Oblika milnega mehurčka nad žično zanko

Žično zanko kvadratne oblike potopimo v milnico tako, da se nanjo napne milna opna. Zanima nas oblika nastalega mehurčka. Naj bo $u(x,y)$ odmik milnega mehurčka v točki $(x,y) \in \mathcal{D} = (-a,a) \times (-a,a)$.

Funkcija u je rešitev *Laplaceove enačbe*

$$\begin{aligned} \Delta u(x,y) &= 0, \quad (x,y) \in \mathcal{D}, \\ u(x,y) &= f(x,y), \quad (x,y) \in \partial\mathcal{D}, \end{aligned} \tag{1}$$

kjer f določa vertikalne odmike funkcije u na robu območja $\mathcal{D}$.

Numerično reševanje parcialne diferencialne enačbe

Kvadrat $[-a,a] \times [-a,a]$ razdelimo na $n+1$ podintervalov v vsaki koordinatni smeri

$$\begin{aligned} -a = x_0 < x_1 < \ldots < x_n < x_{n+1} = a, \\ -a = y_0 < y_1 < \ldots < y_n < y_{n+1} = a, \end{aligned}$$

kjer je $h = \Delta x_i = x_{i+1} - x_i = \Delta y_j = y_{j+1} - y_j$. Iščemo približke $u_{i,j}$ za točne vrednosti $u(x_i, y_j)$, $i = 0,1,\ldots,n+1$, $j = 0,1,\ldots,n+1$. Zaradi robnih pogojev velja

$$\begin{aligned} u_{i,0} &= f(x_i, -a), \quad i = 0,1,\ldots,n+1, \\ u_{i,n+1} &= f(x_i, a), \quad i = 0,1,\ldots,n+1, \\ u_{0,j} &= f(-a, y_j), \quad j = 0,1,\ldots,n+1, \\ u_{n+1,j} &= f(a, y_j), \quad j = 0,1,\ldots,n+1, \end{aligned}$$

Za n^2 neznank $u_{i,j}$ izpeljemo sistem linearnih enačb. Z uporabo simetričnih diferenc za drugi odvod diskretiziramo Laplaceov operator Δ:

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2},$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}.$$

Ko zgornje aproksimacije uporabimo v enačbi (1), dobimo

$$u_{i+1,j} + u_{i-1,j} - 4u_{i,j} + u_{i,j+1} + u_{i,j-1} = 0, \quad i,j = 1,\ldots,n. \tag{2}$$

Če želimo sistem zapisati v matrični obliki, moramo neznanke $u_{i,j}$ oštevilčiti. Zberimo sistem enačb leksikografsko po vrsticah. Neznanke zberemo v vektor x, pri čemer je

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

Dobimo matrični zapis $Ax = b$, kjer je

$$A = \begin{bmatrix} T & I & & \\ I & T & \ddots & \\ & \ddots & \ddots & I \\ & & I & T \end{bmatrix} \in \mathbb{R}^{n^2 \times n^2}.$$

Pri tem je I identiteta dimenzije n in $T \in \mathbb{R}^{n \times n}$ tridiagonalna matrika

$$T = \begin{bmatrix} -4 & 1 & & \\ 1 & -4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{bmatrix}.$$

Elementi desne strani b so bodisi nič, bodisi povezani z robnimi vrednostmi:

Spodnji rob

$$\begin{aligned} b_1 &= -f(-a, y_1) - f(x_1, -a), \\ b_i &= -f(x_i, -a), \quad i = 2, \ldots, n-1, \\ b_n &= -f(a, y_1) - f(x_n, -a). \end{aligned}$$

Zgornji rob

$$\begin{aligned} b_{(n-1)n+1} &= -f(-a, y_n) - f(x_1, a), \\ b_{(n-1)n+i} &= -f(x_i, a), \quad i = 2, \ldots, n-1, \\ b_{n^2} &= -f(a, y_n) - f(x_n, a). \end{aligned}$$

Levi rob

$$b_{(i-1)n+1} = -f(-a, y_i), \quad i = 2, \ldots, n-1$$

Desni rob

$$b_{in} = -f(a, y_i), \quad i = 2, \ldots, n-1$$

Direktno reševanje sistema preko LU razcepa je potratno, saj je časovna zahtevnost enaka $\mathcal{O}(n^6)$, prostorska pa $\mathcal{O}(n^4)$. Zato se reševanja lotimo s pomočjo iterativnih metod, pri katerih tvorimo zaporedje približkov $\{x^{(r)}\}$, ki konvergirajo proti točni rešitvi sistema $Ax = b$. V nadaljevanju si bomo ogledali *Jacobijevo* in *Gauss–Seidlovo* metodo.

Ideja iterativnih metod je podobna navadni iteraciji za reševanje nelinearnih sistemov. Sistem $Ax = b$ zapišemo v ekvivalentni obliki $x = Rx + c$ in ga rešujemo iterativno

$$x^{(r+1)} = Rx^{(r)} + c.$$

Matriko R imenujemo *iteracijska matrika*. O konvergenci iterativnih metod odloča iteracijska matrika. Če za spektralni radij iteracijske matrike R velja $\rho(R) = \max_i |\lambda_i(R)| < 1$, potem zaporedje $x^{(r+1)} = Rx^{(r)} + c$ konvergira za vsak začetni približek x_0.

En način, kako pridemo do iteracijske matrike, je, da matriko A zapišemo kot $A = M + N$, potem pa sistem $Ax = b$ zapišemo kot $Mx = -Nx + b$ in dobimo $R = -M^{-1}N$. Iteracije rešujemo v obliki

$$Mx^{(r+1)} = -Nx^{(r)} + b, \quad r = 0, 1, \ldots$$

matriko M pa izberemo tako, da znamo sistem z matriko M rešiti hitreje od polnega sistema.

Jacobijeva in Gauss–Seidlova metoda

Če zapišemo $A = L + D + U$, kjer je L spodnji trikotnik matrike A brez diagonale, D diagonala, U pa zgornji trikotnik matrike A brez diagonale, potem pri Jacobijevi metodi vzamemo $M = D$ in $N = L + U$. Rešitev sistema $Ax = b$ z Jacobijevo iteracijo dobimo kot zaporedje približkov

$$Dx^{(r+1)} = b - (L + U)x^{(r)}, \quad r = 0, 1, \ldots,$$

Ko po vrsti računamo $x_1^{(r+1)}, x_2^{(r+1)}, \ldots, x_n^{(r+1)}$, bi lahko pri računanju $x_k^{(r+1)}$ uporabili že izračunane vrednosti $x_1^{(r+1)}, x_2^{(r+1)}, \ldots, x_{k-1}^{(r+1)}$. Tako dobimo Gauss–Seidlovo metodo, pri kateri je

$$M = L + D \text{ in } N = U.$$

V matričem zapisu je torej

$$(D + L)x^{(r+1)} = b - Ux^{(r)}, \quad r = 0, 1, \ldots,$$

Pri Jacobijevi metodi je potrebno na vsakem koraku iteracije rešiti sistem linearnih enačb z diagonalno matriko, pri Gauss–Seidlovi pa s spodnjo trikotno.

Reševanje sistema direktno na mreži

Ker je za naš problem matrika posebne oblike, lahko iterativno reševanje izvedemo kar na mreži vrednosti u_{ij}. Enačba (2) pove, da korak Jacobijeve iteracije pomeni računanje povprečja notranjih elementov mreže na podlagi štirih sosedov. Vrednosti na robu so določene s funkcijo f. Korak Gauss–Seidlove metode je podoben koraku Jacobijeve, le da pri izračunu povprečja upoštevamo že izračunane elemente. Zato Gauss–Seidlovo metodo uporabimo direktno na mreži, medtem ko pri Jacobijevi uporabimo pomožno mrežo.

Prostorska zahtevnost je namesto $\mathcal{O}(n^4)$ sedaj $\mathcal{O}(n^2)$, časovna pa namesto $\mathcal{O}(n^6)$ spet $\mathcal{O}(n^2)$ (če je iteracij bistveno manj kot n).

Naloge

1. Sestavite funkciji `jacobi.m` in `gauss_seidl.m`, ki implementirata Jacobijevo in Gauss-Seidlovo metodo direktno na mreži:

```
function U = jacobi(U,tol)
% JACOBI izvaja Jacobijevo metodo na mrezi U, ki predstavlja
% diskretizacijo kvadrata [-a,a] x [-a,a]. Na vsakem koraku
% iteracije je vsak element izracunan kot povprecje njegovih
% stirih sosedov. Pri tem je U matrika z niclami v notranjosti
% in vrednostmi na robu, dolocenimi z robnimi pogoji.
% Pri Jacobijevi metodi potrebujemo pomozno mrezo.
% tol je toleranca, ki doloca natancnost izracunane resitve.

function U = gauss_seidl(U,tol)
% GAUSS_SEIDL izvaja Gauss-Seidlovo metodo direktno
% na mrezi U, ki predstavlja diskretizacijo kvadrata
% [-a,a] x [-a,a]. V vsakem koraku iteracije je vsak element
% izracunan kot povprecje njegovih stirih sosedov.
% Pri tem je U matrika z niclami v notranjosti in
% vrednostmi na robu, dolocenimi z robnimi pogoji.
% tol je toleranca, ki doloca natancnost izracunane resitve.
```

2. Dopolnite funkciji tako, da vam med reševanjem iterativna metoda na vsakem koraku izriše trenutno rešitev. Pri tem dobite animacijo, ki riše mehurček od začetnega stanja do končne rešitve.

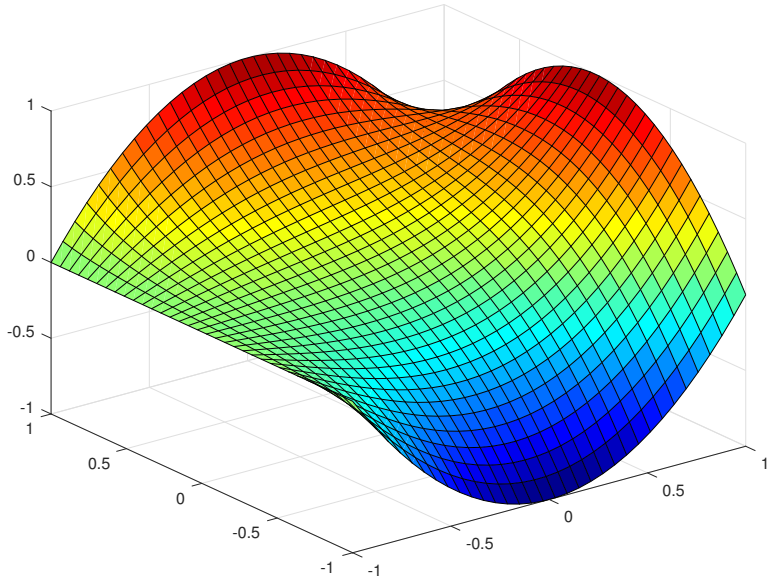
3. Sestavite funkcijo `milnica.m`, ki izračuna obliko mehurčka nad žično zanko:

```
function milnica(a,n,f_spodaj,f_zgoraj,f_levo,f_desno,tol,metoda)
% MILNICA izracuna obliko milnice na kvadratu
% [-a,a] x [-a,a], kjer so podane robne vrednosti
% s funkcijami f_i. Pri tem je:
% n+2 je stevilo delilnih tock na eni koordinatni osi
% f_i so stiri funkcije ene spremenljivke, ki dolocajo
% vrednosti na robu.
% tol je toleranca pri iterativni metodi
% metoda je stikalo, ki doloca iterativno metodo:
%'Jacobi' = Jacobijevo iteracijo
%'Gauss-Seidel' = Gauss-Seidelovo iteracijo
```

Delovanje programa preverite na podatkih $n = 32$, $a = 1$, $tol = 10^{-3}$ in

(a)
```
f_levo = @(y) zeros(size(y));
f_desno = @(y) 1-y.^2;
f_zgoraj = @(x) 1-x.^2;
f_spodaj = @(x) x.^2-1;
```

Rešitev:



(b)
```
f_levo = f_desno =  @(y) 1-y.^2;
f_zgoraj = f_spodaj = @(x) 1-x.^2;
```

Rešitev: