Numerički algoritmi za računanje svojstvenih vrijednosti

Kokotec-Lovrek, Alen

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:695069

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Alen Kokotec-Lovrek

NUMERIČKI ALGORITMI ZA RAČUNANJE SVOJSTVENIH VRIJEDNOSTI

DIPLOMSKI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Alen Kokotec-Lovrek

Matični broj: 0016113056

Studij: Informacijsko i programsko inženjerstvo

NUMERIČKI ALGORITMI ZA RAČUNANJE SVOJSTVENIH VRIJEDNOSTI

DIPLOMSKI RAD

Mentor :

doc. dr. sc. Bojan Žugec

Varaždin, rujan 2020.

Alen Kokotec-Lovrek

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom radu objašnjen je problem određivanja svojstvenih vrijednosti. Prvo su navedeni potrebni pojmovi vezani za matrice i dan je kratak uvod u svojstveni problem. Središnji dio rada su obrađeni numerički algoritmi za računanje svojstvenih vrijednosti: QR algoritam, Jacobijev algoritam, algoritam po principu "podijeli pa vladaj" i metoda potencija - kroz osnovnu matematičku pozadinu pojedinog algoritma, objašnjenja, pseudokod, za važne stavke primjer izračuna i kroz samu implementaciju u programskom jeziku *Python*. Naposljetku su algoritmi testirani, dani su kratak pregled i zaključak.

Ključne riječi: svojstveni problem; svojstvene vrijednosti; numerički algoritmi; QR algoritam; Jacobijev algoritam; algoritam "podijeli pa vladaj"; metoda potencija;

Sadržaj

1. Uvod

Svojstveni problem (na engleskom jeziku *eigenproblem*, za svojstvene vrijednosti može se reći i *vlastite vrijednosti*) je problem pronalaženja svojstvenih vektora i svojstvenih vrijednosti - skalara. Svojstveni vektor je različit od nulvektora, a kada ga se linearno transformira, rezultantni vektor je zapravo jednak umnošku tog svojstvenog vektora i pripadne svojstvene vrijednosti. Rezultantni vektor ima isti smjer kao i početni vektor. Razmatrat ćemo konačnodimenzionalne vektorske prostore pa vektore i linearne operatore prikazivati matricama, dakle računati svojstvene vrijednosti kvadratnih matrica.

Svojstveni problem počinje se spominjati u 17. stoljeću, u 18. stoljeću Leonhard Euler proučava rotacije krutog tijela. Tek početkom 19. stoljeća Augustin-Louis Cauchy spominje pojam karakteristične vrijednosti kod proučavanja spektralne teorije.

Kako je teško izravno računati svojstvene vrijednosti, posebno velikih matrica, potrebno je imati numeričke algoritme za izračun. Njima treba transformirati matrice da se mogu "pročitati" svojstvene vrijednosti. Prvi takav algoritam potječe iz 1929. godine, a radi se o metodi potencija koju je objavio Richard von Mises. Sljedeći važan numerički algoritam potječe iz 1961./1962. godine čiji su autori John G. F. Francis i Vera N. Kublanovskaya koji su radili zasebno.

Numerički algoritmi mogu računati samo svojstvene vrijednosti ili svojstvene vrijednosti zajedno sa svojstvenim vektorima. Također se mogu računati sve svojstvene vrijednosti ili samo neke. Značajni algoritmi osim prije navedenih su Jacobijev algoritam, algoritmi po principu "podijeli pa vladaj", metoda bisekcije, inverzna iteracija i drugi. Spomenuti su često osnova za nadogradnju i poboljšanje izračuna u algoritamskom računanju svojstvenih vrijednosti.

Svojstvene vrijednosti imaju raznoliku primjenu, značajniju na području matematike i fizike. Važne su za rotacije ravnine. U fizici se svojstvene vrijednosti često povezuju s vibracijama. Shodno tome, koriste se kod frekvencija titranja građevina, mostova, instrumenata. Nadalje, svojstvene vrijednosti korespondiraju razinama energije koje molekule mogu zauzeti. U području računalne znanosti, svojstvene vrijednosti mogu se uvidjeti kod procesuiranja slika, primjerice kod prepoznavanja lica osobe. To su samo neki primjeri korištenja.

U ovom su radu obrađeni osnovni algoritmi za računanje svojstvenih vrijednosti.

2. Metode i tehnike rada

Ovaj rad proizlazi iz proučavanja literature na temu svojstvenog problema, točnije numeričkih algoritama kojima se računaju svojstvene vrijednosti. Može se uočiti da su značajniji numerički algoritmi za računanje svojstvenih vrijednosti QR algoritam, Jacobijev algoritam, Cuppenov "podijeli pa vladaj" algoritam i metoda potencija. Spomenuti algoritmi su i obrađeni u radu. Glavni gradivni elementi matematičke teorije su definicije i teoremi. Uz to, napisana su obrazloženja, opisi i dani su primjeri korištenja ili izračuna. Algoritmi su testirani na pojedinim primjerima. Za implementaciju algoritama korišten je programski jezik *Python*. Također za prikaz koda i popratnih rezultata korišten je *Jupyter Notebook*.

Sam rad za prijelom koristi LaTeX.

3. Matrice

Definicija 3.0.1. Neka su $M = \{1, 2, ..., m\}$ i $N = \{1, 2, ..., n\}$. Realna matrica A tipa (m, n) je funkcija

$$A : M \times N \to \mathbb{R},$$

odnosno kompleksna matrica $A : M \times N \to \mathbb{C}$, gdje se funkcijska vrijednost $A(i, j)$ smješta u redak i i stupac j. Matrica ima m redaka i n stupaca:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Korištenje matrica može se uvidjeti u prikazu i obradi podataka, računalnoj grafici, prikazu transformacija, različitim modeliranjima, rješavanjima sustava linearnih jednadžbi, nadalje u elektromagnetizmu, kvantnoj mehanici, statistici itd.

Osnovne operacije s matricama su transponiranje, zbrajanje i množenje, čija svojstva ovdje nećemo navoditi. Definirat ćemo sve pojmove vezane uz matrice koji su potrebni za rješavanje svojstvenog problema. Neka je $\mathcal{M}_{m,n}$ oznaka vektorskog prostora matrica tipa (m, n), a $\mathcal{M}_n$ oznaka vektorskog prostora kvadratnih matrica reda n.

Definicija 3.0.2. Neka je dana $A \in \mathcal{M}_{m,n}(\mathbb{F})$ i $S_1 = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix}, S_2 = \begin{bmatrix} a_{21} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix}, ..., S_n = \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} \in \mathcal{M}_{m1}(\mathbb{F})$ njezini stupci. Rang matrice definira se formulom

$$r(A) = \dim(\{S_1, S_2, ..., S_N\})[1].$$

Drugim riječima, ako je f linearni operator, rang je dimenzija njegove slike, $r(f) = \dim(\mathrm{Im}(f))$.

Definicija 3.0.3. Trag matrice, $\mathrm{tr}(A)$, jednak je zbroju elemenata na glavnoj dijagonali.

3.1. Oblici matrica

Navedimo bitnije oblike važne za daljnju obradu [2].

Definicija 3.1.1. Neka je matrica $A \in \mathcal{M}_{mn}(\mathbb{R})$. $A^T \in \mathcal{M}_{nm}(\mathbb{R})$ je transponirana matrica, svaki redak A^T jednak je odgovarajućem stupcu A.

Nulmatrica je ona čiji su svi elementi jednaki 0. Kvadratna matrica je matrica koja ima isti broj redaka i stupaca. Takve matrice su ovdje posebno bitne i glavnina njih će nadalje biti takve. Jedinična matrica je kvadratna matrica čiji su elementi na glavnoj dijagonali jedinice, a izvan

glavne dijagonale su 0, na primjer

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Definicija 3.1.2. Matrica je simetrična ako vrijedi

$$A^T = A,$$

odnosno antisimetrična ako vrijedi

$$A^T = -A.$$

Simetrična matrica ima elemente $a_{ij} \in \mathbb{R}$ za $i, j = 1, ..., n$ raspoređene simetrično s obzirom na glavnu dijagonalu:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix}.$$

Slično je za antisimetričnu matricu, za $a_{ij} \in \mathbb{R}$ za $i, j = 1, ..., n$ vrijedi da je $a_{ii} = 0$:

$$\begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ -a_{12} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ -a_{1n} & -a_{2n} & \cdots & 0 \end{bmatrix}.$$

Dijagonalna matrica je ona za koju je $d_{ij} = 0, i \neq j$, za $i, j = 1, ..., n$, dakle svi elementi izvan glavne dijagonale su 0,

$$D = \begin{bmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & d_{nn} \end{bmatrix} \in \mathcal{M}_n(\mathbb{R}).$$

Trokutasta matrica je ona kojoj su svi elementi s jedne strane dijagonale jednaki 0, gornje trokutasta ako su nule ispod dijagonale i donje trokutasta ako su nule iznad dijagonale. Navedimo primjer gornje trokutaste matrice:

$$\begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}.$$

Matrica u tridijagonalnoj formi ima elemente različite od nule na glavnoj dijagonali i dvjema

sporednima, prikazano primjerom:

$$\begin{bmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

Definicija 3.1.3. Neka je dana matrica $A = [a_{ij}] \in \mathcal{M}_{mn}(\mathbb{C})$. Hermitski adjungirana matrica $A^* \in \mathcal{M}_{nm}(\mathbb{C})$ je

$$A^* = \begin{bmatrix} \overline{a}_{11} & \overline{a}_{21} & \cdots & \overline{a}_{m1} \\ \overline{a}_{12} & \overline{a}_{22} & \cdots & \overline{a}_{m2} \\ \vdots & \vdots & & \vdots \\ \overline{a}_{1n} & \overline{a}_{2n} & \cdots & \overline{a}_{nm} \end{bmatrix}.$$

U realnom se slučaju hermitsko adjungiranje svodi na transponiranje. Zabilježimo to i definicijom.

Definicija 3.1.4. Neka je dana matrica $A \in \mathcal{M}_{mn}(\mathbb{R})$. Adjungirana matrica $A^* \in \mathcal{M}_{nm}(\mathbb{R})$ je

$$A^* = A^T.$$

Definicija 3.1.5. Matrica $A \in \mathcal{M}_n(\mathbb{C})$ je unitarna ako vrijedi $A^*A = I$.

Definicija 3.1.6. Matrica $A \in \mathcal{M}_n(\mathbb{R})$ je ortogonalna ako vrijedi $A^*A = A^TA = I$.

Realne unitarne matrice su ortogonalne.

Definicija 3.1.7. Matrica $A \in \mathcal{M}_n(\mathbb{F})$ je normalna ako vrijedi $AA^* = A^*A$.

Definicija 3.1.8. Matrica $A \in \mathcal{M}_n(\mathbb{C})$ je hermitska ako vrijedi $A^* = A$.

Nadalje, ako su elementi matrice iz skupa realnih brojeva, radi se o simetričnoj matrici. Hermitske i unitarne matrice su normalne.

Definicija 3.1.9 (Slične matrice)**.** Neka su matrice A i B takve da vrijedi

$$B = S^{-1}AS.$$

Matrice A i B su slične matrice, a matrica S je sličnost.

3.2. Determinante

Definicija 3.2.1. Determinanta matrice $A \in \mathcal{M}_n(\mathbb{F})$ je

$$\det A = \sum_p (-1)^{I(p)} a_{1p(1)} a_{2p(2)} \cdots a_{np(n)}$$

gdje $p = (p(1), p(2), ..., p(n))$ prolazi kroz sve permutacije skupa $\{1, 2, ..., n\}$ i $(-1)^{I(p)} = \pm 1$, ovisno je li p parna ili neparna permutacija.

Determinanta se dakle definira samo za kvadratne matrice i svakako je skalar. Kratko ćemo navesti neka svojstva determinanti. Determinanta transponirane matrice jednaka je determinanti polazne, a determinanta jedinične matrice jednaka je 1. Bitno je za spomenuti da ako su svi elementi jednog reda ili stupca matrice jednaki nuli ili ako matrica ima dva jednaka stupca ili retka, tada je determinanta jednaka nuli. Nadalje, determinanta trokutaste matrice jednaka je produktu dijagonalnih elemenata (glavna dijagonala). Takve matrice su bitne i kod svojstvenih vrijednosti.

Zamijene li se mjesta dva reda ili stupca, determinanta novodobivene matrice jednaka je suprotnoj vrijednosti polazne. Ako se neki red ili stupac zamijeni zbrojem njega i drugog reda ili stupca, ili ako se produkt reda ili stupca i nekog realnog broja doda drugom redu ili stupcu determinanta se ne mijenja [3].

Definicija 3.2.2. Matrica $A \in \mathcal{M}_n(\mathbb{F})$ je regularna ako postoji matrica $B \in \mathcal{M}_n(\mathbb{F})$ takva da je $AB = BA = I$. B je multiplikativni inverz matrice A i označava se s A^{-1}. Ukoliko A nema multiplikativni inverz, tada je A singularna matrica.

Kvadratna matrica koja je regularna ima determinantu različitu od nule.

Teorem 3.2.1 (Binet-Cauchy)**.** Ako su A i B kvadratne matrice istog reda, tada vrijedi

$$\det (A \cdot B) = \det A \cdot \det B.$$

4. Problem svojstvenih vrijednosti

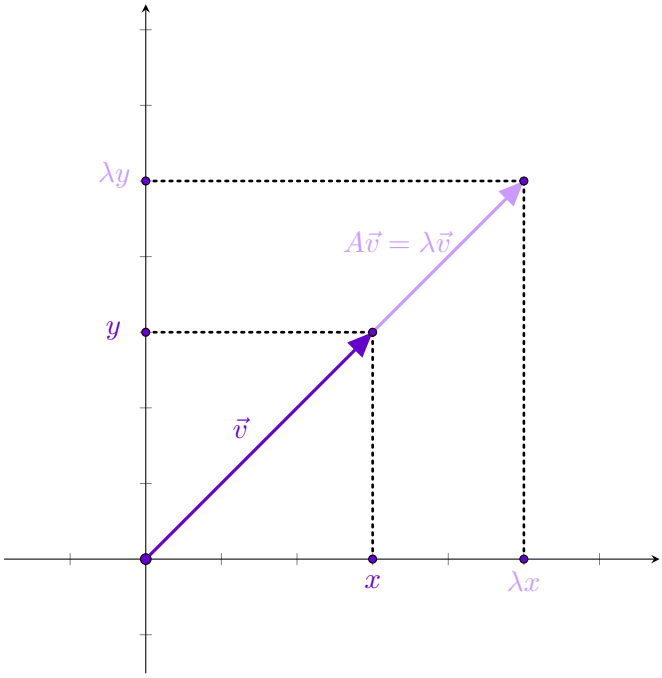
Neka je $\mathbb{V}$ vektorski prostor nad poljem $\mathbb{F}$, a $f : \mathbb{V} \to \mathbb{V}$ linearni operator. Problem svojstvenih vrijednosti je problem određivanja skalara λ i vektora $\mathbf{x}$ za koje je

$$f(\mathbf{x}) = \lambda \mathbf{x}.$$

Skalar $\lambda \in \mathbb{F}$ je svojstvena vrijednost operatora ako je pripadni $\mathbf{x} \in \mathbb{V}$ različit od nulvektora. Takav vektor naziva se svojstveni vektor.
Ograničimo se nadalje na polje kompleksnih brojeva. Dakle, linearni operatori iz konačnodimenzionalnog vektorskog prostora imaju svoj matrični zapis.

Definicija 4.0.1 (Svojstveni par)**.** Par $(\lambda, \mathbf{x})$, gdje je $\mathbf{x} \in \mathbb{C}^n$ svojstveni vektor i $\lambda \in \mathbb{C}$ svojstvena vrijednost, je svojstveni par matrice $A \in \mathcal{M}_n(\mathbb{C})$ ako vrijedi:

$$A\mathbf{x} = \lambda \mathbf{x}, \quad \mathbf{x} \neq \Theta.$$



Slika 4.0.1: Prikaz djelovanja svojstvene vrijednosti

Napišimo još jednom prema slici:

$$A\vec{v} = \lambda\vec{v},$$

gdje je A matrica transformacije, a λ svojstvena vrijednost i vektor $\vec{v}$. Primjerice, ako mijenjamo bazu, vektori mijenjaju svoj prikaz (koordinate). U odnosu na početni vektor, kod promjene baze smjer određenih vektora se ne mijenja, već dolazi do kontrakcije odnosno dilatacije samog

vektora gdje je faktor svojstvena vrijednost, naravno skalar. Napišimo λ u obliku matrice:

$$\begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix},$$

odnosno nakon izlučivanja

$$\lambda \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix},$$

što je jedinična matrica. Sređivanjem dobivamo:

$$A\vec{v} = (\lambda I)\vec{v}$$

$$A\vec{v} - \lambda I\vec{v} = \vec{0}$$

$$(A - \lambda I)\vec{v} = \vec{0}$$

Neka je $S(\lambda)$ skup svih svojstvenih vektora pridruženih svojstvenoj vrijednosti λ. Ako taj skup promatramo kao svojstveni potprostor operatora, dimenzija tog potprostora je geometrijska kratnost svojstvene vrijednosti λ. Označimo još skup svih svojstvenih vrijednosti određenog operatora sa $\sigma(f)$. Taj skup naziva se spektar.

4.1. Karakteristični polinom

Definicija 4.1.1 (Karakteristična matrica)**.** Neka je $A \in \mathcal{M}_n(\mathbb{C})$, I jedinična matrica.

$$C = A - \lambda I,$$

odnosno

$$C = \begin{bmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{bmatrix}$$

je karakteristična ili svojstvena matrica za matricu A. $\lambda \in \mathbb{C}$ je varijabilni parametar.

Definicija 4.1.2 (Karakteristični polinom)**.** Neka je $A \in \mathcal{M}_n(\mathbb{C})$, I jedinična matrica. Polinom

$$k_A(\lambda) = \det(A - \lambda I) = \det(C)$$

je karakteristični ili svojstveni polinom matrice A.

Pripadna jednadžba $k_A(\lambda) = 0$ naziva se karakteristična ili svojstvena jednadžba. Korijeni te

jednadžbe su svojstvene vrijednosti. Dakle, vektor $\mathbf{x} \neq \Theta$ koji zadovoljava $A\mathbf{x} = \lambda\mathbf{x}$ je desni svojstveni vektor matrice A za svojstvenu vrijednost λ. Nadalje, vektor $\mathbf{y} \neq \Theta$ koji zadovoljava $\mathbf{y}^*A = \lambda\mathbf{y}^*$ je lijevi svojstveni vektor matrice A za svojstvenu vrijednost λ. Kratnost svojstvene vrijednosti kao nultočke svojstvene jednadžbe jest njezina algebarska kratnost.

Matrica ne mora imati svojstvenih vektora, niti realnih karakterističnih vrijednosti. Ako postoji kompleksna karakteristična vrijednost, tada postoji i njezin konjugirano kompleksni par [4].

Teorem 4.1.1 (**Osnovni teorem algebre**)**.** Svaki kompleksni polinom stupnja n ima n kompleksnih korijena.

Prema tome, regularna matrica reda n ima n svojstvenih vrijednosti, vodeći računa da mogu biti višestruke.

Primjer 4.1.1. Prikažimo primjer izravnog izračuna svojstvenih vrijednosti. Neka je zadana matrica

$$M = \begin{bmatrix} -2 & -4 & 2 \\ -2 & 1 & 2 \\ 4 & 2 & 5 \end{bmatrix}.$$

Potrebno je riješiti karakterističnu jednadžbu $\det(M - \lambda I) = 0$.

$$\begin{vmatrix} -2-\lambda & -4 & 2 \\ -2 & 1-\lambda & 2 \\ 4 & 2 & 5-\lambda \end{vmatrix} = 0$$

Raspišemo li po Sarrusovom pravilu, dobivamo jednadžbu:

$$(-2-\lambda)(1-\lambda)(5-\lambda) - 32 - 8 - 8(1-\lambda) - 4(-2-\lambda) - 8(5-\lambda) = 0,$$

što nakon sređivanja izgleda:

$$\lambda^3 - 4\lambda^2 - 27\lambda + 90 = 0,$$

i nakon faktorizacije je

$$(\lambda - 3)(\lambda + 5)(\lambda - 6) = 0.$$

To daje rješenja jednadžbe odnosno svojstvene vrijednosti $\sigma = \{3, -5, 6\}$. Odredimo još svojstvene vektore. Za svojstvenu vrijednost -5 potrebno je riješiti matričnu jednadžbu

$$\begin{bmatrix} -2-(-5) & -4 & 2 \\ -2 & 1-(-5) & 2 \\ 4 & 2 & 5-(-5) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Sređivanjem dolazimo do sustava jednadžbi:

$$\begin{bmatrix} 3 & -4 & 2 \\ -2 & 6 & 2 \\ 4 & 2 & 10 \end{bmatrix} \begin{matrix} \\ \\ \mid :2 \end{matrix}, \begin{bmatrix} 3 & -4 & 2 \\ -1 & 3 & 1 \\ 2 & 1 & 5 \end{bmatrix} \begin{matrix} \longleftarrow + \\ \longleftarrow + \\ \cdot(-3) \end{matrix} \begin{matrix} \\ \\ \cdot 4 \end{matrix}, \begin{bmatrix} 11 & 0 & 22 \\ -7 & 0 & -14 \\ 2 & 1 & 5 \end{bmatrix} \begin{matrix} \\ \mid :11 \\ \mid :(-7) \end{matrix}, \begin{bmatrix} 1 & 0 & 2 \\ 1 & 0 & 2 \\ 2 & 1 & 5 \end{bmatrix} \begin{matrix} \\ \cdot(-2) \\ + \end{matrix},$$

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{bmatrix} , \begin{cases} x + 2z = 0 \\ y + z = 0 \end{cases} , \begin{cases} x = -2z \\ y = -z \end{cases} .$$

Neka je $p \in \mathbb{R}$ parametar. Rješenje je tada $(-2p, -p, p)$, odnosno $p(-2, -1, 1)$ što je svojstveni vektor. Analogno, mogu se dobiti vektori $(-2, 3, 1)$ za svojstvenu vrijednost 3 i $(\frac{1}{16}, \frac{3}{8}, 1)$ za svojstvenu vrijednost 6.

4.2. Minimalni polinom

Definicija 4.2.1 (Minimalni polinom)**.** Neka je $\mathbb{F}$ polje i matrica A reda n s vrijednostima iz $\mathbb{F}$. Minimalni polinom $m_A(\lambda) \neq 0$ je svaki onaj najvišeg stupnja za koji je $m_A(A) = 0$.

Pripadna jednadžba $m_A(\lambda) = 0$ se sada zove minimalna jednadžba. Iskažimo dvije propozicije bez dokaza.

Propozicija 4.2.1. Karakteristični polinom kvadratne matrice djeljiv je njezinim minimalnim polinomom.

Propozicija 4.2.2. Slične matrice imaju iste minimalne polinome.

5. Algoritmi za svojstvene vrijednosti

Algoritmi za izračun svojstvenih vrijednosti mogu se klasificirati po nekoliko kategorija. Prema vrsti svojstvenog problema algoritmi mogu biti:

- za simetrični svojstveni problem,
- za nesimetrični svojstveni problem,
- za matrične parove,
- za svojstveni problem velikih rijetkih matrica,
- za svojstveni problem matrica specijalne strukture.

Nadalje, prema broju svojstvenih vrijednosti koje će biti izračunate algoritmi se mogu podijeliti na:

- algoritme koji računaju sve svojstvene vrijednosti,
- algoritme koji računaju najmanju ili najveću svojstvenu vrijednost,
- algoritme koji računaju nekoliko najmanjih ili najvećih svojstvenih vrijednosti,
- algoritme koji računaju svojstvene vrijednosti najbliže nekoj zadanoj vrijednosti,
- algoritme koji računaju svojstvene vrijednosti iz nekog intervala.

6. Kanonske forme matrica

U ovom ćemo poglavlju objasniti forme matrica bitne za svojstvene vrijednosti. Idealan slučaj je matrica u dijagonalnoj formi, no ne mogu se sve matrice dijagonalizirati. Nadalje, kod trokutastih matrica svojstvene vrijednosti se također nalaze na glavnoj dijagonali. Potrebno je dakle pronaći transformacije koje čuvaju spektralne elemente matrice i dovode ju u formu iz koje se mogu pročitati svojstvene vrijednosti.

Definicija 6.0.1. Realna blok-gornjetrokutasta forma ima oblik

$$T = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1b} \\ & A_{22} & \cdots & A_{2b} \\ & & \ddots & \vdots \\ & & & A_{bb} \end{bmatrix}.$$

Vrijedi

$$\det(T - \lambda I) = \prod_{k=1}^{b} \det(A_{kk} - \lambda I).$$

Ova forma ima na glavnoj dijagonali blokove 1×1 za realne svojstvene vrijednosti i blokove 2×2 za konjugirano kompleksne parove.

Propozicija 6.0.1. Slične matrice imaju jednake svojstvene vrijednosti.

Dokaz. Neka su A i B slične matrice. Tada postoji regularna matrica S takva da je

$$B = S^{-1}AS.$$

Raspišimo karakteristični polinom matrice B:

$$\begin{aligned} k_B(\lambda) &= \det(B - \lambda I) = \\ &= \det[S^{-1}AS - \lambda(S^{-1}S)] = \\ &= \det[S^{-1}AS - S^{-1}(\lambda I)S] = \\ &= \det[S^{-1}(A - \lambda I)S] = \\ &= \det S^{-1} \cdot \det(A - \lambda I) \cdot \det S = \\ &= \det S^{-1} \cdot \det S \cdot \det(A - \lambda I) = \\ &= \det(S^{-1}S) \cdot \det(A - \lambda I) = \\ &= \det I \cdot \det(A - \lambda I) = \\ &= \det(A - \lambda I) = k_A(\lambda) \end{aligned}$$

□

Obrat potonjega ne vrijedi, što se lako pokaže protuprimjerom.

Teorem 6.0.1 (Jordanova forma)**.** Neka je $A \in \mathcal{M}_n(\mathbb{F})$. Postoji nesingularna matrica $S \in \mathcal{M}_n(\mathbb{F})$ takva da je

$$S^{-1}AS = J = \operatorname{diag}(J_1, J_2, ..., J_k),$$

pri čemu je Jordanov blok

$$J_i = J_{n_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix} \in \mathcal{M}_{n_i}(\mathbb{F}).$$

λ_i je svojstvena vrijednost. Jordanova matrica J jedinstvena je do na permutaciju dijagonalnih blokova [5].

Broj Jordanovih blokova za jednu svojstvenu vrijednost daje geometrijsku kratnost, a zbroj redova n_i daje njezinu algebarsku kratnost. Ako su sve svojstvene vrijednosti u J jednostruke, tada se matrica A može dijagonalizirati. Još se mogu dijagonalizirati i normalne matrice ako i imaju iste svojstvene vrijednosti.
U numeričkom se računanju ne koristi Jordanova forma.

Teorem 6.0.2. Matrica je normalna akko se može dijagonalizirati unitarnom matricom.

Teorem 6.0.3 (Schurova kanonska forma)**.** Neka je dana matrica $A \in \mathcal{M}_n(\mathbb{C})$. Postoje unitarna matrica $U \in \mathcal{M}_n(\mathbb{C})$ i gornje trokutasta matrica T tako da vrijedi:

$$U^*AU = T.$$

Svojstvene vrijednosti matrice A su dijagonalni elementi matrice T [5][6].

Dokaz. Matematička indukcija po redu matrice A.

Baza indukcije. Za $n = 1$, očito je $U = U^* = 1$.

Pretpostavka indukcije. U traženoj formi možemo napisati svaku matricu do reda $n - 1$.

Korak indukcije. Pokažimo za matricu A reda n. Neka je svojstveni par $(\lambda, \mathbf{v})$ matrice A i $\|\mathbf{v}\| = 1$. Dopunimo svojstveni vektor do unitarne matrice $U_1 = [\mathbf{v}, \overline{U}]$. Tada je

$$(\blacktriangle) \qquad U_1^*AU_1 = \begin{bmatrix} \mathbf{v}^* \\ \overline{U}^* \end{bmatrix} A[\mathbf{v}, \overline{U}] = \begin{bmatrix} \mathbf{v}^*A\mathbf{v} & \mathbf{v}^*A\overline{U} \\ \overline{U}^*A\mathbf{v} & \overline{U}^*A\overline{U} \end{bmatrix} = \begin{bmatrix} \lambda & \mathbf{v}^*A\overline{U} \\ 0 & \overline{U}^*A\overline{U} \end{bmatrix},$$

Vrijedi

$$\begin{cases} \mathbf{v}^*A\mathbf{v} = \mathbf{v}^*\lambda\mathbf{v} = \lambda \\ \overline{U}^*A\mathbf{v} = \overline{U}^*\lambda\mathbf{v} = 0 \end{cases},$$

jer je $\mathbf{v}$ jedinični vektor za λ i također su stupci od $\overline{U}$ ortogonalni na $\mathbf{v}$.
Prema pretpostavci, $\overline{U}^*A\overline{U}$ se također može napisati u traženoj formi. Neka su $\widetilde{U}$ unitarna

matrica i $\widetilde{T}$ gornje trokutasta matrica tako da je $\overline{U}^* A \overline{U} = \widetilde{U}^* \widetilde{T} \widetilde{U}$. Uvrštavanjem u (▲) možemo dobiti:

$$U_1^* A U_1 = \begin{bmatrix} \lambda & \mathbf{v}^* A \overline{U} \widetilde{U} \widetilde{U}^* \\ 0 & \widetilde{U}^* \widetilde{T} \widetilde{U} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \widetilde{U} \end{bmatrix} \begin{bmatrix} \lambda & \mathbf{v}^* A \overline{U} \widetilde{U} \\ 0 & \widetilde{T} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \widetilde{U}^* \end{bmatrix}.$$

□

Korolar 6.0.3.1. Trokutasta matrica T u Schurovoj dekompoziciji $A = UTU^*$ je dijagonalna akko je matrica A normalna. Specijalno vrijedi i:

- Schurova forma hermitske matrice je realna dijagonalna matrica,
- Schurova forma anti-hermitske matrice je dijagonalna s čisto imaginarnim dijagonalnim elementima,
- Schurova forma unitarne matrice je dijagonalna s $|\lambda_i| = 1, i = 1, ..., n$ [7].

Numeričkim računanjem može se doći do kompleksnog spektra i time se povećava korištenje računalnih resursa. Prema tome, uvodi se realna Schurova forma kako bi se izbjeglo računanje kompleksnim brojevima.

Lema 6.0.4. Neka je $(\lambda, \mathbf{v})$ svojstveni par tako da je $\lambda \in \mathbb{C}$ i A realna matrica. Konjugiranjem dobivamo:

$$A\mathbf{v} = \lambda \mathbf{v}, \overline{A\mathbf{v}} = \overline{A}\overline{\mathbf{v}} = A\overline{\mathbf{v}} = \overline{\lambda \mathbf{v}} = \overline{\lambda}\overline{\mathbf{v}}.$$

Prema tome, $(\overline{\lambda}, \overline{\mathbf{v}})$ je također svojstveni par.

Lema 6.0.5. Matrica $\begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix}$, $\alpha, \beta \in \mathbb{R}$, ima svojstvene vrijednosti $\alpha \pm \beta i$.

Teorem 6.0.6 (Realna Schurova forma)**.** Neka je dana matrica $A \in \mathcal{M}_n(\mathbb{R})$. Postoje ortogonalna matrica $Q \in \mathcal{M}_n(\mathbb{R})$ i realna blok-gornjetrokutasta matrica T tako da vrijedi:

$$Q^T A Q = T, \qquad T = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ & R_{22} & \cdots & R_{2m} \\ & & \ddots & \vdots \\ & & & R_{mm} \end{bmatrix},$$

gdje su R_{ii} blokovi 1×1 kod svojstvenih vrijednosti iz $\mathbb{R}$ ili 2×2 kod svojstvenih vrijednosti iz $\mathbb{C}$ [5].

Dokaz. Matematička indukcija po broju konjugirano kompleksnih parova k svojstvenih vrijednosti.

Baza indukcije. Za $k = 0$ matrica A ima svojstvene vrijednosti i svojstvene vektore iz $\mathbb{R}$ i može se iskoristiti dokaz iz Teorema 6.0.3. Dakle, svi dijagonalni blokovi su tada 1×1.
Dalje razmatramo matrice za $k > 0$.

Pretpostavka indukcije. Matrica A s brojem konjugirano kompleksnih parova manjim od k može se napisati u obliku $Q^TAQ = T$.

Korak indukcije. Neka matrica A ima k konjugirano kompleksnih parova. Neka je $\lambda \in \mathbb{C}$ svojstvena vrijednost, $\lambda = \alpha + \beta i, \beta \neq 0$, pa je i svojstveni vektor nužno kompleksan: $\mathbf{v} = a + bi$. Neka je $v_1 = \mathtt{Re}(\mathbf{v})$ i $v_2 = \mathtt{Im}(\mathbf{v})$.

$$v_1 = \frac{1}{2}(\mathbf{v} + \overline{\mathbf{v}}), v_2 = \frac{1}{2i}(\mathbf{v} - \overline{\mathbf{v}}),$$

pa $\mathbf{v}$ i $\overline{\mathbf{v}}$ razapinju isti potprostor kao i v_1 i v_2. Taj prostor je invarijantan.
Neka je $W = [v_1, v_2]$ i izaberimo $\widetilde{W}$ tako da dopunimo do ortogonalne baze

$$U = [W, \widetilde{W}].$$

Neka je matrica $C \in \mathcal{M}_2(\mathbb{R})$ nesingularna takva da je $[v_1, v_2] = [a, b]C$.
Raspišimo:

$$\begin{aligned} AW &= A[a,b]C = \\ &= A[a,b]\begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} C = \\ &= WC^{-1}\begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} C = \\ &= WB \end{aligned}$$

Matrice B i $\begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix}$ su slične pa imaju iste svojstvene vrijednosti.
Dalje računajmo

$$\begin{aligned} U^TAU &= \begin{bmatrix} W^T \\ \widetilde{W}^T \end{bmatrix} A[W, \widetilde{W}] = \begin{bmatrix} W^TAW & W^TA\widetilde{W} \\ \widetilde{W}^TAW & \widetilde{W}^TA\widetilde{W} \end{bmatrix} = \\ &= \begin{bmatrix} W^TWB & W^TA\widetilde{W} \\ \widetilde{W}^TWB & \widetilde{W}^TA\widetilde{W} \end{bmatrix} = \begin{bmatrix} B & W^TA\widetilde{W} \\ 0 & \widetilde{W}^TA\widetilde{W} \end{bmatrix}. \end{aligned}$$

Prema pretpostavci, postoji ortogonalna matrica $Q_2 \in \mathcal{M}_{n-2}(\mathbb{R})$ tako da je

$$Q_2^T(\widetilde{W}^TA\widetilde{W})Q_2$$

realna blok-gornjetrokutasta matrica. Dakle, Q transformira matricu A do realne blok-gornjetrokutaste forme.

□

Korolar 6.0.6.1. Matrica $A \in \mathcal{M}_n(\mathbb{R})$ je normalna akko postoji realna ortogonalna matrica Q i blok dijagonalna matrica T tako da je

$$Q^T A Q = \begin{bmatrix} T_{11} & & \\ & \ddots & \\ & & T_{jj} \end{bmatrix},$$

gdje su blokovi $T_{ii}, i = 1, ..., j$ oblika 1×1 za svojstvene vrijednosti iz $\mathbb{R}$ ili oblika 2×2 za konjugirano kompleksne parove [7].

7. QR algoritam

QR algoritam jedan je od poznatijih numeričkih algoritama za računanje svojstvenih vrijednosti i svojstvenih vektora matrica. Autori su John G. F. Francis i Vera N. Kublanovskaya koji su radili zasebno. Algoritam potječe iz 1961./1962. godine. Osnovna ideja je provoditi QR faktorizaciju da se dođe do forme iz koje se mogu pročitati svojstvene vrijednosti.

7.1. QR dekompozicija matrice

Teorem 7.1.1 (QR dekompozicija)**.** Neka je matrica $A \in \mathcal{M}_{mn}(\mathbb{R})$. Postoje ortogonalna matrica $Q \in \mathcal{M}_m(\mathbb{R})$ i gornje trokutasta matrica $R \in \mathcal{M}_{mn}(\mathbb{R})$ takve da je

$$A = QR.$$

U našim slučajevima matrica A će biti kvadratna. Postoje razni načini za određivanje QR dekompozicije matrice, spomenimo sljedeće:

- Gram-Schmidtova ortogonalizacija,
- Givensove rotacije,
- Householderovi reflektori.

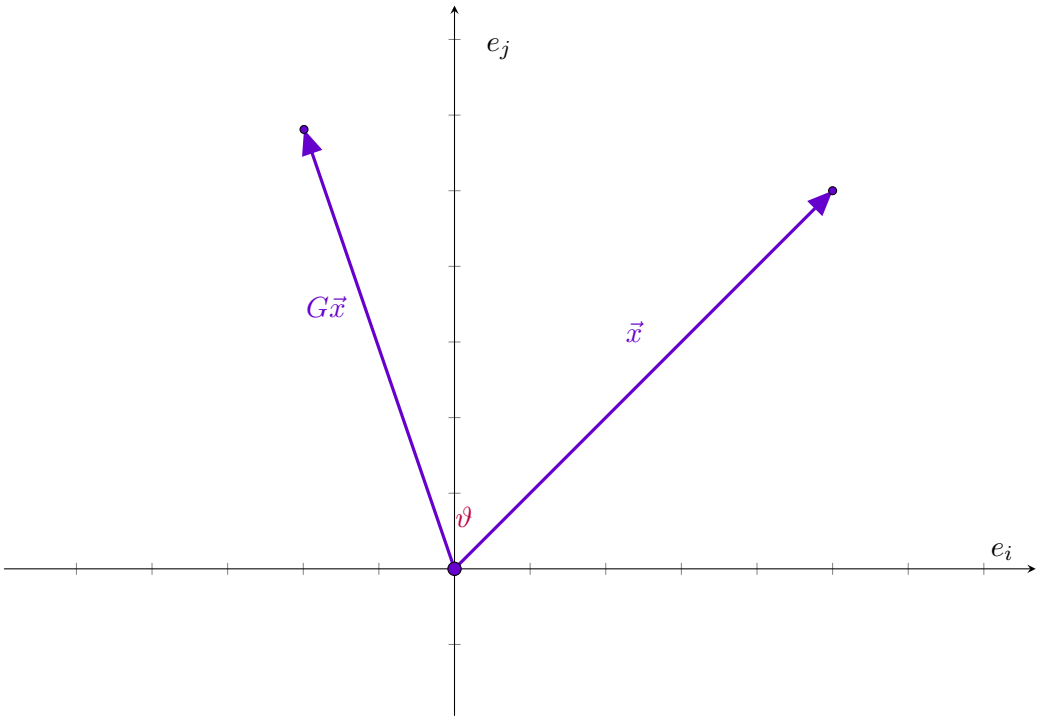
Definicija 7.1.1. Givensove rotacije su

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

gdje je $c = \cos(\vartheta)$ i $s = \sin(\vartheta)$ [8].

Givensove rotacije predstavljaju rotaciju ϑ radijana u (i,j). Neka je $\mathbf{y} = G(i,j,\vartheta)\mathbf{x}, \mathbf{x} \in \mathbb{R}^n$,

$$y_k = \begin{cases} cx_i - sx_j, & k = i \\ sx_i + cx_j, & k = j \\ x_j, & k \neq i,j \end{cases}.$$



Slika 7.1.1: Djelovanje Givensove rotacije

Da bi y_k bio 0 potrebno je postaviti

$$c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad s = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}.$$

Tako redom možemo poništavati elemente matrice ispod i-tog.

Primjer 7.1.1. Provedimo QR faktorizaciju za matricu $A = \begin{bmatrix} 3 & 1 & 2 \\ -1 & 5 & 4 \\ 2 & 7 & -3 \end{bmatrix}$ korištenjem Givensovih rotacija.

Radi preglednosti, rezultate ćemo zaokruživati. Na početku želimo poništiti element matrice na mjestu $(3, 1)$. Stoga je potrebno izračunati Givensovu rotaciju $G_1 = G(1, 3, \varphi_1)$.

$$\cos\varphi_1 = \frac{3}{\sqrt{3^2 + 2^2}} \approx 0.8321$$

$$\sin\varphi_1 = \frac{2}{\sqrt{3^2 + 2^2}} \approx -0.5547$$

Slijedi rotacija

$$G_1 = \begin{bmatrix} 0.8321 & 0 & 0.5547 \\ 0 & 1 & 0 \\ -0.5547 & -0 & 0.8321 \end{bmatrix}.$$

Nakon množenja dobivamo:

$$A_1 = G_1 A = \begin{bmatrix} 3.606 & 4.715 & 0 \\ -1 & 5 & 4 \\ 0 & 5.27 & -3.606 \end{bmatrix}.$$

Analogno dalje tražimo za poništavanje elementa $(2,1)$ rotaciju

$$G_2 = G(1,2,\varphi_2) = \begin{bmatrix} 0.9636 & -0.2672 & 0 \\ 0.2672 & 0.9636 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Dobivamo:

$$A_2 = G_2 A_1 = G_2 G_1 A = \begin{bmatrix} 3.742 & 3.207 & -1.069 \\ 0 & 6.078 & 3.854 \\ 0 & 5.27 & -3.606 \end{bmatrix}.$$

Za poništavanje elementa $(3,2)$ je rotacija

$$G_3 = G(2,3,\varphi_3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.7555 & 0.6551 \\ 0 & -0.6551 & 0.7555 \end{bmatrix}.$$

Na kraju je tada

$$R = A_3 = G_3 G_2 G_1 A = \begin{bmatrix} 3.742 & 3.207 & -1.069 \\ 0 & 8.044 & 0.549 \\ 0 & 0 & -5.249 \end{bmatrix}.$$

Matrica Q se lako izračuna množenjem $G_1^T G_2^T G_3^T$.

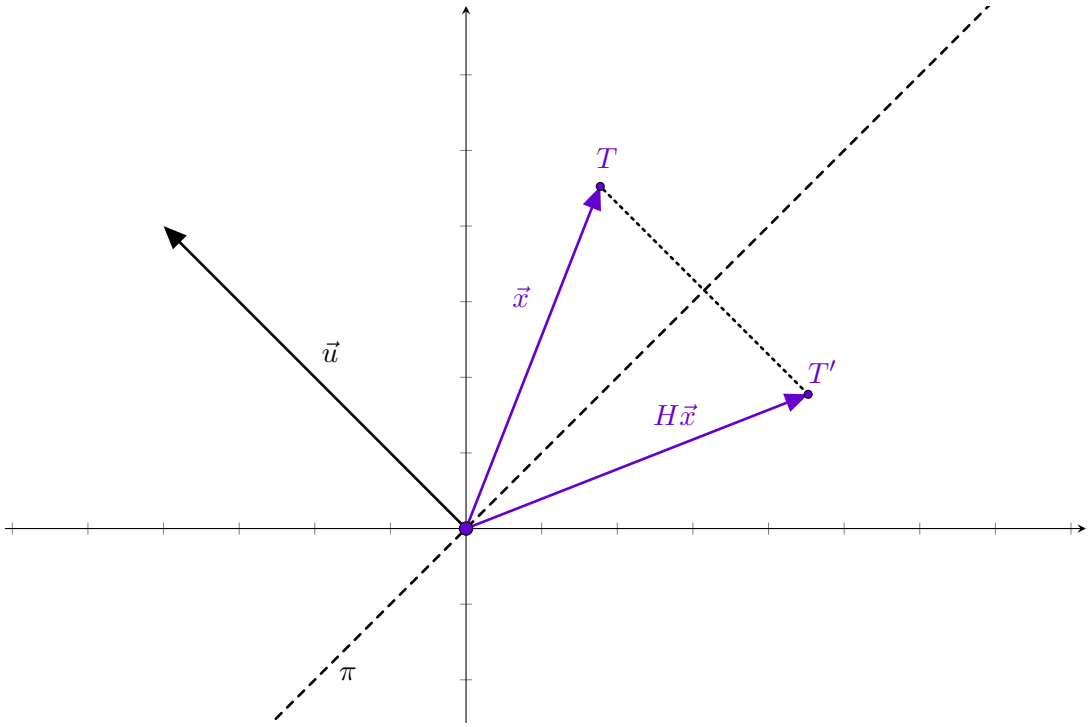
Definicija 7.1.2. Householderov reflektor je matrica $H \in \mathcal{M}_n(\mathbb{C})$ oblika

$$H = H(\mathbf{u}) := I - 2\mathbf{u}\mathbf{u}^*, \mathbf{u} \in \mathbb{C}^n, \|\mathbf{u}\| = 1 \text{[8][9].}$$

Householderov reflektor je hermitska matrica, pa je $H = H^*$.
Promatrajmo vektor $\mathbf{a} \in \mathcal{M}_{n1}$ kao stupac zadane matrice A. Neka je tada vektor $\mathbf{b} := H\mathbf{a}$ konstruiran tako da se komponente do dijagonale podudaraju, a dalje poništavaju. Neka je $\varphi = \pm 1$ predznak od komponente vektora a_k.
Raspišimo:

$$\begin{aligned} \|\mathbf{a} - \mathbf{b}\|^2 &= (a_k + \varphi\|\mathbf{a}\|)^2 + a_{k+1}^2 + ... + a_n^2 = \\ &= a_k^2 + 2\varphi\|\mathbf{a}\|a_k + \|\mathbf{a}\|^2 + a_{k+1}^2 + ... + a_n^2 = \\ &= 2\|\mathbf{a}\|\varphi a_k + 2\|\mathbf{a}\|^2 = \\ &= 2\|\mathbf{a}\|(\|\mathbf{a}\| + |a_k|). \end{aligned}$$

Primijetimo da je $\|\mathbf{a}\| = \sqrt{a_k^2 + ... + a_n^2}$ i $\varphi a_k = |a_k|$.



Slika 7.1.2: Householderov vektor

Slijedi oblik vektora:

$$\mathbf{u} = \frac{\mathbf{a} - \mathbf{b}}{\|\mathbf{a} - \mathbf{b}\|} = \frac{1}{\sqrt{2\|\mathbf{a}\|(\|\mathbf{a}\| + |a_k|)}} \begin{bmatrix} a_k + \varphi\|\mathbf{a}\| \\ a_{k+1} \\ \vdots \\ a_n \end{bmatrix}.$$

Na Slici 7.1.2 vidljivo je kako je vektor $\vec{u}$ okomit na π. Zadani vektor $\vec{x}$ množen Householderovim reflektorom H daje njemu simetrični vektor s obzirom na π. U geometrijskom smislu, Householderovi reflektori predstavljaju zrcaljenje prostora.

Primjer 7.1.2. Provedimo QR faktorizaciju za matricu $A = \begin{bmatrix} 3 & 2 & 5 \\ 6 & -10 & -10 \\ 6 & -15 & 23 \end{bmatrix}$ korištenjem Householderovih reflektora.

1^o Stavimo $k = 1$, $\mathbf{a} = \begin{bmatrix} 3 \\ 6 \\ 6 \end{bmatrix}$. Označimo $\|\mathbf{a}\|$ s γ.

Sada je $\gamma = \sqrt{3^2 + 6^2 + 6^2} = 9$, $\mathbf{u}_1 = \frac{1}{\sqrt{2 \cdot 9(9+3)}} \begin{bmatrix} 3+9 \\ 6 \\ 6 \end{bmatrix} = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$.

Izračunajmo $H_1 = I - 2\mathbf{u}_1\mathbf{u}_1{}^T$ i $A_1 = H_1 A$.

$$H_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - 2 \cdot \frac{1}{6} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \end{bmatrix} = -\frac{1}{3} \begin{bmatrix} 1 & 2 & 2 \\ 2 & -2 & 1 \\ 2 & 1 & -2 \end{bmatrix}$$

$$A_1 = -\frac{1}{3} \begin{bmatrix} 1 & 2 & 2 \\ 2 & -2 & 1 \\ 2 & 1 & -2 \end{bmatrix} \begin{bmatrix} 3 & 2 & 5 \\ 6 & -10 & -10 \\ 6 & -15 & 23 \end{bmatrix} = \begin{bmatrix} -9 & 16 & \frac{-31}{3} \\ 0 & -3 & \frac{-53}{3} \\ 0 & -8 & \frac{43}{3} \end{bmatrix}$$

2^o Stavimo $k = 2$, $\mathbf{a} = \begin{bmatrix} 16 \\ -3 \\ -8 \end{bmatrix}$.

Sada je $\gamma = \sqrt{(-3)^2 + (-8)^2} = \sqrt{73}$, $\mathbf{u}_2 = \frac{1}{\sqrt{73+3\sqrt{73}}} \begin{bmatrix} 0 \\ -3 - \sqrt{73} \\ -8 \end{bmatrix}$.

Izračunajmo $H_2 = I - 2\mathbf{u}_2\mathbf{u}_2{}^T$ i $A_2 = H_2 A_1$.

$$H_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - 2 \cdot \frac{1}{146 + 6\sqrt{73}} \begin{bmatrix} 0 \\ -3 - 3\sqrt{73} \\ -8 \end{bmatrix} \begin{bmatrix} 0 & -3 - 3\sqrt{73} & -8 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{73 + 3\sqrt{73}} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 82 + 6\sqrt{73} & 24 + 8\sqrt{73} \\ 0 & 24 + 8\sqrt{73} & 64 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 9 + 3\sqrt{73} & 24 + 8\sqrt{73} \\ 0 & 24 + 8\sqrt{73} & -9 - 3\sqrt{73} \end{bmatrix}$$

$$A_2 = -\frac{1}{73 + 3\sqrt{73}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 9 + 3\sqrt{73} & 24 + 8\sqrt{73} \\ 0 & 24 + 8\sqrt{73} & -9 - 3\sqrt{73} \end{bmatrix} \begin{bmatrix} -9 & 16 & \frac{-31}{3} \\ 0 & -3 & \frac{-53}{3} \\ 0 & -8 & \frac{43}{3} \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -9 & 16 & \frac{-31}{3} \\ 0 & 219 - 73\sqrt{73} & 209 + \frac{209}{3}\sqrt{73} \\ 0 & 0 & -562 - \frac{562}{3}\sqrt{73} \end{bmatrix}$$

Kako je $A_2 = H_2 H_1 A$, slijedi da je $Q = H_1 H_2$ (što se lako izračuna), a $R = A_2$.

7.2. Osnovni QR algoritam

Algoritam 7.2.1: Osnovni QR algoritam

Ulaz: $A \in \mathbb{C}^{n\times n}$
Rezultat: U, T: $A = UTU^*$
$A_0 \leftarrow A$;
$U_0 \leftarrow I$;
za $i = 1, 2, ...$ *(do konvergencije)* **čini**
  $A_{i-1} \leftarrow Q_i R_i$;
  $A_i \leftarrow R_i Q_i$;
  $U_i \leftarrow U_{i-1} Q_i$;
kraj
$T \leftarrow A_i$;
$U \leftarrow U_i$;

Osnovni algoritam jednostavno koristi QR dekompoziciju matrice. Kada se primijeni na matrici, ostaje samo iz dijagonalnih elemenata matrice T pročitati svojstvene vrijednosti [9].

Propozicija 7.2.1. Matrice A_i dobivene osnovnim QR algoritmom unitarno su slične početnoj A.

Dokaz. Kako je $Q_i^* A_i = R_i$, slijedi $A_{i+1} = Q_i^* A_i Q_i$.

□

Tako je

$$A_i = Q_i^* A_{i-1} Q_i = Q_i^* Q_{i-1}^* A_{i-2} Q_{i-1} Q_i = ... = Q_i^* \cdots Q_1^* A_0 \underbrace{Q_1 \cdots Q_i}.$$

A_i teži k gornje trokutastoj matrici, $U_i = Q_1 \cdots Q_i$ konvergira matrici Schurovih vektora.

Prikažimo implementaciju osnovnog QR algoritma.

```
import pprint
import numpy as np
import scipy.linalg as la
```

Prilog 1: Biblioteke za implementaciju

Korištene su biblioteke *NumPy* i *SciPy Linear algebra*. Bit će korištene i u nadolazećim implementacijama. *NumPy (Numeric Python)* je biblioteka za programski jezik *Python* koja podržava rad s redovima, matricama, operacije s matricama i sadrži potrebne matematičke funkcije za izračun. *SciPy (Scientific Python)* s modulom linearne algebre dodaje potrebne funkcije iz tog područja.

```
A=np.array([[-2, -4, 2], [-2, 1, 2], [4, 2, 5]])

def konvergencija(M):
    for i in range(len(M)):
        for j in range(len(M[i])):
            if (i != j and abs(M[i][j]) > 0.001):
                return False
    return True

def QR(M):
    Ai = M
    i = 0
    print("A_" + str(i))
    pprint.pprint(Ai)
    while(not konvergencija(Ai)):
        i += 1
        Q, R = la.qr(Ai)
        Ai = np.dot(R, Q)
        print("A_" + str(i))
        pprint.pprint(Ai)
```

Prilog 2: Implementacija osnovnog QR algoritma

Argument je matrica za koju je potrebno izračunati svojstvene vrijednosti. Provodi se QR faktorizacija u petlji. Ovdje je uvjet stajanja konvergencija, no ne mora uvijek doći do nje. Za QR dekompoziciju koristi se gotova funkcija.
Dodatna funkcija provjerava konvergenciju, vraća informaciju treba li stajati nakon određenog koraka, u ovom slučaju ako su elementi ispod glavne dijagonale vrlo mali brojevi.

Primjer 7.2.1. Pogledajmo rezultate izvršavanja algoritma na matrici

$$A = \begin{bmatrix} -2 & -4 & 2 \\ -2 & 1 & 2 \\ 4 & 2 & 5 \end{bmatrix}.$$

Svojstvene vrijednosti su dakako $\{-5, 3, 6\}$.

```
A_0
array([[-2, -4,  2],
       [-2,  1,  2],
       [ 4,  2,  5]])
A_1
array([[-1.16666667,  2.37417868,  5.58290526],
       [ 2.14625752,  2.24458874, -1.97231642],
       [ 4.18717895,  0.47717333,  2.92207792]])
A_2
array([[-0.53191489, -4.37377493,  2.20860579],
       [-4.10886846,  1.63717805,  0.61166824],
       [ 5.01955861,  0.44323785,  2.89473684]])
A_3
array([[ 0.96383727,  5.97404747,  2.93772462],
       [ 4.07782168,  1.20783558, -2.43536058],
       [ 2.27909446, -0.44274068,  1.82832715]])

...

A_13
array([[ 5.52345748e+00,  3.55316846e+00, -1.26135039e+00],
       [ 1.41290533e+00, -4.52078895e+00, -1.36285988e+00],
       [ 2.91415573e-03, -1.34493048e-02,  2.99733147e+00]])
A_14
array([[ 6.09809590e+00,  8.32289496e-01,  1.56365601e+00],
       [-1.30413477e+00, -5.09900455e+00,  9.84561203e-01],
       [-1.53388536e-03, -8.34964254e-03,  3.00090865e+00]])
A_15
array([[ 5.70455423e+00,  3.14496865e+00, -1.31869058e+00],
       [ 1.00605970e+00, -4.70366615e+00, -1.30267417e+00],
       [ 7.37701926e-04, -4.91552083e-03,  2.99911191e+00]])

...

A_48
array([[ 6.00049843e+00,  2.13552243e+00,  1.43672097e+00],
       [-2.56750109e-03, -5.00049843e+00,  1.16807718e+00],
       [-8.78693163e-14, -2.39582765e-10,  3.00000000e+00]])
A_49
array([[ 5.99958376e+00,  2.14022916e+00, -1.43622104e+00],
       [ 2.13922842e-03, -4.99958376e+00, -1.16869182e+00],
       [ 4.39310047e-14, -1.43742830e-10,  3.00000000e+00]])
A_50
array([[ 6.00034625e+00,  2.13630700e+00,  1.43663766e+00],
       [-1.78293749e-03, -5.00034625e+00,  1.16817964e+00],
       [-2.19670249e-14, -8.62491150e-11,  3.00000000e+00]])

...
```

Prilog 3: Rezultat izvršavanja osnovnog QR algoritma

Primjer 7.2.2. Neka su matrice A_{49} i A_{50} jednake matricama iz Priloga 3. Prikažimo matricu $A_{50/49}$ čiji su elementi $a_{ij} = \frac{(a_{50})_{ij}}{(a_{49})_{ij}}$ za $i, j = 1, 2, 3$. Svaki kvocijent zaokružimo na 3 decimale i dobivamo:

$$\begin{bmatrix} 1.000 & 0.998 & -1.000 \\ * & 1.000 & -1.000 \\ * & * & 1.000 \end{bmatrix}.$$

Vidljivo je kako su dijagonalni elementi jednaki 1.

Primjer 7.2.3. Provedimo osnovni QR algoritam nad simetričnom matricom

$$A = \begin{bmatrix} 2 & 0 & 2 & 4 \\ 0 & 1 & -1 & 0 \\ 2 & -1 & 1 & 5 \\ 4 & 0 & 5 & 3 \end{bmatrix}.$$

Simetrični slučaj garantira da su svojstvene vrijednosti iz skupa realnih brojeva, a također postoji i ortonormirana baza svojstvenih vektora.
U ovom primjeru vrlo se brzo uočavaju jako male vrijednosti ispod glavne dijagonale pa nakon 10. koraka algoritam prestaje s radom i mogu se pročitati svojstvene vrijednosti. Prikažimo rezultat.

```
A_10
array(
[[ 9.65182505e+00, -2.38483684e-04, -4.08230603e-08, 9.52853478e-13],
 [-2.38483684e-04, -3.47689198e+00, -9.30472045e-04, 8.60122527e-09],
 [-4.08230607e-08, -9.30472045e-04,  1.25310641e+00, -1.00242377e-05],
 [ 9.52573129e-13,  8.60122525e-09, -1.00242377e-05, -4.28039479e-01]]
 )
```

Prilog 4: Rezultat izvršavanja osnovnog QR algoritma na primjeru simetrične matrice

7.3. Redukcija do Hessenbergove forme

Definicija 7.3.1 (Hessenbergova forma)**.** Matrica A je u gornjoj Hessenbergovoj formi ako za njezine elemente vrijedi

$$a_{ij} = 0, i - j \geqslant 2.$$

Kada je matrica u gornjoj Hessenbergovoj formi to znači da je gornje trokutasta i ima još jednu sporednu dijagonalu ispod one glavne,

$$\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ & * & * & * \\ & & * & * \end{bmatrix}.$$

Prikažimo ideju redukcije do Hessenbergove forme. Neka je zadana matrica reda 5,

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}.$$

Biramo Householderov reflektor tako da je

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & \\ & I_4 - 2\mathbf{u}_1\mathbf{u}_1^* \end{bmatrix}.$$

Kako je $P = P^*$, slijedi

$$P_1AP_1 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

Dalje tražimo $P_2 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & H_2 \end{bmatrix}$, pa nakon množenja slijedi

$$P_2P_1AP_1P_2 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix}.$$

Na kraju, analogno dobivano matricu u Hessenbergovoj formi

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} = P_3P_2P_1AP_1P_2P_3 = H.$$

Ako je matrica bila simetrična, simetrična je i njezina Hessenbergova forma. Tako se radi zapravo o tridijagonalizaciji [4][10].

Algoritam 7.3.1: Redukcija do Hessenbergove forme

Ulaz: $A \in \mathbb{C}^{n\times n}$,
Rezultat: H, Hessenbergova matrica
za $k = 1, 2, ..., n-2$ **čini**
  generiraj Householderov reflektorP_k;
  $A_{k+1:n,k:n} = A_{k+1:n,k:n} - 2\mathbf{u}_k(\mathbf{u}_k^* A_{k+1:n,k:n})$;
  $A_{1:n,k+1:n} = A_{1:n,k+1:n} - 2(A_{1:n,k+1:n}\mathbf{u}_k)\mathbf{u}_k^*$;
kraj
$H = A$;

Radi uštede računalnih resursa, algoritam koristi istu matricu, tako da se brojevi prepisuju jedni preko drugih, transformira se matrica dio po dio. Na kraju se na mjestu početne matrice nalazi matrica u Hessenbergovoj formi. Prikažimo implementaciju.

```
def Hessenberg(A):
    H = A
    n = len(H)
    for k in range (0, n-2):
        x = H[k+1:n,k:k+1]
        nx = la.norm(x)

        e = np.append([[nx+x[0][0]]], x[1:], axis=0)

        u = e / math.sqrt(2 * nx * (nx+x[0][0]) )

        h = np.identity(n-k-1) - 2 * np.dot(u,np.transpose(u))

        H[k+1:n,k:n] = np.dot(h,H[k+1:n,k:n])
        H[k:n,k+1:n] = np.dot(H[k:n,k+1:n],h)

    return H
```

Prilog 5: Implementacija redukcije do Hessenbergove forme matrice

Teorem 7.3.1. Ako je matrica A u gornjoj Hessenbergovoj formi, tada je Q u QR faktorizaciji $A = QR$ također u gornjoj Hessenbergovoj formi.

Dokaz. (Skica)
Uzmimo matricu reda 4. Neka je matrica u Hessenbergovoj formi

$$H = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}.$$

Djelovanjem Givensovim rotacijama, dobivamo sljedeće:

$$G(1,2,\vartheta_1)^* H = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} = H_1,$$

$$G(2,3,\vartheta_2)^* H_1 = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} = H_2,$$

$$G(3,4,\vartheta_3)^* H_2 = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} = R.$$

Prema tome za matricu reda n je

$$G_{n-1}^* \cdots G_1^* H = R \quad \Leftrightarrow \quad H = QR$$

$$\overline{H} = RQ = RG_1 \cdots G_{n-1}$$

Dakle, množenjem rotacijama s desna dobivamo:

$$RG(1,2,\vartheta_1) = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} = R_1,$$

$$R_1 G(2,3,\vartheta_2) = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} = R_2,$$

$$R_2 G(3,4,\vartheta_3) = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} = \overline{H}.$$

□

7.4. QR algoritam s pomacima

Da bi se ubrzala konvergencija algoritma, potrebno je dodati pomake. Neka su $\lambda_i \in \mathbb{C}$ svojstvene vrijednosti matrice $A \in \mathcal{M}_n(\mathbb{C})$. Tada su svojstvene vrijednosti matrice $A - \sigma I$ jednake $\lambda_i - \sigma$, gdje je I jedinična matrica i $\sigma \in \mathbb{C}$ pomak. Navedimo lemu bez dokaza.

Lema 7.4.1. Neka je $H \in \mathcal{M}_n(\mathbb{C})$ ireducibilna Hessenbergova matrica i njena QR faktorizacija je $H = QR$. Dijagonalni elementi od R su $|r_{k,k}| > 0$, za $k < n$. Ako je H singularna, tada je $r_{n,n} = 0$.

Za odabir savršenog pomaka bilo bi potrebno znati svojstvene vrijednosti zadane matrice. Prema tome, pomak trebamo odabirati heuristički. Algoritmom se može raditi deflacija, ako je zadnji element donje dijagonale vrlo mali proglasi se jednak 0, tako da se smanji matrica s kojom se računa pa se tako i ovdje smanji memorijska potreba. Prikazano primjerom

$$H = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \epsilon & \times \end{bmatrix},$$

$\epsilon = 0$ se pojavljuje na mjestu $(n, n-1)$, što je ovdje $(5, 4)$, ako je pomak bio svojstvena vrijednost te matrice. Opišimo sada pomake.

Definicija 7.4.1 (Rayleighov kvocijent)**.** Pomak je jednak zadnjem dijagonalnom elementu matrice $A = [h_{ij}] \in \mathcal{M}_n(\mathbb{C}), i, j = 1, ..., n$ u k-tom koraku algoritma,

$$\sigma_k := h_{n,n}^{(k-1)}.$$

Ovakav pomak može se rabiti kod algoritama s jednostrukim pomakom.

Definicija 7.4.2 (Wilkinsonov pomak)**.** Neka je matrica $A = [h_{ij}] \in \mathcal{M}_n(\mathbb{C}), i, j = 1, ..., n$. Pomak σ_k u k-tom koraku algoritma je svojstvena vrijednost od bloka

$$\begin{bmatrix} h_{n-1,n-1}^{(k-1)} & h_{n-1,n}^{(k-1)} \\ \\ h_{n,n-1}^{(k-1)} & h_{n,n}^{(k-1)} \end{bmatrix}$$

koja je najbliže $h_{n,n}^{(k-1)}$.

Wilkinsonov pomak može se rabiti kod simetričnog tridijagonalnog algoritma.

Definicija 7.4.3 (Francisov pomak)**.** Neka je matrica $A = [h_{ij}] \in \mathcal{M}_n(\mathbb{C}), i, j = 1, ..., n$. Pomaci $(\sigma_k)_{1,2}$ u k-tom koraku algoritma su svojstvene vrijednosti od bloka

$$\begin{bmatrix} h_{n-1,n-1}^{(k-1)} & h_{n-1,n}^{(k-1)} \\ \\ h_{n,n-1}^{(k-1)} & h_{n,n}^{(k-1)} \end{bmatrix}.$$

Francisov pomak je svakako za algoritme s dvostrukim pomakom.

Propozicija 7.4.1. Matrice A_{i+1} i A_i dobivene QR algoritmom s pomakom unitarno su slične.

Dokaz. Vrijedi $Q_i R_i = A_i - \sigma_i I$, pa množenjem dobivamo $R_i = Q_i^*(A_i - \sigma_i I)$.
Slijedi:

$$\begin{aligned} A_{i+1} &= R_i Q_i + \sigma_i I = \\ &= [Q_i^*(A_i - \sigma_i I)]Q_i + \sigma_i I = \\ &= Q_i^* A_i Q_i. \end{aligned}$$

□

Iskažimo implicitni Q teorem.

Teorem 7.4.2 (Implicitni Q teorem)**.** Neka je $A \in \mathcal{M}_{n,n}(\mathbb{R})$ i $Q = [\mathbf{q_1}, ..., \mathbf{q_n}]$, $V = [\mathbf{v_1}, ..., \mathbf{v_n}]$ ortogonalne matrice tako da je $H = [h_{ij}] = Q^* A Q$ i $G = [g_{ij}] = V^T A V$. Neka je k najmanji iz $\mathbb{N}^+$ za koji je $h_{k+1,k} = 0$, a $k = n$ ako je H ireducibilna matrica.
Ako je $\mathbf{q}_1 = \mathbf{v}_1$, tada je $\mathbf{q}_i = \pm \mathbf{v}_i$ i $|h_{i,i-1}| = |g_{i,i-1}|$ za $i = 2, ..., k$.
Ako je $k < n$, tada je $g_{k+1,k} = 0$.

Može se vidjeti kako se računa Q_i u iteraciji QR algoritma. Prvi stupac te matrice je paralelan s prvim stupcem matrice $A_i - \sigma_i I$ i norma mu je 1. Ostali stupci računaju se tako da Q_i bude ortogonalna [8][11].

7.4.1. Jednostruki pomak

Algoritam 7.4.1: QR algoritam s jednostrukim pomakom (Rayleigh)

Ulaz: $H \in \mathbb{C}^{n \times n}$, Hessenbergova matrica
Rezultat: $H = UTU^*$
$i \leftarrow 0$;
za $k = n, n-1, ..., 2$ **čini**
  ponavljaj
    $i \leftarrow i + 1$;
    $\sigma_i \leftarrow h_{k,k}^{(i-1)}$;
    $H_{i-1} - \sigma_i I \leftarrow Q_i R_i$;
    $H_i \leftarrow R_i Q_i + \sigma_i I$;
    $U_i \leftarrow U_{i-1} Q_i$;
  dok ne bude $|h_{k,k-1}^{(i)}|$ *jako mali*;
kraj
$T \leftarrow H_i$;

Dio matrice koji se razmatra postepeno se smanjuje. Prvo je to cijela matrica, nakon toga se ne razmatraju posljednji red i posljednji stupac, pa tako dalje redom. U svakom koraku ponavlja se

gore opisana ideja algoritma sve dok posljednji element na sporednoj dijagonali nije dovoljno mali da bi se mogao proglasiti nulom. Prikažimo implementaciju algoritma.

```
def QRshift(H, tol=0.001):
    I = np.zeros((len(H), len(H)), int)
    np.fill_diagonal(I, 1)
    Hi = H
    n = len(H) - 1
    i = 0
    Ui = I
    for k in range(n,0,-1):
        while (abs(Hi[k][k-1]) > tol):
            i = i + 1
            shift = Hi[k][k]
            Q, R = la.qr(Hi-shift*I)
            Hi = np.dot(R,Q) + shift*I
            Ui = np.dot(Ui,Q)
            print("H_" + str(i))
            pprint.pprint(Hi)
```

Prilog 6: Implementacija QR algoritma s jednostrukim pomakom

7.4.2. Dvostruki pomak

Problem nastaje kada matrica ima kompleksne svojstvene vrijednosti, a za takav slučaj nije pogodan jednostruki pomak. Za dvostruki pomak odabrat ćemo Francisov pomak. Neka je $F \in \mathcal{M}_2(\mathbb{R})$, za matricu u Hessenbergovoj formi $H = [h_{ij}] \in \mathcal{M}_n(\mathbb{R})$ prikažimo

$$F = \begin{bmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{bmatrix}.$$

Ako je $\sigma_1 \in \mathbb{C} \setminus \mathbb{R}$, tada je $\sigma_2 = \overline{\sigma_1}$. Izaberimo $\sigma_{1,2}$ kao pomake. Provedimo prva dva koraka QR dekompozicije pa je na početku

$$\begin{cases} H_0 - \sigma_1 I = Q_1 R_1 \\ H_1 = R_1 Q_1 + \sigma_1 I \end{cases},$$

$$\begin{cases} H_1 - \sigma_2 I = Q_2 R_2 \\ H_2 = R_2 Q_2 + \sigma_2 I \end{cases}.$$

Slijedi $Q_1 \cdot /$ $\qquad R_1Q_1 + (\sigma_1 - \sigma_2) = Q_2R_2 \qquad / \cdot R_1.$

$$\begin{aligned} Q_1R_1Q_1R_1 + (\sigma 1 - \sigma_2)Q_1R_1 &= Q_1R_1[Q_1R_1 + (\sigma_1 - \sigma_2)] = \\ &= (H_0 - \sigma_1 I)(H_0 - \sigma_2 I) = \\ &= Q_1Q_2R_2R_1 \end{aligned}$$

Neka je $M := (H_0 - \sigma_1 I)(H_0 - \sigma_2 I)$.

$$M = H^2 - sH + tI$$

tako da je

$$\begin{cases} s = h_{n-1,n-1} + h_{n,n} = \operatorname{tr}(F) \in \mathbb{R}, \\ t = h_{n-1,n-1}h_{n,n} - h_{n-1,n}h_{n,n-1} = \det(F) \in \mathbb{R}. \end{cases}$$

M ima jednu dijagonalu više od zadane matrice, oblika je

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \end{bmatrix}.$$

Namjestimo matricu Z tako da je realna ortogonalna $Z := Q_1Q_2$, pa slijedi da je

$$H_2 = Q_2^* H_1 Q_2 = Q_1^* Q_2^* H_0 Q_1 Q_2 = Z^T H_0 Z.$$

Sada je potrebno izbjeći račun s kompleksnim brojevima. Da bi se proveo dvostruki pomak, potrebno je odabrati Householderov reflektor P_0 oblika

$$P_0 = \begin{bmatrix} \times & \times & \times & & & \\ \times & \times & \times & & & \\ \times & \times & \times & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix}.$$

Množenjem dobivamo

$$P_0^T H P_0 = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ \bigstar & \times & \times & \times & \times & \times & \times \\ \bigstar & \bigstar & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{bmatrix}.$$

Pojavljuju se "izbočine" (elementi matrice koji su prije transformacije bili 0, no djelovanjem više nisu) koje je potrebno istjerivati dolje dijagonalno. Dalje je

$$P_1 = \begin{bmatrix} 1 & & & & & & \\ & \times & \times & \times & & & \\ & \times & \times & \times & & & \\ & \times & \times & \times & & & \\ & & & & \ddots & & \\ & & & & & 1 & \end{bmatrix},$$

$$P_1^T P_0^T H P_0 P_1 = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times \\ 0 & \bigstar & \times & \times & \times & \times & \times \\ & \bigstar & \bigstar & \times & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{bmatrix}.$$

Analogno dalje do zadnjeg koraka je rezultantna matrica oblika

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times & \times \\ & & & \times & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & 0 & \times & \times \end{bmatrix}.$$

Prikažimo algoritam u pseudokodu [5].

Algoritam 7.4.2: Implicitni QR algoritam s dvostrukim pomakom (Francis)

Ulaz: $H \in \mathbb{R}^{n\times n}$, Hessenbergova matrica

Rezultat: $H = UTU^*$

$p \leftarrow n$;

dok $p > 2$ **čini**

$q \leftarrow p-1$;

$s \leftarrow H_{qq} + H_{pp}$;

$t \leftarrow H_{qq}H_{pp} - H_{qp}H_{pq}$;

$x \leftarrow H_{11}^2 + H_{12}H_{21} - sH_{11} + t$;

$y \leftarrow H_{21}(H_{11} + H_{22} - s)$;

$z \leftarrow H_{21}H_{32}$;

za $k = 0, ..., p-3$ **čini**

Householder $P, P^T[x, y, z]^T$;

$r \leftarrow \max\{1, k\}$;

$H_{k+1:k+3,r:n} \leftarrow P^T H_{k+1:k+3,r:n}$;

$r \leftarrow \min\{k+4, p\}$;

$H_{1:r,k+1:k+3} \leftarrow H_{1:r,k+1:k+3}P$;

$x \leftarrow H_{(k+2)(k+1)}$;

$y \leftarrow H_{(k+3)(k+1)}$;

ako $k < p-3$ **onda**

$z \leftarrow H_{(k+4)(k+1)}$;

kraj

kraj

Givens $P, P^T[x, y]^T$;

$H_{q:p,p-2:n} \leftarrow P^T H_{q:p,p-2:n}$;

$H_{1:p,p-1:p} \leftarrow H_{1:p,p-1:p}P$;

ako $|H_{pq}| < \epsilon(|H_{qq}| + |H_{pp}|)$ **onda**

$H_{pq} \leftarrow 0$;

$p \leftarrow p-1$;

$q \leftarrow p-1$;

kraj

inače ako $|H_{(p-1)(q-1)}| < \epsilon(|H_{(q-1)(q-1)}| + |H_{qq}|)$ **onda**

$H_{(p-1)(q-1)} \leftarrow 0$;

$p \leftarrow p-2$;

$q \leftarrow p-1$;

kraj

kraj

```
def QRdoubleShift(M, e=0.0001):
    H = M

    n = len(H)
    p = n - 1

    while (p > 1):
        q = p - 1
        s = H[q][q] + H[p][p]
        t = H[q][q] * H[p][p] - H[q][p] * H[p][q]

        x = H[0][0] ** 2 + H[0][1] * H[1][0] - s * H[0][0] + t
        y = H[1][0] * (H[0][0] + H[1][1] - s)
        z = H[1][0] * H[2][1]

        for k in range (-1,p-2):
            x = np.array([[x],[y],[z]])
            P = house(x,p-1)
            r = max(0,k)
            H[k+1:k+4,r:n] = np.dot(np.transpose(P),H[k+1:k+4,r:n])
            r = min(k+4,p)
            H[0:r+1,k+1:k+4] = np.dot(H[0:r+1,k+1:k+4],P)


            x = H[k+2][k+1]
            y = H[k+3][k+1]
            if (k < p - 3):
                z = H[k+4][k+1]

        P = givens(x,y)
        H[q:p+1,p-2:n] = np.dot(np.transpose(P),H[q:p+1,p-2:n])
        H[0:p+1,p-1:p+1] = np.dot(H[0:p+1,p-1:p+1],P)

        if (abs(H[p][q]) < e*(abs(H[q][q]) + abs(H[p][p]))):
            H[p][q] = 0
            p = p - 1
            q = p - 1
        elif (abs(H[p-1][q-1]) < e*(abs(H[q-1][q-1]) + abs(H[q][q]))):
            H[p-1][q-1] = 0
            p = p - 2
            q = p - 1

    return H
```

Prilog 7: Implementacija implicitnog QR algoritma s dvostrukim pomakom

7.5. Simetrični tridijagonalni QR algoritam

Neka je simetrična tridijagonalna matrica

$$A = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & & & \\ & \ddots & \ddots & & \\ & \ddots & a_{n-1} & b_{n-1} & \\ & & b_{n-1} & a_n \end{bmatrix} \in \mathcal{M}_n(\mathbb{R}).$$

Provođenjem QR algoritma, forma ostaje očuvana. Hermitske matrice imaju realni spektar, pa se možemo ograničiti na provođenje jednostrukih pomaka. Redukcija pune hermitske matrice do Hessenbergove forme daje hermitsku Hessenbergovu matricu koja je zapravo realna simetrična tridijagonalna matrica.

Prikažimo na primjeru djelovanje Givensovih rotacija i tjeranje "izbočine" (elemenata matrice koji su prije transformacije bili 0, no djelovanjem više nisu) dijagonalno [5][11]. Neka je simetrična tridijagonalna matrica

$$A = \begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

Odaberimo Givensove rotacije u i-tom koraku $G_{i-1} = G(i, i+1, \vartheta_{i-1}), i = 1, ..., 4$. Slijedi

$$G_0^* A G_0 = \begin{bmatrix} \times & \times & \bigstar & & \\ \times & \times & \times & & \\ \bigstar & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix},$$

$$G_1^* G_0^* A G_0 G_1 = \begin{bmatrix} \times & \times & 0 & & \\ \times & \times & \times & \bigstar & \\ 0 & \times & \times & \times & \\ & \bigstar & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

Naposljetku je

$$G_3^* G_2^* G_1^* G_0^* A G_0 G_1 G_2 G_3 = \overline{A} = \begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & 0 \\ & & \times & \times & \times \\ & & 0 & \times & \times \end{bmatrix}.$$

Dakle, općenito je

$$\overline{A} = Q^* A Q, \qquad Q = G_0 G_1 \cdots G_{n-2}$$

za matricu reda n. Prikažimo algoritam u pseudokodu [5].

Algoritam 7.5.1: Simetrični tridijagonalni QR algoritam

Ulaz: $A \in \mathbb{R}^{n\times n}$ simetrična tridijagonalna, dijagonalni elementi su $a = a_1, ..., a_n$, vandijagonalni elementi su $b = b_2, ..., b_n$

Rezultat: svojstvene vrijednosti $a = \lambda_1, ..., \lambda_n$, korespondirajući svojstveni vektori $\mathbf{q}_1, ..., \mathbf{q}_n$ u Q

$m \leftarrow n$;

dok $m > 1$ **čini**

  $d \leftarrow \frac{a_{m-1} - a_m}{2}$;

  ako $d = 0$ **onda**

    $s \leftarrow a_m - |b_m|$;

  kraj

  inače

    $s \leftarrow a_m - \frac{b_m^2}{d + \text{sign}\,(d)\sqrt{d^2 + b_m^2}}$;

  kraj

  $x \leftarrow a_1 - s; \quad y \leftarrow b_2$;

  za $k = 1, ..., m-1$ **čini**

    ako $m > 2$ **onda**

      $[c, s] \leftarrow$ givens(x, y) ;

    kraj

    inače

      $[c, s]$ takvi da je dijagonalna $\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a_1 & b_2 \\ b_2 & a_2 \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$;

    kraj

    $w \leftarrow cx - sy; \quad d \leftarrow a_k - a_{k+1}; \quad z \leftarrow (2cb_{k+1} + ds)s;$

    $a_k \leftarrow a_k - z; \quad a_{k+1} \leftarrow a_{k+1} + z; \quad b_{k+1} \leftarrow dcs + (c^2 - s^2)b_{k+1}; \quad x \leftarrow b_{k+1}$;

    ako $k > 1$ **onda**

      $b_k \leftarrow w$;

    kraj

    ako $k < m - 1$ **onda**

      $y \leftarrow -sb_{k+2}; \quad b_{k+2} \leftarrow cb_{k+2}$;

    kraj

    $Q_{1:n,k:k+1} \leftarrow Q_{1:n,k:k+1} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$;

  kraj

  ako $|b_m| < \epsilon(|a_{m-1}| + |a_m|)$ **onda**

    $m \leftarrow m - 1$;

  kraj

kraj

```
def QRsymetricTridiagonal(M,e=0.00001):
    a = np.append([0],M.diagonal())
    b = np.append([0,0],M.diagonal(1))
    a=np.array(a,dtype=float)
    b=np.array(b,dtype=float)
    n = len(M)
    m = n

    while(m > 1):
        d = (a[m-1] - a[m]) / 2
        if (d == 0):
            s = a[m] - abs(b[m])
        else:
            s=a[m]-b[m]**2/(d+np.sign(d)*math.sqrt(d**2+b[m]**2))

        x = a[1] - s
        y = b[2]

        for k in range (1,m):
            if (m > 2):
                c, s = givens1(x,y)
            else:
                c, s = givens1(a[1],b[2])

            w = c * x - s * y
            d = a[k] - a[k+1]
            z = (2 * c * b[k+1] + d * s) * s

            a[k] = a[k] - z
            a[k+1] += z
            b[k+1] = d * c * s + (c ** 2 - s ** 2) * b[k+1]

            x = b[k+1]

            if (k > 1):
                b[k] = w
            if (k < m-1):
                y = -s * b[k+2]
                b[k+2] = c * b[k+2]

        if (abs(b[m]) < e*(abs(a[m-1]) + abs(a[m]))):
            m -= 1

    return a
```

Prilog 8: Implementacija simetričnog tridijagonalnog QR algoritma

8. Jacobijev algoritam

Ideju za računanje svojstvenih vrijednosti predložio je Carl Gustav Jacob Jacobi još 1846. godine, a do izražaja je došla tek kasnije zbog uporabe računala. Potrebno je dijagonalizirati matricu kako bi se pročitale svojstvene vrijednosti. Jacobijev algoritam namijenjen je za realne simetrične matrice.

8.1. Dijagonalizacija rotacijom u ravnini

Definicija 8.1.1. Frobeniusova norma matrice $A = [a_{ij}]$ je

$$\|A\|_{\mathbb{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\operatorname{tr}(A^*A)}\text{[8].}$$

Cilj je postići da bude što manja vandijagonalna Frobeniusova norma

$$\operatorname{off}(A) = \sqrt{\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}a_{ij}^2},$$

odnosno da bude mala do mjere da se vandijagonalni elementi mogu proglašavati nulom. Razradimo primjer matrice malog reda. Neka je dana simetrična matrica drugog reda,

$$A = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{bmatrix}.$$

Postavimo skraćene oznake $s := \sin(\theta)$, $c := \cos(\theta)$, $t := \operatorname{tg}(\theta)$.
Želimo poništiti vandijagonalne elemente, tako da je potrebno dobiti

$$A' = \begin{bmatrix} a'_{ii} & 0 \\ 0 & a'_{jj} \end{bmatrix} = R(\theta)AR(\theta)^*,$$

gdje je $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ ravninska rotacija.

Sredimo izraz:

$$\begin{aligned} R(\theta)AR(\theta)^* &= \begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{bmatrix}\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \\ &= \begin{bmatrix} a_{ii}c^2 - 2a_{ij}sc + a_{jj}s^2 & (c^2-s^2)a_{ij} + (a_{ii}-a_{jj})sc \\ (c^2-s^2)a_{ij} + (a_{ii}-a_{jj})sc & a_{ii}s^2 + 2a_{ij}sc + a_{jj}c^2 \end{bmatrix} \end{aligned}$$

Potrebno je pronaći kut za koji vrijedi

$$(c^2 - s^2)a_{ij} + (a_{ii} - a_{jj})sc = 0.$$

$$\begin{cases} c^2 - s^2 = \cos(2\theta) \\ 2sc = \sin(2\theta) \end{cases} \Rightarrow \cos(2\theta)a_{ij} + \tfrac{1}{2}\sin(2\theta)(a_{ii} - a_{jj}) = 0 \quad / : \sin(2\theta)$$

$$\operatorname{ctg}(2\theta)a_{ij} + \frac{1}{2}(a_{ii} - a_{jj}) = 0 \quad / : a_{ij}$$

$$\operatorname{ctg}(2\theta) + \frac{a_{ii} - a_{jj}}{2a_{ij}} = 0$$

$$\operatorname{ctg}(2\theta) = \frac{a_{jj} - a_{ii}}{2a_{ij}}$$

Također je

$$\varsigma := \operatorname{ctg}(2\theta) = \frac{\cos(2\theta)}{\sin(2\theta)} = \frac{c^2 - s^2}{2cs} : \frac{c^2}{c^2} = \frac{1 - t^2}{2t}.$$

Uzevši u obzir da znamo kotangens kuta, potrebno je riješiti jednadžbu:

$$\varsigma = \frac{1 - t^2}{2t} \quad / \cdot 2t$$

$$t^2 + 2\varsigma t - 1 = 0$$

$$t_{1,2} = -\varsigma \pm \sqrt{\varsigma^2 + 1}$$

Biramo po apsolutnoj vrijednosti manji kut pa je tako $t = \begin{cases} -\varsigma + \sqrt{\varsigma^2 + 1}, & \varsigma \geq 0 \\ -\varsigma - \sqrt{\varsigma^2 + 1}, & \varsigma < 0 \end{cases}$.

Deracionalizacijom izbjegavanjem kraćenja dobivamo

$$t = \frac{\operatorname{sign}(\varsigma)}{|\varsigma| + \sqrt{\varsigma^2 + 1}}.$$

Vrijedi $s^2 + c^2 = 1 \quad / : c^2$, pa je

$$t^2 + 1 = \frac{1}{c^2}, \quad c^2 = \frac{1}{1 + t^2}, \quad c = \frac{1}{\sqrt{1 + t^2}}.$$

Kako je $t = \frac{s}{c}$, slijedi da je

$$s = ct = \frac{t}{\sqrt{1 + t^2}}.$$

Za dijagonalne elemente od A je:

$$\begin{aligned}
a'_{ii} &= a_{ii}c^2 - 2a_{ij}sc + a_{jj}s^2 = \\
&= a_{ii}c^2 + a_{ii}s^2 - a_{ii}s^2 - 2a_{ij}sc + a_{jj}s^2 = \\
&= a_{ii} + (a_{jj} - a_{ii})s^2 - 2a_{ij}sc = \\
&= a_{ii} + 2a_{ij}s^2\varsigma - 2a_{ij}sc = \\
&= a_{ii} + 2a_{ij}s^2\frac{c^2 - s^2}{2sc} - a_{ij}sc = \\
&= a_{ii} + a_{ij}t(c^2 - s^2) - 2a_{ij}sc = \\
&= a_{ii} - a_{ij}t(s^2 - c^2 + 2c^2) = \\
&= a_{ii} - a_{ij}t
\end{aligned}$$

Slično je i

$$a'_{jj} = a_{jj} + a_{ij}t.$$

Definicija 8.1.2. Jacobijeve rotacije su

$$J(p,q,\theta) = \begin{array}{c} \\ p \\ \\ q \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccc} & & p & & q & \end{array} \\ \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & \cdots & -s & \\ & & \vdots & \ddots & \vdots & \\ & & s & \cdots & c & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix} \end{array},$$

gdje je $c = \cos(\theta)$ i $s = \sin(\theta)$.

Jacobijeve rotacije odgovaraju Givensovim rotacijama. Kako je prikazano dijagonaliziranje matrica reda 2, tako je potrebno za matricu reda n iterativno provoditi postupak.

Prije i poslije transformiranja za matricu $A = [a_{ij}] \in \mathcal{M}_n(\mathbb{R})$ vrijedi

$$(a'_{ik})^2 + (a'_{jk})^2 = a_{ik}^2 + a_{jk}^2.$$

Za $k \neq i, j$ je

$$\begin{cases} a'_{ik} = ca_{ik} - sa_{jk} \\ a'_{ki} = ca_{ki} + sa_{kj} \\ a'_{jk} = sa_{ik} + ca_{jk} \\ a'_{kj} = sa_{ki} + ca_{kj} \end{cases}.$$

Za $k, l \neq i, j$ je

$$a'_{kl} = a_{kl}.$$

Prikažimo pseudokod računanja objašnjenog kosinus-sinus para [8].

Algoritam 8.1.1: Simetrična Schurova dekompozicija

Ulaz: $A \in \mathbb{R}^{n\times n}$, $p, q \in \mathbb{Z}, 1 \leq p < q \leq n$

Rezultat: c, s, tako da je $B = J(p, q, \theta)^T AJ(p, q, \theta)$ i $b_{pq} = b_{qp} = 0$

$c \leftarrow 1$;

$s \leftarrow 0$;

ako $A_{pq} \neq 0$ **onda**

  $\tau \leftarrow \frac{A_{qq}-A_{pp}}{2A_{pq}}$;

  ako $\tau \geq 0$ **onda**

    $t \leftarrow \frac{1}{\tau+\sqrt{1+\tau^2}}$;

  kraj

  inače

    $t \leftarrow \frac{1}{\tau-\sqrt{1+\tau^2}}$;

  kraj

  $c \leftarrow \frac{1}{\sqrt{1+t^2}}$;

  $s \leftarrow tc$;

kraj

8.2. Algoritam - Poništavanje najvećeg vandijagonalnog elementa

Prikažimo Jacobijev algoritam kod kojeg je redoslijed poništavanja vandijagonalnog elementa tako da se odabere najveći po apsolutnoj vrijednosti [8] [12].

Algoritam 8.2.1: Jacobijev algoritam - poništavanje najvećeg vandijagonalnog elementa

Ulaz: $A \in \mathbb{R}^{n\times n}$, simetrična matrica

Rezultat: $A = V^T AV$, gdje je V ortogonalna i $\operatorname{off}(V^T AV) \leq \operatorname{tol}\|A\|_F$

$V \leftarrow I_n$;

$\delta \leftarrow \operatorname{tol}\|A\|_F$;

dok $\operatorname{off}(A) > \delta$ **čini**

  $(p, q) : |a_{pq}| = \max_{i\neq j} |a_{ij}|$;

  $[c, s] \leftarrow \mathsf{SymSchur}(A, p, q)$;

  $A \leftarrow J(p, q, \theta)^T AJ(p, q, \theta)$;

  $V \leftarrow VJ(p, q, \theta)$;

kraj

Korak algoritma ponavlja se sve dok se ne ispuni zadani uvjet o iznosu objašnjene norme. Odabere se vandijagonalni element najveće apsolutne vrijednosti, nađe se kosinus-sinus par i naposljetku primijeni rotacija da bi došlo do poništavanja. Prikažimo implementaciju ove pivotne strategije.

```
def Jacobi(M,tol):
    A = M
    n = len(A)
    V = np.identity(n)
    d = tol * off(A)
    while (off(A) > d):
        p, q = maxElement(A)
        c, s = symSchur(A,p,q)
        J = np.identity(n)
        J[p][p] = c
        J[q][p] = -s
        J[p][q] = s
        J[q][q] = c

        A = np.dot(np.dot(np.transpose(J),A),J)
        V = np.dot(V,J)

    return A
```

Prilog 9: Implementacija Jacobijevog algoritma

Ulazne vrijednosti su sama matrica kojoj će se računati svojstvene vrijednosti i željena tolerancija. Posebno je dodana funkcija koja pronalazi po apsolutnoj vrijednosti najveći vandijagonalni element matrice.

8.3. Algoritam - Ciklički obilazak

Sljedeća pivotna strategija je ciklički obilazak. Prikažimo pseudokod takvog Jacobijevog algoritma [8].

Algoritam 8.3.1: Jacobijev algoritam - ciklički obilazak

Ulaz: $A \in \mathbb{R}^{n\times n}$, simetrična matrica
Rezultat: $A = V^T AV$, gdje je V ortogonalna i $\operatorname{off}(V^T AV) \leq \operatorname{tol}\|A\|_F$
$V \leftarrow I_n$;
$\delta \leftarrow \operatorname{tol}\|A\|_F$;
dok $\operatorname{off}(A) > \delta$ **čini**
  za $p = 1, ..., n-1$ **čini**
    za $q = p+1, ..., n$ **čini**
      $[c, s] \leftarrow \text{SymSchur}(A, p, q)$;
      $A \leftarrow J(p, q, \theta)^T AJ(p, q, \theta)$;
      $V \leftarrow VJ(p, q, \theta)$;
    kraj
  kraj
kraj

Sam postupak algoritma i uvjet stajanja s obzirom na vandijagonalnu normu je isti kao i u prošlom algoritmu. Ovdje se odabire element za izračun kosinus-sinus para prolazeći vandijagonalne elemente po recima, a moguće je prolaziti i po stupcima. Prikažimo primjer implementacije i takvog obilaska.

```
def cyclicJacobi(M,tol):
    A = M
    n = len(A)
    V = np.identity(n)
    d = tol * off(A)
    while (off(A) > d):
        for p in range (0,n-1):
            for q in range (p+1,n):
                c, s = symSchur(A,p,q)
                J = np.identity(n)
                J[p][p] = c
                J[q][p] = -s
                J[p][q] = s
                J[q][q] = c

                A = np.dot(np.dot(np.transpose(J),A),J)
                V = np.dot(V,J)

    return A
```

Prilog 10: Implementacija Jacobijevog algoritma - ciklički obilazak

Ulazne vrijednosti su sama matrica kojoj će se računati svojstvene vrijednosti i željena tolerancija.

9. Algoritam "podijeli pa vladaj"

Ovaj način izračuna svojstvenih vrijednosti osmislio je J. J. M. Cuppen 1981. godine. Radi se o brzom algoritmu za računanje svojstvenih vrijednosti tridijagonalnih matrica rekurzivno. M. Gu i S. C. Eisenstat su 1995. godine iznijeli "podijeli pa vladaj" algoritam za stabilno računanje svojstvenih vektora.

9.1. Podjela problema

"Podijeli pa vladaj" pristup u informatici označava podjelu većeg problema na manje, jednostavnije. Rješava se često rekurzivno. Opišimo u tri koraka ideju računanja svojstvenih vrijednosti simetrične tridijagonalne matrice:

1. podjela početnog problema na manje tridijagonalne probleme, sve do kada se ne dođe do malog problema kojeg je moguće riješiti jednostavno, direktno,
2. konkretno rješavanje manjih problema,
3. davanje rješenja za cjelokupni problem kao kombinacije spomenutih manjih rješenja.

Radi uštede računalnih resursa može se spriječiti daljnje korištenje rekurzije od trenutka kada se podjelom dođe do matrica dovoljno malog reda čije će se svojstvene vrijednosti tada računati drugim algoritmom, primjerice QR.

Neka je zadana matrica $T \in \mathcal{M}_n(\mathbb{R})$, simetrična tridijagonalna. Raspišimo ju na sljedeći način [4][11]:

$$T = \left[\begin{array}{cccc|cccc} a_1 & b_1 & & & & & & \\ b_1 & \ddots & \ddots & & & & & \\ & \ddots & a_{m-1} & b_{m-1} & & & & \\ & & b_{m-1} & a_m & b_m & & & \\ \hline & & & b_m & a_{m+1} & b_{m+1} & & \\ & & & & b_{m+1} & \ddots & \ddots & \\ & & & & & \ddots & a_{n-1} & b_{n-1} \\ & & & & & & b_{n-1} & a_n \end{array}\right] =$$

$$= \left[\begin{array}{cccc|cccc} a_1 & b_1 & & & & & & \\ b_1 & \ddots & \ddots & & & & & \\ & \ddots & a_{m-1} & b_{m-1} & & & & \\ & & b_{m-1} & a_m \mp b_m & & & & \\ \hline & & & & a_{m+1} \mp b_m & b_{m+1} & & \\ & & & & b_{m+1} & \ddots & \ddots & \\ & & & & & \ddots & a_{n-1} & b_{n-1} \\ & & & & & & b_{n-1} & a_n \end{array}\right] + \left[\begin{array}{cc|cc} & & & \\ & & \pm b_m & b_m \\ \hline & & b_m & \pm b_m \\ & & & \\ & & & \end{array}\right] =$$

$$= \left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + \rho \mathbf{u}\mathbf{u}^T,$$

gdje su $\rho = \pm b_m$ i $\mathbf{u}^T = \left[\begin{array}{c|c} 0, ..., 0, 1 & 1, 0, ..., 0 \end{array}\right]$.

Neka su rastavi svojstvenih problema:

$$T_i = Q_i \Lambda_i Q_i^T, \quad Q_i^T Q_i = I, \qquad i = 1, 2.$$

Prema tome raspišimo

$$\left[\begin{array}{c|c} Q_1^T & \\ \hline & Q_2^T \end{array}\right] \left(\left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + \rho \mathbf{u}\mathbf{u}^T \right) \left[\begin{array}{c|c} Q_1 & \\ \hline & Q_2 \end{array}\right] =$$

$$= \left[\begin{array}{c|c} \Lambda_1 & \\ \hline & \Lambda_2 \end{array}\right] + \rho \mathbf{v}\mathbf{v}^T,$$

tako da je $\mathbf{v} = \begin{bmatrix} Q_1^T & \\ & Q_2^T \end{bmatrix}$ i $\mathbf{u} = \begin{bmatrix} \pm\text{posljednji red } Q_1 \\ \text{prvi red } Q_2 \end{bmatrix}$.

9.2. Konstrukcija algoritma

Podjelom dolazimo do svojstvenog problema

$$(D + \rho \mathbf{v}\mathbf{v}^T)\mathbf{x} = \lambda \mathbf{x}, \qquad D = \Lambda_1 \oplus \Lambda_2 = \operatorname{diag}(\lambda_1, ..., \lambda_n).$$

Dakle, D je dijagonalna matrica, neka je i $\widehat{D} = D + \rho \mathbf{v}\mathbf{v}^T$ tako da su dijagonalni elementi $d_1 > d_2 > ... > d_n$.

Propozicija 9.2.1. Neka su dani vektori $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Vrijedi

$$\det(I + \mathbf{x}\mathbf{y}^T) = 1 + \mathbf{y}^T \mathbf{x}.$$

Dokaz. Znamo da je

$$\det(I + \mathbf{x}\mathbf{y}^T) = \prod_{k=1}^{n} \lambda_k(I + \mathbf{x}\mathbf{y}^T),$$

$$\lambda_k(I + \mathbf{x}\mathbf{y}^T) = 1 + \lambda_k(\mathbf{x}\mathbf{y}^T), \quad k = 1, ..., n.$$

$$\det(I + \mathbf{x}\mathbf{y}^T) = \prod_{k=1}^{n}(1 + \lambda_k(\mathbf{x}\mathbf{y}^T))$$

Potrebno je pronaći svojstvene vrijednosti od matrice $\mathbf{x}\mathbf{y}^T$, što se dijeli na dva slučaja.
1^o Za $\mathbf{x} = 0$ ili $\mathbf{y} = 0$ je $\mathbf{x}\mathbf{y}^T = 0$, $\mathbf{y}^T\mathbf{x} = 0$ i $\det I = 1$ pa tvrdnja vrijedi.
2^o Uzmimo da je $\mathbf{x}, \mathbf{y} \neq 0$. $r(\mathbf{x}\mathbf{y}^T) = 1$ i neka je λ_0 jedina svojstvena vrijednost koja nije 0. Vrijedi

$$\lambda_0 = \operatorname{tr}(\mathbf{x}\mathbf{y}^T) = \mathbf{y}^T\mathbf{x}.$$

Naposljetku slijedi da je

$$\det(I + \mathbf{x}\mathbf{y}^T) = 1 + \lambda_0 = 1 + \mathbf{y}^T\mathbf{x}.$$

□

Lema 9.2.1. Neka je dana matrica $D = \operatorname{diag}(d_1, ..., d_n) \in \mathcal{M}_n(\mathbb{R})$ i $d_1 > ... > d_n$. $\rho \neq 0, \rho \in \mathbb{R}$ i $\mathbf{v} \in \mathbb{R}^n$ ne sadrži 0. Ako je $(D + \rho\mathbf{v}\mathbf{v}^T)\mathbf{x} = \lambda\mathbf{x}, \mathbf{x} \neq 0$, tada je $\mathbf{v}^T\mathbf{x} \neq 0$ i $D - \lambda I$ je nesingularna.

Dokaz. Ako je λ svojstvena vrijednost od D znači da za neki k je $\lambda = d_k$ i

$$e_k^T((D - \lambda I)\mathbf{x} + \rho\mathbf{v}^T\mathbf{x}\mathbf{v}) = \rho\mathbf{v}^T\mathbf{x}v_k = 0.$$

Kako je $\rho, v_k \neq 0$ slijedi da je $\mathbf{v}^T\mathbf{x} = 0$ i $D\mathbf{x} = \lambda\mathbf{x}$. Prema pretpostavci D ima različite svojstvene vrijednosti pa je prethodna tvrdnja kontradiktorna.
Prema tome, D i $D + \rho\mathbf{v}\mathbf{v}^T$ nemaju zajedničkih svojstvenih vrijednosti i $\mathbf{v}^T\mathbf{x} \neq 0$.

□

Teorem 9.2.2. Neka je dana matrica $D = \operatorname{diag}(d_1, ..., d_n) \in \mathcal{M}_n(\mathbb{R})$ i $d_1 > ... > d_n$. $\rho \neq 0, \rho \in \mathbb{R}$ i $\mathbf{v} \in \mathbb{R}^n$ ne sadrži 0. Neka je $Q \in \mathcal{M}_n(R)$ ortogonalna matrica tako da $Q^T(D + \rho\mathbf{v}\mathbf{v}^T)Q = \operatorname{diag}(\lambda_1, ..., \lambda_n), \lambda_1 \geqslant ... \geqslant \lambda_n$ i $Q = [\mathbf{x}_1|...|\mathbf{x}_n]$. Tada vrijedi:

- λ_i su nultočke od $f(\lambda) = 1 + \rho\mathbf{v}^T(D - \lambda I)^{-1}\mathbf{v}$,
- za $\rho > 0$ je $\lambda_1 > d_1 > \lambda_2 > ... > \lambda_n > d_n$,
  za $\rho < 0$ je $d_1 > \lambda_1 > d_2 > ... > d_n > \lambda_2$,
- svojstveni vektor $\mathbf{x}_i$ je proporcionalan $(D - \lambda_i I)^{-1}\mathbf{v}$.

Dokaz. Kako je $(D + \rho\mathbf{v}\mathbf{v}^T)\mathbf{x} = \lambda\mathbf{x}$. Raspišimo karakterističnu jednadžbu:

$$\begin{aligned} k_{D+\rho\mathbf{v}\mathbf{v}^T} &= \det(D + \rho\mathbf{v}\mathbf{v}^T - \lambda I) = \\ &= \det\left[(D - \lambda I)(I + \rho(D - \lambda I)^{-1}\mathbf{v}\mathbf{v}^T)\right] \end{aligned}$$

Kako je $D - \lambda I$ nesingularna i λ predstavlja svojstvenu vrijednost $D + \rho \mathbf{v}\mathbf{v}^T$ slijedi:

$$\det\left(I + \underbrace{\rho(D - \lambda I)^{-1}}\,\mathbf{v}\,\underbrace{\mathbf{v}^T}\right) = 0$$

$$1 + \rho \mathbf{v}^T (D - \lambda I)^{-1} \mathbf{v} = 0$$

Nadalje, funkcija i derivacija su

$$f(\lambda) = 1 + \rho\left(\frac{v_1^2}{d_1 - \lambda} + \cdots + \frac{v_n^2}{d_n - \lambda}\right),$$

$$f'(\lambda) = \rho\left(\frac{v_1^2}{(d_1 - \lambda)^2} + \cdots + \frac{v_n^2}{(d_n - \lambda)^2}\right).$$

Funkcija f je monotona između čvorova d_k, pa ima n korijena, svakog u jednom intervalu. U $\langle d_n, d_{n-1}\rangle, ..., \langle d_2, d_1\rangle, \langle d_1, \infty\rangle$ za $\rho > 0$ ili u $\langle -\infty, d_n\rangle, \langle d_n, d_{n-1}\rangle, ..., \langle d_2, d_1\rangle$ za $\rho < 0$.

Kako je $(D - \lambda I)\mathbf{x} + \rho \mathbf{v}\mathbf{v}^T\mathbf{x} = 0$, množenjem s lijeva s $(D - \lambda I)^{-1}$ i sređivanjem dobivamo $\mathbf{x} = -\rho(D - \lambda I)^{-1}\mathbf{v}(\mathbf{v}^T\mathbf{x})$ što pokazuje proporcionalnost.

□

Potrebno je riješiti karakterističnu jednadžbu za $\widehat{D}$. Dakle znamo da je

$$k_{\widehat{D}} = \det(\widehat{D} - \lambda I) = 1 + \rho \mathbf{v}^T (D - \lambda I)^{-1} \mathbf{v} = 0.$$

Napišimo izraz kao

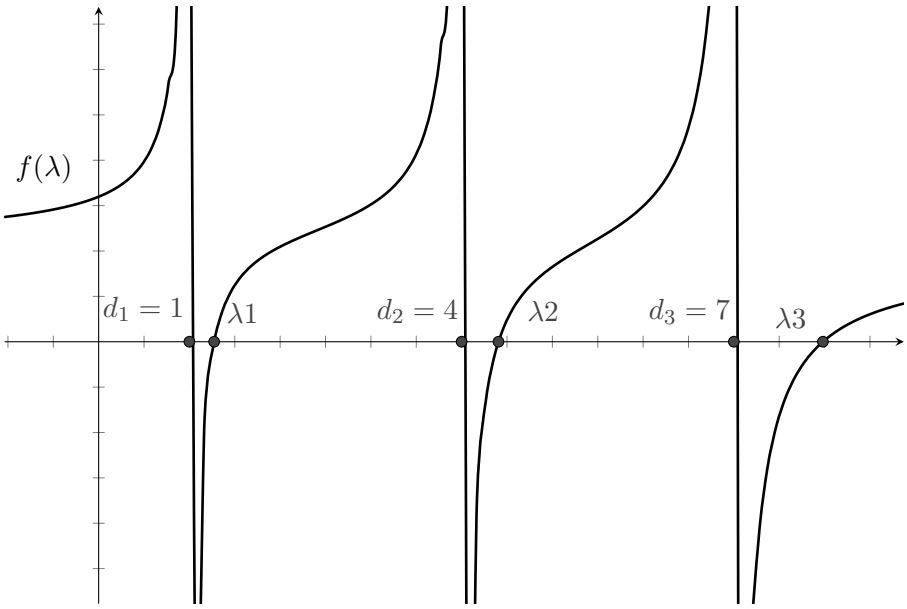
$$1 + \rho \sum_{k=1}^{n} \frac{v_k^2}{d_k - \lambda} =: f(\lambda) = 0.$$

Nultočke ove jednadžbe su svojstvene vrijednosti od $D + \rho \mathbf{v}\mathbf{v}^T$, dok su d_k svojstvene vrijednosti od D.

Primjer 9.2.1. Prikažimo grafički funkciju

$$f(\lambda) = 1 + \frac{0,6^2}{1 - \lambda} + \frac{0,7^2}{4 - \lambda} + \frac{0,9^2}{7 - \lambda}.$$

Nultočke funkcije su svojstvene vrijednosti λ_1, λ_2 i λ_3. One se nalaze na intervalima $\langle 1, 4\rangle$, $\langle 4, 7\rangle$ i $\langle 7, +\infty\rangle$.



Slika 9.2.1: Graf funkcije $f(\lambda) = 1 + \frac{0{,}6^2}{1-\lambda} + \frac{0{,}7^2}{4-\lambda} + \frac{0{,}9^2}{7-\lambda}$

Prikažimo pseudokod rekurzivnog algoritma "podijeli pa vladaj".

Algoritam 9.2.1: "Podijeli pa vladaj" algoritam

Ulaz: $T \in \mathbb{R}^{n \times n}$, simetrična tridijagonalna matrica
Rezultat: $T = Q\Lambda Q^T$, Λ dijagonalna, Q ortogonalna matrica
Funkcija `CDC(`T`)` **:**
  ako $T : 1 \times 1$ **onda**
    vrati $\Lambda = T, Q = 1$;
  kraj
  inače
    formiraj $T_1, T_2 : T = \begin{bmatrix} T_1 & \\ & T_2 \end{bmatrix} + \rho \mathbf{u}\mathbf{u}^T$;
    $\Lambda_1, Q_1 \leftarrow$ `CDC(`T_1`)` ;
    $\Lambda_2, Q_2 \leftarrow$ `CDC(`T_2`)` ;
    formiraj $\widehat{D} = D + \rho \mathbf{v}\mathbf{v}^T$;
    $\Lambda \leftarrow$ svojstvene vrijednosti od $\widehat{D}$;
    $Q' \leftarrow$ svojstveni vektori od $\widehat{D}$;
    $Q \leftarrow \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} \cdot Q'$;
    vrati Λ, Q;
  kraj

Funkciji se daje realna simetrična tridijagonalna matrica. Ako je ona veća od 1×1, tada se rastavlja na sumu dva manja tridijagonalna problema i modifikacije matricom ranga 1. Nad dva manja spomenuta problema rekurzivno se poziva funkcija. Nadalje se formira matrica $\widehat{D}$

za koju se postavlja karakteristična jednadžba. Potrebno je izračunati svojstvene vrijednosti i svojstvene vektore od $\widehat{D}$. U implementaciji se nećemo zamarati samim izračunom jer je jednostavan zbog rekurzivnog pozivanja algoritma. Sidreni uvjet rekurzije je upravo veličina dobivene matrice: ako se radi o 1×1 matrici, vraćene vrijednosti su ta matrica i 1.

```
def CDC(M):
    T = M
    if (len(T) == 1):
        L = T
        Q = 1
        return L, Q
    else:
        n = int(len(T))
        m = n / 2 - 1
        m = int(m)
        b = T[m][m+1]
        T[m][m] -= b
        T[m+1][m+1] -= b

        T_1 = T[0:m+1,0:m+1]
        T_2 = T[m+1:n,m+1:n]

        L_1, Q_1 = CDC(T_1)
        L_2, Q_2 = CDC(T_2)

        v = la.block_diag(np.transpose(Q_1),np.transpose(Q_2))
        D = la.block_diag(L_1,L_2) + b * np.dot(v,np.transpose(v))

        val, vec = la.eigh(D)

        L = np.diag(val)
        Q = np.dot(la.block_diag(Q_1,Q_2),vec)

        return L, Q
```

Prilog 11: Implementacija početnog "podijeli pa vladaj" algoritma

10. Metoda potencija

Ideja računanja svojstvenih vrijednosti metodom potencija je jednostavna. Metoda potječe iz 1929. godine, objavio ju je Richard von Mises. Ovdje se izračunava samo po apsolutnoj vrijednosti najveća svojstvena vrijednost i pripadni svojstveni vektor.

10.1. Jednostavna metoda potencija

Neka je dana matrica $A \in \mathcal{M}_n(\mathbb{C})$ tako da se može zapisati $A = S\Lambda S^{-1}$,

$$\Lambda = \operatorname{diag}(\lambda_1, ..., \lambda_n)$$

gdje su svojstvene vrijednosti

$$|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|.$$

Neka je matrica za potenciju k jednaka

$$A^k = S\Lambda^k S^{-1} = \quad B^k \quad + \quad C^k \quad =$$
$$= S \begin{bmatrix} \lambda_1^k & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} S^{-1} + S \begin{bmatrix} 0 & & & \\ & \lambda_2^k & & \\ & & \ddots & \\ & & & \lambda_n^k \end{bmatrix} S^{-1}.$$

Za dovoljno veliki k, A^k je blizu ranga 1 i može se dobiti informacija o svojstvenom vektoru matrice A.

$$\frac{\|A^k - B^k\|_2}{\|A^k\|_2} \leq \|S\|_2 \|S^{-1}\|_2 \left|\frac{\lambda_2}{\lambda_1}\right|^k \quad \to 0$$

Kako metoda vraća po modulu najveću svojstvenu vrijednost, neće raditi ako je ta vrijednost kompleksna. Brzina konvergencije ovisi o $\frac{|\lambda_2|}{|\lambda_1|}$, sporija je ako je omjer veći [7][13].

Algoritam 10.1.1: Jednostavna metoda potencija

Ulaz: $A \in \mathbb{C}^{n \times n}$

Rezultat: svojstvena vrijednost najveća po modulu

odaberi $\mathbf{x}^{(0)}, \|\mathbf{x}^{(0)}\| = 1$;

$k \leftarrow 0$;

ponavljaj

  $k \leftarrow k + 1$;

  $\mathbf{y}^{(k)} \leftarrow A\mathbf{x}^{(k-1)}$;

  $\mathbf{x}^{(k)} \leftarrow \frac{\mathbf{y}^{(k)}}{\|\mathbf{y}^{(k)}\|}$;

dok ne bude *zadovoljena konvergencija*;

Osnovni algoritam metode potencija dobiva na ulazu matricu za koju će se računati po modulu dominantna svojstvena vrijednost i pripadni svojstveni vektor. Na početku se generira vektor norme 1 pa nakon toga slijedi iteracija kao što je prikazano. U svakoj iteraciji vrši se normiranje. Kada algoritam staje, nakon što imamo vektor potrebno je još samo izračunati i svojstvenu vrijednost. Prikažimo primjer implementacije metode potencija.

```
def powerMethod(A,i):

    k = 0
    x_0=np.random.uniform(size=(len(A),1))
    x_k = x_0 / la.norm(x_0)

    while(k < i):
        k = k + 1
        y_k = np.dot(A,x_k)
        n = la.norm(y_k)
        x_k = y_k / n

    val = np.dot(np.dot(np.transpose(x_k),A),x_k) /
                  np.dot(np.transpose(x_k),x_k)
    vec = x_k

    return val, vec
```

Prilog 12: Implementacija jednostavne metode potencija

11. Primjeri izračuna

Primjer 11.0.1. Odredimo svojstvene vrijednosti generirane simetrične matrice 7×7.

Učinimo to Jacobijevim algoritmom tako da je pivotna strategija najveći vandijagonalni element po svojstvenoj vrijednosti i tridijagonalnim QR algoritmom.

Prikažimo generiranu matricu.

```
B=array([
[-65.394,  16.092,  -0.952,  10.949,  36.001, -69.077,  48.2  ],
[ 16.092, -66.455,  14.244, -37.892, -18.563,   3.589,  75.129],
[ -0.952,  14.244,  96.287,  46.814,  18.084, -36.444,  46.564],
[ 10.949, -37.892,  46.814,  46.142, -23.752,   4.44 ,  66.798],
[ 36.001, -18.563,  18.084, -23.752,  60.706,  48.301, -21.95 ],
[-69.077,   3.589, -36.444,   4.44 ,  48.301,  14.099,  27.093],
[ 48.2  ,  75.129,  46.564,  66.798, -21.95 ,  27.093,  36.772]
])
```

Prilog 13: Generirana simetrična matrica za primjer izračuna

Za QR algoritam matrica je dodatno svedena do Hessenbergove forme. Primijetimo da je matrica simetrična tridijagonalna.

```
B_hess=array([
[-65.394, -93.652,  -0.   ,   0.   ,   0.   ,  -0.   ,  -0.   ],
[-93.652, -19.38 ,  77.169,   0.   ,  -0.   ,  -0.   ,   0.   ],
[  0.   ,  77.169, 132.03 ,  42.574,   0.   ,  -0.   ,   0.   ],
[  0.   ,   0.   ,  42.574,   2.338, -83.62 ,   0.   ,  -0.   ],
[  0.   ,   0.   ,   0.   , -83.62 , -69.162, -36.315,  -0.   ],
[  0.   ,   0.   ,   0.   ,   0.   , -36.315,  53.51 ,  -8.764],
[  0.   ,   0.   ,   0.   ,   0.   ,   0.   ,  -8.764,  88.217]
])
```

Prilog 14: Matrica svedena do Hessenbergove forme za primjer izračuna

Prikažimo svojstvene vrijednosti dobivene funkcijom *scipy.linalg.eigh*.

```
array(
[-148.693,  -130.445,  14.583,  41.432,  73.332,  91.475,  180.474]
)
```

Prilog 15: Dobivene svojstvene vrijednosti sa *scipy.linalg.eigh*

Sada pozovimo implementirane funkcije nad matricom.

```
r1=Jacobi(B,0.0001)
r2=QRsymetricTridiagonal(B_Hess)
pprint.pprint(r1)
pprint.pprint(r2)
```

Prilog 16: Poziv funkcija za primjer računanja

Zaokruženi rezultati poziva slijede.

```
array([
[-1.48693e+02,  0.0,  0.0,   0.0,   0.0,   0.002,   0.0],
[ 0.0,  -1.30445e+02,  0.0,  -0.0,   0.0,   0.01,  -0.0],
[ 0.0,  0.0,   1.80474e+02,   0.0, 0.004, -0.0,  -0.008],
[-0.0, -0.0,  0.0,   1.45830e+01,  0.005,   0.004,  0.0],
[ 0.0,  0.0,  0.004,  0.005, 9.14750e+01,   0.0,    0.0],
[ 0.002,  0.01,  0.0,  0.004, 0.0,  7.33320e+01,  0.007],
[ 0.0,  -0.0,  -0.008,  0.0,  0.0,  0.007,  4.14320e+01]])
```

Prilog 17: Rezultat izvršavanja Jacobijevog algoritma

```
array(
[-148.693, -130.445, 180.474, 14.583, 41.432, 73.332, 91.475]
)
```

Prilog 18: Rezultat izvršavanja simetričnog tridijagonalnog QR algoritma

Čitajući dijagonalne elemente ispisane matrice i ispisane glavne dijagonale za simetrični tridijagonalni QR algoritam, čitamo dobivene svojstvene vrijednosti. Za usporedbu možemo prikazati tablično.

Tablica 11.0.1: Pregled izračuna po algoritmima

scipy.linalg.eigh	JACOBIJEV ALGORITAM	QR ALGORITAM
-148.693	-148.693	-148.693
-130.445	-130.445	-130.445
14.583	14.583	14.583
$41,432$	$41,432$	$41,432$
73.332	73.332	73.332
91.475	91.475	91.475
180.474	180.474	180.474

Uočljivo je kako su na ovom testu oba algoritma dali zadovoljavajuće točne rezultate. U tablici su zelenom bojom označene one koje se podudaraju.

Primjer 11.0.2. Odredimo svojstvene vrijednosti generirane matrice 5×5.

Učinimo to QR algoritmom s dvostrukim pomakom.

Prikažimo generiranu matricu.

```
M=array([
       [2.614, 2.019, 4.765, 2.888, 2.593],
       [2.517, 7.66 , 2.389, 3.244, 0.13 ],
       [3.943, 7.748, 6.272, 9.488, 5.817],
       [6.899, 8.284, 8.616, 2.193, 5.311],
       [9.992, 3.037, 1.741, 3.988, 4.076]
       ])
```

Prilog 19: Generirana matrica za primjer izračuna

Za QR algoritam matrica je dodatno svedena do Hessenbergove forme. Ova matrica nije simetrična tridijagonalna.

Prikažimo svojstvene vrijednosti dobivene funkcijom *scipy.linalg.eig*. Ovdje ne moraju sve svojstvene vrijednosti biti realne.

```
array(
[22.846+0.j, 0.656+3.826j, 0.656-3.826j, 6.137+0.j, 3.173+0.j]
)
```

Prilog 20: Dobivene svojstvene vrijednosti sa *scipy.linalg.eig*

Sada pozivamo implementiranu funkciju QR algoritma s dvostrukim pomakom nad matricom. Slijede zaokruženi rezultati poziva.

```
array([
[22.846, -3.929, -2.638, -2.827,  8.078],
[ 0.   ,  0.226,  3.481, -1.596,  2.678],
[-0.   , -4.259,  1.086, -2.178,  1.617],
[ 0.   ,  0.   ,  0.   ,  6.137,  0.129],
[ 0.   ,  0.   ,  0.   , -0.   ,  3.173]
])
```

Prilog 21: Dobivena matrica QR algoritmom s dvostrukim pomakom

Mogu se čitati svojstvene vrijednosti

$$\{22.846, 6.137, 3.173\}.$$

Vidljivo je da postoje kompleksne svojstvene vrijednosti jer se javlja "izbočina" na poziciji $(3, 2)$ i to konjugirano kompleksni par. Potrebno je naći svojstvene vrijednosti matrice

$$M_1 = \begin{bmatrix} 0.226 & 3.481 \\ -4.259 & 1.086 \end{bmatrix}.$$

Dakle, rješenje je još i

$$\{0.656 \pm 3.826i\}.$$

12. Usporedba algoritama

U ovom poglavlju dat ćemo kratak pregled obrađenih algoritama za računanje svojstvenih vrijednosti. Prikažimo tablicu popisa algoritama s naznakama s kakvim matricama rade i što mogu izračunati.

Tablica 12.0.1: Pregled algoritama

ALGORITAM	ULAZ	IZLAZ
QR algoritam	matrica u Hessenbergovoj formi iz $\mathbb{C}^{n\times n}$ ili $\mathbb{R}^{n\times n}$	sve svojstvene vrijednosti (i svi svojstveni vektori)
simetrični tridijagonalni QR algoritam	simetrična tridijagonalna matrica iz $\mathbb{R}^{n\times n}$	sve svojstvene vrijednosti (i svi svojstveni vektori)
Jacobijev algoritam	simetrična matrica iz $\mathbb{R}^{n\times n}$	sve svojstvene vrijednosti (i svi svojstveni vektori)
algoritam "podijeli pa vladaj"	simetrična tridijagonalna matrica iz $\mathbb{R}^{n\times n}$	sve svojstvene vrijednosti (i svi svojstveni vektori)
metoda potencija	matrica iz $\mathbb{C}^{n\times n}$	najveća svojstvena vrijednost po apsolutnoj vrijednosti (i pripadni svojstveni vektor)

Svakom je algoritmu potrebno dati matricu za kakvu je predviđen. Za osnovni QR algoritam je konvergencija naročito spora ako su svojstvene vrijednosti izrazito blizu jedna drugoj. Dodavanjem pomaka ubrzava se konvergencija. QR algoritam s jednim pomakom ima više utroška nego s dvostrukim pomakom.

Jacobijev algoritam je vrlo pouzdan i odabire se kada je bitna točnost rezultata. S tim algoritmom je moguća ušteda resursa. Može se uvidjeti kvadratična konvergencija za jednostruke svojstvene vrijednosti, ako se radi o višestrukim onda do takve konvergencije dolazi ako su one u bloku (jedna kraj druge).

"Podijeli pa vladaj" algoritam smatra se jako brzim. Tek kasnije je unaprijeđeno računanje svojstvenih vektora s obzirom na početni algoritam. Može se implementirati rekurzivno.

Metoda potencija je prvi numerički algoritam za računanje svojstvenih vrijednosti i to je jednostavnija metoda u odnosu na ostale koje se ovdje razmatraju. Ne zahtijeva posebne oblike matrica. Dolazi do problema ako postoje kompleksne svojstvene vrijednosti ili ako su realne svojstvene vrijednosti suprotnog predznaka (iste apsolutne vrijednosti) ili približno jednake. Tada ne dolazi do konvergencije ili je jako spora.

13. Zaključak

Svojstvene vrijednosti imaju široku primjenu, najviše u matematici i fizici. Njihovo izravno računanje nije lako pa je stoga potrebno imati numeričke algoritme. Nakon uvoda u svojstveni problem izloženi su najpoznatiji numerički algoritmi. Radi se o QR algoritmu, Jacobijevom algoritmu, Cuppenovom "podijeli pa vladaj" algoritmu i metodi potencija. Prikazana je njihova matematička pozadina. Algoritmi su dani u pseudokodu i implementirani su u programskom jeziku *Python*.

Za izračun svojstvenih vrijednosti bira se algoritam ovisno o potrebama. Treba prije razmotriti veličinu samog problema. Nadalje, iz kojeg su skupa elementi matrice, radi li se o simetričnim matricama ili simetričnim tridijagonalnim matricama. Svakako je pitanje što se želi dobiti: sve svojstvene vrijednosti i svojstveni vektori ili samo neki. Za vrlo velike matrice često nije potrebno računati sve. Numerički algoritmi za računanje svojstvenih vrijednosti razlikuju se po konvergenciji, brzini izračuna, točnosti kojom aproksimiraju rezultate. Algoritme je potrebno implementirati (i osmisliti) tako da se štede računalni resursi. Tipični primjeri za to su spremanje samo polovice simetrične matrice, razmatranje matrice dio po dio, pisanje novih izračunatih elemenata na mjesta postojećih.

Postoje drugi numerički algoritmi koji ovdje nisu razmatrani, dok se osnovne ideje ovih razmatranih mogu unaprijediti. Primjerice "podijeli pa vladaj" algoritam nadograđen je tako da se stabilno računaju svojstveni vektori, Jacobijev algoritam može se unaprijediti korištenjem faktorizacije Choleskog tako da je precizniji i brži od QR algoritma. Razmatranje konvergencije kod izračuna nije trivijalno. Cilj je imati što brže algoritme sa što manjim utroškom računalnih resursa i sa što većom točnošću izračuna.

Popis literature

[1] D. Bakić, *Linearna algebra*. Školska knjiga, 2008.

[2] S. Barnett, *Matrices - Methods and Aplications*. Oxford applied mathematics i computing science series, 1990.

[3] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2016.

[4] L. N. Trefethen i D. Bau, *Numerical Linear Algebra*. SIAM, 1997.

[5] P. Arbenz, *Lecture Notes on Solving Large Scale Eigenvalue Problems*. Computer Science Department ETH Zürich, 2016.

[6] Z. Drmač, V. Hari, M. Marušić, M. Rogina, S. Singer i S. Singer, *Numerička analiza*. Sveučilište u Zagrebu, PMF - Matematički odjel, 2003.

[7] Z. Drmač, *Numerička analiza 1*. Sveučilište u Zagrebu, PMF - Matematički odjel, 2008.

[8] G. H. Golub i C. F. Van Load, *Matrix Computations (4ed)*. The Johns Hopkins University Press, 2013.

[9] E. Jarlebring, *QR algorithm, Lecture notes in numerical linear algebra*. KTH, 2016.

[10] D. Bindel, *Matrix Computations (Lectures)*. Cornell CIS, 2012.

[11] J. W. Demmel, *Applied Numerical Linear Algebra*. SIAM, 1997.

[12] V. Hari, *Linearna algebra*. Sveučilište u Zagrebu, PMF - Matematički odjel, 2005.

[13] J. H. Mathews i K. D. Fink, *Numerical Methods Using MATLAB (3ed)*. Prentice Hall, 1999.

Popis slika

Popis tablica

Popis algoritama

Popis priloga