Računanje svojstvenih vrijednosti i svojstvenih vektora realne matrice u programskom jeziku C

Tomić, Bernarda

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

RAČUNANJE SVOJSTVENIH VRIJEDNOSTI I SVOJSTVENIH VEKTORA REALNE MATRICE U PROGRAMSKOM JEZIKU C

Završni rad

Bernarda Tomić

Osijek, 2018.

SADRŽAJ

1. UVOD

Zadatak ovog završnog rada je stvoriti program u programskom jeziku C koji će računati svojstvene vrijednosti i svojstvene vektore realne matrice. Sam naslov rada govori nam sve što je potrebno obraditi unutar njega.

Kao polaznu temu uzet ćemo pojam matrica jer je razumijevanje osnovnih definicija i svojstava matrica nužno za daljnje rješavanje zadatka rada. Nadalje, teorijski ćemo objasniti ključne pojmove - svojstvene vrijednosti i svojstvene vektore. Nakon što objasnimo teorijski dio počet ćemo s njegovom primjenom na zadatcima. Najprije ćemo riješiti primjer računanja svojstvenih vrijednosti i svojstvenih vektora na matrici dimenzija 2x2, a zatim na primjeru matrice 3x3. Neki od važnijih postupaka prilikom rješavanja bit će detaljnije objašnjeni.

Sljedeća tema biti će implementacija zadatka u programskom jeziku, a to je ujedno i temeljni zadatak rada. Unutar ove teme obradit ćemo osnove programskog jezika C nakon čega slijede programi napisani u njemu. Prikazat ćemo implementaciju dvaju programa (za matrice dimenzija 2x2 i 3x3) ispod kojih će biti detaljna objašnjenja gotovo svake linije koda.

Naposljetku rada prikazat ćemo rezultate programa nakon pokretanja, tj. testirati ćemo jesu li programi doista funkcionalni. Na samom kraju završnog rada slijedi zaključak.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je napisati program za računanje svojstvenih vrijednosti i svojstvenih vektora u programskom jeziku C.

2. SVOJSTVENE VRIJEDNOSTI I SVOJSTVENI VEKTORI REALNE MATRICE

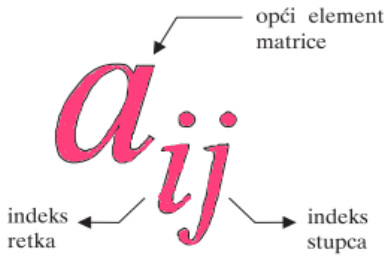
U ovom poglavlju reći ćemo nešto o matricama, svojstvenim vrijednostima i svojstvenim vektorima te ćemo to znanje primijeniti prilikom rješavanja zadataka.

2.1. Općenito o matricama (definicije i primjeri)

Realna matrica je svaka pravokutna shema brojeva sljedećeg oblika:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} ,$$

gdje su $a_{ij} \in \mathbf{R}$, $i = 1 , \dots , m$; $j = 1 , \dots , n$. **A** je opći oblik matrice koja ima m redaka i n stupaca pa kažemo da je tipa (m, n). Brojeve a_{ij} nazivamo elementi matrice gdje indeks i predstavlja broj retka, a indeks j broj stupca u kojem se element nalazi [1]. To je prikazano na slici 2.1:



Slika 2.1. Element matrice [1]

Kraće matricu zapisujemo: $\mathbf{A} = (a_{ij})$.

Matrice označavamo velikim slovima:

$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ -3 & 5 \end{bmatrix}$ $\mathbf{B} = \begin{bmatrix} -2 & 2 \\ 3 & 5 \\ 9 & 1 \end{bmatrix}$ $\mathbf{C} = \begin{bmatrix} 2 & 0 & 1 & 7 \\ 3 & 4 & 6 & 2 \end{bmatrix}$ $\mathbf{D} = \begin{bmatrix} 2 & 5 & -1 \\ 3 & 4 & 5 \\ 0 & 2 & 8 \end{bmatrix}$

Matrica **A** ima 2 retka i 2 stupca (2x2), matrica **D** ima 3 retka i 3 stupca (3x3). Prilikom rješavanja zadatka ovog rada (računanja našeg svojstvenih vrijednosti i svojstvenih vektora realne matrice) koristiti ćemo matrice upravo tih dimenzija. Matrica **B** ima 3 retka i 2 stupca, matrica **C** ima 2 retka i 4 stupca.

Ako je broj redaka matrice jednak broju stupaca, tj. $m = n$, tada je zovemo *kvadratna matrica* reda n . Matricu tipa (1, 1) poistovjećujemo sa realnim brojem. Matrica kojoj su svi elementi jednaki nuli ($a_{ij} = 0, \forall_i, \forall_j$) zove se *nul matrica* i označava se s 0. [1]

Za kvadratne matrice definiramo dijagonalu matrice – dijagonala sadrži elemente s jednakim indeksima: $a_{11}, a_{22}, \dots, a_{nn}$. Ako su matrici svi elementi izvan dijagonale jednaki nuli, tada je zovemo *dijagonalna matrica* . Primjeri dijagonalnih matrica:

$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \begin{bmatrix} 6 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 7 \end{bmatrix}, \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{mn} \end{bmatrix}$. [1]

Dijagonalna matrica kojoj su svi dijagonalni elementi jednaki 1 ($a_{ii} = 1$, $i = 1, \dots, n$) zove se *jedinična matrica* i označava s **I.** Primjeri jediničnih matrica:

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$. [1]

Kvadratna matrica je *gornja trokutasta* ako su svi njeni elementi *ispod* dijagonale jednaki nuli. Primjeri gornje trokutaste matrice:

$\begin{bmatrix} 2 & 3 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 5 & 2 & 4 \\ 0 & 1 & -2 \\ 0 & 0 & 7 \end{bmatrix}, \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix}$. [1]

Kvadratna matrica je *donja trokutasta* ako su svi njeni elementi *iznad* dijagonale jednaki nuli. Primjeri donje trokutaste matrice:

$$\begin{bmatrix} 2 & 0 \\ 1 & 4 \end{bmatrix}, \quad \begin{bmatrix} 5 & 0 & 0 \\ 2 & -1 & 0 \\ 0 & 3 & 7 \end{bmatrix}, \quad \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}. \qquad [1]$$

2.2. Teorijska podloga

Opisat ćemo kako se traži najprikladnija baza vektorskog prostora *X*, odnosno baza u kojoj će linearni operator $\mathcal{A}: X \to X$ imati najjednostavniji prikaz. *X* će nam predstavljati n-dimenzionalni vektorski prostor i svi će operatori biti definirani i uzimati vrijednosti u istome prostoru *X* . Za razumijevanje sljedećih objašnjenja bitno je definirati i pojam kolinearnosti - *kolinearni vektori* su oni koji pripadaju istom ili paralelnim pravcima.

Razmotrimo sljedeći primjer. Uzmimo da je $\mathcal{A}$ operator simetrije s obzirom na pravac *p*. Najprikladnija baza za opis ovoga operatora je ona koju sačinjava vektor smjera pravca *p* i vektor okomit na njega. Vektore označimo s *e* i *f* . Pri tome vrijedi: $\mathcal{A}(e) = e$, $\mathcal{A}(f) = -f$: slike vektora *e* i *f* su paralelne na same vektore. Oni (i njima kolinearni vektori) su i jedini vektori s tim svojstvima. Ako uzmemo bili koji drugi vektor koji nije kolinearan s jednim od ova dva, $x = \alpha e + \beta f$, tad je $\mathcal{A}(x) = = \alpha e - \beta f$ i taj vektor nije kolinearan s x.

Vektor $v \neq 0$ zovemo **svojstvenim vektorom** operatora $\mathcal{A}$ ako postoji skalar λ takav da vrijedi $\mathcal{A}\mathrm{v} = \lambda\mathrm{v}$.

Skalar λ nazivamo **svojstvena vrijednost** operatora $\mathcal{A}$, koja odgovara svojstvenom vektoru *v* . Dakle, po definiciji vidimo da je i αe ($\alpha \in \mathbf{R}$, $\alpha \neq 0$) svojstveni vektor, čim je *v* svojstveni vektor, a svi ti vektori odgovaraju istoj svojstvenoj vrijednosti.

Neka su *x, y* dva svojstvena vektora (ako postoje, ali ne nužno kolinarana) koji odgovaraju istoj svojstvenoj vrijednosti λ, tada vrijedi:
$\mathcal{A}(\alpha x + \beta y) = \alpha\mathcal{A}(x) + \beta\mathcal{A}(y) = \alpha\lambda x + \beta\lambda y = \lambda(\alpha x + \beta y)$
te je $\alpha x + \beta y$ svojstveni vektor za istu svojstvenu vrijednosti (ako je različit od 0).

Poopćimo li to razmatranje, možemo za svaku svojstvenu vrijednost λ promotriti potprostor Ker (λ**I** – **A**), jezgru operatora λ**I** – **A**. Svaki vektor, koji je različit od nule, iz toga potprostora je svojstveni vektor operatora $\mathcal{A}$. Naime, (λ**I** – **A**) (*v*) = 0 povlači **A**(*v*) = λ*v* , te se taj potprostor naziva svojstveni potprostor koji pripada svojstvenoj vrijednosti λ.

Primjer: Neka je operator $\mathcal{A}$ zadan sljedećom matricom: **A**= $\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}$, nađimo njegove svojstvene vrijednosti i vektore!

Pomoću jednadžbe $\mathcal{A}\mathrm{v} = \lambda\mathrm{v}$ stvorimo sljedeći sustav:

$$x_1 + x_2 = \lambda x_1$$

$$x_2 = \lambda x_2$$

pa iz druge jednadžbe iščitamo da je $\lambda = 1$ ili $x_2 = 0$. Ako je $\lambda = 1$, iz prve jednadžbe slijedi da je $x_1 + x_2 = x_1$ te je $x_2 = 0$, x_2 je bilo kakav. Ako je $x_2 = 0$ onda iz prve jednadžbe vidimo da je $\lambda = 1$ i opet x_1 bilo kakav (ali različit od 0). Zato postoji jedna svojstvena vrijednost $\lambda = 1$ i jednodimenzionalni svojstveni potprostor $\alpha \begin{bmatrix}1\\0\end{bmatrix}$ koji odgovaraju toj svojstvenoj vrijednosti.

Umjesto da govorimo o jednodimenzionalnom potprostoru, izabrat ćemo jedan njegov vektor i govoriti i svojstvenom vektoru koji pripada toj svojstvenoj vrijednosti.

Nakon što smo pronašli svojstvenu vrijednost, slijedi nalaženje svojstvenih vektora. Nalaženje svojstvenih vektora svodi se na rješavanje homogenog linearnog sustava. Jednadžba $\mathcal{A}\mathrm{v} = \lambda\mathrm{v}$ je ekvivalentna s (λ**I** – **A**) (*v*) = 0. Stoga, *v* je svojstveni vektor ako i samo ako pripada jezgri operatora λ**I** – **A** . Taj uvjet govori nam o načinu na koji se mora birati skalar λ.

Da bi jednadžba (λ**I** – **A**) (*v*) = 0 imala netrivijalno rješenje, operator λ**I** – **A** ne smije biti regularan. Neka je **A** matrica operatora $\mathcal{A}$ u nekoj bazi. Matrica jediničnog operatora (u svakoj bazi) je jedinična matrica **I** . Zato operatoru λ**I** – **A** odgovara matrica λ**I – A,** a s obzirom da ta matrica nije ragularna, njezina determinanta mora biti jednaka 0:

$$\left|\lambda\mathbf{I} - \mathbf{A}\right| = \begin{vmatrix} \lambda - a_{11} & -a_{12} & \dots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \dots & -a_{2n} \\ \vdots & & & \\ -a_{n1} & -a_{n2} & \dots & \lambda - a_{nn} \end{vmatrix} = 0.$$

Ova determinanta je polinom po nepoznanici λ, stupnja *n*. Nazivamo ga karakteristični polinom operatora $\mathcal{A}$ (ili matrice **A**) i označavamo ga s $k_A(\lambda)$.

$$k_A(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A}) .$$

Vodeći koeficijent ovoga polinoma, uz potenciju λ^n je 1 pa zato on ima oblik:

$$k_A(\lambda) = \lambda^n - \sigma_1 \lambda^{n-1} - \ldots - \sigma_{n-1} \lambda - \sigma_n .$$

Jednadžba $k_A(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A}) = 0$ se naziva karakteristična jednadžba operatora $\mathcal{A}$ (matrice **A**). Njena rješenja su svojstvene vrijednosti operatora $\mathcal{A}$.

Karakteristični polinom ne ovisi o izboru baze, on se računa preko determinante matrice koja odgovara operatoru $\lambda \mathbf{I} - \mathbf{A}$. Ta matrica ovisi o izabranoj bazi, ali njezina determinanta ne! Svake takve dvije matrice su slične i zato imaju istu determinantu.

S obzirom da su svojstene vrijednosti nul-točke karakterističnog polinoma niti one ne ovise o izboru baze. Zbog toga pri računanju svojstvenh vrijednosti možemo uzeti bilo koju bazu za prikaz operatora $\mathcal{A}$.

2.2.1. Računanje svojstvenih vrijednosti

Svojstvene vrijednosti su nul-točke polinoma stupnja *n*. Kako bismo ih odredili, moramo prvo odrediti taj polinom. S obzirom da je on determinanta matrice reda *n*, suočeni smo s dva problema:

1. Kako odrediti determinantu matrice reda *n*, čiji elementi nisu numerički, nego se u njoj pojavljuje i nepoznanica λ?
2. Nakon što izračunamo taj polinom (na neki način!), kako odrediti njegove nul-točke?

Na prvo pitanje se ne može dati zadovoljavajući odgovor. Postoji nekoliko načina za određivanje koeficijenta karakterističnog polinoma koji ne koriste direktno računanje determinanti, ali svi su oni efikasni samo za matrice malog reda.

Što se tiče nalaženja svojstvenih vrijednosti, nul-točke polinoma velikoga stupnja se mogu računati samo približnim metodama. Razlog za to je što eksplicitne formule za nalaženje nul-točaka polinoma stupnja većeg od četiri ne postoje. Za polinome stupnja tri i četiri, formule postoje ali su praktički beskorisne.

Sve ovo nam govori da se svojstvene vrijednosti (i vektori) matrica velikoga reda nalaze potpuno drugačijim metodama. Tim se problemom bavi posebno područje matematike koje se zove numerička linearna algebra.

2.2.2. Značenje kompleksnih brojeva

Polje realnih brojeva nije dostatno u problemu nalaženja svojstvenih vrijednosti. Razlog za to je što polinom (čak i onaj s realnim koeficijentima) ne mora imati niti jedan realni korijen. Ako je to karakteristični polinom onda odgovarajući operator nema (realnih) svojstvenih vrijednosti. S druge strane, prema osnovnom stavku algebre svaki polinom stupnja n ima točno n kompleksnih nul-točaka (brojeći njihovu višestrukost). Zato je korisno pri nalaženju svojstvenih vrijednosti dozvoliti račun u polju kompleksnih brojeva. Na taj način će svaki operator imati barem jednu svojstvenu vrijednost i barem jedan svojstveni vektor (koji ne mora imati geometrijsku interpretaciju). [2]

2.3. Primjer zadatka za 2x2 matricu

Prije rješavanja zadataka ponovimo najprije teorijski dio ukratko!

Neka je X vektorski prostor. Za skalar $\lambda \in \mathbf{R}$ kažemo da je *svojstvena vrijednost* linearnog operatora $\mathcal{A}: X \to X$ ako postoji vektor $v \in X$ koji je različit od 0 (nulvektora) tako da vrijedi $\mathcal{A}v = \lambda v$. Tada taj vektor nazivamo *svojstveni vektor*. Karakteristični polinom kvadratne matrice **A** ($k_A(\lambda)$) je determinanta matrice ($\lambda I - \mathbf{A}$). $k_A(\lambda) = 0$

Zadatak:

Odredite svojstvene vrijednosti i svojstvene vektore linearnog operatora $\boldsymbol{\mathcal{A}}: \boldsymbol{v^2} \to \boldsymbol{v^2}$ ako je $\boldsymbol{\mathcal{A}\vec{x}} = (2x_1 - 5x_2)\,\boldsymbol{\vec{e_1}} + (x_1 - 4x_2)\,\boldsymbol{\vec{e_2}}$.

Rješenje:

Najprije iz zadane jednadžbe odredimo matricu iz koje ćemo izračunati sve ostalo. Odredimo ju tako što brojeve uz x_1 pišemo u prvi stupac, a brojeve uz x_2 pišemo u drugi stupac.

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

Nakon što smo odredili osnovnu kvadratnu matricu možemo početi s računanjem njenog karakterističnog polinoma.

$k_A(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A})$

$k_A(\lambda) = 0$

$k_A(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A}) = \det\left(\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} - \begin{bmatrix} 2 & -5 \\ 1 & -4 \end{bmatrix} \right) = \begin{vmatrix} \lambda - 2 & 5 \\ -1 & \lambda + 4 \end{vmatrix}$

$(\lambda - 2)(\lambda + 4) - (-1 \cdot 5) = \lambda^2 + 4\lambda - 2\lambda - 8 + 5 = 0$

$\lambda^2 + 2\lambda - 3 = 0 \qquad (a=1, b=2, c=-3)$

$\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-2 \pm \sqrt{4 - 12}}{2}$, iz toga slijedi da su **svojstvene vrijednosti**:

$\lambda_1 = -3$

$\lambda_2 = 1$

Stvaranje sustava jednadžbi:

$\mathcal{A}\vec{v} = \lambda\vec{v}$, $\vec{v} = x_1\vec{e_1} + x_2\vec{e_2}$

$\begin{bmatrix} 2 & -5 \\ 1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$\begin{bmatrix} 2x_1 & -5x_2 \\ x_1 & -4x_2 \end{bmatrix} = \begin{bmatrix} \lambda x_1 \\ \lambda x_2 \end{bmatrix}$

_____

* $2x_1 - 5x_2 = \lambda x_1$

* $x_1 - 4x_2 = \lambda x_2$

Za svaki λ pripadni vektor dobivamo kao rješenje sustava jednadžbi. Imamo dva rješenja za λ što znači da ćemo imati dva slučaja.

1. SLUČAJ:

Za $\lambda_1 = -3$, uvrstimo λ u gore navedene jednadžbe (*) te rješavamo sustave.

$2x_1 - 5x_2 = -3x_1$

$x_1 - 4x_2 = -3x_2$

_____

$2x_1 - 5x_2 + 3x_1 = 0$

$x_1 - 4x_2 + 3x_2 = 0$

_____

$5x_1 - 5x_2 = 0 \quad / :5$ homogen sustav jednadžbi

$x_1 - x_2 = 0$ (obje imaju jednaka rješenja)

_____

$x_1 = x_2$

$x_1 = t \in \mathbf{R}$

$(t, t) = t\,(1, 1)$

$\begin{bmatrix} t \\ t \end{bmatrix} = t \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Iz toga slijedi da je **svojstveni vektor**:

$\vec{v_1} = \vec{e_1} + \vec{e_2}$

$\mathcal{A}\vec{v_1} = -3\vec{v_1}$

2. SLUČAJ

Za $\lambda_2 = 1$, rješavamo analogno kao prvi slučaj.

$2x_1 - 5x_2 = x_1$

$x_1 - 4x_2 = x_2$

$2x_1 - 5x_2 - x_1 = 0$

$x_1 - 4x_2 - x_2 = 0$

$x_1 - 5x_2 = 0$

$x_1 - 5x_2 = 0$

$x_1 = 5x_2$

$x_2 = t \in \mathbf{R}$

$(5t, t) = t\,(5, 1)$

$\begin{bmatrix} 5t \\ t \end{bmatrix} = t \begin{bmatrix} 5 \\ 1 \end{bmatrix}$

Svojstveni vektor:

$\overrightarrow{v_2} = 5\overrightarrow{e_1} + \overrightarrow{e_2}$

$\mathcal{A}\overrightarrow{v_2} = \overrightarrow{v_2}$

$\mathbf{A}(e) = \begin{bmatrix} 2 & -5 \\ 1 & -4 \end{bmatrix}$

$(v) = (\overrightarrow{v_1}, \overrightarrow{v_2})$

$\mathbf{A}(v) = \begin{bmatrix} -3 & 0 \\ 0 & 1 \end{bmatrix}$ Matrica je dijagonalna jer su na dijagonali svojstvene vrijednosti.

2.4. Primjer zadatka za 3x3 matricu

Zadatak:

Odredite svojstvene vrijednosti i svojstvene vektore linearnog operatora $\mathcal{A}: \boldsymbol{v^3} \rightarrow \boldsymbol{v^3}$ kojemu je u bazi *(e)* = $(\overrightarrow{e_1}, \overrightarrow{e_2}, \overrightarrow{e_3})$ pridružena matrica:

$$\mathbf{A}(e) = \begin{bmatrix} 4 & -2 & 1 \\ -2 & 1 & 2 \\ 1 & 2 & 4 \end{bmatrix}$$

Rješenje:

Najprije računamo karakteristični polinom zadane matrice.

$k_A(\lambda) = \det(\lambda\mathbf{I} - \mathbf{A})$

$k_A(\lambda) = 0$

$$k_A(\lambda) = \det(\lambda\mathbf{I} - \mathbf{A}) = \det\left(\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \begin{bmatrix} 4 & -2 & 1 \\ -2 & 1 & 2 \\ 1 & 2 & 4 \end{bmatrix}\right) = \begin{vmatrix} \lambda - 4 & 2 & -1 \\ 2 & \lambda - 1 & -2 \\ -1 & -2 & \lambda - 4 \end{vmatrix}$$

2.4.1. Računanje determinante n-tog reda

Računanje determinante matrice n-tog reda (u ovom slučaju dimenzija 3x3) je nešto složenije nego što je to bilo kod matrice 2x2. Iz toga razloga postoje različite metode kojima se determinanta brže izračuna, a najopćenitija metoda je Laplaceov razvoj determinante. Laplaceov razvoj determinante se može provoditi po bilo kojem retku ili stupcu matrice. Neka je matrica **A** reda n. Ako u toj matrici izostavimo i-ti redak i j-ti stupac dobit ćemo matricu čiju determinantu zovemo subdeterminanta ili minora i označavamo je sa Mij. Algebarski komplement ili kofaktor elementa a_{ij} je broj

$\mathbf{A}_{ij} = (-1)^{i+j} M_{ij}$. [3]

Primjer, razvoj po prvom retku (a_{11}, a_{12}, a_{13}) :

$$\det(\mathbf{A}) = |\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \cdot (-1)^{1+1} \cdot \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} + a_{12} \cdot (-1)^{1+2} \cdot \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$$

$$+ a_{13} \cdot (-1)^{1+3} \cdot \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Prateći navedno objašnjenje možemo nastaviti s računanjem determinante zadane u našem zadatku.

$$\begin{vmatrix} \lambda-4 & 2 & -1 \\ 2 & \lambda-1 & -2 \\ -1 & -2 & \lambda-4 \end{vmatrix} = (\lambda-4)\cdot(-1)^{1+1}\cdot\begin{vmatrix} \lambda-1 & -2 \\ -2 & \lambda-4 \end{vmatrix} + 2\cdot(-1)^{1+2}\cdot\begin{vmatrix} 2 & -2 \\ -1 & \lambda-4 \end{vmatrix}$$

$$+ (-1)\cdot(-1)^{1+3}\cdot\begin{vmatrix} 2 & \lambda-1 \\ -1 & -2 \end{vmatrix} = 0$$

$$(\lambda-4)[(\lambda^2-4\lambda-\lambda+4)-4\,]-2\,(2\lambda-8-2)-1(-4+\lambda-1)=0$$

$$(\lambda-4)(\,\lambda^2-5\lambda)-2(2\lambda-10)-1(\lambda-5)=0$$

$$(\lambda-5)\,[(\lambda\,(\lambda-4)-4-1]=0$$

$$(\lambda-5)(\,\lambda^2-4\lambda-5)=0$$

1. $\lambda-5=0 \rightarrow \lambda_3=5$
2. $\lambda^2-4\lambda-5=0 \rightarrow \lambda_2=5\,,\ \lambda_1=-1$

Dakle, svojstvene vrijednosti su:

$\boldsymbol{\lambda_1=-1}$

$\boldsymbol{\lambda_2=5}$

$\boldsymbol{\lambda_3=5}$

Stvaranje sustava jednadžbi:

$\mathcal{A}\vec{v}=\lambda\vec{v}\,,\ \ \vec{v}=\boldsymbol{x_1}\overrightarrow{\boldsymbol{e_1}}+\boldsymbol{x_2}\overrightarrow{\boldsymbol{e_2}}+\boldsymbol{x_3}\overrightarrow{\boldsymbol{e_3}}$

$$\begin{bmatrix} 4 & -2 & 1 \\ -2 & 1 & 2 \\ 1 & 2 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\lambda\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\begin{bmatrix} 4x_1 & -2x_2 & x_3 \\ -2x_1 & x_2 & 2x_3 \\ x_1 & 2x_2 & 4x_3 \end{bmatrix}=\begin{bmatrix} \lambda x_1 \\ \lambda x_2 \\ \lambda x_3 \end{bmatrix}$$

$$4x_1-2x_2+\ x_3=\ \lambda x_1$$

$$-2x_1+x_2+\ 2x_3=\ \lambda x_2$$

$$x_1+2x_2+\ 4x_3=\ \lambda x_3$$

Za svaki λ pripadni vektor dobivamo kao rješenje sustava jednadžbi. Imamo tri rješenja za λ što znači da bi trebali imati tri slučaja, međutim $\lambda_2=\ \lambda_3=5$ pa će nam to biti samo jedan slučaj. Dakle, bez obzira na tri rješenja za λ, ipak postoje samo dva slučaja.

1. SLUČAJ

Za λ_1= -1 :

$4x_1 - 2x_2 + x_3 = -x_1$

$-2x_1 + x_2 + 2x_3 = -x_2$

$x_1 + 2x_2 + 4x_3 = -x_3$

$5x_1 - 2x_2 + x_3 = 0 \rightarrow x_3 = -5x_1 + 2x_2$ (to uvrstimo u drugu jednadžbu)

$-2x_1 + 2x_2 + 2x_3 = 0$

$x_1 + 2x_2 + 5x_3 = 0$

$-2x_1 + 2x_2 + 2(-5x_1 + 2x_2) = 0$

$-2x_1 + 2x_2 - 10x_1 + 4x_2 = 0$

$-12x_1 + 6x_2 = 0$

$-12x_1 = -6x_2$ / : (-6)

$\boldsymbol{x_2 = 2x_1}$ (to uvrstimo u treću jednadžbu)

$x_1 + 2 \cdot 2x_1 + 5x_3 = 0$

$x_1 + 4x_1 + 5x_3 = 0$

$5x_1 + 5x_3 = 0$

$5x_1 = -5x_3$ / : (-5)

$\boldsymbol{x_3 = -x_1}$

Izrazili smo x_2 i x_3 preko x_1 .

$x_1 = t \in \mathbf{R}$

(t, 2t, -t) = t (1, 2, -1)

$\begin{bmatrix} t \\ 2t \\ -t \end{bmatrix} = t \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$

Svojstveni vektor: $\boldsymbol{\overrightarrow{v_1} = \overrightarrow{e_1} + 2\overrightarrow{e_2} - \overrightarrow{e_3}}$

$\mathcal{A}\overrightarrow{v_1} = -\overrightarrow{v_1}$

2. SLUČAJ

Za $\lambda_2 = \lambda_3 = 5$

$4x_1 - 2x_2 + x_3 = 5x_1$

$-2x_1 + x_2 + 2x_3 = 5x_2$

$x_1 + 2x_2 + 4x_3 = 5x_3$

$-x_1 - 2x_2 + x_3 = 0 \quad \rightarrow x_3 = x_1 + 2x_2$

$-2x_1 - 4x_2 + 2x_3 = 0 \,/ : 2 \quad \rightarrow x_3 = x_1 + 2x_2$

$x_1 + 2x_2 - x_3 = 0 \quad \rightarrow x_3 = x_1 + 2x_2$

Izlučivanjem varijable x_3 iz svake jednadžbe dobili smo isto rješenje stoga možemo zaključiti da su sve tri jednadžbe linearno zavisne ("jednake").

$x_1 = t \in \mathbf{R}$

$x_2 = s \in \mathbf{R}$

(t, s, t + 2s)

$$\begin{bmatrix} t \\ s \\ t+2s \end{bmatrix} = \begin{bmatrix} t \\ 0 \\ t \end{bmatrix} + \begin{bmatrix} 0 \\ s \\ 2s \end{bmatrix} = t\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + s\begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

Svojstveni vektor: $\overrightarrow{v_2} = \overrightarrow{e_1} + \overrightarrow{e_3}$

$\mathcal{A}\overrightarrow{v_2} = 5\overrightarrow{v_2}$

Svojstveni vektor: $\overrightarrow{v_3} = \overrightarrow{e_2} + 2\overrightarrow{e_3}$

$\mathcal{A}\overrightarrow{v_3} = 5\overrightarrow{v_3}$

$$\mathbf{A}(e) = \begin{bmatrix} 4 & -2 & 1 \\ -2 & 1 & 2 \\ 1 & 2 & 4 \end{bmatrix}$$

$(v) = (\overrightarrow{v_1}, \overrightarrow{v_2}, \overrightarrow{v_3})$

$$\mathbf{A}(v) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

3. IMPLEMENTACIJA U PROGRAMSKOM JEZIKU

U ovome poglavlju implementirat ćemo 2 programa za računanje svojstvenih vrijednosti i svojstvenih vektora. Prvi program će biti za matrice veličine 2x2, a drugi za matrice 3x3. Oba programa ćemo implementirati u programskom jeziku C++ te pored pojedinih linija koda navoditi komentare koji pobliže objašnjavaju postupak rješavanja.

Gotovo svaka linija predstavlja određenu formulu te je svaka od njih korak prema konačnom izračunu traženog rezultata – svojstvenih vrijednosti i svojstvenih vektora. Ostale linije koda, koje nisu formule, su neke funkcije, naredbe, zaglavlja itd. koje su nužne za pravilno pisanje i izvođenje programa. O njima ćemo reći nešto više u sljedećem podnaslovu (programski jezik C).

3.1. C programski jezik

Najjednostavniji oblik C programa koji na ekran ispisuje „Hello, world“ (Slika 3.1.) često se koristi kako bi se prikazala osnovna sintaksa programa. Na temelju takvog programa moguće je naučiti osnovni način funkcioniranja i pisanja C programa stoga ćemo takav jednostavan program objasniti u nastavku. Svaku liniju ovoga osnovnog programa možemo pronaći kasnije i u našim programima za računanje svojstvenih vrijednosti i svojstvenih vektora samo u složenijem kontekstu.

main.c

```
1  #include <stdio.h>
2  main(){
3  printf("Hello, World\n");
4  return 0;
5  }
6
```

Slika 3.1. „Hello, world“ program u C programu

Prva linija programa, *#include <stdio.h>* govori računalu da u program uključi informacije o standardnoj ulazno/izlaznoj datoteci. Datoteke s ekstenzijom .h nazivaju se datoteke zaglavlja i njihovo stavljanje u oštre zagrade < > govori računalo da se radi o standardnim datotekama

zaglavlja, koje se nalaze na unaprijed određenim mjestima u datotečnom sustavu. Dakle, datoteka stdio.h sadrži informacije koje su nužne za korištenje funkcije printf.

Dalje slijedi funkcija *main* koja mora biti prisutna u svakom programu. U ovom slučaju ona je definirana kao funkcija koja ne očekuje nikakve argumente, što je predstavljeno praznom listom ( ). Općenito, zadaća funkcije main je pozivanje drugih funkcija.

Naredbe funkcije nalaze se unutar velikih zagrada {}. U našem slučaju funkcija main ima samo jednu naredbu *printf("Hello, World\n");* . Funkcija se poziva tako da se navede njezino ime iza kojega idu oble zagrade ( ) s listom argumenata koji se predaju funkciji. Funkcija *printf* dobiva samo jedan argument, a to je niz znakova *"Hello, World\n"* . Funkcija printf je iz biblioteke koja ispisuje podatke na ekran, u našem slučaju to je niz znakova između navodnika.

Posljednja naredba u tijelu funkcije main je *return 0,* kojom će pozivatelju biti vraćena 0 kao poruka da su sve naredbe funkcije main izvršene ispravno.

Nadalje, kada smo opisali osnovnu sintaksu C programa možemo objasniti i ostale osnovne pojmove koji su bitni za razumijevanje programa, a nisu prikazani u prethodnom primjeru. Za pojedine pojmove navest ćemo i primjere gdje i kako su korišteni u našem kodu.

TIPOVI PODATAKA:

U C-u postoji nekoliko osnovnih tipova podataka:

char: predstavlja jedan byte, jedan znak iz skupa znakova

*int:*to je cjelobrojna vrijednost i odnosi se na prirodnu veličinu cijelih brojeva na računalu

float: predstavlja realni broj s jednostrukom točnošću

double: predstavlja realni broj s dvostrukom točnošću [4]

Od navedenih tipova podataka u našim programima koristit ćemo samo float.

Primjer: `float trace = A[0][0] + A[1][1];`

NAREDBE I PETLJE:

Naredbe *if, if-else i swich* samo jednom izvršavaju uvjet i naredbe. Često se događa da naredbe moramo izvršiti više od jednog puta pa u takvim slučajevima kada istu naredbu ponavljamo više puta koristimo petlje. Petlje predstavljaju naredbu ili niz naredbi koje će se izvršavati više puta za redom sve dok se ne ispuni neki unaprijed zadan uvjet. U C programu koriste se tri petlje: *for, while i do-while*.

Primjer za for petlju:
```
for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 3; j++) {
```

Primjer za if-else naredbu:
```
if(r <= -1) {
  phi = PI/3;
} else if(r >= 1) {
  phi = 0;
 } else {
phi = acos(r) / 3
```

Naredba *if - else* koristi se za donošenje određenih odluka. Formalna sintaksa jest:

if(izraz)

naredba1

else

naredba2

Izraz se provjerava. Ako je istinit (ima vrijednost koja nije nula), izvršava se naredba1. Ako nije istinit (ima vrijednost nula) i ako postoji else dio, izvršava se naredba2. [4]

DATOTEKE:

Uključivanje datoteke olakšava manipulaciju funkcijama, deklaracijama i #define skupovima. Svaka izvorna linija oblika #include "ime datoteke" ili #include < ime datoteke > se mijenja sadržajem imena datoteke. Naredba #include najbolja je za međusobno povezivanje deklaracija velikog programa. Ona jamči da će sve izvorne datoteke vidjeti definicije i deklaracije varijabli, čime će eliminirati neke poteškoće. Kad se promatrana datoteka izmjeni, sve datoteke koje je spominju u izvornom kodu, moraju se ponovo prevoditi.

Primjer:
```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
```

MATEMATIČKE FUNKCIJE:

Više je od dvadeset matematičkih funkcija deklarirano u zaglavlju, a ovdje su navedene neke češće korištene. Svaka treba jedan ili dva argumenta.

sin(x) sinus od x, x je izražen u radijanima

cos(x) kosinus od x, x je izražen u radijanima

atan2(y,x) arkus tangens od y/x, u radijanima

exp(x) eksponencijalna funkcija

pow(x,y) xy

sqrt(x) drugi korijen od x (x>0) [4]

Od navedenih matematičkih funkcija u našim programima koristit ćemo pow i sqrt.

Primjer za *pow*: `float Delta = pow( trace , 2) - 4*det ;`

Power diže bazu na n-ti eksponent, n>=0 .

Primjer za *sqrt*: `float p = sqrt(p3/6);`

Funkcija sqrt je deklarirana u <math.h>

3.2. Implementacija za 2x2 matricu

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbf{R}^{2x2}$$

main.cpp

```
1  #include <stdio.h>
2  #include <math.h>
3
4  void eigenvaluesvectors2(float A[2][2]) {
5      float trace = A[0][0] + A[1][1];
6      float det = A[0][0] * A[1][1] - A[0][1] * A[1][0];
7
8      float Delta = pow( trace , 2) - 4*det ;
9
10     if ( Delta < 0) {
11        printf ( " Svojstvene vrijednosti su kompleksni brojevi! \n" ) ;
12        return ;
13       }
14
15     float lambda1 = (trace + sqrt(trace*trace - 4 * det))/2;
16     float lambda2 = (trace - sqrt(trace*trace - 4 * det))/2;
17
18     float v1[2] = {A[0][0] - lambda2, A[1][0]};
19     float v2[2] = {A[0][0] - lambda1, A[1][0]};
20
21     printf("lambda1 = %f, lambda2 = %f \n", lambda1, lambda2);
22     printf("v1 = [%f, %f], v2 = [%f, %f]",v1[0],v1[1],v2[0],v2[1]);
23 }
24
25 int main() {
26     float A[2][2] = {{1.0,2.0},{3.0,4.0}};
27
28     eigenvaluesvectors2(A);
29
30     return 0; }
```

Slika 3.2. Kod za računanje svojstvenih vrijednosti i svojstvenih vektora matrice dimenzija 2x2 u C programu

LINIJA 5: `float trace = A[0][0] + A[1][1];`

trag matrice **A**, zbroj elemenata na glavnoj dijagonali kvadratne matrice :

$$\mathrm{tr}(\mathbf{A}) = a + d$$

LINIJA 6: `float det = A[0][0] * A[1][1] - A[0][1] * A[1][0];`

determinanta matrice, $\det(\mathbf{A}) = ad - bc$

LINIJA 8: `float Delta = pow( trace , 2) - 4*det ;`

diskriminanta karakterističnog polinoma; naime, svojstvene vrijednosti računaju se kao nultočke karakterističnog polinoma $k_A(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A})$, a to je

$$k_A(\lambda) = \begin{vmatrix} \lambda - a & b \\ c & \lambda - d \end{vmatrix} = (\lambda - a)(\lambda - d) - bc = \lambda^2 - \lambda(a + d) + ad - bc = \lambda^2 - tr(A)\,\lambda + \det(A)$$

Diskriminanta polinoma $P(x) = ax^2 + bx + c$ računa se kao $\Delta = b^2 - 4ac$, pa je za k_A diskriminanta jednaka $\Delta = tr^2(\mathbf{A}) - 4\det(\mathbf{A})$.

```
if ( Delta < 0) {
      printf ( " Svojstvene vrijednosti su kompleksni brojevi! \n" ) ;
```

Ako je $\Delta < 0$, onda su svojstvene vrijednosti kompleksni brojevi.

LINIJA 15:
```
float lambda1 = (trace + sqrt(trace*trace - 4 * det))/2;
float lambda2 = (trace - sqrt(trace*trace - 4 * det))/2;
```

Nultočke polinoma $k_A(\lambda) = \lambda^2 - tr(\mathbf{A})\,\lambda + \det(\mathbf{A})$:

$$\lambda_{1,2} = \frac{\mathrm{tr(A)} \pm \sqrt{tr^2(\mathrm{A}) - 4\det(A)}}{2}$$

LINIJA 18:
```
float v1[2] = {A[0][0] - lambda2, A[1][0]};
float v2[2] = {A[0][0] - lambda1, A[1][0]};
```

vrijedi $(\mathbf{A} - \lambda_1\mathbf{I})(\mathbf{A} - \lambda_2\mathbf{I}) = 0$, pa za svojstveni vektor koji pripada svojstvenoj vrijednosti λ_1 možemo uzeti prvi stupac (ili drugi, svejedno je) matrice $(\mathbf{A} - \lambda_2\mathbf{I})$, tj.

$$v_1 = \begin{bmatrix} a - \lambda_2 \\ c \end{bmatrix}$$

jer tada imamo $(\mathbf{A} - \lambda_1\mathbf{I})v_1 = 0$, tj. $Av_1 = \lambda_1 v_1$ što je definicija svojstvenog vektora. Slično, jer vrijedi $(\mathbf{A} - \lambda_2\mathbf{I})(\mathbf{A} - \lambda_1\mathbf{I}) = 0$, za svojstveni vektor koji pripada svojstvenoj vrijednosti λ_2 možemo uzeti prvi stupac (ili drugi, svejedno je) matrice $(\mathbf{A} - \lambda_1\mathbf{I})$, tj.

$$v_2 = \begin{bmatrix} a - \lambda_1 \\ c \end{bmatrix}$$

LINIJA 21: `printf("lambda1 = %f, lambda2 = %f \n", lambda1, lambda2);`

ispis svojstvenih vrijednosti na ekran (λ_1, λ_2)

LINIJA 22: `printf("v1 = [%f,%f], v2 =[%f,%f]",v1[0],v1[1],v2[0],v2[1]);`

ispis svojstvenih vektora na ekran (v_1, v_2)

LINIJA 26: `float A[2][2] = {{1.0,2.0},{3.0,4.0}};`

definiramo matricu koja ima 2 retka i 2 stupca, zadajemo vrijednosti njenih elemenata

LINIJA 28: `eigenvaluesvectors2(A);`

pozivamo funkciju za računanje svojstvenih vrijednosti i svojstvenih vektora;
kao argument joj predajemo prethodno definiranu matricu **A**

3.3. Implementacija za 3x3 matricu

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \in \mathbf{R}^{3x3}$$

main.cpp

```
1  #include <stdio.h>
2  #include <math.h>
3  #define PI 3.14159265358979323846
4
5  void eigenvaluesvectors2(float A[2][2]) {
6      float trace = A[0][0] + A[1][1];
7      float det = A[0][0] * A[1][1] - A[0][1] * A[1][0];
8
9      float Delta = pow(trace, 2) - 4*det;
10
11     if(Delta < 0) {
12         printf("Svojstvene vrijednosti su kompleksni brojevi!\n");
13         return;
14     }
15
16     float lambda1 = (trace + sqrt(trace*trace - 4 * det))/2;
17     float lambda2 = (trace - sqrt(trace*trace - 4 * det))/2;
18
19     float v1[2] = {A[0][0] - lambda2, A[1][0]};
20     float v2[2] = {A[0][0] - lambda1, A[1][0]};
21
22     printf("lambda1 = %f, lambda2 = %f \n", lambda1, lambda2);
23     printf("v1 = [%f, %f], v2 = [%f, %f]\n",v1[0],v1[1],v2[0],v2[1]);
24 }
25
26 void eigenvaluesvectors3(float A[3][3]) {
27     float lambda1, lambda2, lambda3;
28     float a = 1;
29     float b = - (A[0][0] + A[1][1] + A[2][2]);
30
```

Slika 3.3. Kod za računanje svojstvenih vrijednosti i svojstvenih vektora matrice dimenzija 3x3 u C programu (linije **1-28**)

main.cpp

```
29      float b = - (A[0][0] + A[1][1] + A[2][2]);
30      float c = A[0][0]*A[1][1]+A[0][0]*A[2][2]+A[1][1]*A[2][2]-A[0][1]*A[1][0]-A[0][2]*A[2][0]-A[1][2]*A[2][1];
31      float d = -(A[0][0]*(A[1][1]*A[2][2]-A[1][2]*A[2][1])-A[0][1]*(A[1][0]*A[2][2]-A[1][2]*A[2][0])+
32                  A[0][2]*(A[1][0]*A[2][1]-A[1][1]*A[2][0]));
33
34      float Delta = pow(b*c,2)-4*a*pow(c,3)-4*pow(b,3)*d-27*pow(a*d,2)+18*a*b*c*d;
35
36      if(Delta < 0) {
37          printf("Svojstvene vrijednosti su kompleksni brojevi!\n");
38          return;
39      }
40
41      float p1 = pow(A[0][1], 2) + pow(A[0][2], 2) + pow(A[1][2], 2);
42
43      if(p1 == 0) { // matrica je donje trokutasta, pa su svojstvene vrijednosti na dijagonali
44          lambda1 = A[0][0];
45          lambda2 = A[1][1];
46          lambda3 = A[2][2];
47      } else {
48          float q = (A[0][0] + A[1][1] + A[2][2])/3; // tr(A) / 3
49          float p2 = A[0][1]*A[1][0] + A[0][2]*A[2][0] + A[1][2]*A[2][1];
50          float p3 = pow(A[0][0] - q, 2) + pow(A[1][1] - q, 2) + pow(A[2][2] - q, 2) + 2*p2;
51          float p = sqrt(p3/6); // sqrt(tr((A-qI)^2)/6)
52
53          float B[3][3]; // B = 1/p (A-qI)
54
55          for(int i = 0; i < 3; i++) {
56              for(int j = 0; j < 3; j++) {
57                  if(i == j) {
58
```

Slika 3.4. Kod za računanje svojstvenih vrijednosti i svojstvenih vektora matrice dimenzija 3x3 u C programu (linije **29-56**)

LINIJA 5-24: Unutar ovog programa za računanje svojstvenih vrijednosti i svojstvenih vektora matrica 3x3 implementirali smo i kod za matrice 2x2 (prethodni program) da bi mogli rezultate oba programa prikazati zajedno.

LINIJA 34: `float Delta = pow(b*c,2)-4*a*pow(c,3)-4*pow(b,3)*d-27*pow(a*d,2)+18*a*b*c*d;`

diskriminanta karakterističnog polinoma $k_A(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A})$, a to je

$$k_A(\lambda) = \begin{vmatrix} \lambda - a & b & c \\ d & \lambda - c & f \\ g & h & \lambda - i \end{vmatrix} = \ldots = \lambda^3 - \mathrm{tr}(\mathbf{A})\,\lambda^2 - \lambda\,\frac{tr(A^2) - tr^2(A)}{2} - \det(\mathbf{A})$$

Diskriminanta polinoma $P(x) = ax^3 + bx^2 + cx + d$ računa se kao $\Delta = b^2c^2 - 4ac^3 - 4b^3d - 27a^2d^2 + 18abcd$.

LINIJA 36:
```
if(Delta < 0) {
    printf("Svojstvene vrijednosti su kompleksni brojevi!\n");
```

Ako je $\Delta < 0$, onda su svojstvene vrijednosti kompleksni brojevi.

LINIJA 28:
```
float a = 1;
    float b = - (A[0][0] + A[1][1] + A[2][2]);
    float c = A[0][0]*A[1][1]+A[0][0]*A[2][2]+A[1][1]*A[2][2]-
A[0][1]*A[1][0]-A[0][2]*A[2][0]-A[1][2]*A[2][1];
    float d = -(A[0][0]*(A[1][1]*A[2][2]-A[1][2]*A[2][1])-
A[0][1]*(A[1][0]*A[2][2]-
A[1][2]*A[2][0])+A[0][2]*(A[1][0]*A[2][1]-A[1][1]*A[2][0]));
```

Kao što smo već izračunali u prethodnom koraku, rješavanjem $k_A(\lambda) = \det(\lambda I - \mathbf{A})$ dobijemo:

$$k_A(\lambda) = \begin{vmatrix} \lambda - a & b & c \\ d & \lambda - c & f \\ g & h & \lambda - i \end{vmatrix} = \ldots = \lambda^3 - \mathrm{tr}(\mathbf{A})\,\lambda^2 - \lambda\frac{tr(A^2) - tr^2(A)}{2} - \det(\mathbf{A})$$

iz toga vidimo je $a = 1$, $b = -\mathrm{tr}(\mathbf{A})$, $c = -\frac{tr(A^2) - tr^2(A)}{2}$ i $d = -\det(\mathbf{A})$.

$b = -\mathrm{tr}(\mathbf{A}) = -(a + e + i)$.

Odredimo c. U matrici $\mathbf{A}^2$ zanimaju nas samo elementi na dijagonali (jer je trag samo zbroj elemenata na dijagonali), pa ostale nećemo računati:

$$\mathbf{A}^2 = \mathbf{A}\cdot\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\cdot\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} a^2 + bd + cg & . & . \\ . & bd + e^2 + fh & . \\ . & . & cg + fh + i^2 \end{bmatrix}.$$

Stoga je $\mathrm{tr}(\mathbf{A}^2) = a^2 + e^2 + i^2 + 2(bd + cg + fh)$.

Nadalje, $\mathrm{tr}^2(\mathbf{A}) = (a + e + i)^2$ = raspišemo pomoću formule za kvadrat trinoma=

$$a^2 + e^2 + i^2 + 2(ae + ai + ei)\,.$$

Kada uvrstimo dobiveni $\mathrm{tr}(\mathbf{A}^2)$ i $\mathrm{tr}^2(\mathbf{A})$ u gore navedenu formulu za c dobijemo konačni

$c = ae + ai + ei - bd - cg - fh$.

Odredimo d. Razvojem po prvom retku:

$$\det(\mathbf{A}) = |\mathbf{A}| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\cdot(-1)^{1+1}\cdot\begin{vmatrix} e & f \\ h & i \end{vmatrix} + b\cdot(-1)^{1+2}\cdot\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\cdot(-1)^{1+3}\cdot\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$= a(ei - fh) - b(di - fg) + c(dh - ge)$

Dakle $d = -(a(ei - fh) - b(di - fg) + c(dh - eg))$

LINIJA 41: `float p1 = pow(A[0][1], 2) + pow(A[0][2], 2) + pow(A[1][2], 2);`

```
    if(p1 == 0) {
        lambda1 = A[0][0];
        lambda2 = A[1][1];
        lambda3 = A[2][2];
    } else {
```

Provjeravamo da li je matrica donja trokutasta, jer ako je onda ima svojstvene vrijednosti na glavnoj dijagonali. Donja torkutasta matrica dimenzija 3x3 ima elemente *b, c* i *f* jednake nuli.
Ako je zbroj tih elemenata iznad glavne dijagonale jednak nuli ($p_1 = b^2 + c^2 + f^2 = 0$), tada će svojstvene vrijednosti biti elementi *a, e* i *i* , odnosno:

$\lambda_1 = a$

$\lambda_2 = e$

$\lambda_3 = i$

Ako uvjet nije zadovoljen nastavljamo računati svojstvene vrijednosti daljnjim postupkom.

LINIJA 48:

Uvodimo nove varijable q, p_2, p_3 (za koji je nužno prethodno izračunati p_2) i p (za koji je nužno prehodno izračunati p_3 i q).
Za tražene varijable koristili smo gotove formule koje su navedene u nastavku.

$$q = tr(\mathbf{A}) / 3 \text{ , odnosno } q = (a + e + i) / 3$$

$$p_2 = bd + cg + fh$$

$$p_3 = (a - q)^2 + (e - q)^2 + (i - q)^2 + 2p_2$$

$$p = \sqrt{\left(\frac{p_3}{6}\right)} = \sqrt{tr\,\frac{(A-qI)^2}{6}}$$

LINIJA 53:

Varijable p i q, koje smo prethodno izračunali, ćemo koristiti da bi stvorili matricu **B**. Stvoriti ćemo ju jer će nam ona korisiti kako bi lakše izračunali svojstvene vrijednosti.

Matrica **B** se u programu predstavlja kao dvodimezionalno polje (MxN elemenata, u našem slučaju 3x3) te zato koristimo dvije for petlje. U prvoj petlji sa indeksom *i* broje se elementi u

retku. Imamo 3 retka, stoga indeks *i* broji od 0 do 3 (tj. 0,1,2). U drugoj petlji sa indeksom *j* broje se elementi u stupcu. Imamo 3 stupca, stoga indeks *j* broji također od 0 do 3. Pomoću naredbe if provjeravamo uvjet (je li broj redaka i broj stupaca jednak), te ako je uvjet ispunjen matricu **B** računamo prema formuli:

$$\mathbf{B} = \frac{1}{p}(\mathbf{A} - q\mathbf{I}) .$$

Ako uvjet nije ispunjen (tj. ako je broj redaka I stupaca različit) matricu **B** računamo kao:

$$\mathbf{B} = \frac{A}{p} .$$

main.cpp

```
57                  if(i == j) {
58                      B[i][i] = (A[i][i] - q)/p;
59                  } else {
60                      B[i][j] = A[i][j]/p;
61                  }
62              }
63          }
64
65          // det B
66          float detB = B[0][0]*(B[1][1]*B[2][2]-B[1][2]*B[2][1])-B[0][1]*(B[1][0]*B[2][2]-B[1][2]*B[2][0])+
67                       B[0][2]*(B[1][0]*B[2][1]-B[1][1]*B[2][0]);
68          float r = detB / 2;
69          float phi;
70
71          if(r <= -1) {
72              phi = PI/3;
73          } else if(r >= 1) {
74              phi = 0;
75          } else {
76              phi = acos(r) / 3;
77          }
78
79          lambda1 = q + 2 * p * cos(phi);
80          lambda3 = q + 2 * p * cos(phi + (2*PI/3));
81          lambda2 = 3 * q - lambda1 - lambda3;
82      }
83
84      float v11 = (A[0][0]-lambda2)*(A[0][0]-lambda3)+A[0][1]*A[1][0]+A[0][2]*A[2][0];
85      float v12 = A[1][0]*(A[0][0]-lambda3)+(A[1][1]-lambda2)*A[1][0]+A[1][2]*A[2][0];
86
```

Slika 3.5. Kod za računanje svojstvenih vrijednosti i svojstvenih vektora matrice dimenzija 3x3 u C programu (linije **57-84**)

Linija 41-82 [5]

LINIJA 66: `det B`

```
        float detB = B[0][0]*(B[1][1]*B[2][2]-B[1][2]*B[2][1])-
B[0][1]*(B[1][0]*B[2][2]-
B[1][2]*B[2][0])+B[0][2]*(B[1][0]*B[2][1]-B[1][1]*B[2][0]);
```

Računamo determinantu novonastale matrice **B**.

$$\det(\mathbf{B}) = |\mathbf{B}| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \cdot(-1)^{1+1} \cdot \begin{vmatrix} e & f \\ h & i \end{vmatrix} + b \cdot(-1)^{1+2} \cdot \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \cdot(-1)^{1+3} \cdot \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

= a (ei - fh) – b (di – fg) + c (dh – ge)

Dakle, det(**B**) = a (ei - fh) – b (di – fg) + c (dh – eg)

LINIJA 68:

Uvodimo nove varijable (r i phi) koje ćemo koristiti u formulama za računanje svojstvenih vrijednosti naposljetku. Ovdje vidimo zašto smo morali stvoriti matricu **B** (jer nam je potrebna za računanje vaijable r).

$r = \frac{\det(B)}{2}$

Pomoću naredbe *if/ else* postavljamo uvjete:

1. Ako je r manje ili jednako -1 tada će phi biti: phi = $\frac{PI}{3}$

   Vrijednost broja PI definirana je odmah na početku koda **linijom 3**:
   `#define PI 3.14159265358979323846`

2. Ako je r veće ili jednako 1 tada će phi biti: phi = 0
3. Za ostale slučajeve (za r = 0) phi će biti: phi = $\frac{\cos(r)}{3}$

   acos = arccos = $\cos^{-1}$

LINIJA 79:

```
lambda1 = q + 2 * p * cos(phi);
lambda3 = q + 2 * p * cos(phi + (2*PI/3));
lambda2 = 3 * q - lambda1 - lambda3;
```

U konačnici, možemo izračunati svojstvene vrijednosti čije računanje i je jedan od ciljeva ovog programa. Korištene formule:

$\lambda_1 = q + 2p \cdot \cos(phi)$

$\lambda_3 = q + 2p \cdot \cos(phi + (2\frac{PI}{3}))$

$\lambda_2 = 3q - \lambda_1 - \lambda_3$

LINIJA 84:
```
float v11 = (A[0][0]-lambda2)*(A[0][0]-
lambda3)+A[0][1]*A[1][0]+A[0][2]*A[2][0];
    float v12 = A[1][0]*(A[0][0]-lambda3)+(A[1][1]-
lambda2)*A[1][0]+A[1][2]*A[2][0];
    float v13 = (A[0][0]-
lambda3)*A[2][0]+A[2][1]*A[1][0]+A[2][0]*(A[2][2]-lambda2);

    float v21 = (A[0][0]-lambda1)*(A[0][0]-
lambda3)+A[0][1]*A[1][0]+A[0][2]*A[2][0];
    float v22 = A[1][0]*(A[0][0]-lambda3)+(A[1][1]-
lambda1)*A[1][0]+A[1][2]*A[2][0];
    float v23 = (A[0][0]-
lambda3)*A[2][0]+A[2][1]*A[1][0]+A[2][0]*(A[2][2]-lambda1);

    float v31 = (A[0][0]-lambda1)*(A[0][0]-
lambda2)+A[0][1]*A[1][0]+A[0][2]*A[2][0];
    float v32 = A[1][0]*(A[0][0]-lambda2)+(A[1][1]-
lambda1)*A[1][0]+A[1][2]*A[2][0];
    float v33 = (A[0][0]-
lambda2)*A[2][0]+A[2][1]*A[1][0]+A[2][0]*(A[2][2]-lambda1);
```

Vrijedi $(\mathbf{A} - \lambda_1\mathbf{I})(\mathbf{A} - \lambda_2\mathbf{I})(\mathbf{A} - \lambda_3\mathbf{I}) = 0$, pa svojstveni vektor koji pripada svojstvenoj vrijednosti λ_1 možemo uzeti kao prvi stupac matrice $(\mathbf{A} - \lambda_2\mathbf{I})(\mathbf{A} - \lambda_3\mathbf{I})$:

$$(\mathbf{A} - \lambda_2\mathbf{I})(\mathbf{A} - \lambda_3\mathbf{I}) = \left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} - \begin{bmatrix} \lambda_2 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix}\right) \cdot \left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} - \begin{bmatrix} \lambda_3 & 0 & 0 \\ 0 & \lambda_3 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}\right)$$

$$= \begin{bmatrix} a-\lambda_2 & b & c \\ d & e-\lambda_2 & f \\ g & h & i-\lambda_2 \end{bmatrix} \cdot \begin{bmatrix} a-\lambda_3 & b & c \\ d & e-\lambda_3 & f \\ g & h & i-\lambda_3 \end{bmatrix}$$

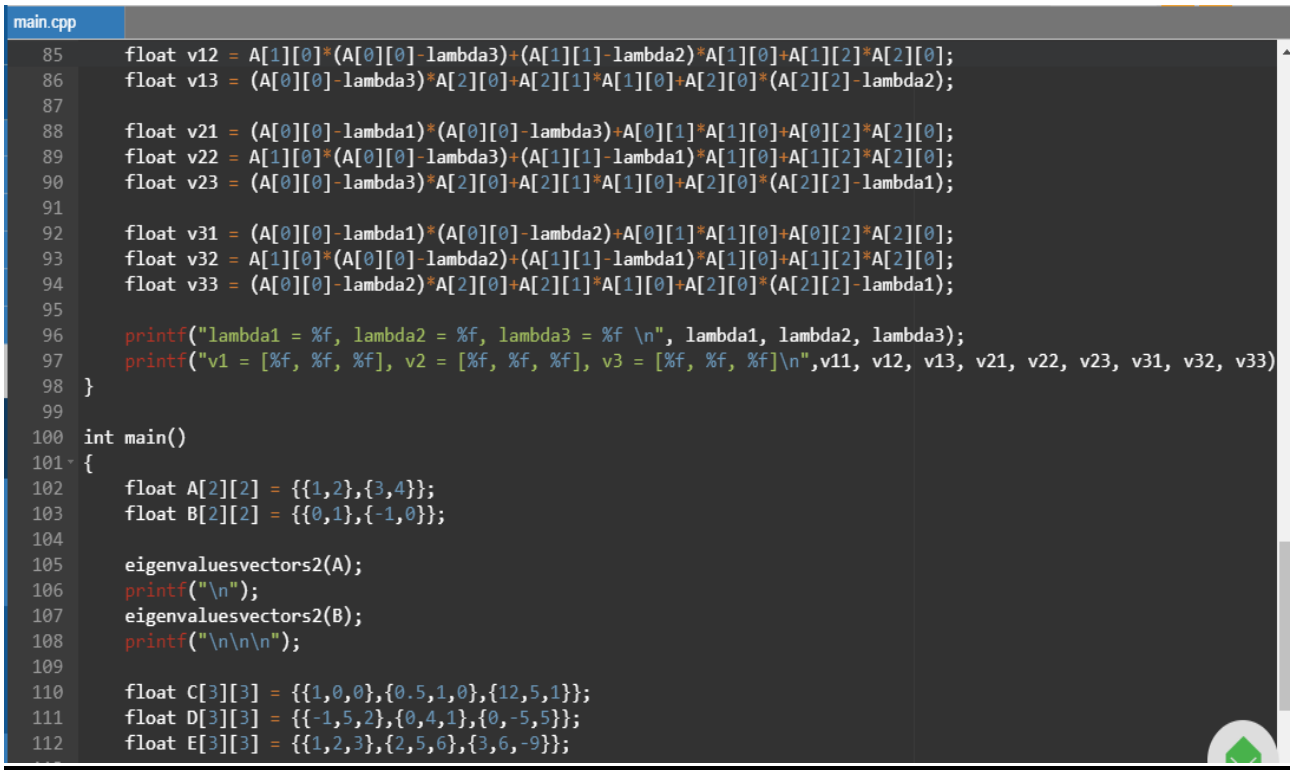
$$= \begin{bmatrix} (a-\lambda_2)(a-\lambda_3)+bd+cg & . & . \\ d(a-\lambda_3)+(e-\lambda_2)d+fg & . & . \\ g(a-\lambda_3)+hd+(i-\lambda_2)g & . & . \end{bmatrix}$$

Dakle: $v_{11} = g(a-\lambda_3)+hd+(i-\lambda_2)g$

$v_{12} = d(a-\lambda_3)+(e-\lambda_2)d+fg$

$v_{13} = g(a-\lambda_3)+hd+(i-\lambda_2)g$

Analogno za λ_2 (stupac matrice $(\mathbf{A} - \lambda_1\mathbf{I})(\mathbf{A} - \lambda_3\mathbf{I})$) i λ_3 (stupac matrice $(\mathbf{A} - \lambda_1\mathbf{I})(\mathbf{A} - \lambda_2\mathbf{I})$) .



Slika 3.6. Kod za računanje svojstvenih vrijednosti i svojstvenih vektora matrice dimenzija 3x3 u C programu (linije **85-111**)

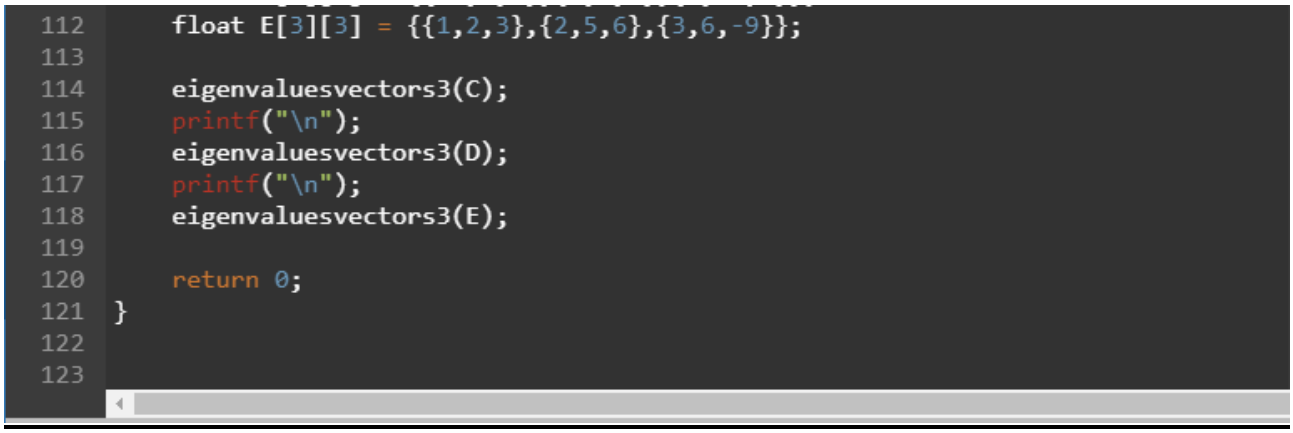
LINIJA 96:
```
printf("lambda1 = %f, lambda2 = %f, lambda3 = %f \n", lambda1, lambda2, lambda3);
    printf("v1 = [%f, %f, %f], v2 = [%f, %f, %f], v3 = [%f, %f, %f]\n",v11, v12, v13, v21, v22, v23, v31, v32, v33);
```

Izlazna funkcija printf pomoću koje ispisujemo svojstvene vrijednosti i svojstvene vektore na ekran.

LINIJA 102:
```
float A[2][2] = {{1,2},{3,4}};
float B[2][2] = {{0,1},{-1,0}};

eigenvaluesvectors2(A);
printf("\n");
eigenvaluesvectors2(B);
```

Definirali smo dvije matrice (**A** i **B**) veličine 2x2 s različitim vrijednostima te pozvali funkciju za računanje njihovih svojstvenih vrijednosti i svojstvenih vektora. To smo već napravili u prethodnom programu, ali ćemo sada taj program ponovo pokrenuti unutar ovoga za matrice 3x3 da bi ih mogli lakše usporediti.



```
112     float E[3][3] = {{1,2,3},{2,5,6},{3,6,-9}};
113
114     eigenvaluesvectors3(C);
115     printf("\n");
116     eigenvaluesvectors3(D);
117     printf("\n");
118     eigenvaluesvectors3(E);
119
120     return 0;
121 }
122
123
```

Slika 3.6. Kod za računanje svojstvenih vrijednosti i svojstvenih vektora matrice dimenzija 3x3 u C programu (linije **112-121**)

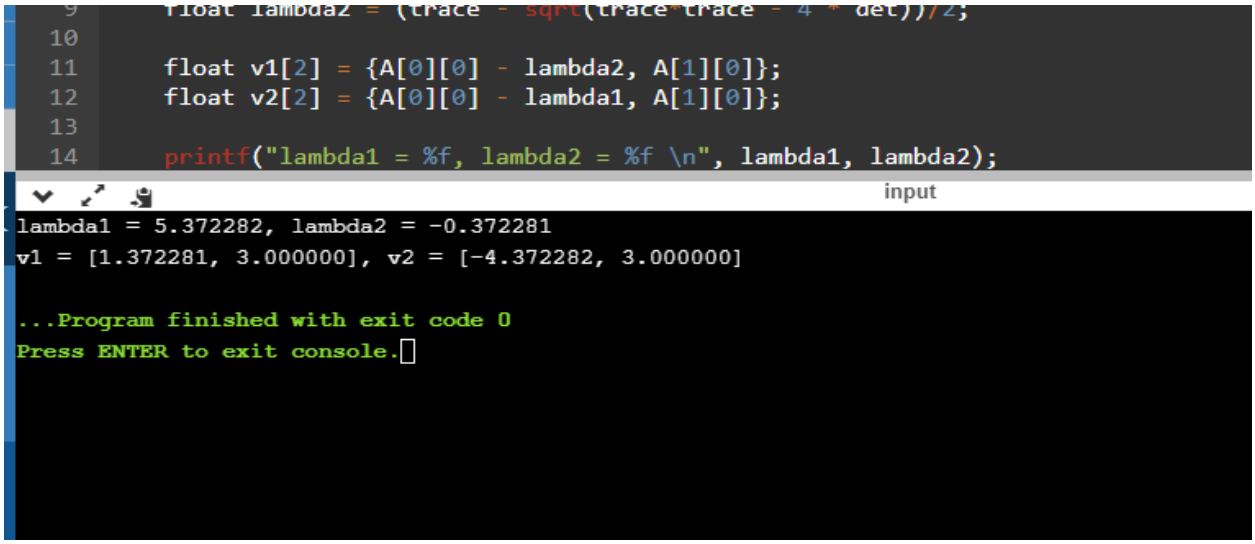
LINIJA 110:
```
    float C[3][3] = {{1,0,0},{0.5,1,0},{12,5,1}};
    float D[3][3] = {{-1,5,2},{0,4,1},{0,-5,5}};
    float E[3][3] = {{1,2,3},{2,5,6},{3,6,-9}};

    eigenvaluesvectors3(C);
    printf("\n");
    eigenvaluesvectors3(D);
    printf("\n");
    eigenvaluesvectors3(E);
```

Definirali smo tri matrice (**C**, **D** i **E**) veličine 3x3 s elementima različitih vrijednosti te pozvali funkciju za računanje njihovih svojstvenih vrijednosti i svojstvenih vektora. Funkcija kao argument prima matricu.

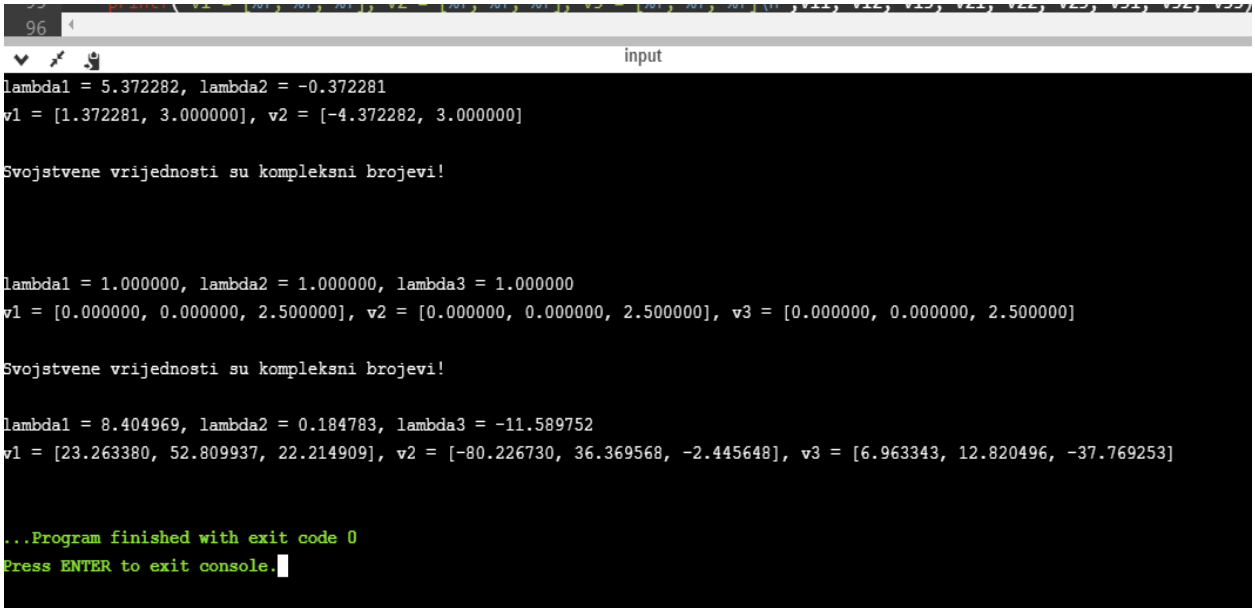
4. TESTIRANJE FUNKCIONALNOSTI PROGRAMA

U ovom poglavlju prikazat ćemo i ukratko analizirati rezultate dobivene nakon pokretanja gore navedenih programa u programskom jeziku C. Testirat ćemo jesu li programi doista funkcionalni, odnosno da li izvršavaju svoju namjenu.



Slika 4.1. Testiranje programa za matricu 2x2

Program je izračunao svojstvene vrijednosti lambda1 i lambda2 te pripadajuće svojstvene vektore v1 i v2 za zadanu matricu.



Slika 4.2. Testiranje programa za matrice 3x3

Kao što smo ranije spomenuli, unutar ovog programa implementiran je i program za matrice 2x2. Program je najprije izračunao svojstvene vrijednosti lambda1 i lambda2 te pripadajuće svojstvene vektore v1 i v2 za zadanu matricu **A**. Nakon toga slijedi računanje za zadanu matricu **B**, ali njene vrijednosti su takve da daju svojstvene vrijednosti koje su kompleksni brojevi.

Nadalje, za zadane matrice (**C** i **E**) dimenzija 3x3 program je izračunao svojstvene vrijednosti lambda1, lambda2 i lambda3 te pripadajuće svojstvene vektore v1, v2 i v3. Matrica **D** ima takve vrijednosti da nakon računanja daje svojstveene vrijednosti koje su kompleksni brojevi.

5. ZAKLJUČAK

U ovom završnom radu detaljno je objašnjen postupak računanja svojstvenih vrijednosti i svojstvenih vektora u programskom jeziku C. Najprije smo objasnili osnovne pojmove koji su nužni za rješavanje ovog problema. Svakim naslovom smo postupno uvodili nove definicije i primjere, upoznavali se s problemom i načinom na koji ćemo ga rješavati, te smo ga u konačnici uspješno riješili. Računanje svojstvenih vrijednosti i svojstvenih vektora je zadatak koji počinje od matrice, stoga smo najprije objasnili upravo pojam matrica, nakon toga mogli smo krenuti na pravu problematiku zadatka pa smo objasnili što u teorijskom smislu znače svojstvene vrijednosti i svojstveni vektori te smo teorijsko znanje prikazali na primjerima zadataka. Na temelju riješenih zadataka u konačnici smo stečeno znanje preveli u programski kod koji smo detaljno objasnili i testirali njegovu funkcionalnost.

Izrađeni su programi za rad s matricama dimenzija 2x2 i 3x3. Vidimo očiglednu razliku u složenosti između ta dva programa. Prvenstveno možemo navesti drastičnu razliku u duljini koda, za matrice 2x2 kod je puno kraći. Sljedeća razlika je u složenosti. Za matricu 2x2 potrebno je riješiti samo 2 sustava jednadžbi s dvije nepoznanice, kod matrica 3x3 to rješavanje je složenije jer se susrećemo s tri jednadžbe s tri nepoznanice. Dakle, proporcionalno s povećanjem dimenzija matrice povećava se broj jednadžbi i broj nepoznanica što uvelike otežava rješavanje zadatka i povećava duljinu programskog koda.

Ako matrica nije simetrična onda može imati kompleksne svojstvene vrijednosti, a ako su svojstvene vrijednosti kompleksne njih nismo dalje koristili (nismo računali svojstvene vektore za njih) samo smo ispisali „Svojstvene vrijednosti su kompleksni brojevi“.

LITERATURA

[1] https://element.hr/artikli/file/1416 (14.8.2018.)

[2] N. Elezović: Linearna algebra
https://www.scribd.com/document/77622336/N-elezovic-Linearna-Algebra (14.8.2018.)

[3] file:///C:/Users/Hp/Downloads/03kece.pdf (14.8.2018.)

[4] Dennis M. Ritchie Brian W. Kernighan: Programski jezik C, Drugo izdanje
file:///C:/Users/Hp/Downloads/programski_jezik_c%20(1).pdf (14.8.2018.)

[5] https://en.wikipedia.org/wiki/Eigenvalue_algorithm#3%C3%973_matrices (14.8.2018.)

SAŽETAK

Zadatak ovog rada bio je opisati i na konkretnim primjerima implementirati računanje svojstvenih vrijednosti i svojstvenih vektora realne matrice u programskom jezik C. Kreiranje tih programa zahtijevalo je osnovno znanje o matricama, rješavanje zadataka sa svojstvenim vrijednostima i svojstvenim vektorima i poznavanje rada u C programu. Rad sadrži dva programa, prvi za računanje s matricama dimenzija 2x2 i drugi za računanje s matricama 3x3.

Ključne riječi: C implementacija, matrice, svojstvene vrijednosti, svojstveni vektori

ABSTRACT

The task of this paper was to describe and in concrete examples implement calculation of eigenvalues and eigenvectors of the real matrix in C programming language. Creating these programs required basic knowledge about matrix, solving tasks with eigenvalues and eigenvectors and knowledge of work in C program. The paper contains two programs, first for computing with matrix dimension 2x2 and the other for computing with 3x3 matrix.

Keywords: C implementation, matrix, eigenvalues, eigenvectors

ŽIVOTOPIS

Bernarda Tomić rođena je 8. svibnja 1995. godine u Požegi. Završila je Osnovnu školu fra Kaje Adžića u Pleternici te je nakon završene osnovne škole upisala Klasičnu gimnaziju u Požegi. Maturirala je 2014. godine, nakon čega je upisala Preddiplomski sveučilišni studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.