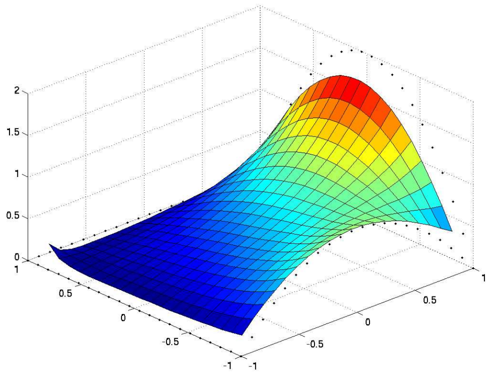
LINEARNI SISTEMI: ITERATIVNE METODE

MARTIN VUK

Naloga. Žično zanko s pravokotnim tlorisom potopimo v milnico, tako da se nanjo napne milna opna.



Slika 1. Ploskev, ki opisuje milni mehurček, napet na žično zanko.

Radi bi poiskali obliko milne opne, razpete na žični zanki. Malo brskanja po fizikalnih knjigah in internetu hitro razkrije, da ploskve, ki tako nastanejo, sodijo med minimalne ploskve `http://en.wikipedia.org/wiki/Minimal_surface`, ki so burile domišljijo mnogim matematikom in nematematikom. Navdušenje nad minimalnimi ploskvami sta pokazala tudi arhitekta Muenchenskega olimpijskega stadiona, kjer ima streha obliko minimalne ploskve.

Slika 2. Muenchenski Olimpijski park (slike so vzete iz wikipedie)

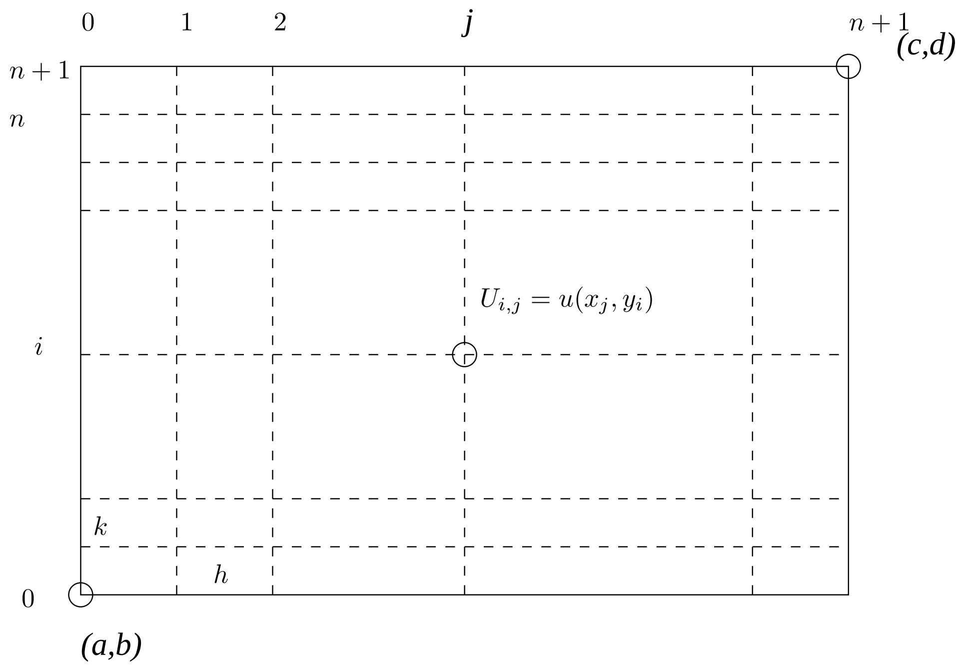
Matematično ozadje. Ploskev lahko predstavimo s funkcijo dveh spremenljivk $u(x,y)$, ki predstavlja višino ploskve nad točko (x,y). Naša naloga bo poiskati funkcijo $u(x,y)$ na tlorisu žične mreže.

Funkcija $u(x,y)$, ki opisuje milno opno, zadošča matematična enačbi, znani pod imenom *Poissonova enačba*

$$\Delta u(x,y) = \rho(x,y). \tag{1}$$

Funkcija $\rho(x,y)$ je sorazmerna tlačni razliki med zunanjo in notranjo površino milne opne. Tlačna razlika je lahko posledica višjega tlaka v notranjosti milnega mehurčka ali pa teže, v primeru opne, napete na žični zanki.

Diskretizacija in linearni sistem enačb. Problema se bomo lotili numerično, zato bomo vrednosti $u(x,y)$ poiskali le v končno mnogo točkah: problem bomo *diskretizirali.* Za diskretizacijo je najpreprosteje uporabiti enakomerno razporejeno pravokotno mrežo točk. Točke na mreži imenujemo *vozlišča.* Zaradi enostavnosti bomo obravnavali le mreže z enakim številom točk v obeh koordinatnih smereh.



Slika 3. Mreža vozlišč na tlorisu žične zanke

Označimo z (a,b) levo spodnje, s (c,d) pa desno zgornje oglišče tlorisa zanke in naj $h = \frac{d-b}{n}$ in $k = \frac{c-a}{n}$ določata gostoto mreže v x oziroma y smeri. V mreži so torej točke s koordinami oblike

$$\begin{aligned} x_j &= a + jh \\ y_i &= b + ik \end{aligned}$$

za $i,j = 0,\ldots,n+1$. Zanimajo nas le vrednosti v točkah na mreži in namesto funkcije $u(x,y)$ iščemo matriko $U = [u_{i,j}]$ dimenzije $n \times n$, v kateri bodo približne vrednosti $u_{i,j} = u(x_j, y_i)$. Obliko žične zanke opisujejo vrednosti funkcije u na robu pravokotnika, ki so podane kot vhodni podatek. Poznane so torej vrednosti $u_{0,j}$, $u_{n+1,j}$, $u_{i,0}$ in $u_{i,n+1}$ za poljubne vrednosti indeksov i in j. Naslednji korak je diskretizacija Poissonove enačbe (1). Namesto enačbe za zvezno funkcijo $u(x,y)$, dobimo n^2 enačb

$$\Delta u_{i,j} = \rho(x_j, y_i)$$

z n^2 neznankami $u_{i,j}$. Diskretni približek za Laplaceov operator

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{2}$$

dobimo z drugimi diferencami

$$\begin{aligned} \frac{\partial^2 u(x_j, y_i)}{\partial x^2} &= \frac{1}{h^2}\left(u(x_j + h, y_i) - 2u(x_j, y_i) + u(x_j - h, y_i)\right) \\ &= \frac{1}{h^2}\left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right) \end{aligned}$$

in podobno za odvod po y. Tako dobimo formulo za Laplaceov operator (2)

$$\Delta u(x_j, y_i) = \frac{1}{h^2}\left(u_{i,j+1} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i-1,j}\right).$$

Če upoštevamo še, da so $u_{i,j}$ podani z robnimi pogoji za $i, j = 0, n+1$, dobimo sistem enačb za $u(x,y)$

$$\begin{aligned} -4u_{1,1} + u_{1,2} + u_{2,1} &= \quad h^2\rho_{1,1} - u_{1,0} - u_{0,1} &&= b_1 \\ u_{1,1} - 4u_{1,2} + u_{1,3} + u_{2,2} &= \quad h^2\rho_{1,2} - u_{0,2} &&= b_2 \\ &\vdots \qquad\qquad \vdots && \vdots \\ u_{n-1,n} + u_{n,n-1} - 4u_{n,n} &= \quad h^2\rho_{n,n} - u_{n,n+1} - u_{n+1,n} &&= b_n \end{aligned} \tag{3}$$

V vsakem notranjem vozlišču mreže dobimo eno linearno enačbo, ki skupaj tvorijo sistem n^2 linearnih enačb z n^2 neznankami. Ne pozabimo, da so vrednosti u znane v robnih vozliščih. Ker bomo sistem reševali z iteracijo, lahko za računanje uporabimo kar matriko $U = [u_{i,j}]_{0 \le i,j \le n+1}$.

Matrična oblika sistema. V primeru, da bi imeli opraviti z nehomogenim ali celo neizotropnim problemom, koeficientov sistema ne bi bilo mogoče vgraditi v program. V tem primeru bi bilo bolj pregledno sistem zapisati v matrični obliki, tako da elemente matrike $u_{i,j}$ zložimo v en sam stolpec. Stolpce matrike $[u_{i,j}]_j$ razvrstimo enega pod drugim in dobimo vektor u dolžine n^2.

$$\begin{aligned} u \quad = \quad & [u_{1,1},\, u_{2,1},\, u_{3,1}, \ldots \\ & \ldots,\quad u_{1,2},\, u_{2,2}, \ldots,\, u_{n,1},\, u_{n,2}, \ldots,\, u_{n,n}]^T. \end{aligned}$$

Povezava med elementi matrike $U = [u_{i,j}]$ in vektorja $u = [u_k]$ je dana z enačbo

$$u_{i,j} = u(j(n-1) + i).$$

Enačbe (3) se sedaj glasijo

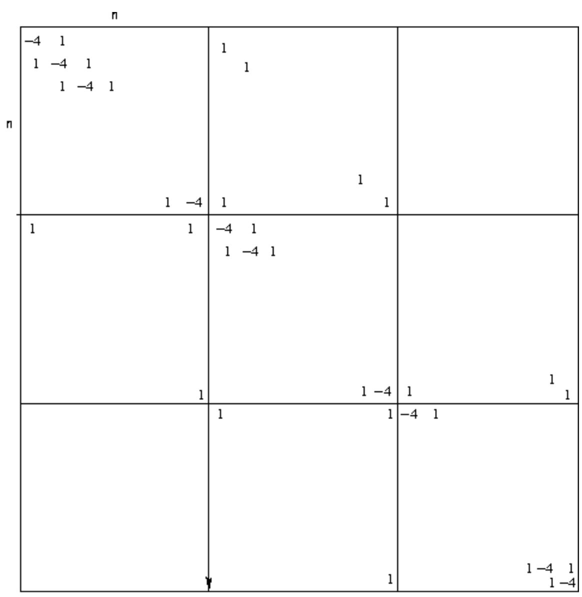
$$\begin{aligned} -4u_1 + u_2 + u_{n+1} &= b_1 \\ u_1 - 4u_2 + u_3 + u_{n+2} &= b_2 \\ &\vdots \\ u_1 - 4u_{n+1} + u_{n+2} + u_{2n+1} &= b_{n+1} \\ &\vdots \\ u_i + u_{n+i-1} - 4u_{n+i} + u_{n+i+1} + u_{2n+i} &= b_{n+i} \\ &\vdots \\ u_{n^2-n} + u_{n^2-1} - 4u_{n^2} &= b_{n^2} \end{aligned}$$

Matrika sistema je bločna

$$A = \begin{pmatrix} L & I & 0 & \cdots & \cdots & 0 \\ I & L & I & 0 & \cdots & 0 \\ 0 & I & L & I & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & I & L & I \\ 0 & \cdots & \cdots & 0 & I & L \end{pmatrix}$$

z bloki velikosti $n \times n$, kjer so matrike I identične $n \times n$ matrike in L Laplaceove matrike oblike

$$L \quad = \quad \begin{pmatrix} -4 & 1 & 0 & \cdots & 0 \\ 1 & -4 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -4 & 1 \\ 0 & \cdots & 0 & 1 & -4 \end{pmatrix}.$$



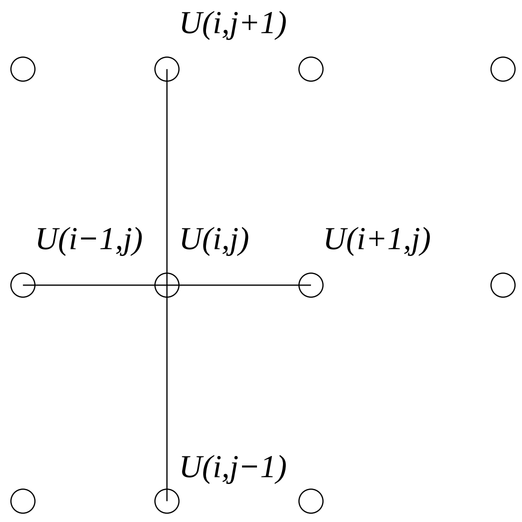
Slika 4. Matrika sistema

Za rešitev sistema potrebujemo še vektor desnih strani, ki ga izračunamo iz robnih pogojev.

$$\begin{aligned} b \quad = \quad & h^2\rho(x,y) - [u_{0,1} + u_{1,0}, u_{0,2}, \ldots, u_{0,n} + u_{1,n+1}, u_{2,0}, \\ & 0, \ldots, 0, u_{2,n+1}, u_{3,0}, \ldots \\ & , u_{n,0} + u_{n+1,1}, u_{n+1,2}, \ldots, u_{n+1,n} + u_{n,n+1}]^T. \end{aligned}$$

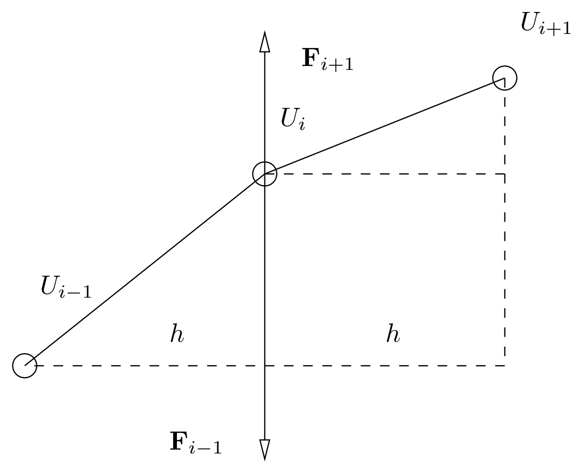
Matrika sistema je precej "prazna", zato jo je smiselno hraniti v kompaktni obliki $5 \times n^2$ podobno kot pri reševanju tridiagonalnega sistema. Algoritem je potrebno ustrezno prilagoditi.

Do istih enačb pridemo tudi fizikalno z "mahanjem rok". Predpostavimo, da na vrednost v $u(i,j)$ vplivajo le sosednja vozlišča v mreži, in sicer $u(i-1,j)$, $u(i,j+1)$, $u(i+1,j)$ in $u(i,j-1)$.



SLIKA 5. Sosednja vozlišča, ki jih v enačbi upoštevamo.

V ravnovesni legi bo vsota vseh sil, ki delujejo na dano točko (x_j, y_i) v mreži, enaka nič. Sila med dvema sosednjima točkama je sorazmerna njuni višinski razliki (glej sliko).



SLIKA 6. Sile med sosednjimi točkami v mreži

Sila, ki je posledica tlačne razlike $\rho(x, y)$, pa je sorazmerna površini malega kvadratka h^2

$$\begin{aligned}((u_{i-1,j} - u_{i,j}) + (u_{i,j+1} - u_{i,j}) + (u_{i+1,j} - u_{i,j}) + (u_{i,j-1} - u_{i,j})) &= h^2 \rho_{i,j} \\ u_{i-1,j} + u_{i,j-1} + u_{i+1,j} + u_{i,j+1} - 4u_{i,j} &= h^2 \rho_{i,j}\end{aligned}$$

Reševanje sistema z Jacobijevo iteracijo. Ker je matrika sistema skoraj prazna, je smiselno uporabiti katero od iteracijskih metod. Predvsem prostorski prihranek bo tako precejšen in bo omogočil reševanje sistema tudi za zelo velike dimenzije, ko bi sicer zmanjkalo pomnilnika. Matrika sistema je *diagonalno dominantna* po vrsticah in po stolpcih, ni pa strogo diagonalno dominantna. Ker je v matriki majhno število različnih elementov, bomo matriko sistema vgradili kar v algoritem.

Rešitev sistema $Ax = b$ z *Jacobijevo* iteracijo dobimo kot zaporedje približkov

$$Dx_{n+1} = b - (S + Z)x_n,$$

kjer je D diagonala matrike A, $S + Z$ pa preostanek matrike A (A brez diagonale). Naslednji približek po komponentah tako dobimo s formulo

$$x_i^{k+1} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{i,j} x_j^k \right) \tag{4}$$

Za našo matriko bodo v vsoti nastopali največ štirje členi

$$\begin{aligned}
x_i^{k+1} &= \frac{1}{-4}\left(b_i - x_{i-1}^k - x_{i+1}^k - x_{i-n}^k - x_{i+n}^k\right), \quad i \neq kn+1, kn-1 \\
x_i^{k+1} &= \frac{1}{-4}\left(b_i - x_{i+1}^k - x_{i-n}^k - x_{i+n}^k\right), \quad i = kn+1 \\
x_i^{k+1} &= \frac{1}{-4}\left(b_i - x_{i-1}^k - x_{i-n}^k - x_{i+n}^k\right), \quad i = kn-1,
\end{aligned}$$

pri čemer so elementi x_j z indeksom $j < 0$ ali $j > n^2$ enaki 0.

Gauss-Seidlova iteracija. Konvergenco iteracije lahko izboljšamo, če pri Jacobijevi iteraciji v formuli (4) uporabimo tiste komponente novega približka $\mathbf{x}^{k+1}$, ki jih že poznamo. Tako dobimo *Gauss-Seidlovo iteracijo*

$$x_i^{k+1} = \frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right) \tag{5}$$

Za našo matriko dobimo formulo za izračun naslednjega približka.

$$x_i^{k+1} = \frac{1}{-4}\left(b_i - x_{i-1}^{k+1} - x_{i+1}^k - x_{i-n}^{k+1} - x_{i+n}^k\right), \quad i \neq kn \pm 1.$$

Podobno naredimo še v ostalih vrsticah.

SOR iteracija. Kot smo se prepričali na zgornjem primeru, je konvergenca Jacobieve in Gauss-Seidlove iteracije včasih kaj klavrna. Zato so pred nekaj desetletji razvili iteracijsko shemo *SOR*. Pri tej metodi približke, dobljene z Gauss-Seidlovo iteracijo, "relaksiramo" s prejšnjim približkom. Formule (5) popravimo, tako da nov približek po Gauss-Seidlu pomnožimo z ω in mu prištejemo $(1 - \omega)$ krat prejšnji približek.

$$x_i^{k+1} = \omega \left(\frac{1}{a_{i,i}} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j} x_j^k \right) \right) + (1-\omega) x_i^k. \tag{6}$$

Tako dobimo celo družino iteracij, parametrizirano z ω. Za $\omega = 1$ dobimo Gauss-Seidlovo iteracijo. Konvergenca je seveda odvisna od izbire ω in mogoče je pokazati, da SOR ne konvergira za $\omega \notin (0, 2)$.

Primer. Poglejmo si še primer, ki je na začetni sliki. Tloris zanke je kvadrat $[-1, 1] \times [-1, 1]$. Ploskev je na robu kvadrata podana s funkcijami $u(x, 1) = 1 - x^2$, $u(1, y) = 2 - 2x^2$ in $u(x, -1) = u(-1, y) = 0$. Slika na začetku je izračunana na mreži 20×20.