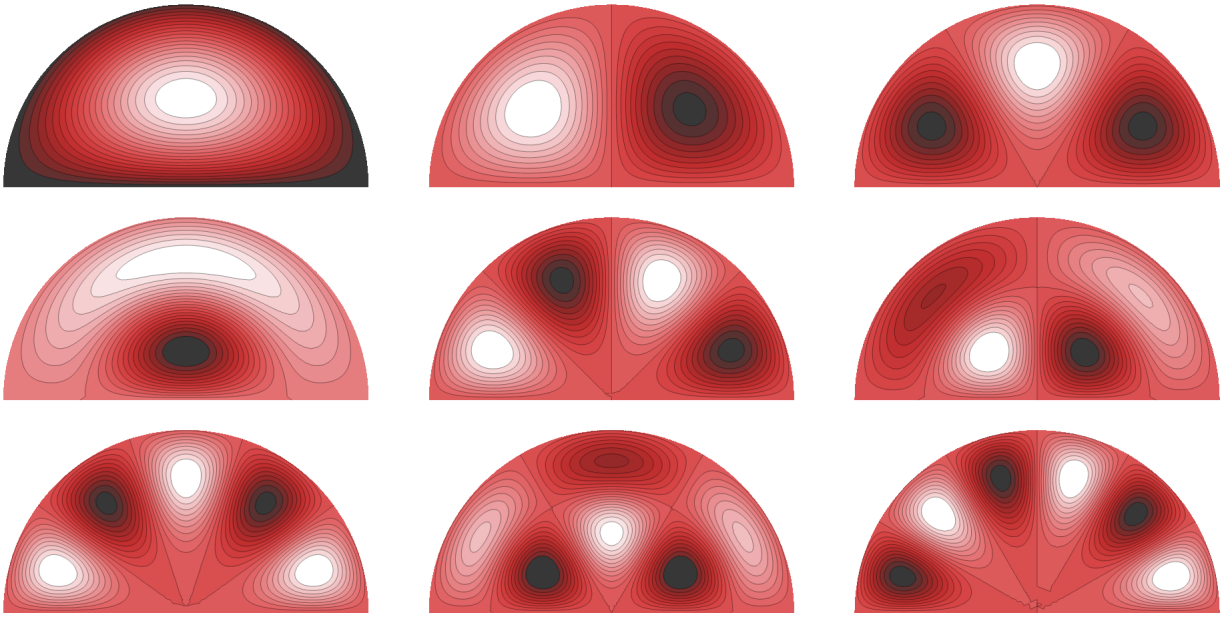
Tudi za ta primer si oglejmo strukturo matrike **A** (levo) in matrike **C** (desno):

Vidimo, da matriki nista veliki in imata relativno malo členov, kjer je $m_{max} = 3$ in $n_{max} = 3$, pa tudi njuna struktura je precej bolj drugačna kot v prejšnjem primeru. Izkaže se, da s takšnimi nastavki ne rabimo veliko členov, saj že par funkcijo dovolj potrebno opiše naše lastne rešitve. Velja celo, da preveč členov pokvari situacijo, saj se pri večjih potencah Galerkinove funkcije vedno bolj prekrivajo med sabo in niso več optimalne za uporabo v razvoju. Oglejmo si najprej natančnost lastnih vrednosti, dobljenih z Galerkinovimi funkcijami, da preverimo, kateri števili m_{max} in n_{max} sta najbolj optimalni:

(m, n)	$(M, N) = (5, 2)$	$(5, 5)$	$(8, 2)$	$(8, 5)$	$(8, 7)$	$(10, 5)$	$y_{m,n}^2$
(1, 1)	-2.95	-7.03	-2.95	-7.03	-5.51	-7.03	14.682
(2, 1)	-2.02	-6.59	-2.02	-6.59	-4.65	-6.59	26.3746
(3, 1)	-1.63	-5.83	-1.63	-5.83	-3.61	-5.83	40.7065
(1, 2)	0.38	-3.57	0.38	-3.57	-5.26	-3.57	49.2185
(4, 1)	-1.47	-4.72	-1.47	-4.72	-2.06	-4.72	57.5829
(2, 2)	0.15	-2.75	0.15	-2.75	-2.94	-2.75	70.85
(5, 1)	-1.45	-4.03	-1.45	-5.68	-1.36	-5.68	76.9389
(3, 2)	-0.206	-2.47	-0.206	-2.47	-1.11	-2.47	95.2776

Opazimo, da je natančnost v splošnem veliko večja kor v prejšnjem delu naloge, še sploh po tem, ko upoštevamo, da imamo v tem primeru veliko manjše matrike. Čas, ki ga je metoda potrebovala v tem primeru, je bil reda stotink ali desetink sekunde, veliko manj kot prej, kar je torej le še dodaten plus.

Oglejmo si še lastne načine nihanja, dobljene iz lastnih vrednosti:



Res vidimo, da smo dobili dobro poznane lastne nihajne načine polkroga. V splošnem bi lahko reševali tudi primer kakšnih drugih dimenzij, vendar je potem vprašanje katera metoda je boljša, saj pri bolj kompliciranih geometriji elementov matrik ni enostavno določiti, če imamo zvezne poskusne funkcije, kot v tem primeru. Za zahtevno geometrijo je ravno zato boljša izbira metoda s trikotno mrežo.