Mobilna aplikacija za rad s kompleksnim matricama

Šarčević, Nikolina

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

Sveučilišni studij

MOBILNA APLIKACIJA ZA RAD S KOMPLEKSNIM MATRICAMA

Završni rad

Nikolina Šarčević

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 01.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Nikolina Šarčević
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R 4429, 01.10.2020.
OIB Pristupnika:	18926255370
Mentor:	doc. dr. sc. Anita Katić
Sumentor:	doc. dr. sc. Krešimir Romić
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna aplikacija za rad s kompleksnim matricama
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rad:	U radu je potrebno izložiti osnovne pojmove i operacije s matricama kojima su elementi kompleksni brojevi te Izraditi mobilnu aplikaciju koja će izvršavati osnovne operacija s matricama kojima su elementi kompleksni brojevi. Treba opisati postupak izrade aplikacije, testirati ju na primjerima i prikazati rezultate. Tema rezervirana za: Nikolina Šarčević Sumentor s FERIT-a: Krešimir Romić
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	01.09.2023.
Datum potvrde ocjene od strane Odbora:	08.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 11.09.2023.

Ime i prezime studenta:	Nikolina Šarčević
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R 4429, 01.10.2020.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna aplikacija za rad s kompleksnim matricama**

izrađen pod vodstvom mentora doc. dr. sc. Anita Katić

i sumentora doc. dr. sc. Krešimir Romić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Matricom se smatraju sve pravokutne tablice koje su napravljene od redaka i stupaca. Retci i stupci su ispunjeni elementima koji su najčešće realni brojevi, ali mogu biti i kompleksni. Elementi mogu predstavljati podatke ili matematičke jednadžbe. Ukoliko matrica ima m redaka i n stupaca zapisujemo ju kraće kao $\mathrm{A} = (\mathrm{a}_{ij})$ ili na sljedeći način:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \text{ te se kaže da je tipa } m \times n.$$

Matrice imaju vrlo široku i raznoliku primjenu u različitim područjima znanosti te su relativno novi i napredan dio matematike. Uz matematiku jako važnu primjenu imaju u računalnim znanostima. Posebno je korisno u grafici pri obradi digitalnih slika jer su digitalne slike u osnovi matrice. Redovi i stupci piksela na slikama se mogu zapisati kao redovi i stupci u matrici, dok numerički unosi odgovaraju bojama na slici. Osim za obradu slika, koristi se i za obradu digitalnog videa i zvuka. Također, svoju primjenu pronalaze i u optici, kriptografiji, ekonomiji, kemiji, geologiji, robotici, bežičnoj komunikaciji, obradi signala i još mnoge druge.

Cilj ovog završnog rada je uspješno izraditi mobilnu aplikaciju koja bi olakšala rješavanje zadataka koji uključuju kompleksne matrice, a njegova struktura je sljedeća. U drugom poglavlju je objašnjeno što su to kompleksne matrice te su definirane osnovne operacije s matricama. Također, prikazana su i postojeća rješenja na ovu temu Treće poglavlje uključuje korištene alate i prikaz korisničkog sučelja. U četvrtom poglavlju se nalazi implementacija funkcionalnosti aplikacije, prikaz programskog rješenja te prikaz rezultata i izlaznih podataka. Na samom je kraju dan zaključak.

1.1. Zadatak završnog rada

U ovom završnom radu potrebno je izložiti osnovne pojmove i operacije s matricama kojima su elementi kompleksni brojevi te izraditi mobilnu aplikaciju koja će izvršavati osnovne operacije s matricama kojima su elementi kompleksni brojevi. Treba opisati postupak izrade aplikacije, testirati ju na primjerima i prikazati rezultate.

2. KOMPLEKSNE MATRICE

Kao što je već spomenuto u uvodu, elementi matrica su najčešće realni brojevi, no u ovom završnom radu se bavimo matricama čiji su elementi kompleksni brojevi.

2.1. Definicija

Kompleksi brojevi su brojevi koje zapisujemo u obliku

$$z = x + yi\,,$$

gdje su x i y elementi skupa realnih brojeva, dok je i imaginarna jedinica pri čemu i zadovoljava jednakost $i^2 = -1$. Broj x zovemo realni dio kompleksnog broja z i označavamo ga s $Re\ z$, dok broj y nazivamo imaginarni dio te ga označavamo s $Im\ z$ [5]. Za matricu kažemo da je kompleksna matrica ako vrijedi da je $a_{ij} \in \mathbb{C}$, za sve elemente a_{ij}, gdje i predstavlja redak, a j stupac u kojem se nalazi element a_{ij}.

Kompleksni brojevi se uvode u matematiku kako bi omogućili rješavanje matematičkih problema kao što su kvadratne jednadžbe, koje nemaju rješenje u polju realnih brojeva $\mathbb{R}$. Time se postiže da takvi problemi uvijek imaju rješenje, za bilo koji izbor koeficijenata. Na primjer, kvadratna jednadžba

$$x^2 + 1 = 0$$

nema rješenja u skupu realnih brojeva, rješiva je u skupu kompleksnih brojeva i njena rješenja su $x = i$ i $x = -i$. Također, još jedan od razloga uvođenja kompleksnih brojeva obrazlaže Bezoutov poučak koji govori da se dvije algebarske krivulje redova m ili n sijeku u točno m ili n točaka (uzimajući u obzir kratnosti i točke u beskonačnosti). Ovaj poučak ne bi vrijedio da nema kompleksnih brojeva jer na primjer pravac

$$y = x + 2$$

u polju realnih brojeva ne siječe kružnicu jednadžbe $x^2 + y^2 = 1$, već ju siječe u točkama $(-1 - \frac{\sqrt{2}}{2}i,\ 1 - \frac{\sqrt{2}}{2}i\,)$ i $(-1 + \frac{\sqrt{2}}{2}i, 1 + \frac{\sqrt{2}}{2}i\,)$. Jednim od prvih koji je uočio potrebu za proširenjem polja realnih brojeva smatra se talijanski matematičar, G. Cardano [7]. On je prilikom rješavanja geometrijskog problema dobio jednakost iz koje se može zaključiti da se množenjem nerealnih brojeva dobije produkt koji je realan broj.

Primjena kompleksnih brojeva podrazumijeva skoro sve grane moderne matematike, a osim toga primjenjuju se i u dijelovima fizike i elektrotehnike [5].

2.2. Osnovne operacije s matricama

Osnovne operacije koje su temeljne za rad s matricama i koje će biti obrađene i implementirane u ovom završnom radu i aplikaciji su: zbrajanje matrica, množenje matrica, množenje matrica skalarom te transponiranje matrice.

2.2.1. Zbrajanje matrica

Da bi se matrice mogle zbrajati moraju biti istog tipa. Zbrajanje matrica se obavlja na način da se elementi matrica na istom mjestu zbrajaju, odnosno

$$(a+b)_{ij} := (a)_{ij} + (b)_{ij}.$$

Rezultat koji dobijemo je ponovno matrica istog tipa. Pravila zbrajanja koja vrijede za matrice realnih brojeva vrijede i za kompleksne matrice, samo što elemente kompleksne matrice zbrajamo prema sljedećem pravilu

$$z_1 \pm z_2 = (x_1 + y_1 i) \pm (x_2 + y_2 i) = (x_1 \pm x_2) + (y_1 \pm y_2)i$$

U sljedećem primjeru prikazano je zbrajanje kompleksnih matrica:

$$\begin{bmatrix} 3+i & 1-i \\ 8-i & 5+i \end{bmatrix} + \begin{bmatrix} i & 1+i \\ 2 & -3-i \end{bmatrix} = \begin{bmatrix} 3+2i & 2 \\ 10-i & 2 \end{bmatrix}.$$

Općenito za zbrajanje kompleksnih matrica vrijedi

$$\begin{bmatrix} x_{1_{11}} + y_{1_{11}} i & \cdots & x_{1_{1n}} + y_{1_{1n}} i \\ \vdots & \ddots & \vdots \\ x_{1_{m1}} + y_{1_{m1}} i & \cdots & x_{1_{mn}} + y_{1_{mn}} i \end{bmatrix} + \begin{bmatrix} x_{2_{11}} + y_{2_{11}} i & \cdots & x_{2_{1n}} + y_{2_{1n}} i \\ \vdots & \ddots & \vdots \\ x_{2_{m1}} + y_{2_{m1}} i & \cdots & x_{2_{mn}} + y_{2_{mn}} i \end{bmatrix} =$$

$$\begin{bmatrix} (x_{1_{11}} + x_{2_{11}}) + (y_{1_{11}} + y_{2_{11}})i & \cdots & (x_{1_{1n}} + x_{2_{1n}}) + (y_{1_{1n}} + y_{2_{1n}})i \\ \vdots & \ddots & \vdots \\ (x_{1_{m1}} + x_{2_{m1}}) + (y_{1_{m1}} + y_{2_{m1}})i & \cdots & (x_{1_{mn}} + x_{2_{mn}}) + (y_{1_{mn}} + y_{2_{mn}})i \end{bmatrix}.$$

Također, za zbrajanje matrica vrijede sljedeća svojstva:

1) asocijativnost zbrajanja $(A+B)+C = A+(B+C)$,
2) postojanje neutralnog elementa za zbrajanje $A+O = O+A = A$, odnosno $O = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$

3) postojanje suprotnog elementa s obzirom na zbrajanje $A + B = B + A = O$, gdje B predstavlja suprotnu matricu, odnosno $B = -A$,
4) komutativnost zbrajanja $A + B = B + A$.

2.2.2. Množenje matrica skalarom

Ako postoji neka kompleksna matrica A i neki realan broj λ , množenjem skalara i kompleksne matrice A, dobijemo kompleksnu matricu istog tipa kao i A, koju označavamo λA. Elementi te matrice su λa_{ij}. Odnosno, kompleksnu matricu množimo skalarom λ tako da svaki njen element pomnožimo brojem λ. Za množenje skalarom vrijede sljedeća svojstva ($\lambda, \mu \in \mathbb{R}$):

1) distributivnost u odnosu na zbrajanje matrica $\lambda(A + B) = \lambda A + \lambda B$,
2) distributivnost u odnosu na zbrajanje skalara $(\lambda + \mu)A = \lambda A + \mu A$,
3) kvaziasocijativnost $(\lambda \cdot \mu)A = \lambda(\mu A)$

2.2.3. Množenje matrica

Da bi se matrice mogle množiti, moraju biti ulančane. To znači da broj stupaca prve matrice A mora odgovarati broju redaka druge matrice B. Množenje matrica se radi na način da se i-ti redak matrice A množi s k-tim stupcem matrice B prema formuli $c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$. Ukoliko je matrica A tipa $i \times n$, a matrica B tipa $n \times k$, kao rezultat, odnosno produkt matrica, dobije se matrica c_{ik} koja je tipa $i \times k$. Ista pravila vrijede i za realne i za kompleksne matrice, samo što je potrebno pripaziti na množenje kompleksnih brojeva koje se radi na sljedeći način

$$z_1 \cdot z_2 = (x_1 + y_1 i) \cdot (x_2 + y_2 i) = (x_1 x_2 - y_1 y_2) + (x_1 y_2 + y_1 x_2) i$$

Svojstva množenja matrica su:

1) $A \cdot (B + C) = AB + AC$,
2) $(A + B) \cdot C = AC + BC$,
3) $(\alpha A) \cdot B = A \cdot \alpha B = \alpha(AB)$,
4) $(AB)C = A(BC)$,
5) $IA = A, AI = A$

Primjer množenja matrica:

$$\begin{bmatrix} 2+i & 8+3i \\ -1+2i & -i \\ 5 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1-i & 3-i & 2i \\ i & 4-2i & 9i & 6 \end{bmatrix}$$

$$= \begin{bmatrix} (2+i)\cdot 2 + (8+3i)\cdot i & \cdots & \cdots & \cdots \\ \cdots & (-1+2i)\cdot(1-i) + (-i)\cdot(4-2i) & \cdots & \cdots \\ \cdots & \cdots & \cdots & 5\cdot 2i + 0\cdot 6i \end{bmatrix}$$

$$= \begin{bmatrix} 1+10i & 41-5i & -20+73i & 46+22i \\ -1+4i & -1-i & 8+7i & -4-8i \\ 10 & 5-5i & 15-5i & 10i \end{bmatrix}$$

2.2.4. Transponiranje matrica

Neka je A kompleksna matrica tipa $m \times n$, njena transponirana matrica A^T je tipa $n \times m$. Ako je $A = (a_{ij})$, a $A^T = (b_{ij})$ za njihove elemente vrijedi $b_{ij} = a_{ji}$. Odnosno, transponiranje matrica se obavlja tako da se element iz i-tog retka i j-tog stupca matrice A mijenja s elementom j-tog retka i i-tog stupca. Ta operacija se obavlja samo na jednoj matrici te se zbog toga naziva unarna operacija matrica.

Primjer transponiranja matrice:

$$\begin{bmatrix} 2+i & -1+2i & 5 \\ 8+3i & -i & 0 \end{bmatrix}^T = \begin{bmatrix} 2+i & 8+3i \\ -1+2i & -i \\ 5 & 0 \end{bmatrix},$$

a opći prikaz transponiranja izgleda ovako:

$$\begin{bmatrix} x_{11}+y_{11}i & \cdots & x_{m1}+y_{m1}i \\ \vdots & \ddots & \vdots \\ x_{1n}+y_{1n}i & \cdots & x_{mn}+y_{mn}i \end{bmatrix}^T = \begin{bmatrix} x_{11}+y_{11}i & \cdots & x_{1n}+y_{1n}i \\ \vdots & \ddots & \vdots \\ x_{m1}+y_{m1}i & \cdots & x_{mn}+y_{mn}i \end{bmatrix}$$

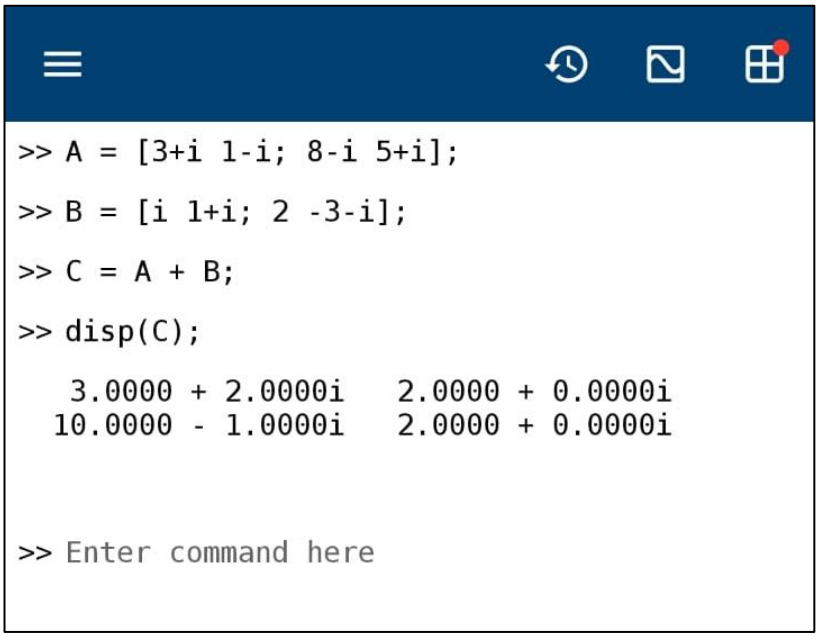
2.3. Postojeća rješenja

U ovom potpoglavlju su prikazana već postojeća, slična, Android rješenja na temu kompleksnih matrica.

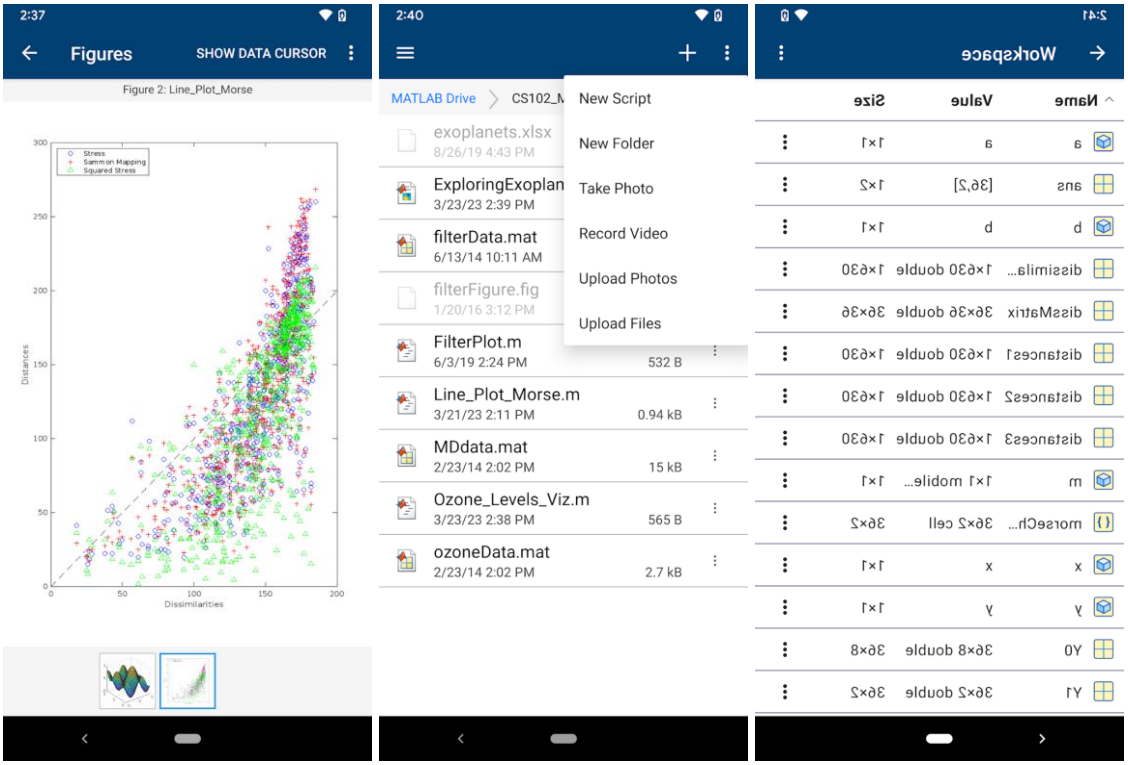
2.3.1. MATLAB Mobile

Ova mobilna aplikacija nije napravljena samo za rad s kompleksnim matricama, već ima i puno drugih funkcionalnosti. Omogućuje rad s osnovnim aritmetičkim operacijama, trigonometrijske funkcije, eksponencijalne funkcije, logaritamske, kompleksne operacije. Također, mogu se pisati

i izvršavati Matlab skripte što omogućuje izradu programa za obradu i analizu podataka. Osim toga, omogućuje i prikazivanje grafikona, vizualizaciju podataka te simulaciju i modeliranje različitih sustava. Funkcionalnosti koje pruža za rad s matricama su transponiranje matrica, množenje, zbrajanje, pronalazak inverza i još mnoge druge funkcionalnosti. Aplikacija je razvijena od strane The MathWorks, Inc., a u nastavku je primjer zbrajanja kompleksnih matrica te prikaz korisničkog sučelja aplikacije.



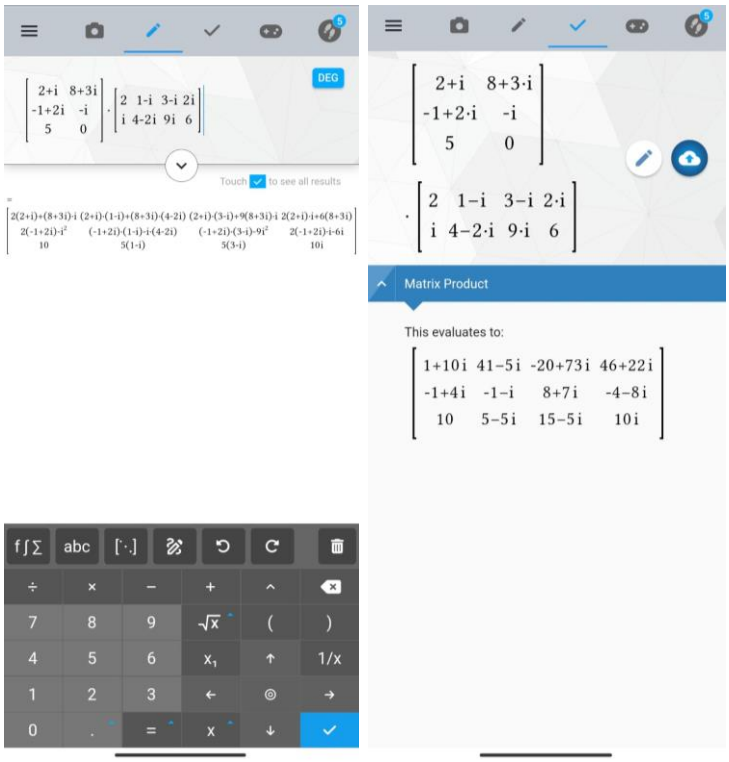
Slika 2.1. Primjer zbrajanja kompleksnih matrica



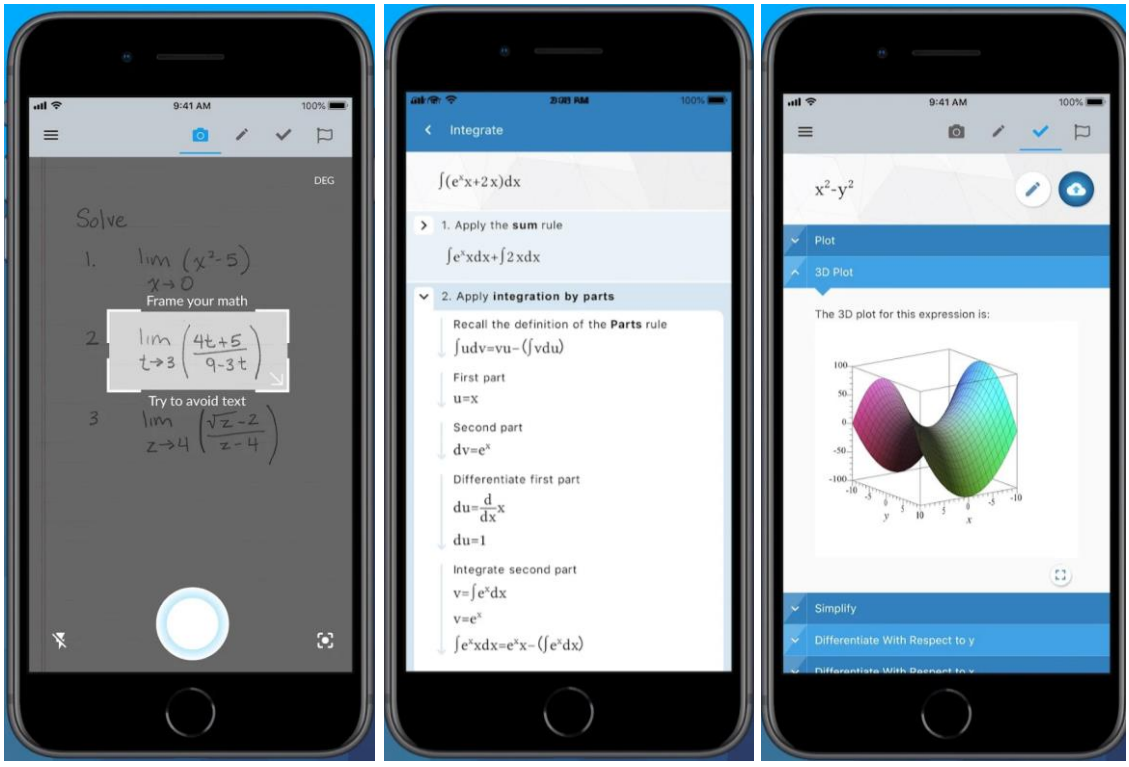
Slika 2.2. Izgled korisničkog sučelja

2.3.2. Maple Calculator

Maple Calculator je mobilna aplikacija koja je vrlo slična Photomath-u. Razvijena je od strane Maplesoft-a te omogućava korisnicima da se služe mnogim funkcionalnostima. Sličnost s Photomath-om je ta da omogućuje fotografiranje matematičkih zadataka koje zatim rješava korak po korak. Kako bi se vidjeli koraci rješavanja potrebno je pretplatiti se na aplikaciju, ali ukoliko želimo vidjeti samo rješenje zadatka to možemo dobiti bez pretplate. Neke od funkcionalnosti su: zbrajanje, množenje, faktorizacija, integriranje, rješavanje jednadžbi, evaluacija izraza, aproksimacije, rad s decimalnim brojevima, grafički prikazi, matrice i vektorske operacije, statistika i analiza podataka te pretvorba mjernih jedinica (duljina, masa, vrijeme, temperatura i sl.). U nastavku je dano korisničko sučelje aplikacije i primjer množenja kompleksnih matrica.



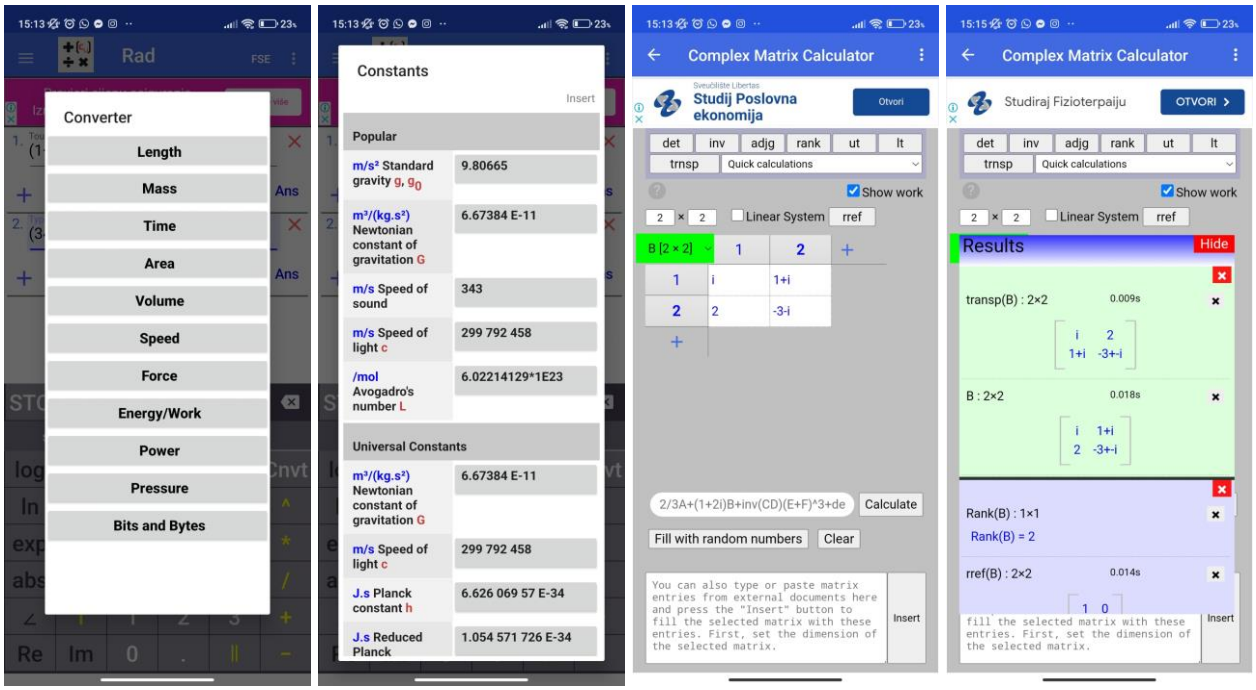
Slika 2.3. Primjer množenja kompleksnih matrica



Slika 2.4. Izgled korisničkog sučelja

2.3.3. Complex Number Calculator

Android aplikacija, Complex Number Calculator napravljena je od strane A.M.I.R. Kao i prethodne dvije aplikacije koje su spomenute može se preuzeti na Google Play-u. Ova aplikacija se razlikuje od prethodne dvije jer nema toliko puno funkcionalnosti. Njene funkcionalnosti su pretvorba mjernih jedinica (vrijeme, masa, volumen, duljina i sl.), rad s kompleksnim matricama (determinanta, inverz, rang, transponirana i adjungirana matrica) te nudi opciju i znanstvenih konstanti. U nastavku je dano korisničko sučelje aplikacije.



Slika 2.5. Korisničko sučelje

3. RAZVOJ PROGRAMSKOG RJEŠENJA

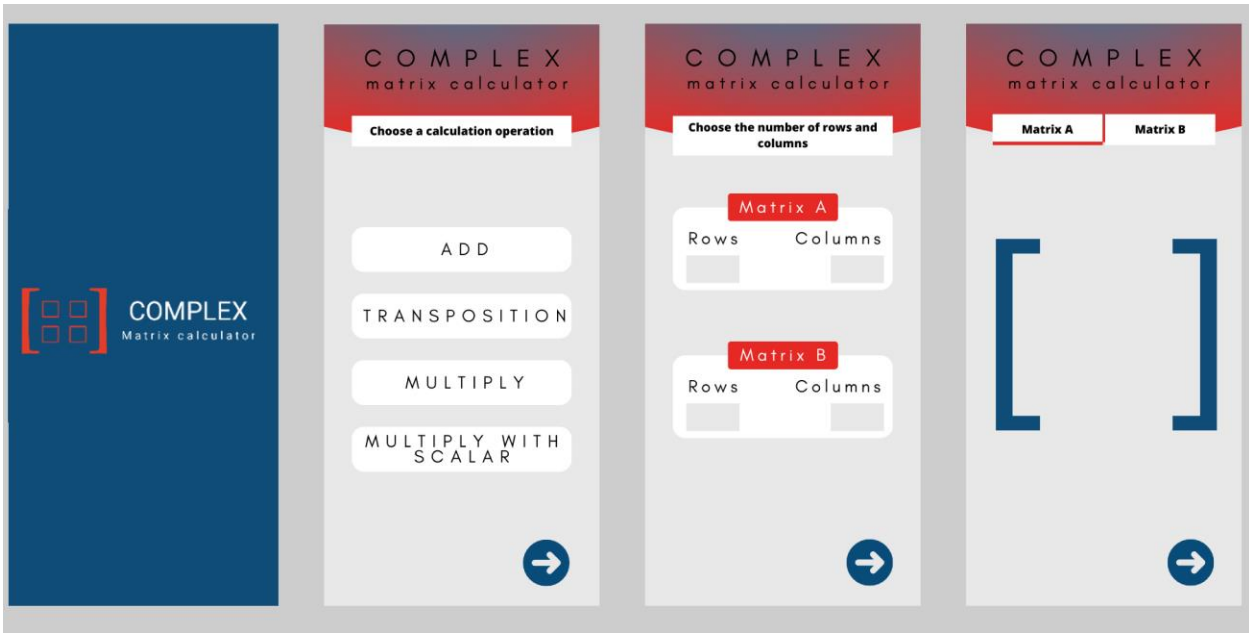
U ovom poglavlju opisani su alati korišteni prilikom implementacije te samo korisničko sučelje aplikacije.

3.1. Korišteni alati

Mobilna aplikacija je razvijena u službenoj integriranoj razvojnoj okolini (IDE), odnosno Android Studiju, koji služi za izradu Android aplikacija. Nudi brojne alate i značajke koje pomažu u poboljšanju produktivnosti u razvoju Android aplikacije [8]. Prototip je rađen u online alatu Figmi, dok je za programski jezik kojim je rađena aplikacija izabran Kotlin. Kotlin je koncizan i siguran programski jezik sličan Javi. Sadrži odlične razvojne alate i okruženje te radi na Java virtualnom stroju. Razvio ga je Ruski programer, radnik JetBrains-a, a ime je dobio po otoku Kotlin, koji se nalazi u blizini Sankt Petersburga. Prema riječima Andreja Breslava (voditelj projektnog tima), može se zaključiti da je Kotlin nastao kako bi se odgovorilo na zahtjeve industrije za „boljim“ programskim jezikom od Jave. Iako njegova sintaksa nije kompatibilna s Java jezikom, Kotlin je dizajniran tako da bude kompatibilan s Java kodom kako bi mnoge tvrtke mogle što lakše prilagoditi kod s Jave na Kotlin [9].

3.2. Korisničko sučelje aplikacije

Bitan korak, prije nego što se započne s programiranjem aplikacije, je izrada dizajna korisničkog sučelja koje je napravljeno pomoću online alata Figma (slika 3.1.). Vrlo je važno da korisničko



Slika 0.1. Izgled korisničkog sučelja u Figmi

sučelje izgleda što jednostavnije kako bi se korisnici znali samoinicijativno služiti samom aplikacijom. Ovaj korak je bitan jer se omogućuje definiranje vizualnog izgleda aplikacije kako bi se na bolji način moglo unaprijed razumjeti korisničko služenje aplikacijom. Osim toga, pruža smjernice za daljnji razvoj implementacije te identificira potencijalne izazove i poboljšanja. Nakon što se pokrene aplikacija, na početnom zaslonu ponuđene su četiri osnovne operacije za rad s kompleksnim matricama. Korisnik treba, dodirivanjem zaslona, izabrati jednu od ponuđenih opcija kako bi mogao prijeći na sljedeći zaslon aplikacije. Nakon što korisnik izabere koju će operaciju koristiti stisne plavu strelicu te pređe na zaslon gdje se upisuju retci i stupci za obje matrice. Nakon toga, ponovno se stisne plava strelica te se pređe na zaslon gdje se upisuju elementi matrica. Pritiskom na plavu strelicu dobije se rješenje. Na zaslonu prije rješenja, prema zadanim postavkama aplikacije najprije se upisuje matrica A, a zatim pritiskom na pravokutnik u kojemu piše *Matrix B* se upisuje B matrica. Ukoliko korisnik pokuša pritisnuti plavu strelicu prije nego što je unijeo sve elemente za obje matrice, dobiti će kratkotrajnu *Toast poruku* koja će ih obavijestiti o tome kako nisu ispunjeni svi uvjeti za izračun.

4. IMPLEMENTACIJA FUNKCIONALNOSTI APLIKACIJE

U ovom poglavlju biti će detaljnije prikazani svi dijelovi aplikacije, način rada aplikacije te dijelovi koda.

4.1. Prikaz programskog rješenja po komponentama

Programsko rješenje sastoji se od nekoliko komponenata koje su opisane u ovom potpoglavlju, a uključuju sljedeće: glavni zaslon, zaslon za unos redaka i stupaca matrice, zaslon za unos elemenata matrice i zaslon za prikaz matrice.

4.1.1. Glavni zaslon

Nakon što korisnik otvori aplikaciju, nailazi na zaslon (slika 3.1.) gdje ima mogućnosti birati između četiri računske operacije. Kada se klikne jedan od ponuđenih gumbova otvara se novi zaslon ovisno o tome što je izabrano. Kako bi se u narednim fragmentima znalo što je korisnik odabrao, pritisak na svaki gumb omogućuje prenošenje podatka s tog fragmenta na sljedeće fragmente. Dio koda je prikazan na slici 3.1.

Slika 4.1. Odabir računskih operacija

```
val addButton = view.findViewById<ImageView>(R.id.add_button)
val transpositionButton = view.findViewById<ImageView>(R.id.transposition_button)
val multiplyButton = view.findViewById<ImageView>(R.id.multiply_button)
val scalarButton = view.findViewById<ImageView>(R.id.multiply_with_scalar_button)


addButton.setOnClickListener {  it: View!
    val chooseRowAndColFragment = AddAndTranspositionMatrixDimensionFragment()
    val bundle = Bundle()
    bundle.putString("operation", "add")

    chooseRowAndColFragment.arguments = bundle

    toAnotherFragment(chooseRowAndColFragment)
}


transpositionButton.setOnClickListener {  it: View!
    val chooseRowAndColFragment = AddAndTranspositionMatrixDimensionFragment()
    val bundle = Bundle()

    bundle.putString("operation", "transposition")
    chooseRowAndColFragment.arguments = bundle

    toAnotherFragment(chooseRowAndColFragment)

}
```

Slika 4.2. Kod gumbova zbrajanja i transponiranja

4.1.2. Unos redaka i stupaca matrice

Nakon što je korisnik odabrao računsku operaciju, nalazi se na sljedećem fragmentu gdje upisuje dimenzije matrica. Ukoliko je odabrana operacija zbrajanja otvorit će se fragment s mogućnošću unosa dimenzija samo za jednu matricu jer matrice moraju biti istog tipa da bi se mogle zbrajati (slika 3.1.). Pri odabiru operacije transponiranja također se otvara fragment s mogućnošću

Slika 4.3. Unos dimenzija matrice za zbrajanje, transponiranje i množenje matrice skalarom

unošenja dimenzija samo za jednu matricu jer nam ne trebaju dvije matrice da bi se neka matrica transponirala. Izgled fragmenta je isti kao i za zbrajanje. Također, i za odabir operacije množenja skalarom izgled fragmenta je isti kao i za zbrajanje i transponiranje. Ako je jedna od odabranih opcija bila množenje otvara se fragment s mogućnošću unosa dimenzija za obje matrice (slika

Slika 4.4. Unos dimenzija matrice za množenje matrica

3.1.). Budući da matrice moraju biti ulančane, tj. da broj stupaca matrice *A* mora odgovarati broju

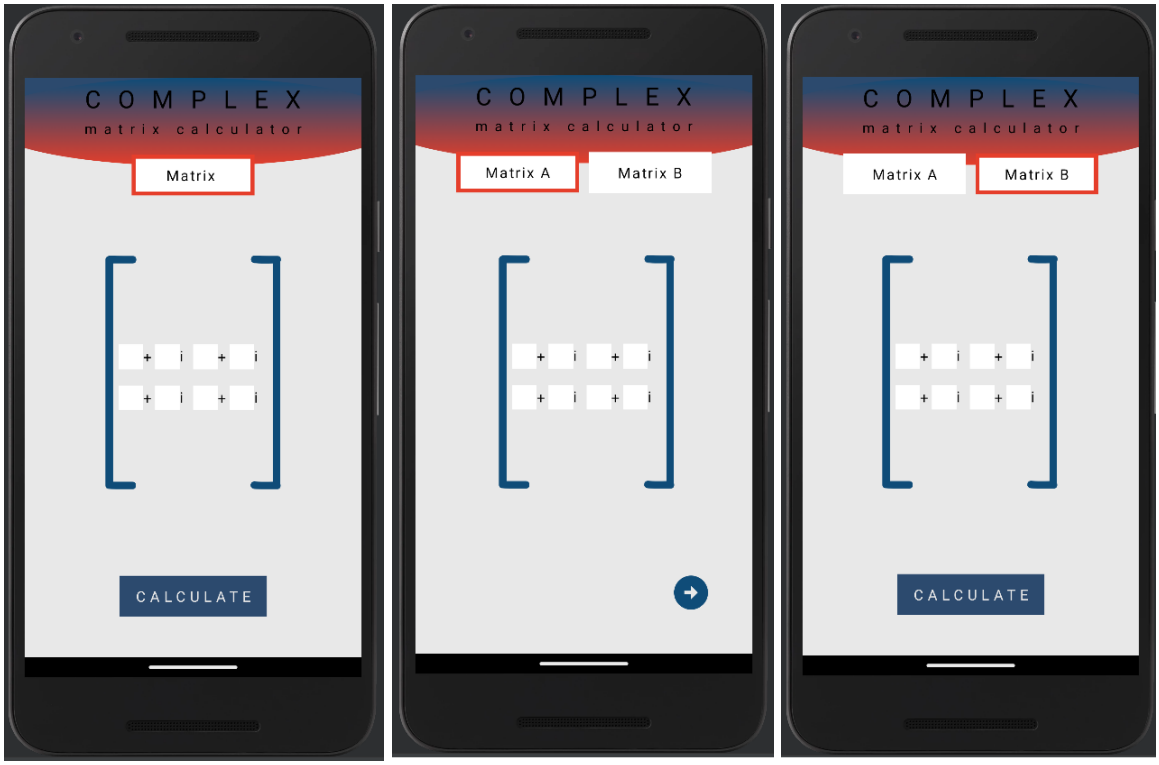
```
} else if(brojStupacaA != brojRedakaB) {
    Toast.makeText(context, text: "Matrice nisu ulancane!", Toast.LENGTH_SHORT).show()
}  else {
```

Slika 4.5. Kod

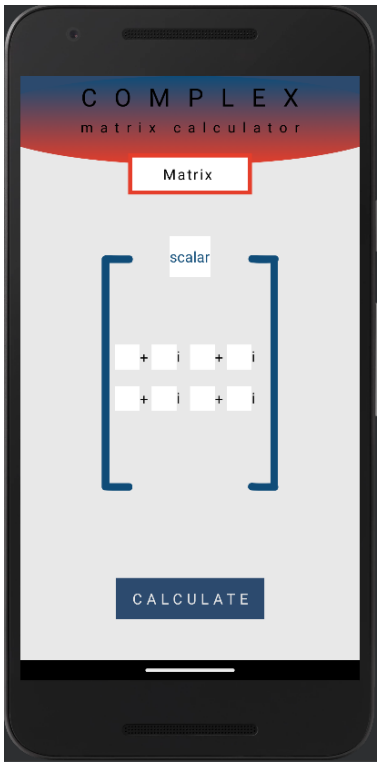
redaka matrice *B*, pomoću *if* grananja i *Toast* poruke je osigurano da se ne unesu krive dimenzije. Kod je prikazan na slici 3.1.

4.1.3. Unos elemenata u matricu

Nakon što je korisnik upisao dimenzije matrice, gumb strelice ga vodi na sljedeći fragment gdje izvršava popunjavanje elemenata kompleksne matrice. Najprije se vrši popunjavanje matrice *A*, a nakon toga, pritiskom na gumb strelice, vrši se i popunjavanje matrice *B*. Izgled fragmenta za unos elemenata dizajnom je jednak za operacije zbrajanja, transponiranja i množenja (slika 3.1.), dok se za operaciju množenja skalarom malo razlikuje jer uz unos elemenata matrice sadrži i *EditText* za unos skalara (slika 3.1.).



Slika 4.6. Unos elemenata matrica (zbrajanje, transponiranje i množenje)



Slika 4.7. Unos elemenata matrica (množenje skalarom)

Unos elemenata se radi tako da se dinamički generiraju *EditText*-ovi unutar *RecyclerView*-a ovisno o unesenom broju redaka i stupaca (slika 3.1.). Unutar *recycler_item*-a se nalaze dva *EditText*-a kako bi se mogli unositi realni i imaginarni dijelovi kompleksnog broja. To u kasnijim dijelovima koda omogućuje lakšu i jednostavniju manipulaciju elementima kompleksne matrice.

```
inner class InputMatrixViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
    private val realPartEditText: EditText = itemView.findViewById(R.id.editTextRealPart)
    private val imaginaryPartEditText: EditText = itemView.findViewById(R.id.editTextImaginaryPart)

    fun bind(rowIndex: Int, colIndex: Int, element: ComplexNumber) {
        realPartEditText.setText("")
        imaginaryPartEditText.setText("")


        realPartEditText.addTextChangedListener(object : TextWatcher {
            override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {}
            override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {}

            override fun afterTextChanged(s: Editable?) {
                val realPartText = s.toString().toIntOrNull() ?: 0
                val imaginaryPartText = imaginaryPartEditText.text.toString().toIntOrNull() ?: 0


                matrix.setElement(rowIndex, colIndex, realPartText, imaginaryPartText)
                printMatrix()
            }
        })
```

Slika 4.8. Dio koda za dinamičko generiranje *EditTextov*-a i spremanje unosa u *matrix*

Nakon što korisnik unese elemente, elementi se spremaju u varijablu *matrix* te se klikom na gumb *CALCULATE* prosljeđuju na zadnji fragment. Ovisno o izabranoj računskoj operaciji s prvog fragmenta pozivaju se funkcije koje su definirane u klasi *ComplexMatrix* te se izvršava određena računska operacija. Dijelovi koda su prikazani na slikama 3.1., 4.10. i 4.11.

```
fun add(other: ComplexMatrix): ComplexMatrix {
    require(rows == other.rows && cols == other.cols)

    val result = ComplexMatrix(rows, cols)
    for (i in 0 ≤ until < rows) {
        for (j in 0 ≤ until < cols) {
            val sum = matrix[i][j] + other.matrix[i][j]
            result.setElement(i, j, sum.real, sum.imaginary)
        }
    }
    return result
}

fun transpose(): ComplexMatrix {
    val transposedMatrix = ComplexMatrix(cols, rows)
    for (i in 0 ≤ until < rows) {
        for (j in 0 ≤ until < cols) {
            val complexNumber = matrix[i][j]
            transposedMatrix.setElement(j, i, complexNumber.real, complexNumber.imaginary)
        }
    }
    return transposedMatrix
}
```

Slika 4.9. Kod za zbrajanje i transponiranje matrica

```
fun multiply(other: ComplexMatrix): ComplexMatrix {
    require(cols == other.rows)

    val result = ComplexMatrix(rows, other.cols)
    for (i in 0 ≤ until < rows) {
        for (j in 0 ≤ until < other.cols) {
            var sum = ComplexNumber( real: 0, imaginary: 0)
            for (k in 0 ≤ until < cols) {
                sum += matrix[i][k] * other.matrix[k][j]
                Log.d( tag: "MatrixMultiplication", msg: "[$i][$k] * [${k}][$j] = ${matrix[i][k]} * ${other.matrix[k][j]} = $sum")
            }
            result.setElement(i, j, sum.real, sum.imaginary)
            Log.d( tag: "MatrixMultiplication", msg: "Result [$i][$j] = ${result.getElement(i, j)}")
        }
    }
    return result
}

fun multiplyWithScalar(scalar: ComplexNumber): ComplexMatrix {
    val result = ComplexMatrix(rows, cols)
    for (i in 0 ≤ until < rows) {
        for (j in 0 ≤ until < cols) {
            val product = matrix[i][j]*scalar
            result.setElement(i, j, product.real, product.imaginary)
        }
    }
    return result
}
```

Slika 4.10. Kod za množenje i množenje skalarom

```
when (operation) {
    "add" -> {
        val newMatrix = matrix.add(matrix2)
        brojRedaka = newMatrix.rows
        brojStupaca = newMatrix.cols
        matrix = newMatrix
    }
    "transposition" -> {
        val newMatrix = matrix.transpose()
        brojRedaka = newMatrix.rows
        brojStupaca = newMatrix.cols
        matrix = newMatrix
    }
    "multiply" -> {
        val newMatrix = matrix.multiply(matrix2)
        brojRedaka = newMatrix.rows
        brojStupaca = newMatrix.cols
        matrix = newMatrix
    }
    "multiplyWithScalar" -> {
        val newMatrix = matrix.multiplyWithScalar(scalar)
        brojRedaka = newMatrix.rows
        brojStupaca = newMatrix.cols
        matrix = newMatrix
    }
    else -> {
        Toast.makeText(context, text: "Nije odabrana ni jedna operacija", Toast.LENGTH_SHORT).show()
    }
}
```

Slika 4.11. Kod prepoznavanja odabira računske operacije

4.1.4. Prikaz matrice na ekranu

Nakon što korisnik klikne na gumb *CALCULATE*, gumb ga vodi na zadnji fragment gdje se prikazuju rezultati. Prikaz rezultata je omogućen na sličan način kao i unos elemenata, odnosno pomoću *RecyclerView*-a. Unutar njega se nalazi *result_recycler_item* koji se sastoji od jednog *TextView*-a gdje se upisuju realni i imaginarni dio kompleksnog broja. Također, pomoću *MatrixAdapter*-a se dinamički generiraju *TextView*-ovi ovisno o broju redaka i stupaca rezultantne matrice. Dijelove koda pogledajte na slikama 3.1. i 4.13.

```
inner class DisplayMatrixViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
    private val complexNumberTextView: TextView = itemView.findViewById(R.id.textViewComplexNumber)

    fun bind(formattedText: String) {
        complexNumberTextView.text = formattedText
    }
}
```

Slika 4.12. Kod iz *MatrixAdapter*-a

```
val recyclerView = view.findViewById<RecyclerView>(R.id.resultView)
val layoutManager = GridLayoutManager(requireContext(), brojStupaca)
recyclerView.layoutManager = layoutManager

val displayAdapter = MatrixAdapter(requireContext(), brojRedaka, brojStupaca, MatrixAdapter.AdapterMode.DISPLAY)
displayAdapter.setMatrix(matrix)
recyclerView.adapter = displayAdapter
```

Slika 4.13. Kod korištenja adapter za prikaz elemenata na ekran

Nakon što korisnik dobije rezultantnu matricu, pritiskom na gumb *NEW CALCULATION* se može vratiti na početni zaslon te napraviti nove izračune (slika 4.13.).

```
newCalculation.setOnClickListener {  it: View!
    val fragmentTransaction: FragmentTransaction? =
        activity?.supportFragmentManager?.beginTransaction()
    fragmentTransaction?.replace(R.id.mainFragment, ChooseOperationFragment())
    fragmentTransaction?.addToBackStack( name: null)?.commit()
}
```

Slika 4.14. Kod prelaska na početni zaslon

4.2. Primjer rada aplikacije

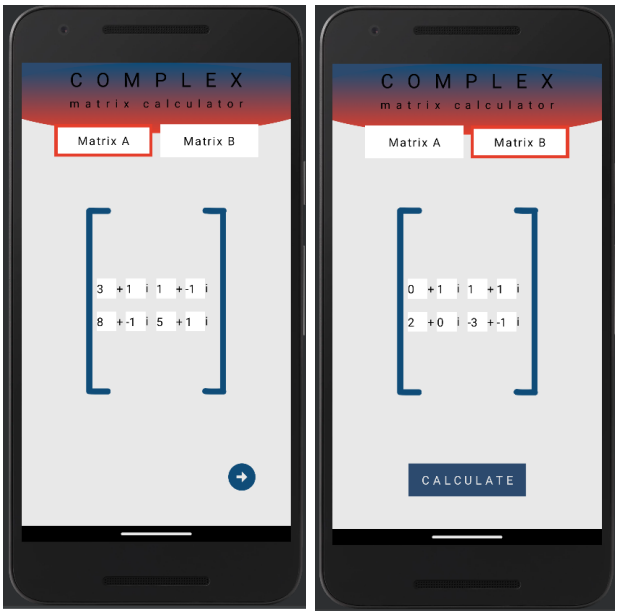
U nastavku će biti prikazani primjeri rada aplikacije za zbrajanje (Slike 4.15 – 4.17.) i transponiranje (Slike 4.18. – 4.20.).

4.2.1. Zbrajanje

1. Korak: Odabir opcije „zbrajanje“
2. Korak: Unos dimenzija matrice

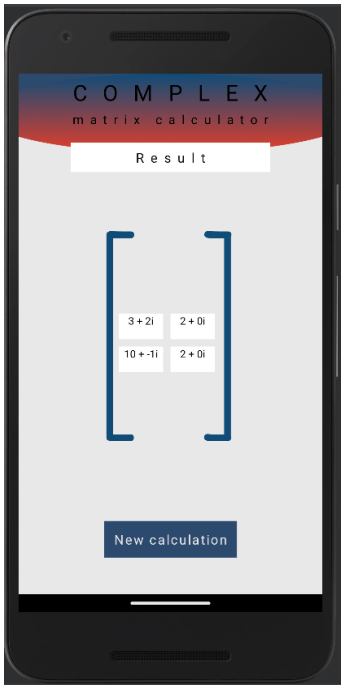
Slika 4.15. Primjer za zbrajanje - 2. korak

3. Korak: Unos elemenata matrice



Slika 4.16. Primjer za zbrajanje - 3. korak

4. Korak: Prikaz rezultata



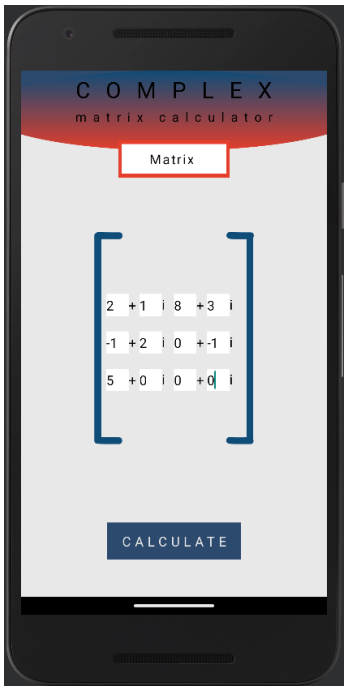
Slika 4.17. Primjer za zbrajanje - 4. korak

4.2.2. Transponiranje

1. Korak: Odabir operacije „transponiranje“
2. Korak: Unos dimenzija matrice

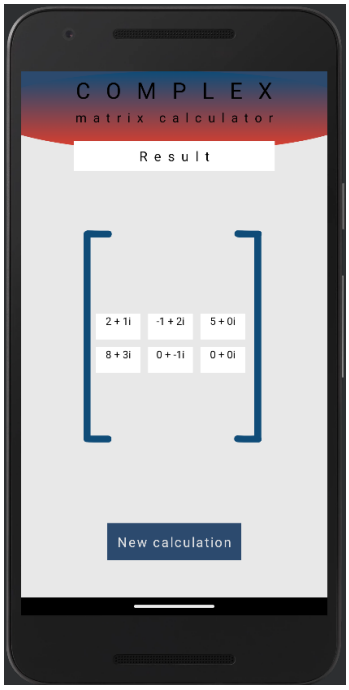
Slika 4.18. Primjer za transponiranje - 2. korak

3. Korak: Unos elemenata matrice



Slika 4.19. Primjer za transponiranje - 3. korak

4. Korak: Prikaz rezultata



Slika 4.20. Primjer za transponiranje - 4. korak

5. ZAKLJUČAK

U ovom radu je izrađena i opisana mobilna aplikacija koja olakšava rješavanje zadataka s kompleksnim matricama kroz funkcionalnosti kao što su zbrajanje, množenje skalarom, množenje i transponiranje. Jednostavno i praktično korisničko sučelje omogućava korisnicima različitih razina znanja, intuitivno korištenje aplikacijom. Takvo korisničko sučelje je jako bitno kako bi aplikacija mogla biti pristupačnija i privlačnija širem krugu korisnika. Izrada aplikacije provedena je u Android studiju u modernom programskom jeziku Kotlinu, koji se razvio kao alternativa Javi.

Dakle, mobilne aplikacije imaju značajnu ulogu u unapređenju matematičkog učenja i istraživanja jer se tako može doprinijeti promicanju matematičke pismenosti te olakšati rad korisnicima koji se susreću s problemima kompleksnih matrica.

LITERATURA

[1] N. Elezović, Aglić Andrea, *Linearna algebra*. Zagreb: Element 2001.

[2] F. Nikšić, Svijet matrica, Playmath, sv. I, izd. 2, str. 11–16, 2003.

[3] D. Bakić, Linearna algebra, školska knjiga, Zagreb 2008.

[4] „Application of Matrices in Science, Commerce and Social Science Fields“, *Vedantu*. https://www.vedantu.com/maths/application-of-matrices (pristupljeno 05. ožujak 2023.).

[5] L. Hardesty, „Explained: Matrices“, *MIT News*, 12. lipanj 2013. https://news.mit.edu/2013/explained-matrices-1206 (pristupljeno 05. ožujak 2023.).

[6] „kompleksni brojevi“, *Hrvatska enciklopedija, mrežno izdanje*, Leksikografski zavod Miroslav Krleža, 2021. https://enciklopedija.hr/natuknica.aspx?ID=32652 (pristupljeno 20. travanj 2023.).

[7] „Iz povijesti matematike“, *Hrvatski matematički elektronski časopis*. http://e.math.hr/povmat/pov1.html (pristupljeno 15. svibanj 2023.).

[8] D. Jukić, R. Scitovski, „Kompleksni brojevi“, u *Matematika I*, Osijek: 2000.

[9] „Android Developers“, *Meet Android Studio*. https://developer.android.com/studio/intro (pristupljeno 18. lipanj 2023.).

[10] M. Arsić, „Startit.rs“, *Uvod u Kotlin — Koje su prednosti novog zvaničnog jezika za razvoj Android aplikacija?*, 16. lipanj 2017. https://startit.rs/uvod-u-kotlin-koje-su-prednosti-zvanicnog-jezika-za-razvoj-android-aplikacija/ (pristupljeno 18. lipanj 2023.).

[11] „Matrice“. https://element.hr/wp-content/uploads/2020/06/unutra-13520.pdf (pristupljeno 30. lipanj 2023.).

SAŽETAK

U ovom radu se izrađivala mobilna aplikacija za rad s kompleksnim matricama. Osim toga, opisana je i teorija matrica te su istražena postojeća rješenja na tu temu. Nakon teorijskog dijela objašnjena je izrada mobilne aplikacije, koji alati su bili korišteni te sam rad aplikacije. Mobilna aplikacija nudi opcije za zbrajanje, množenje skalarom, množenje te transponiranje. Funkcionalnost same aplikacije je ispitana na raznim primjerima kompleksnih matrica.

Ključne riječi: Android, aplikacija, kompleksne matrice, kompleksni brojevi, Kotlin

ABSTRACT

MOBILE APPLICATION FOR WORKING WITH COMPLEX MATRICES

In this paper, a mobile aplicaation for working with complex matrices was created. In addition, the theory of matrices is ddescribed and existing solutions on that topic are investigated. After the theoretical part, the creation of the mobile application offers options for adding, scalar multiplication, multiplication and transposition. The functionality of the application itself was tested on various examples of complex matrices.

Keywords: Android, application, complex matrices, complex numbers, Kotlin