Numerična matematika - Domača naloga 1

Veronika Matek

Marec 2025

1 SOR iteracija za razpršene matrike

Definiramo tip *RedkaMatrika*, ki začetno kvadratno matriko *nxn* pretvori v 2 novi matriki velikosti *nxm*. Prva matrika V hrani neničelne elemente v istih vrsticah po vrsti, druga matrika I pa hrani po enakih pozicijah kot V indekse stolpcev, kjer se ta element nahaja.

Primer:

$$A = \begin{bmatrix} 3 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 4 \end{bmatrix}, \quad V = \begin{bmatrix} 3 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 0 \\ 2 & 1 & 4 \end{bmatrix}, \quad I = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 4 & 0 \\ 3 & 0 & 0 \\ 1 & 2 & 4 \end{bmatrix}$$

1.1 Definirane funkcije

1.1.1 getindex

Za pridobivanje indeksa na poljubni lokaciji (i,j) se preprosti sprehodimo po vrstici i po matriki I, kjer želimo ugotoviti, če obstaja indeks j. Če ta ne obstaja, potem gre za ničelni element v vrstici, saj teh ne hranimo v matrikah V in I. Drugače pa enostavno vrnemo element iz matrike V, ki se nahaja na isti vrstici i in najdenem indeksu stolpca j.

1.1.2 setindex!

Nastavljanje indeksa na lokaciji (i,j) poteka različno v primerih, če je trenutni element na tisti lokaciji 0 ali ne. Če gre za ničelni element moramo se sprehodimo po vrstici matrike in preverimo, če obstaja kakšen prazen prostor (ničla), kamor lahko element zapišemo. Če ta ne obstaja in je vrstica polna, potem obe matriki povečamo za 1 stolpec ničel in element vpišemo.

V primeru, da gre za neničelni element, podobno got getindeks najdemo ustrezni indeks stolpca v I in prepišemo trenutni element.

1.1.3 firstindex

Firstindex metoda vedno vrne 1, saj je v programskem jeziku Julia za preproste matrike, kot je naša, prvi indeks vedno enak 1.

1.1.4 lastindex

Lastindeks kot po dokumentaciji razdelimo na 2 metodi. Prva sprejme samo element RedkaMatrika in vrne pozicijo zadnjega element torej kar $n * n$.

Druga metoda sprejme še indeks, ki nam pove dimenzijo, po kateri želimo pridobiti zadnji indeks, vendar pa je ta vedno enak n, saj imamo kvadratno matriko.

1.1.5 *

Množenje redke matrike z vektorjem naredimo preprosto z uporabo metode getindeks, kjer se preprosto sprehodimo po vrsticah in izračunamo vsoto produktov.

1.2 SOR

Korak SOR iteracije izvedemo preprosto po formuli, dano na predavanjih. Nato izvedemo več korakov te iteracije in jo predčasno ustavimo, če je absolutni maksimalni element $A * x - b$ manjši od neke tolerance tol.

1.3 Vložitev grafa v ravnino

Za pripravo vložitve grafa uporabimo že napisane funkcije in pridobimo graf, njegovo matriko A ter desno stran b. Nato izvajamo SOR iteracije do konvergence, za pridobitev rešitve x.

1.4 Optimalna omega

Za pridobitev optimalne omege razdelimo interval omeg med 0 in 2 na 20 segmentov in za vsak segment izračunamo število iteracij, ki so bile potrebne za konvergenco SOR metode. Ugotovimo, da je za začetni približek samih ničel optimalna omega enaka 0.947368 in sicer z njo opravimo 18 iteracij. Vse segmente in hitrost konvergence si lahko ogledamo na grafu 1, kjer vrednost 1000 predstavlja nekonvergenco.

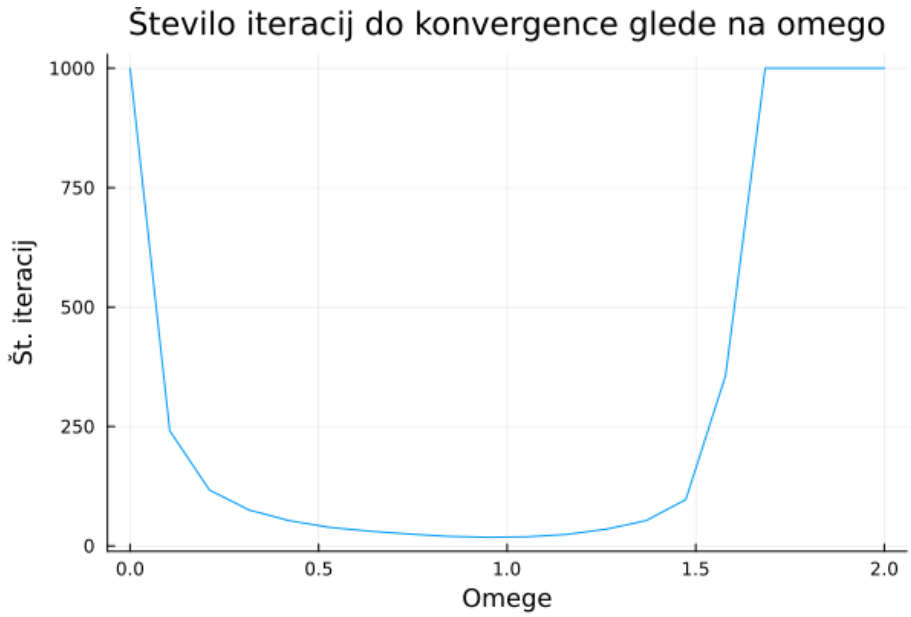

Figure 1: Primerjava vpliva omege na hitrost konvergence SOR metode.