NUMERIČNA MATAMETIKA - 1. domača naloga

Ana Knafelc

Šolsko leto 2023/2024

QR razcep simetrične tri-diagonalne matrike

V linearni algebri je QR razcep proces razdelitve matrike A v produkt matrike Q (ortogonalna matrika) in R (zgornje-trikotna matrika) tako, da velja:

$$A = Q \cdot R$$

Poznamo več metod računanja QR razcepa matrik, med njimi sta najbolj znani metodi QR razcepa z Givensovimi rotacijami in QR razcep matrike po Gram–Schmidt metodi. V tej domači nalogi se bomo posvetili računanju QR razcepa z Givensovimi rotacijami, Gram–Schmidt metodo pa bomo uporabili za primerjavo izračunanih vrednosti.

Matriko R (zgornje-trikotna matrika) izračunamo s pomočjo zaporednih Givensovih rotacij, ki izničijo ne-ničelne elemente pod glavno diagonalo.

Z matriko Givensove rotacije $G(i, j, \theta)$,

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

kjer je $c = cos(\theta)$ in $s = sin(\theta)$, lahko vektor osnovne matrike A z množenjem z matriko $G(i, j, \theta)$ zasukamo preko (i, j) ravnine za kot θ.

Izvedba Givensove rotacije $G(i, j, \theta)$ na matriki A,

$$G = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \quad A = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

kjer je $r = \sqrt{a^2 + b^2}$, $c = \frac{a}{r}$ in $s = \frac{-b}{r}$.
Iz zgornjega primera je razvidno, da smo se z Givensovo rotacijo znebili ne-ničelnega elementa matrike b.

Če sledeči postopek izvedemo za vse elemente, ki ležijo pod glavno diagonalo poljubno velike simetrične matrike A, kot rezultat dobimo matriko R, ki predstavlja prvi element QR razcepa matrike A.

$G_{i_{N-1}j_{N-1}} \cdot ... \cdot G_{i_1j_1} \cdot A = R$

$G^T_{i_1j_1} \cdot ... \cdot G^T_{i_{N-1}j_{N-1}} = Q$

$G^T_{i_1j_1} \cdot ... \cdot G^T_{i_{N-1}j_{N-1}} \cdot R = Q \cdot R$

Primerjava pridobljenih rezultatov

QR razcep matrike

Funkcija za QR razvoj matrike je napisana v programskem jeziku python, s pomočjo knjižnic numpy in math. Funkcija je razvita po zgoraj navedenem principu pridobivanja Q in R matrike s pomočjo Givensovih rotacij.

Funkcija $QR\_Decomposition\_using\_Givens\_Rotations$, kot vhodni parameter sprejema kvadratno matriko poljubne velikosti (NxN), kot izhod pa vrne matriko Q, matriko R tipa $ZgornjaDvodiagonalna$ in parameter Q_givens, ki v sebi hrani zaporedje Givensovih rotacij in indekse vrstic, na katere te rotacije vplivajo.

Kot primer najprej izračunajmo QR razcep simetrične tri-diagonalne matrike A.

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$R = G_{3,2} \cdot G_{1,1} \cdot A$$

$$G_{1,1} \cdot A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2} & \sqrt{2} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 1 \end{bmatrix}$$

$$R = G_{3,2} \cdot G_{1,1} \cdot A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \sqrt{2} & \sqrt{2} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2} & \sqrt{2} & \frac{1}{\sqrt{2}} \\ 0 & 1 & 1 \\ 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$Q = G^T_{1,1} \cdot G^T_{3,2} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix}$$

$$A = Q \cdot R = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \sqrt{2} & \sqrt{2} & \frac{1}{\sqrt{2}} \\ 0 & 1 & 1 \\ 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Primerjajmo zgornje rezultate z rezultati pridobljeni s funkcijo $QR\_Decomposition\_using\_Givens\_Rotations(A)$,

```
QR razcep z Givensovimi rotacijami:

Matrika Q
[[ 0.70711  0.       0.70711]
 [ 0.70711  0.      -0.70711]
 [ 0.       1.       0.     ]]

Matrika R
[[ 1.41421  1.41421  0.70711]
 [ 0.       1.       1.     ]
 [ 0.       0.      -0.70711]]

A = Q * R
[[ 1.  1. -0.]
 [ 1.  1.  1.]
 [ 0.  1.  1.]]
```

s funkcijo $Gram\_Schmidt\_QR\_decomposition(A)$,

```
QR razcep z Gram-Schmidt metodo:

Matrika Q
[[ 0.70711  0.      -0.70711]
 [ 0.70711  0.       0.70711]
 [ 0.       1.      -0.     ]]

Matrika R
[[ 1.41421  1.41421  0.70711]
 [ 0.       1.       1.     ]
 [-0.      -0.       0.70711]]

A = Q * R
[[1. 1. 0.]
 [1. 1. 1.]
 [0. 1. 1.]]
```

in s funkcijo $np.linalg.qr(A)$.

```
QR razcep s pomočjo numpy knjižnjice -> np.linalg.qr(A):

Matrika Q
[[-0.70711  0.      -0.70711]
 [-0.70711  0.       0.70711]
 [-0.       1.       0.     ]]

Matrika R
[[-1.41421 -1.41421 -0.70711]
 [ 0.       1.       1.     ]
 [ 0.       0.       0.70711]]

A = Q * R
[[ 1.  1. -0.]
 [ 1.  1.  1.]
 [ 0.  1.  1.]]
```

V vseh primerih smo prišli do pravilnih vrednosti matrik Q in R, kar je razvidno iz pravilnega končnega rezultata $A = Q \cdot R$.

Lastne vrednosti in lastni vektorji matrike

Za podano matriko A lahko s pomočjo QR razcepa z Givensovimi rotacijami izračunamo tudi njene lastne vrednost in lastne vektorje. Postopek pridobivanja lastnih vrednosti in vektorjev je iterativen in vključuje večkraten izračun QR razcep matrik.

V prvem koraku najprej izračunamo QR razcep matrike A_0 in izračunamo matriko A_1 :

$$A_0 = Q_0 \cdot R_0$$

$$Q_0 = Q_0$$

$$A_1 = R_0 \cdot Q_0$$

V drugem koraku ponovimo postopek QR razcepa vendar za matriko A_1

$$A_1 = Q_1 \cdot R_1$$

$$Q_1 = Q_0 \cdot Q_1$$

$$A_2 = R_1 \cdot Q_1$$

V koraku N izvedemo QR razcep za matriko A_{N-1}

$$A_{N-1} = Q_{N-1} \cdot R_{N-1}$$

$$Q_{N-1} = Q_0 \cdot Q_1 \cdot ... \cdot Q_{N-1}$$

$$A_N = R_N \cdot Q_N$$

Ko postopek ponavljamo za več N korakov, začne diagonala matrike A_N konvergirati proti lastnim vrednostim osnovne matrike A_0. Hkrati začne matrika Q_N prevzemati vrednosti lastnih vektorjev.

Primerjajmo pridobljene rezultate za lastne vrednosti matrike A s funkcijo $Eigenvalues\_Eigenvectors\_Givens$,

```
Lastne vrednosti in vektorji pridobljeni z Givensovimi rotacijami:

Lastne vredsnoti
[[ 2.41421 -0.      -0.     ]
 [ 0.       1.      -0.     ]
 [ 0.       0.      -0.41421]]

Lastni vektorji
[[ 0.5     -0.70711  0.5    ]
 [ 0.70711 -0.      -0.70711]
 [ 0.5      0.70711  0.5    ]]
```

s funkcijo $Eigenvalues\_Eigenvectors\_Gram\_Schmidt$,

```
Lastne vrednosti in vektorji pridobljeni z Gram-Schmidt metodo:

Lastne vredsnoti
[[ 2.41421 -0.      -0.     ]
 [-0.       1.      -0.     ]
 [ 0.       0.      -0.41421]]

Lastni vektorji
[[ 0.5     -0.70711  0.5    ]
 [ 0.70711 -0.      -0.70711]
 [ 0.5      0.70711  0.5    ]]
```

in s funkcijo $np.linalg.eig(A)$:

```
Lastne vrednosti in vektorji pridobljeni z numpy funkcijo -> np.linalg.eig(A):

Lastne vredsnoti
[-0.41421  1.       2.41421]

Lastni vektorji
[[ 0.5      0.70711  0.5    ]
 [-0.70711  0.       0.70711]
 [ 0.5     -0.70711  0.5    ]]
```

Pravilnost izračuna lastnih vrednosti in lastnih vektorjev matrike A lahko preverimo z naslednjo enačbo:

$$A \cdot lastni\_vektor_i = lastna\_vrednost_i \cdot lastni\_vektor_i$$

```
Lastni vektor 0 in lastna vrednost 0:
[[1. 1. 0.]
 [1. 1. 1.]
 [0. 1. 1.]] * [0.5     0.70711 0.5    ] = [0.5     0.70711 0.5    ] * 2.4142135623730954
[1.20711 1.70711 1.20711] = [1.20711 1.70711 1.20711]

Lastni vektor 1 in lastna vrednost 1:
[[1. 1. 0.]
 [1. 1. 1.]
 [0. 1. 1.]] * [-0.70711 -0.       0.70711] = [-0.70711 -0.       0.70711] * 1.000000000000001
[-0.70711 -0.       0.70711] = [-0.70711 -0.       0.70711]

Lastni vektor 2 in lastna vrednost 2:
[[1. 1. 0.]
 [1. 1. 1.]
 [0. 1. 1.]] * [ 0.5     -0.70711  0.5    ] = [ 0.5     -0.70711  0.5    ] * -0.4142135623730952
[-0.20711  0.29289 -0.20711] = [-0.20711  0.29289 -0.20711]
```

Iz zgornjega izračuna je razvidno, da so lastne vrednosti in vektorji iz vseh izračunov pravilni, saj izpolnjujejo enačbo:

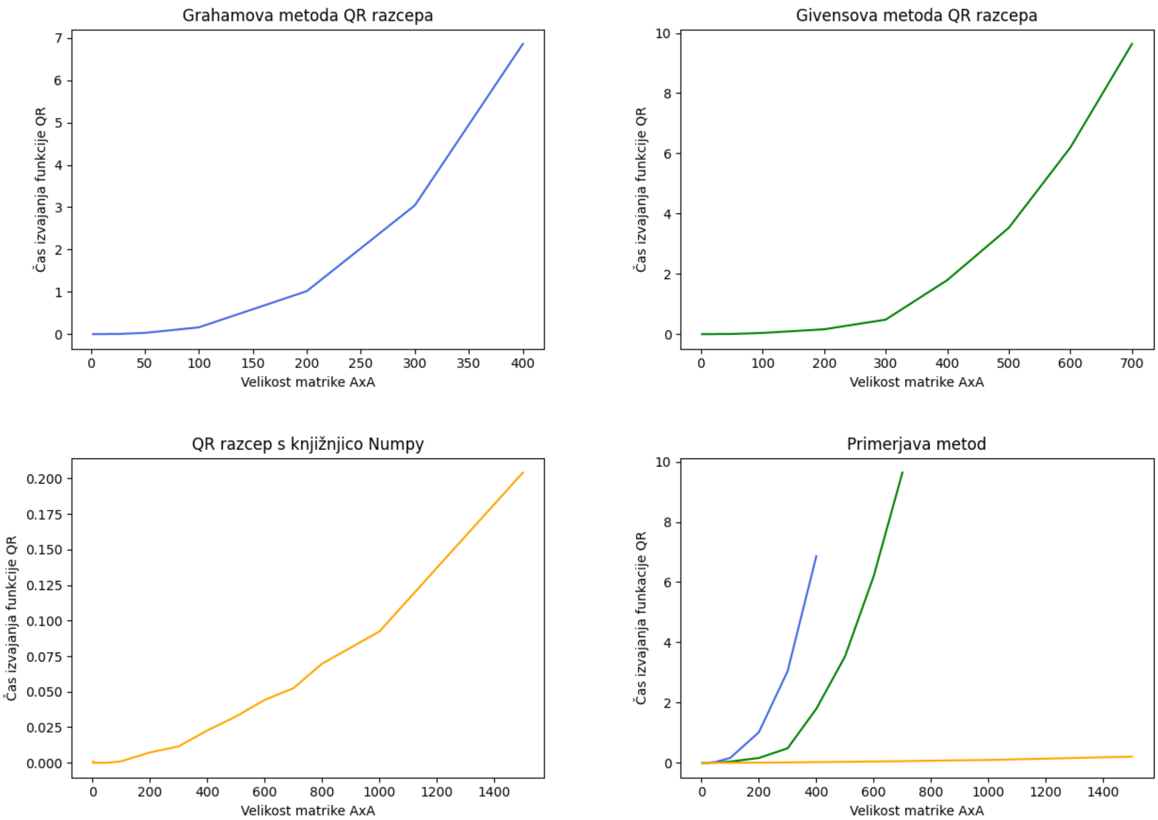
$$A \cdot lastni\_vektor_i = lastna\_vrednost_i \cdot lastni\_vektor_i$$

Časovna zahtevnost funkcij

QR razcep matrike

Pri QR razcepu s pomočjo Givensovih rotacij, se pri računanju matrike R osredotočimo na računanje Givensovih matrik za elemente v spodnjem kvadrantu. To pomeni, da za poljubno matriko A (velikosti nxn) izračunamo največ $n^2/2$ Givensovih matrik. Pri tem je zahtevnost računanja matrike Q v vsakem koraku enaka $O(n^3)$. Enako velja za matriko R.

Poglejmo, kako velikost matrike vpliva na čas izvajanja funkcij:

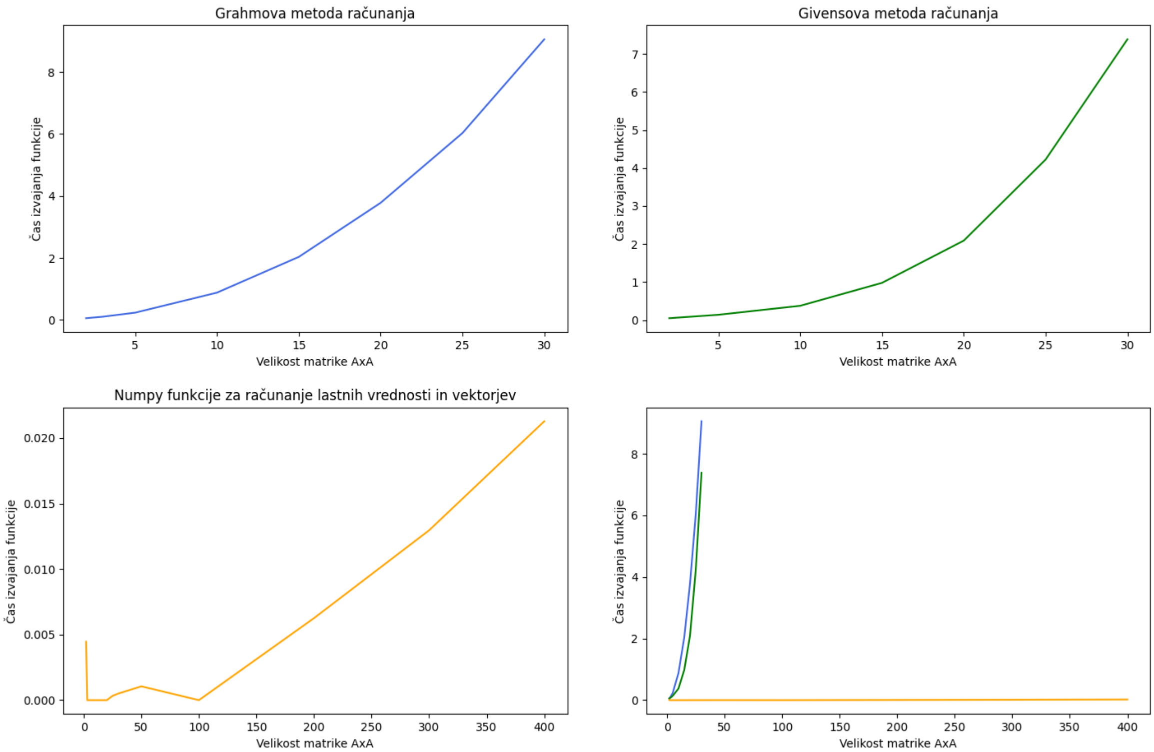


Funkcija, ki izvaja QR razcep matrike s pomočjo Givensovih rotacij je nekoliko hitrejša kot funkcija, ki temelji na Gram-Schmidt metodi, vendar sta obe znatno potratnejši, kot optimizirana funkcija iz knjižnjice $numpy$.

Lastne vrednosti in lastni vektorji matrike

Računanje lastnih vrednosti in lastnih vektorjev matrike A je ciklični proces računanja QR razcepa matrik, zato funkcija za izračun lastnih vrednosti in vektorjev prevzame veliko lastnosti funkcije QR razcepa.
V vsaki iteracij funkcije za izračun lastnih vrednosti in vektorjev moramo izračunati dva posamezna množenja med dvema matrikama in en QR razcep. Vsako množenje dveh matrik ima časovno zahtevnost $O(n^3)$.

Poglejmo, kako velikost matrike A vpliva na čas izvajanja funkcij:



Zgornji test računanja lastnih vrednosti in vektorjev je bil izveden s 1000 iteracijami.

Tudi v tem primeru je računanje lastnih vrednosti in vektorjev hitrejše po metodi Givens kot po metodi Gram-Schmidt, sej je računanje vrednosti neposredno vezano na hitrost funkcije QR razcepa. Ponovno vidimo, da se naše funkcije ne morejo primerjati s hitrostjo funkcije za računanje lastnih vrednosti in vektorjev iz knjižnice $numpy$.

Test pokritosti kode

Test pokritosti kode je bil izveden s pomočjo knjižnice *coverage* in skripte $00\_Pokritost\_kode.py$, ki vsebuje vse funkcije in njihove metode. V skripti so izvedene vse funkcije in metode iz mape *src*.

Rezultati testa pokritosti kode:

```
Name                                          Stmts   Miss  Cover   Missing
---------------------------------------------------------------------------
DN_1\src\Data_type.py                            87      5    94%   13, 32, 54, 71, 107
DN_1\src\Eigenvalues_Eigenvectors.py             38      0   100%
DN_1\src\QR_decomposition_Givens_rotations.py    37      1    97%   14
DN_1\src\QR_decomposition_Gram_Schmidt.py        25      1    96%   9
DN_1\src\Random_Matrix.py                         9      0   100%
DN_1\tests\00_Pokritost_kode.py                 140      0   100%
---------------------------------------------------------------------------
TOTAL                                           336      7    98%
```

S testom smo preverili 98% delovanja kode.