

UNIVERZITET CRNE GORE,

PRIRODNO-MATEMATIČKI FAKULTET

**PARALELIZACIJA ALGORITMA STEPENE ITERACIJE ZA NALAŽENJE DOMINANTNE SVOJSTVENE VRIJEDNOSTI MATRICE**

PROJEKTNI IZVJEŠTAJ IZ PREDMETA PARALELNO PROGRAMIRANJE

|  |  |
| --- | --- |
| Profesor:  Dr Igor Jovančević | Studenti:  Đorđe Čabarkapa 5/24 |
| Asistent:  Mr Nikola Pižurica | Miletić Rajan 7/24 |

Podgorica, 2025. godine

**SADRŽAJ**

[1. UVOD 1](#_Toc199142998)

[2. OPIS IMPLEMENTACIJE 2](#_Toc199142999)

[3. REZULTATI 10](#_Toc199143000)

[3.1 Tabela rezultata 10](#_Toc199143001)

[3.2 Grafički prikaz rezultata 10](#_Toc199143002)

[4. ZAKLJUČAK 13](#_Toc199143003)

# UVOD

Power Iteration metoda, opisana u seminarskom radu, predstavlja ključni numerički postupak za brzo i skalabilno nalaženje dominantne (najveće po apsolutnoj vrijednosti) svojstvene vrijednosti i pripadajućeg vektora kvadratne matrice. U praksi, ova tehnika se koristi u oblastima poput PageRank algoritma, analize stabilnosti dinamičkih sistema i PCA za redukciju dimenzionalnosti. Iako je teoretski jednostavna, njena sekvencijalna implementacija na velikim, gusto popunjenim matricama postaje računski zahtjevna zbog O(n²) kompleksnosti po iteraciji.

Ovaj projekat nadovezuje se na seminarski rad tako što donosi uporednu studiju dvije verzije Power Iteration algoritma:

Sekvencijalna implementacija u C programskom jeziku, pogodna za srednje dimenzije, ali ograničena na jedno procesorsko jezgro.

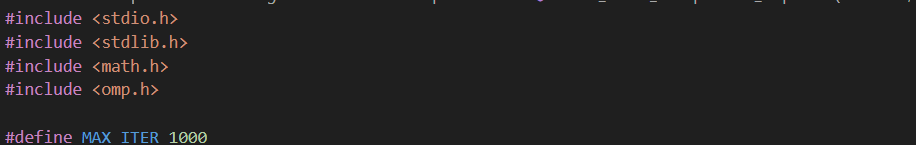
Paralelna implementacija pomoću OpenMP biblioteke u C programskom jeziku, koja raspodjeljuje množenje matrice i vektora između više procesora radi smanjenja vremena obrade, ali uvodi troškove komunikacije.

Cilj je da, kroz eksperimentalna mjerenja performansi na matricama različitih dimenzija, kvantifikujemo dobit od paralelizacije, izdvojimo one faze algoritma koje najviše doprinose ukupnom vremenu izvođenja i definišemo prag dimenzija od kojeg korišćenje paralelne verzije ima prednost nad sekvencijalnom.

# OPIS IMPLEMENTACIJE

U ovom poglavlju detaljno je objašnjen svaki segment implementacije algoritma za pronalaženje dominantne svojstvene vrijednosti matrice korišćenjem metode stepene iteracije, uz paralelizaciju pomoću OpenMP, u programskom jeziku C. Kod je modularno organizovan, sa jasno odvojenim funkcijama za osnovne matematičke operacije nad matricama i vektorima, kao i sam glavni algoritam. Funkcionalnost svih djelova koda je pojedinačno opisana u naredom dijelu.

Ove četiri biblioteke čine osnovu za rad sa matricama, matematičkim operacijama, rad sa dinamičkom memorijom i omogućavaju OpenMP paralelizaciju (slika 1).



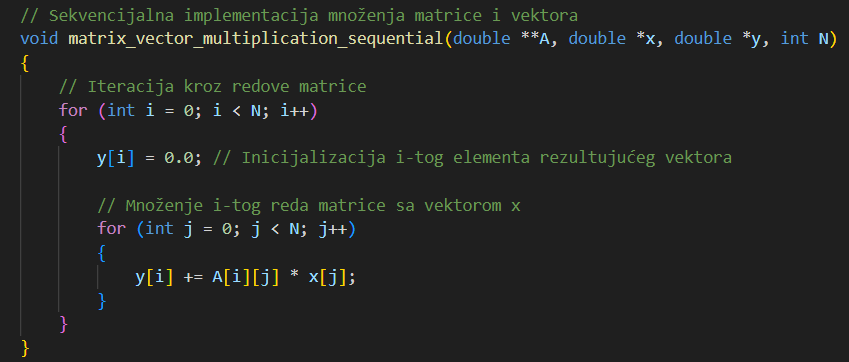
Slika : Uključivanje biblioteka

Vrijednosti na slici 2 podešavaju parametre algoritma: broj dozvoljenih iteracija, prihvatljivu grešku i koliko puta se test izvodi radi pouzdanosti mjerenja.



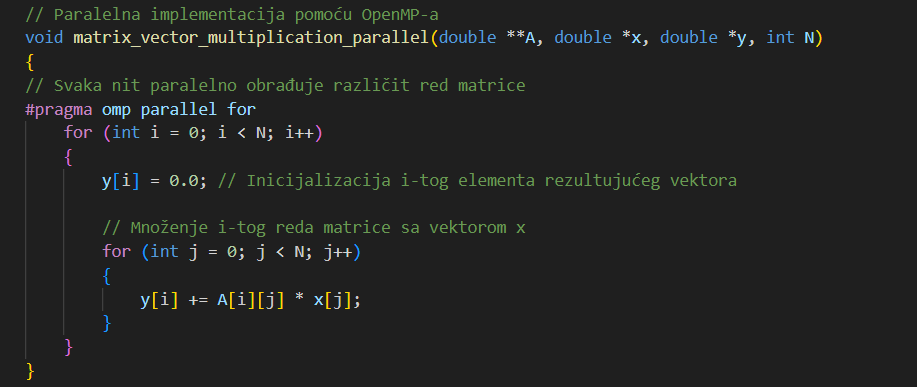
Slika : Definisanje konstanti

Funkcija na slici 3 implementira standardno sekvencijalno matrično-vektorsko množenje oblika **y = A·x** za kvadratne matrice.



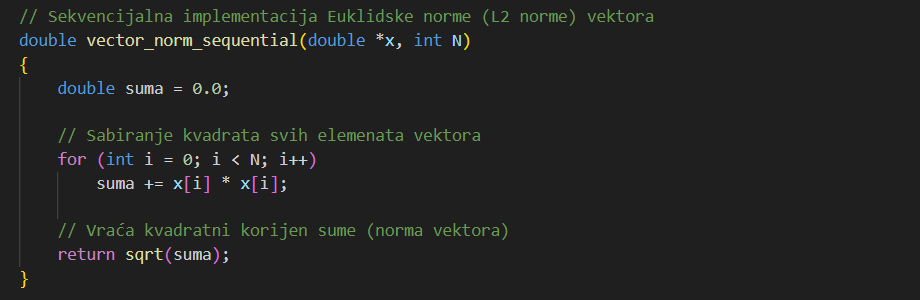
Slika : Množenje matrice i vektora (sekvencijalno)

U kodu na slici 4 se koristi OpenMP direktiva za automatsko raspoređivanje redova matrice na više procesorskih jezgara.



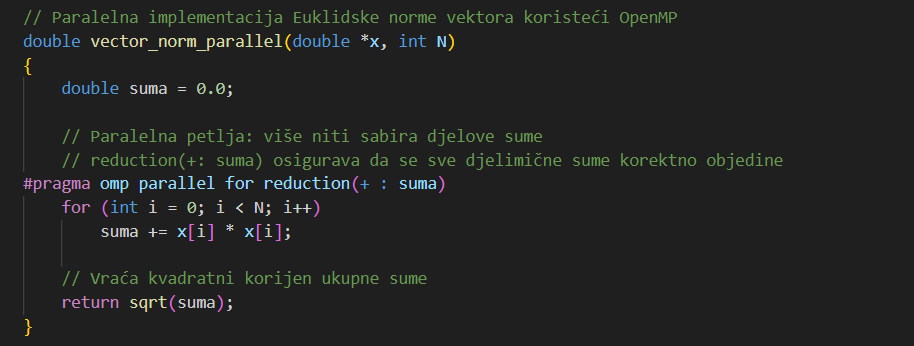
Slika : Množenje matrice i vektora (paralelno)

Funkcija na slici 5 računa standardnu sekvencijalnu Euklidsku normu vektora, tj. dužinu vektora.



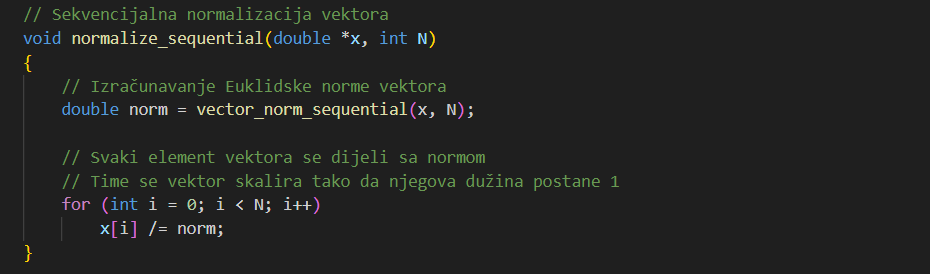
Slika : Izračunavanje Euklidske (L2) norme vektora (sekvencijalno)

Na slici 6 se nalazi paralelna implementacija računanja Euklidske norme. Sa reduction direktivom, djelimični rezultati sumiranja se automatski objedine po završetku petlje.



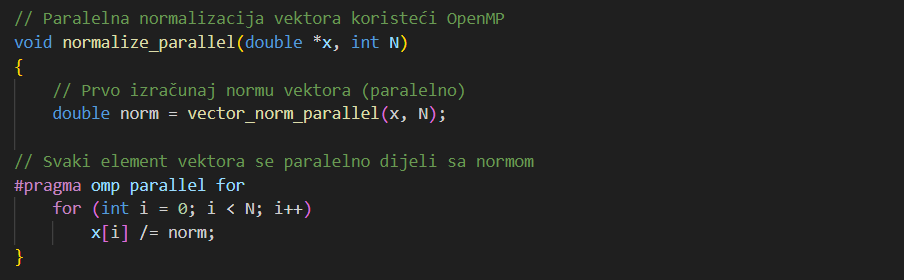
Slika : Euklidska norma vektora (paralelno)

Zatim, na slici 7 je prikazana sekvencijalna normalizacija vektora. Svi elementi vektora se skaliraju tako da njegova norma postane 1 (pravac vektora ostaje isti).



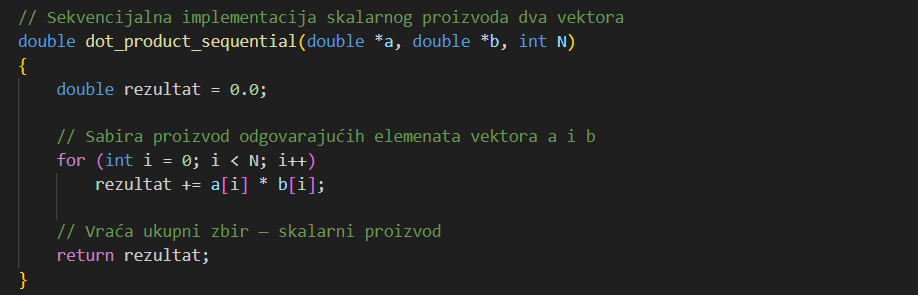
Slika : Normalizacija vektora (sekvencijalno)

Na slici 8 je data paralelna normalizacija vektora korišćenjem OpenMP. Norma vektora je već izračunata i ista je za sve elemente. Zato se svi elementi mogu normalizovati istovremeno — nema zavisnosti ni preklapanja između izračunavanja x[0], x[1], ... x[N-1]. Svaki x[i] se može podijeliti sa normom potpuno nezavisno od x[j] (za bilo koje j ≠ i). Zato se svi elementi mogu normalizovati istovremeno — nema zavisnosti ni preklapanja između izračunavanja x[0], x[1], ... x[N-1]. Svaka nit uzima svoj indeks (ili više njih) i nezavisno ažurira odgovarajući element x[i]. Pošto nema upisa u iste memorijske lokacije od strane više niti, nema rizika od "race condition" i nema potrebe za sinhronizacijom između niti u ovoj petlji.



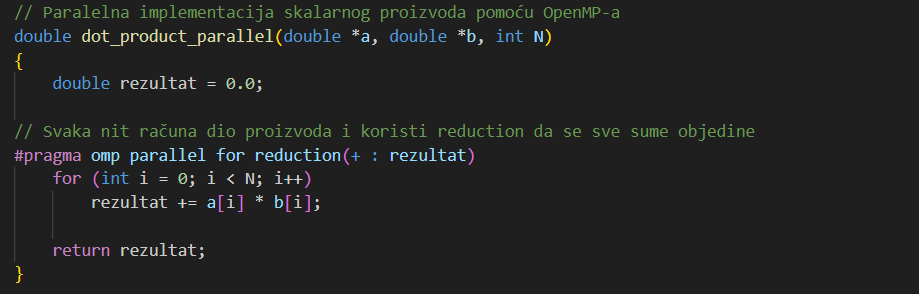
Slika : Normalizacija vektora (paralelno)

Na slici 9 je prikazan sekvencijalni kod za računanje skalarnog proizvoda dva vektora.



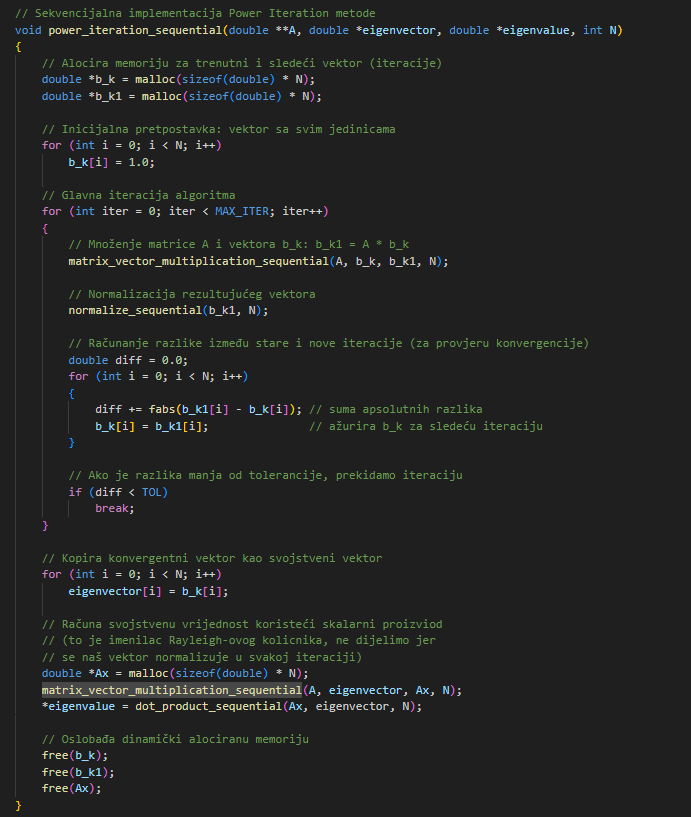
Slika : Skalarni proizvod vektora (sekvencijalno)

Dalje, na slici 10 je paralelna implementacija računanja skalaranog proizvoda. Koristi se OpenMP reduction da bi više jezgara istovremeno sabiralo djelimične proizvode.



Slika : Skalarni proizvod vektora (paralelno)

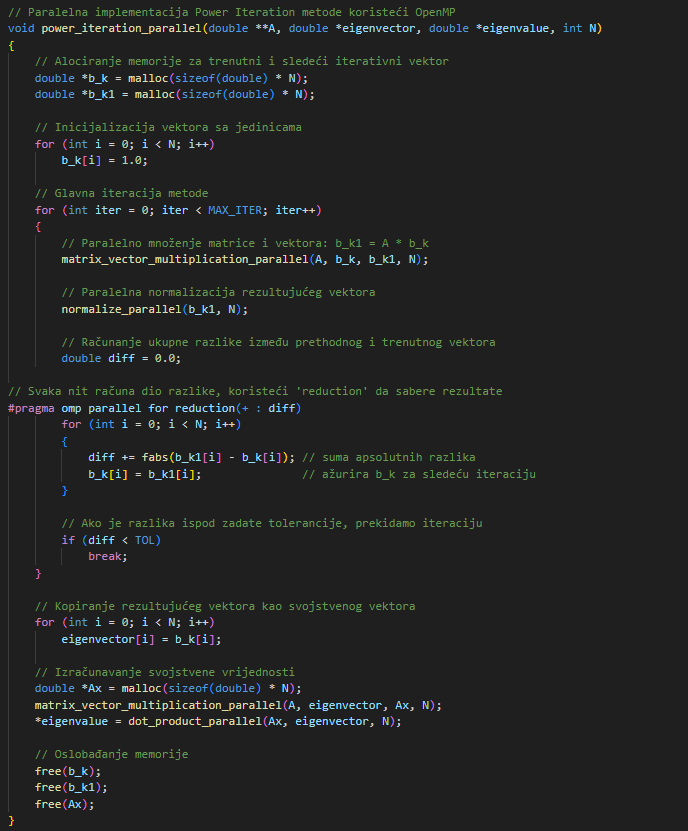
Na slici 11 je data sekvencijalno implementirana cjelokupna metoda stepene iteracije. Na početku se vrši alokacija vektora. Alociraju se dva vektora veličine N. U b\_k se nalazi trenutna iteracija, u b\_k1 se smješta rezultat nakon množenja i normalizacije. Algoritam počinje od proizvoljnog vektora — ovdje je to vektor pun jedinica. Glavna iteracija metode se izvodi najviše MAX\_ITER puta, ili manje ako dostigne konvergenciju. Zatim se koriste funkcije definisane na slici 3 i slici 7, za množenje matrice i normalizaciju rezultujućeg vektora. Nakon toga se računa suma apsolutnih razlika između stare i nove iteracije i prekopira se nova vrijednost u b\_k za sledeći prolaz. Ako je ukupna promjena između dvije iteracije manja od zadate tolerancije, smatra se da je algoritam konvergirao i prekida se dalja iteracija. Na kraju, u izlazni vektor se smješta rezultujući (dominantni) svojstveni vektor. Zatim se alocira novi privremeni vektor Ax dužine N. Ovdje će biti smješten rezultat množenja matrice i vektora. Potom se izračunava proizvod matrice A i pronađenog svojstvenog vektora uz pomoć formule iz slike 3. Rezultat (A·x) se upisuje u vektor Ax. Slijedi, računanje skalarnog proizvoda između vektora Ax i eigenvector — to je aproksimacija dominantne svojstvene vrijednosti λ, jer matematički: λ ≈ (A·x, x^T). Ova formula daje broj koji predstavlja koliko se vektor „skalira“ kada ga pomnožimo matricom A — i to je upravo definicija svojstvene vrijednosti. Na kraju se oslobađa memorija korišćena za skladištenje vektora.



Slika : Metoda stepene iteracije (sekvencijalna)

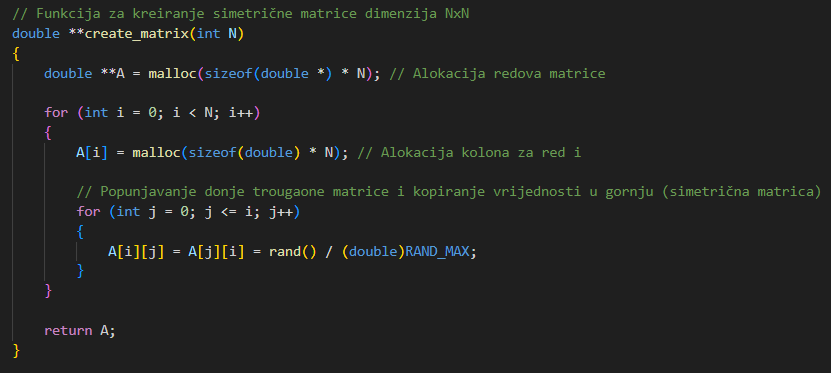
Na slici 12 imamo paralelizovanu implementaciju metode stepene iteracije za nalaženje dominantne svojstvene vrijednosti. Ovdje se umjesto sekvencijalnih koriste paralelne verzije svih funkcija. Razlika je u OpenMP direktivama i efikasnosti na višeprocesorskim sistemima. Takođe, paralelizuje se i računanje ukupne razlike između vektora iz dvije uzastopne iteracije i ažuriranje vektora. U ovom segmentu koda, koristi se paralelna petlja kako bi se efikasno izračunala suma apsolutnih razlika između elemenata vektora iz prethodne (b\_k) i trenutne (b\_k1) iteracije. Za svaki indeks i, računa se fabs(b\_k1[i] - b\_k[i]), što predstavlja koliko se svaki pojedinačni element promijenio. Ove razlike se sabiraju u zajedničku varijablu diff koristeći OpenMP reduction, što omogućava svim nitima da paralelno računaju svoje djelove sume i da na kraju dobiju ispravan zbir. Istovremeno, svaka nit ažurira vrijednost u vektoru b\_k novom vrijednošću iz b\_k1 za sljedeću iteraciju.

Ovaj postupak omogućava da se provjeri da li je algoritam konvergirao — ako je ukupna razlika manja od zadate tolerancije, iteracija se prekida. Paralelizacija ove provjere dodatno doprinosi ubrzanju izvođenja za velike dimenzije vektora.



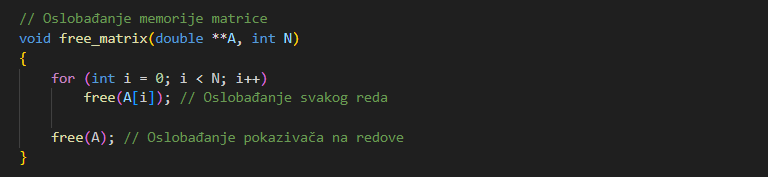
Slika : Metoda stepene iteracije (paralelna)

Ova funkcija stvara novu NxN matricu koja je simetrična (A[i][j] = A[j][i]), ispunjenu slučajnim brojevima u intervalu [0, 1] (slika 13).



Slika : Generisanje simetrične matrice

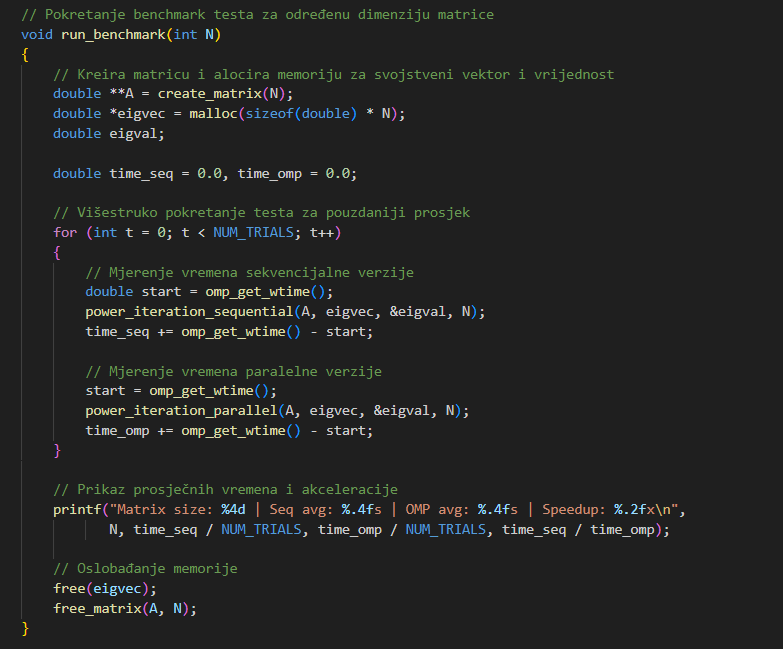
Funkcija na slici 14 vrši potpuno oslobađanje memorije koja je bila rezervisana za dvodimenzionalnu matricu.



Slika : Oslobađanje memorije

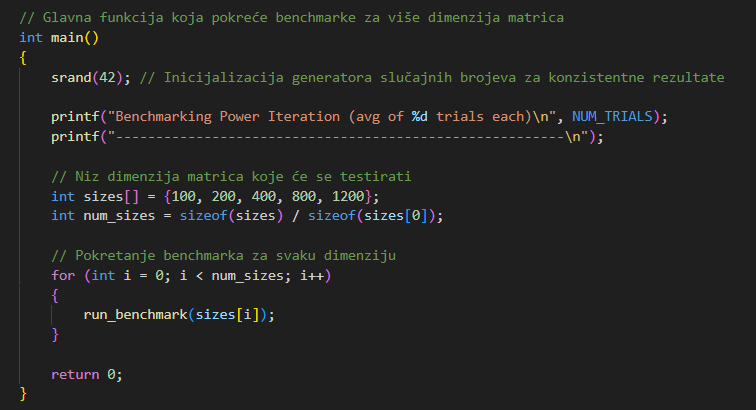
Na slici 15 je funkcija koja automatski testira performanse i sekvencijalne i paralelne verzije algoritma stepene iteracije za računanje dominantne svojstvene vrijednosti matrice, i to za svaku zadatu dimenziju matrice. Prvo se generiše nasumična kvadratna matrica i alocira memorija za vektor i svojstvenu vrijednost. Zatim, funkcija u petlji višestruko ponavlja izvođenje oba algoritma (sekvencijalnog i paralelnog) kako bi se dobili pouzdani prosječni rezultati mjerenja vremena. Za svaku verziju mjeri se vrijeme izvođenja pomoću OpenMP funkcije omp\_get\_wtime(), sabiraju se ukupna vremena i računa se prosjek. Po završetku testiranja, rezultati se prikazuju u jasno formatiranom obliku: veličina matrice, prosječno vrijeme za sekvencijalnu i paralelnu verziju, kao i ubrzanje (speedup). Na kraju se sva privremena memorija pravilno oslobađa.

Na ovaj način, funkcija daje objektivnu procjenu koristi od paralelizacije za različite dimenzije matrica, omogućava lako poređenje performansi i identifikuje za koje veličine podataka paralelna verzija daje najviše benefita.



Slika : Benchmark funkcija (mjerenje performansi)

Za kraj, prikazana je glavna funkcija na slici 16. Ona pokreće testiranje za više dimenzija matrica, obezbjeđuje ponovljivost rezultata i prikazuje rezultate korisniku.



Slika : Glavna funkcija

# REZULTATI

Svi eksperimenti su izvođeni na računaru MacBook Pro 13" (M1, 2020) sa ukupno 8 jezgara (4 performance i 4 efficiency jezgra) i 16 GB RAM-a. Ove karakteristike treba uzeti u obzir pri interpretaciji rezultata, budući da brzina izvođenja paralelnih algoritama direktno zavisi od broja fizičkih i logičkih jezgara, kao i ukupnih resursa računarskog sistema.

Testiranje je sprovedeno tako što su, za niz različitih dimenzija kvadratnih matrica (100, 200, 400, 800, 1200, 1600, 2000, 4000, 10000), izvođene i sekvencijalna i paralelna verzija algoritma stepene iteracije. Svaka verzija je pokrenuta više puta (prema vrijednosti konstante NUM\_TRIALS) da bi se dobili pouzdani prosjeci vremena izvršavanja. Svi eksperimenti su rađeni na istim slučajno generisanim matricama, a pomoću fiksiranog „seed“-a (srand(42)) je osigurana ponovljivost rezultata.

Za svaku dimenziju matrice, mjereno je: prosječno vrijeme izvođenja sekvencijalne verzije, prosječno vrijeme izvođenja paralelne verzije (OpenMP) i faktor ubrzanja („speedup“), koji je odnos sekvencijalnog i paralelnog vremena.

## Tabela rezultata

U tabeli 1 su prikazani dobijeni rezultati mjerenja vremena izvršavanja sekvencijalne i paralelne implementacije metode stepene iteracije za različite dimenzije matrica. U tabeli su navedeni prosječni iznosi vremena za obje verzije, kao i faktor ubrzanja koji je izračunat kao odnos sekvencijalnog i paralelnog vremena.

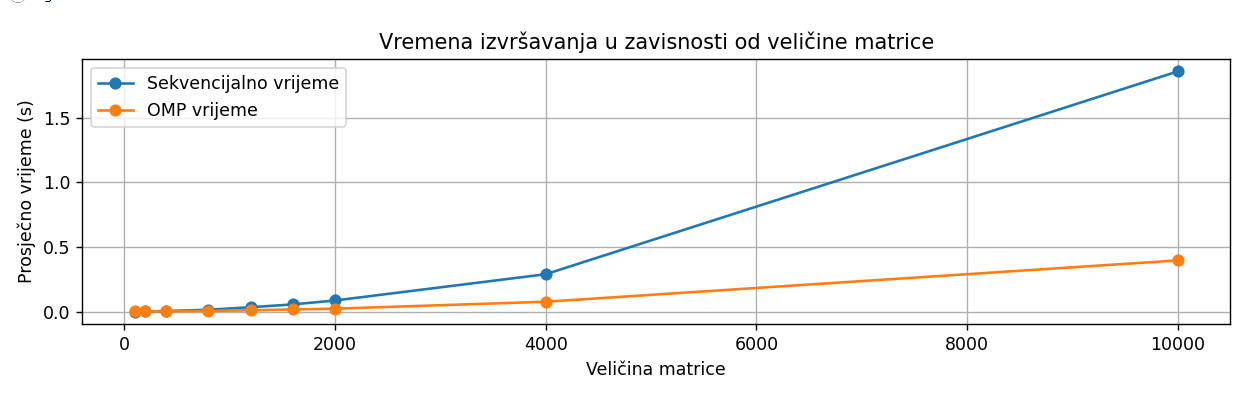
| **Dimenzija matrice (N)** | **Sekvencijalno vrijeme (s)** | **Paralelno vrijeme (s)** | **Ubrzanje** |
| --- | --- | --- | --- |
| 100 | 0.0003 | 0.0013 | 0.24x |
| 200 | 0.0010 | 0.0013 | 0.75x |
| 400 | 0.0041 | 0.0024 | 1.72x |
| 800 | 0.0138 | 0.0047 | 2.95x |
| 1200 | 0.0336 | 0.0099 | 3.40x |
| 1600 | 0.0552 | 0.0160 | 3.44x |
| 2000 | 0.0856 | 0.0223 | 3.84x |
| 4000 | 0.2890 | 0.0760 | 3.80x |
| 10000 | 1.8595 | 0.3960 | 4.70x |

Tabela : Rezultati

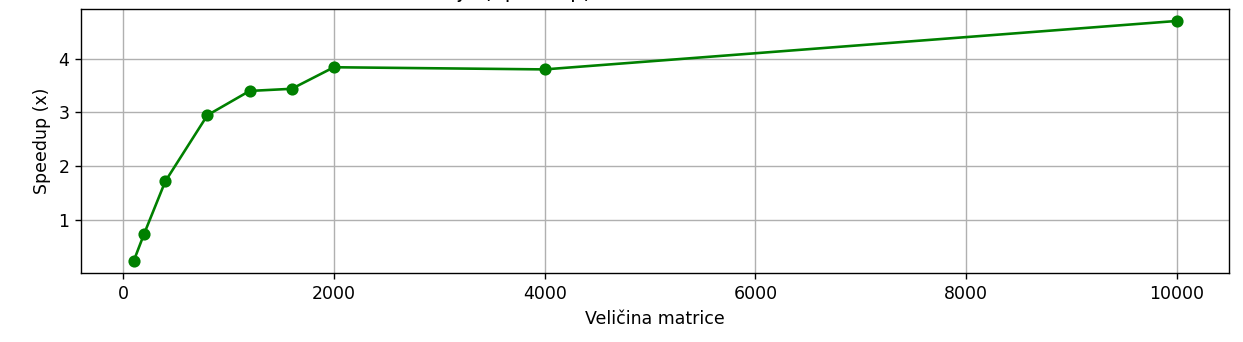
## Grafički prikaz rezultata

Radi lakšeg poređenja performansi sekvencijalne i paralelne implementacije metode stepene iteracije, rezultati su prikazani i grafički. Na grafikonu 1 su na X-osi prikazane različite dimenzije matrica, dok je na Y-osi prikazano vrijeme izvršavanja u sekundama. Prikazane su vrijednosti za obje verzije algoritma, što omogućava vizuelno uočavanje razlike u performansama, kao i trendove ubrzanja kod većih dimenzija.

Takođe, prikazan je i grafikon faktora ubrzanja (speedup), koji pokazuje odnos između vremena izvršavanja sekvencijalnog i paralelnog algoritma za svaku veličinu matrice (grafikon 2). Iz grafikona je jasno vidljivo da paralelna verzija daje značajno bolje rezultate za veće matrice, dok kod manjih matrica prednost nije toliko izražena.



Grafikon :Vremena izvršavanja u zavisnosti od veličine matrice



Grafikon : Ubrzavanje (Speedup) u zavisnosti od veličine matrice

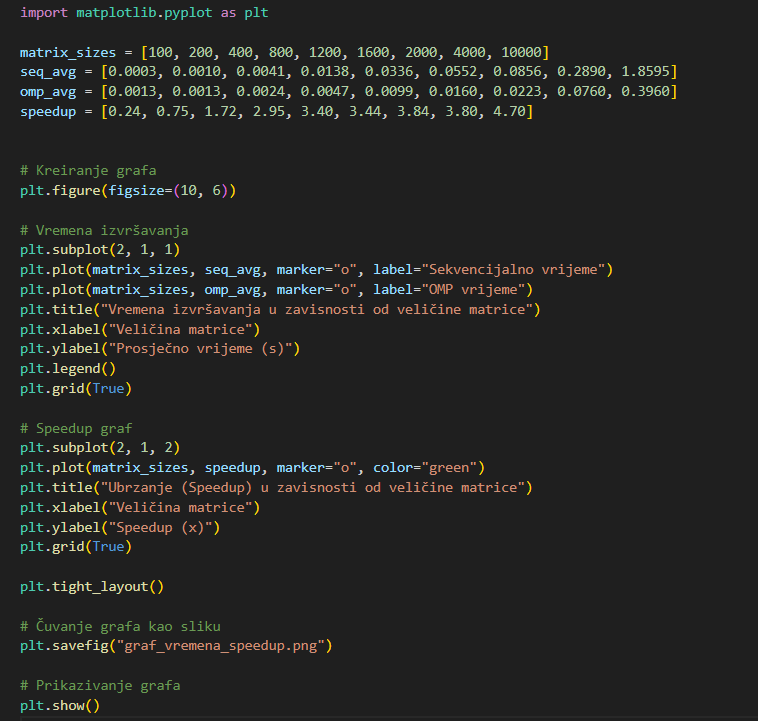
Grafički prikaz na ovaj način omogućava brzu i intuitivnu analizu efikasnosti paralelizacije, kao i identifikaciju veličina problema za koje je paralelni pristup najisplativiji.

Na osnovu rezultata vidljivo je da paralelna implem1entacija značajno ubrzava izvođenje algoritma u odnosu na sekvencijalnu, posebno za veće dimenzije matrica. Faktor ubrzanja (speedup) je približno konstantan i proporcionalan broju dostupnih jezgara na procesoru. Za najmanje matrice ubrzanje je manje izraženo zbog dodatnog troška paralelizacije, dok kod većih matrica prednosti paralelnog izvođenja postaju dominantne.

Ovi rezultati potvrđuju efikasnost OpenMP paralelizacije za problem nalazenja dominantne svojstvene vrijednosti matrice, i pokazuju da se Power Iteration metoda može značajno optimizovati za rad sa velikim podacima korišćenjem modernih višejezgarnih procesora.

Ispod, na slici 17 prikazan je kod koji je korišćen za dobijanje grafičkog prikaza rezultata. Ovaj python kod koristi biblioteku matplotlib za vizualizaciju rezultata mjerenja vremena izvršavanja sekvencijalne i paralelne verzije metode stepene za različite dimenzije matrica.

Ovaj kod se sastoji iz nekoliko logičkih cjelina koje zajedno omogućavaju automatsko crtanje i čuvanje grafikona na osnovu eksperimentalnih rezultata. Prvo se definišu podaci — dimenzije matrica, izmjerena vremena izvršavanja i dobijeni speedup. Zatim se kreira glavni prozor odgovarajuće veličine i iscrtava prvi grafikon koji prikazuje poređenje vremena izvršavanja sekvencijalnog i paralelnog algoritma. Drugi grafikon prikazuje ubrzanje (speedup) koje donosi paralelizacija, jasno označen zelenom bojom radi lakšeg poređenja. Automatskim sređivanjem rasporeda obezbjeđuje se preglednost svih elemenata grafikona, a potom se oba grafikona zajedno čuvaju kao slika.



Slika : Kod za grafički prikaz rezultata

# ZAKLJUČAK

U ovom projektu je uspješno implementirana i analizirana metoda stepene iteracije za nalaženje dominantne svojstvene vrijednosti matrice, u sekvencijalnoj i paralelnoj verziji korišćenjem OpenMP.

Rezultati eksperimentalnog testiranja pokazuju da paralelizacija donosi značajno poboljšanje performansi kod većih matrica, gdje je ubrzanje višestruko u odnosu na sekvencijalnu implementaciju. Za manje matrice, dobitak je minimalan ili ga čak nema već je vrijeme izvršavanja paralelne verzije sporije, što ukazuje da se trošak upravljanja nitima i organizacije paralelnog rada isplati tek kod većih problema. Dobijeni rezultati potvrđuju opravdanost primjene paralelnog programiranja za numeričke algoritme na savremenim višejezgarnim procesorima. Korišćenje alata poput OpenMP-a omogućava jednostavno i efikasno iskorišćenje hardverskih resursa, uz relativno male izmjene postojeće sekvencijalne logike. Na kraju, grafička analiza rezultata jasno pokazuje kada i koliko paralelizacija doprinosi ubrzanju algoritma, što je od ključnog značaja za praktičnu primjenu u obradi velikih podataka.

U budućnosti se očekuje dalji razvoj paralelnih tehnika koji će omogućiti još veću efikasnost metode stepene iteracije, naročito u aplikacijama poput analize velikih skupova podataka, neuronskih mreža, ili računarskih simulacija na superračunarima.