Android aplikacija za numeričko rješavanje integrala

Kojčinović, Bojan

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

Sveučilišni studij

ANDROID APLIKACIJA ZA NUMERIČKO RJEŠAVANJE INTEGRALA

Završni rad

Bojan Kojčinović

Osijek, 2024.

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

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Bojan Kojčinović
Studij, smjer:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. pristupnika, god.	4914, 21.09.2020.
JMBAG:	0165088257
Mentor:	izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	doc. dr. sc. Anita Katić
Sumentor iz tvrtke:	
Naslov završnog rada:	Android aplikacija za numeričko rješavanje integrala
Znanstvena grana završnog rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rada:	a

Datum prijedloga ocjene završnog rada od strane mentora:	17.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	27.09.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 27.09.2024.

Ime i prezime Pristupnika:	Bojan Kojčinović
Studij:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. Pristupnika, godina upisa:	4914, 21.09.2020.
Turnitin podudaranje [%]:	7

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za numeričko rješavanje integrala**

izrađen pod vodstvom mentora izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora doc. dr. sc. Anita Katić

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 pristupnika:

SADRŽAJ

1. UVOD

Numeričko rješavanje integrala je postupak koji se u praksi koristi kad nije moguće primijeniti Newton-Leibnizovu formulu u svrhu dobivanja točnog rezultata ili je postupak rješavanja prezahtjevan. Kao rezultat numeričkog rješavanja integrala ne dobiva se točna numerička vrijednost određenog integrala, osim u nekim rijetkim slučajevima, već vrijednost koja ju aproksimira uz određenu pogrešku.

Ovaj završni rad sastoji se iz dva dijela, prvi dio je teorija numeričke integracije, a drugi izrada Android aplikacije za numeričko rješavanje integrala.

Prvi dio opisuje teoriju iza numeričke integracije i metode koje predstavljaju osnovu koju je potrebno poznavati za izradu i rukovanje aplikacijom.

Drugi dio opisuje samu izradu aplikacije, pri čemu je korišteno razvojno okruženje *Android Studio* i programski jezik *Kotlin*. Uz to, bio je potreban *library* za grafički prikaz funkcija, a za to je korišten *GraphView library*. Ovdje će se primjenjivati teorija objašnjena u prethodnom dijelu, a cijeli proces će biti objašnjen u obliku teksta, slika i primjera.

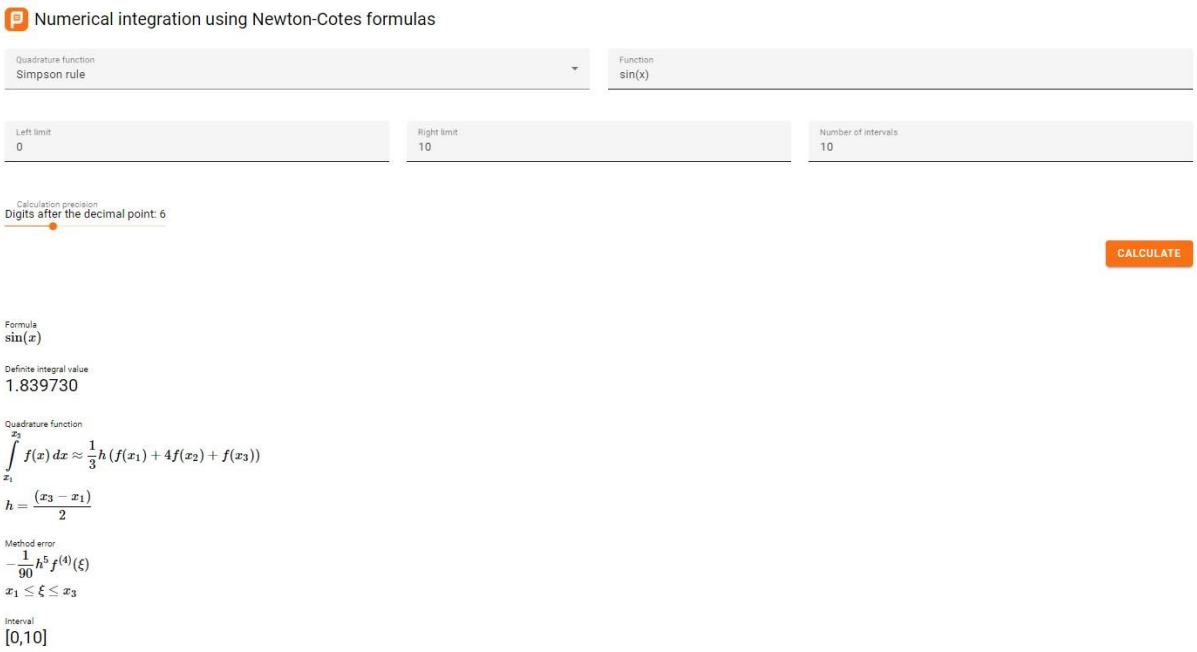
1.1. Zadatak završnog rada

Potrebno je izraditi *Android* aplikaciju koja će primjenom trapezne i Simpsonove formule izračunati približnu vrijednost integrala zadane funkcije, dobivene vrijednosti usporediti s točnom vrijednosti integrala te grafički ju prikazati. Također je potrebno opisati postupak izrade aplikacije, testirati ju na primjerima i prikazati rezultate.

2. PREGLED PODRUČJA TEME

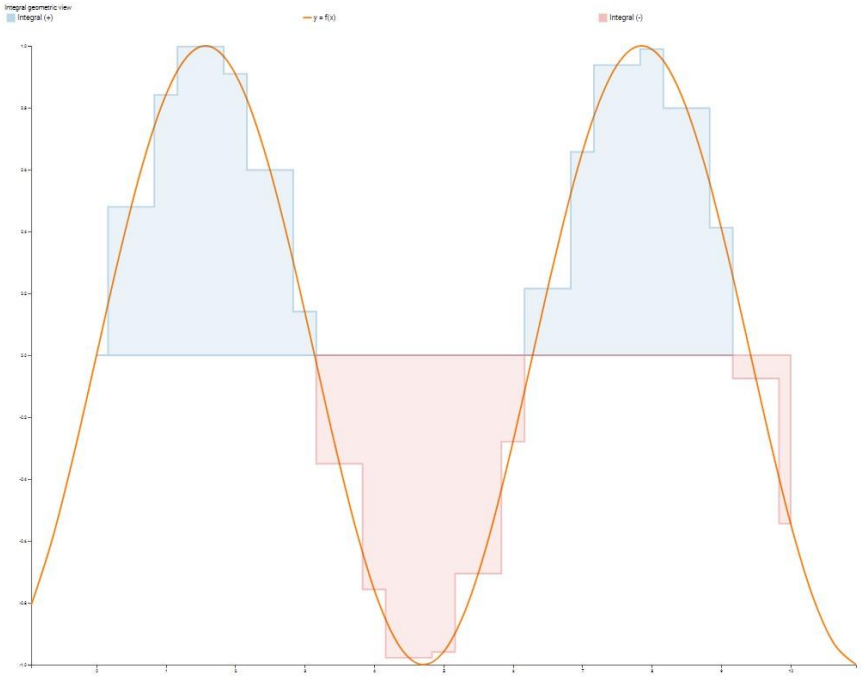
Pretragom uz pomoć internet preglednika, na temu numeričke integracije, dobivamo brojne rezultate. Među njima su stranice koje sadrže svu bitnu teoriju iza numeričke integracije i njenih metoda, različiti edukativni videozapisi na istu temu te kalkulatori, koji rade na sličan princip kao i aplikacija koja je zadatak ovog završnog rada. Iz tog razloga sam kao primjer odabrao dvije web stranice sa takvim kalkulatorima.

Prvi kalkulator nalazi se na poveznici „https://planetcalc.com/5494/“. Ovaj kalkulator nudi mogućnost unosa bilo koje željene funkcije, granica integriranja, broja segmenata kod metoda numeričke integracije, odabira pravila uz pomoć kojeg će se izvršiti numerička integracija, rezultat integrala dobiven odabranim pravilom, iznos pogreške te grafički prikaz odabrane funkcije i pravila.



Slika 2.1. Primjer unosa podataka u kalkulator

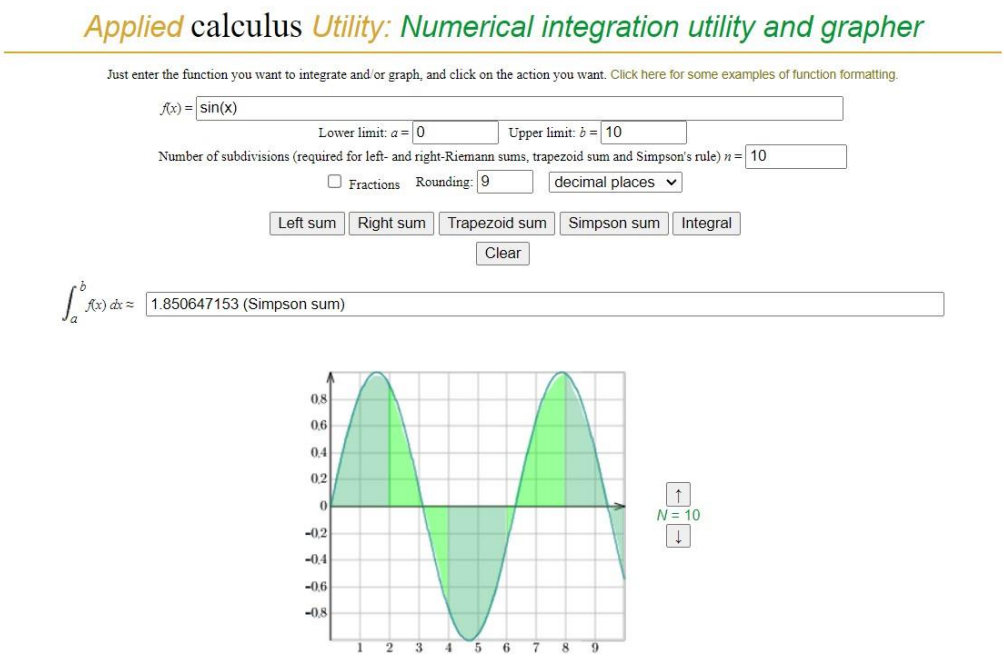
Na slici 2.1. prikazan je izgled dijela stranice gdje se unose podaci za računanje, rezultat integrala, izraz kojim se dobiva rezultat te izraz koji označava pogrešku pri računanju.



Slika 2.2. Grafički prikaz funkcije i odabranog pravila primjenjenog na tu funkciju

Cijela stranica i sam kalkulator su vrlo dobro izrađeni, za odabir pravila za numeričku integraciju ponuđeno je 19 različitih pravila, među kojima su i trapezoidno i Simpsonovo pravilo, koje koristi i aplikacija u ovom završnom radu. Rezultati su brzo vidljvi, bez dužeg čekanja i grafički prikaz je precizan i čitljiv.

Na poveznici: „https://www.zweigmedia.com/RealWorld/integral/integral.html“ nalazi se drugi kalkulator. Ovdje je za razliku od 19 metoda numeričke integracije ponuđeno 5, no svakako je kvalitetno izrađen. Također je moguć unos funkcije, granica integriranja, broja segmenata pri izračunu preko metoda numeričke integracije te postoji i grafički prikaz funkcije i metode integracije. Prikaz ovog kalkulatora je na slici 2.3.



Slika 2.3. Primjer unosa podataka u kalkulator, rješenje i grafički prikaz

3. TEORIJA NUMERIČKE INTEGRACIJE

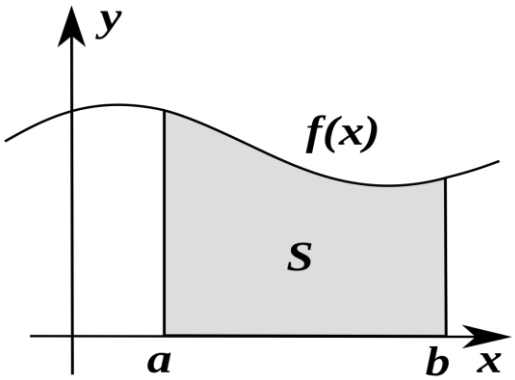
3.1. Određeni integral

Određeni integral je matematički koncept koji predstavlja pripadnost površine ispod krivulje u određenom intervalu. Koristi se za izračunavanje ukupne promjene ili akumulirane vrijednosti funkcije na zadanom intervalu. Označava se kao:

$$\int_{a}^{b} f(x)\,dx \tag{3-1}$$

Funkcija koja se integrira označava se sa f, $[a, b]$ predstavlja interval na kojem se vrši integracija, a dx naznačava da se integrira po varijabli x.

Prema [1], rezultat određenog integrala je broj koji predstavlja površinu ispod krivulje funkcije f na intervalu $[a, b]$:



Sl. 3.1. Površina kao rezultat određenog integrala

Rezultat određenog integrala dobiva se uz pomoć Newton-Leibnizove formule:

$$\int_{a}^{b} f(x)\,dx = F(b) - F(a) \tag{3-2}$$

U ovoj formuli F predstavlja primitivnu funkciju od funkcije f. Bitno je napomenuti da se Newton-Leibnizova formula može primijeniti samo ako je funkcija neprekidna i ako vrijedi:

$$f : [a, b] \rightarrow \mathbb{R} \tag{3-3}$$

3.2. Numerička integracija i njena pravila

U ovome potpoglavlju bit će riječ o matematičkoj teoriji iza numeričke integracije. Prvi dio je vezan za opće informacije o toj metodi, dok se u ostatku poglavlja detaljnije opisuju dva pravila koja su bitna i koja su korištena u izradi aplikacije: trapezno i Simpsonovo pravilo.

3.2.1. Općenito o numeričkoj integraciji

Kao što je u uvodu navedeno, ne može se uvijek primijeniti Newton-Leibnizova formula za računanje određenog integrala, a razlozi su sljedeći:

1. podintegralna funkcija poznata je samo u nekim određenim točkama, ne u svim, npr. u točkama dobivenim metodom uzorkovanja,
2. moguće je naći primitivnu funkciju podintegralne funkcije, ali je njeno računanje prezahtjevno pa je puno jednostavnije naći numeričku aproksimaciju,
3. nije moguće naći primitivnu funkciju, npr. $f(x) = \exp(-x^2)$, primitivna funkcija ove funkcije ne može biti zapisana u elementarnom obliku

Numerička integracija temelji se na ideji da se integral izračuna pomoću vrijednosti funkcije *f* na diskretnom skupu točaka umjesto kontinuirane funkcije.

Opća integracijska formula sljedećeg je oblika:

$$I(f) = I_m(f) + E_m(f) \tag{3-4}$$

Pri tome, $m + 1$ predstavlja broj točaka koje se koriste, $I_m(f)$ je aproksimacija integrala koja se dobiva, a $E_m(f)$ označava grešku koja se javlja. Ove metode za aproksimaciju integrala funkcija jedne varijable na jednodimenzionalnom području često se nazivaju kvadraturnim formulama zbog tumačenja integrala kao površine ispod krivulje.

3.2.2. Trapezna formula

Za aproksimativni izračun određenog integrala, prema (3-1), uz pomoć trapezne formule, funkcija f mora biti neprekinuta na intervalu $[a, b]$. Područje integriranja se dijeli na n jednakih dijelova i odabire se korak izračunavanja h.

$$h = \frac{b-a}{n} \tag{3-5}$$

Prema [2,3], ako vrijedi da su $x_i = x_0 + ih$ $(x_0 = a, x_n = b, i = 0, 1, 2, \dots, n)$ apscise djelišta, a $y_i = f(x_i)$ pripadne vrijednosti podintegralne funkcije $y = f(x)$, tada vrijedi sljedeća formula:

$$\int_a^b f(x)dx \approx h(\frac{y_0 + y_n}{2} + y_1 + y_2 + \cdots + y_{n-1}) \tag{3-6}$$

Ista formula može se zapisati i kao:

$$\int_a^b f(x)dx \approx \frac{h}{2}(\, y_0 + 2y_1 + 2y_2 + \cdots + 2y_{n-1} + y_n) \tag{3-7}$$

s apsolutnom pogreškom:

$$R_n \leq \frac{h^2}{12}(b-a) \cdot M_2 \tag{3-8}$$

gdje M_2 predstavlja:

$$M_2 = max\,|f''(x)|\ \ za\ a \leq x \leq b \tag{3-9}$$

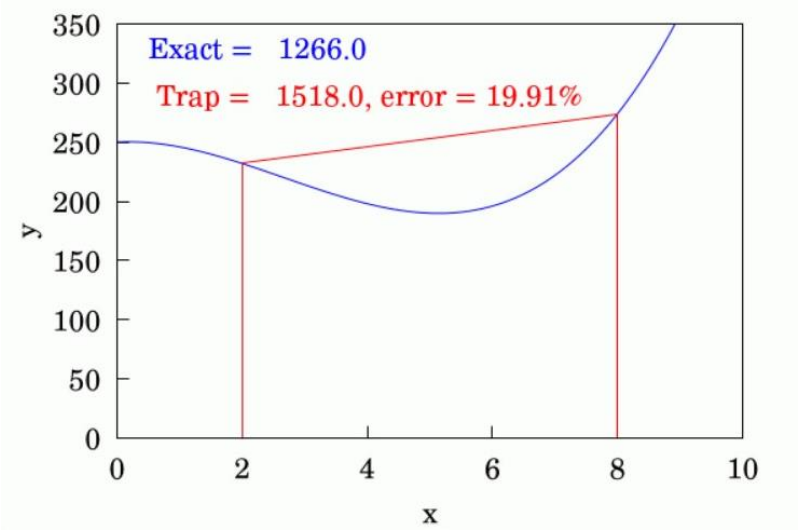
Točnost računa se označava sa grčkim slovom ε (*epsilon*) te ako se želi postići određena točnost, računa se korak izračunavanja h iz sljedeće nejednadžbe:

$$h^2 \leq \frac{12\varepsilon}{(b-a)M_2} \tag{3-10}$$

Dobivena vrijednost h zaokružuje se na manju vrijednost, kako bi broj podintervala n bio cijeli broj, a on se računa sljedećim izrazom:

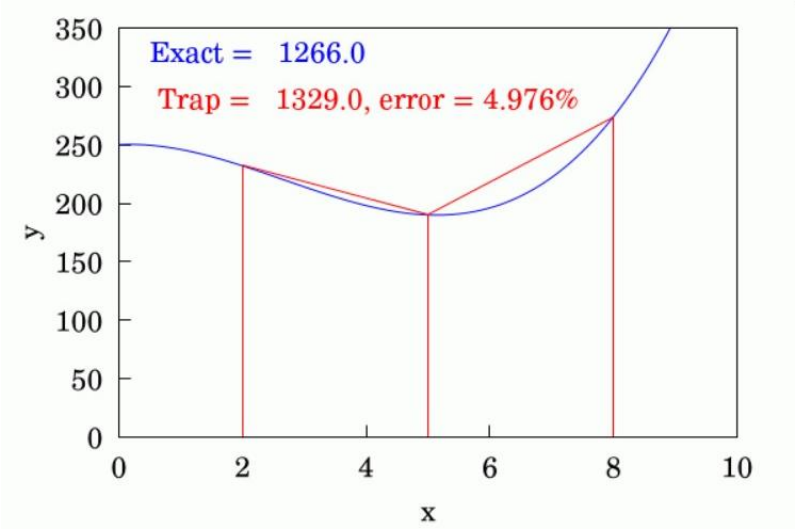
$$\frac{b-a}{h} = n \tag{3-11}$$

Smanjenjem koraka izračunavanja h, područje integriranja [a, b] se dijeli na više podintervala. Primjena trapezne formule nad svakim od tih podintervala omogućuje veću preciznost, tj. aproksimativna vrijednost se približava stvarnoj vrijednosti određenog integrala. Prikaz toga je na sljedećim slikama.

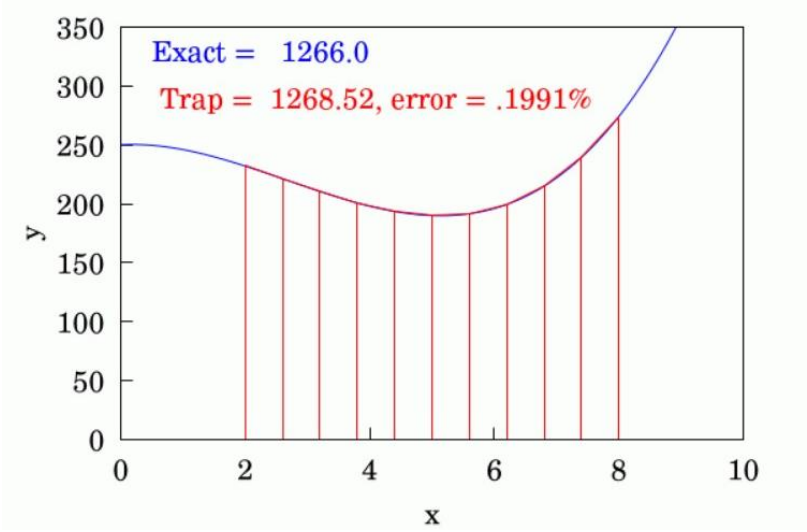


Sl. 3.2. Primjena trapezne formule bez podjele na podintervale

Na slici 3.2. interval integriranja nije podijeljen na manje podintervale te je trapezna formula primjenjena samo jednom. To je rezultiralo velikom pogreškom pri izračunu.



Sl. 3.3. Primjena trapezne formule s dva podintervala



Sl. 3.4. Primjena trapezne formule s deset podintervala

Slike 3.3. i 3.4. prikazuju rezultate višestruke primjene trapezne formule, tj. smanjenja koraka izračunavanja h ili povećanja broja podintervala n na istoj funkciji f. Pogreška se smanjila i time dobiveni rezultati bolje aproksimiraju stvarnu vrijednost. To ukazuje da je odabir koraka izračunavanja, a samim time točnosti ε, bitan faktor.

3.2.3. Simpsonova formula

Prema [2,4], ako je broj podintervala n paran, tada vrijedi i Simpsonova formula koja glasi:

$$\int_a^b f(x)dx \approx \frac{h}{3}[(y_0 + y_n) + 4(y_1 + y_3 + \cdots + y_{n-1}) + 2(y_2 + y_4 + \cdots + y_{n-2})] \tag{3-12}$$

s apsolutnom pogreškom:

$$R_n \leq \frac{h^4}{180}(b-a)M_4 \tag{3-13}$$

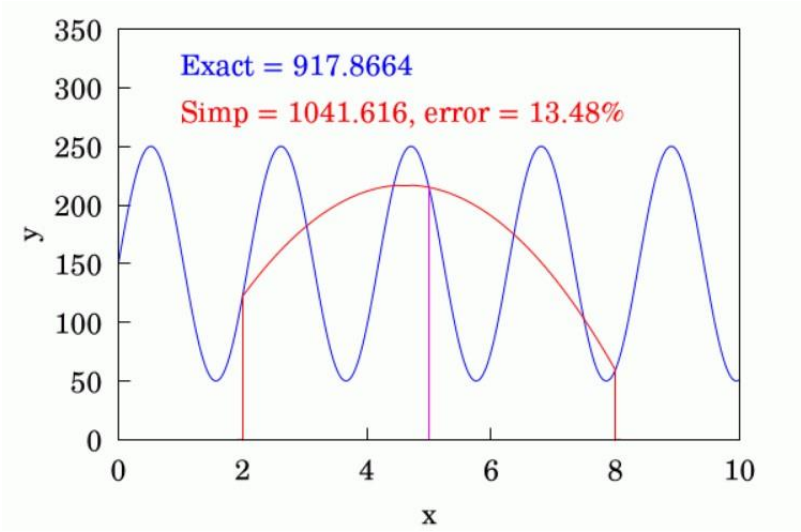
gdje M_4 predstavlja:

$$M_4 = max|f^{IV}(x)| \ \ za\ a \leq x \leq b \tag{3-14}$$

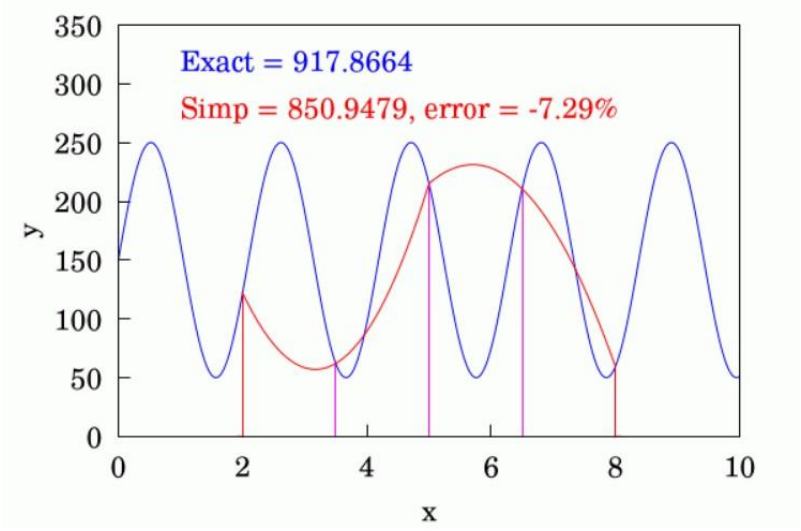
Nakon što se odredi tražena točnost ε, korak računanja h se računa prema izrazu:

$$\frac{h^4}{180}(b-a)M_4 \leq \varepsilon \tag{3-15}$$

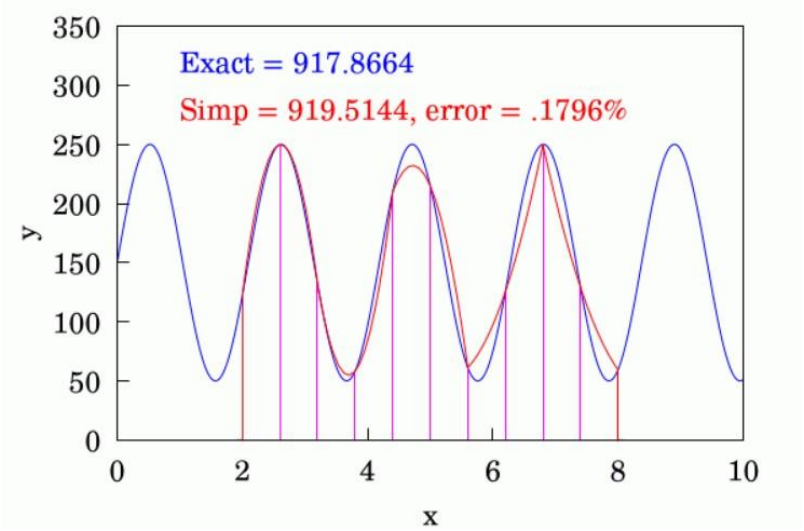
Kao i kod trapezne formule i ovdje je bitno odrediti pogodan korak računanja:



Sl. 3.5. Primjena Simpsonove formule s 2 podintervala



Sl. 3.6. Primjena Simpsonove formule s 4 podintervala



Sl. 3.7. Primjena Simpsonove formule s 10 podintervala

Simpsonova formula, grafički gledano, radi na principu aproksimiranja glavne funkcije parabolama na području svaka dva susjedna podintervala. Preciznije, ako su $[a, b], [b, c]$ promatrani podintervali, tada su bitne sljedeće vrijednosti funkcije f: $f(a), f(b), f(c)$. Glavna funkcija će tada na intervalu $[a, c]$ biti aproksimirana parabolom koja prolazi kroz te tri vrijednosti. Na kraju, kad se površine ispod svih parabola zbroje, dobije se aproksimirani rezultat. Kao i kod trapezne formule, manjim korakom izračuna h, tj. većim brojem podintervala n osigurava se precizniji rezultat, kao što je vidljivo kad se usporede slike 3.5., 3.6. i 3.7.

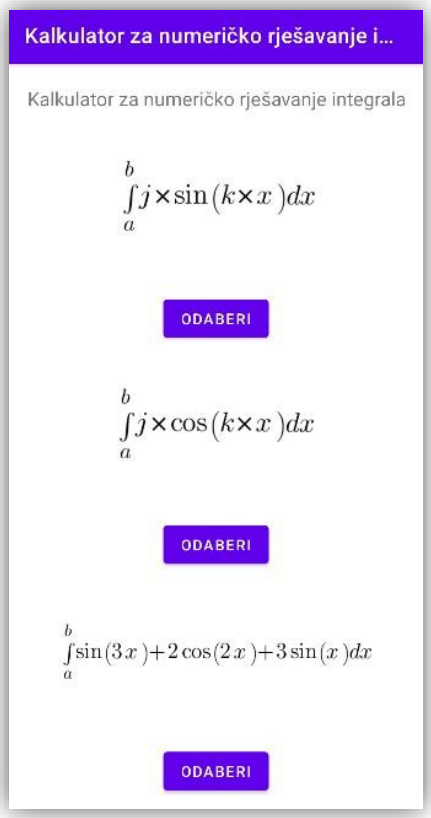
4. Izrada aplikacije

U ovome poglavlju opisuje se cijeli postupak izrade aplikacije u razvojnom okruženju Android Studio. Radi preglednosti, podijeljeno je na tri dijela: postupak izrade *Layout*-a aplikacije, implementiranje rješenja prema matematičkoj teoriji te na testiranje točnosti rješenja.

4.1. Layout (izgled) aplikacije

Kod izrade aplikacije prvi korak je bio napraviti layout aplikacije. Kao što je ranije navedeno, za samu izradu aplikacije koristi se razvojno okruženje Android Studio.

Aplikacija je osmišljena tako da se sastoji od jednog *Activity*-a u kojem se će se mijenjati različiti fragmenti uz pomoć *FragmentContainerView*-a. To znači da je da za svaki fragment potrebno napraviti *xml* datoteku koja definira izgled fragmenta.



Slika 4.1. Izgled glavnog fragmenta

Pri pokretanju aplikacije otvara se glavni fragment (Slika 4.1.), gdje korisnik može odabrati za koju od ponuđenih funkcija želi računati integral. Na slici se može vidjeti kako se glavni fragment sastoji od jednog *TextView* elementa, tri *ImageView* elementa te tri *Button* elementa.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/frameLayout"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainFragment">
9
10    <androidx.constraintlayout.widget.ConstraintLayout
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"
13         tools:context=".MainActivity">
14
15         <TextView
16             android:id="@+id/textViewTitle"
17             android:layout_width="wrap_content"
18             android:layout_height="wrap_content"
19             android:layout_marginTop="20dp"
20             android:text="@string/app_name"
21             android:textSize="18sp"
22             app:layout_constraintEnd_toEndOf="parent"
23             app:layout_constraintStart_toStartOf="parent"
24             app:layout_constraintTop_toTopOf="parent" />
```

Slika 4.2. Kod vezan za *TextView* element

```
26          <ImageView
27                android:id="@+id/imageSine"
28                android:layout_width="198dp"
29                android:layout_height="126dp"
30                android:layout_marginTop="50dp"
31                android:layout_marginBottom="10dp"
32                android:contentDescription="@string/ImageSineDesc"
33                app:layout_constraintBottom_toTopOf="@+id/ChoiceButton1"
34                app:layout_constraintEnd_toEndOf="parent"
35                app:layout_constraintStart_toStartOf="parent"
36                app:layout_constraintTop_toTopOf="parent"
37                app:srcCompat="@drawable/sine" />
38
39           <Button
40                android:id="@+id/ChoiceButton1"
41                android:layout_width="wrap_content"
42                android:layout_height="wrap_content"
43                android:text="@string/ChoiceButtonText"
44                app:layout_constraintBottom_toTopOf="@+id/imageCosine"
45                app:layout_constraintEnd_toEndOf="parent"
46                app:layout_constraintHorizontal_bias="0.498"
47                app:layout_constraintStart_toStartOf="parent"
48                app:layout_constraintTop_toBottomOf="@+id/imageSine"
49                tools:ignore="DuplicateSpeakableTextCheck" />
```

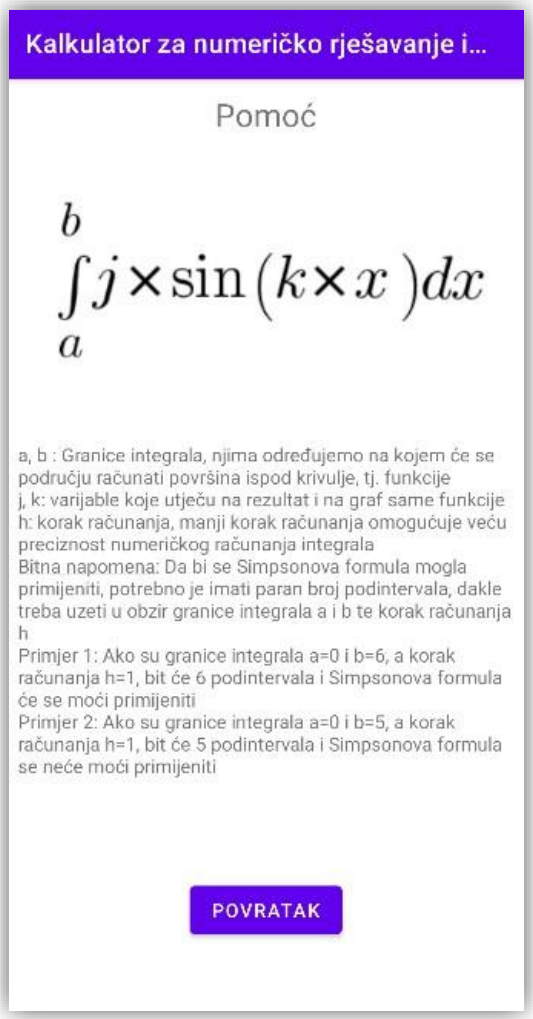
Slika 4.3. Kod vezan za *ImageView* i *Button* elemente

Na slikama 4.2. i 4.3. vidljiv je dio *xml* koda za glavni fragment koji se odnosi na *TextView*, jedan *ImageView* i jedan *Button* element.

Nadalje, za svaku od funkcija postoji poseban fragment na kojemu će se računati integrali istih, iz razloga što im je izgled i kod veoma sličan, bit će prikazan izgled i kod tih fragmenata samo za sinusnu funkciju.

Slika 4.4. Izgled fragmenta za računanje

Na slici 4.4. se vidi izgled fragmenta na kojem će se računati određeni integral sinusne funkcije. U layoutu se nalazi pet *EditText* elemenata koji se koriste za unos granica a i b, varijabli j i k te koraka računanja h. Pritiskom na *Button* Izračunaj ispod će se pojaviti rezultati numeričkih metoda rješavanja integrala te točno rješenje. Funkcija te primjena trapezoidne i Simpsonove formule grafički će se prikazati na elementu *GraphView* koji se nalazi na dnu fragmenta. Osim toga tu se nalazi i *Button* Pomoć koji vodi do fragmenta koji sadrži kratke upute za korištenje aplikacije (Slika 4.5.) te *Button* Povratak koji korisnika vraća na glavni izbornik.



Slika 4.5. Izgled fragmenta s uputama

4.2. Kod i funkcionalnost aplikacije

Za početak je bilo potrebno omogućiti mijenjanje fragmenata. Kada se otvori aplikacija prikazan je glavni fragment (Slika 4.1.), a dodirom na jedan od tri *Button* elementa, prelazimo na jedan od tri fragmenta za računanje, ovisno kojeg korisnik odabere.

Na slici 4.6. prikazan je način promjene fragmenata, na *choiceButton1*, koji predstavlja *Button* za odabir računanja sa sinusnom funkcijom, postavljen je *OnClickListener*. Dodirom ili klikom na taj element izvest će se gore napisani kod. Inicijaliziran je novi fragment koji predstavlja novi

fragment za računanje sa sinusnom funkcijom te se izvršava transakcija, tako da on zamijeni trenutni fragment koji se nalazi u *fragmentContainerView* elementu.

```
23 choiceButton1.setOnClickListener {
24     val fragment = SineFragment()
25
26     val fragmentTransaction: FragmentTransaction? =
27         activity?.supportFragmentManager?.beginTransaction()
28     fragmentTransaction?.replace(R.id.fragmentContainerView, fragment)
29     fragmentTransaction?.commit()
30 }
```

Slika 4.6. Kod za zamjenu fragmenata

Na isti način se prelazi u fragment za računanje sa kosinusnom funkcijom, fragment za računanje sa složenom funkcijom te u pomoćne fragmente iz bilo kojega iz prethodno navedena tri fragmenta.

Sljedeći korak je bio spremiti podatke koje korisnik unosi u *EditText* elemente koji se nalaze u fragmentu za računanje (Slika 4.4.) i omogućiti njihovo korištenje za daljnje računanje. Kod za to je vidljiv na slici 4.7. ispod. Sav kod od ovog dijela nalazi se unutar *OnClickListener*-a koji je postavljen na *Button* Izračunaj u fragmentu za računanje, tako da se izvodi tek nakon klika na taj element.

```
54 val a = view.findViewById<EditText>(R.id.editTextA1).text.toString().toInt()
55 val b = view.findViewById<EditText>(R.id.editTextB1).text.toString().toInt()
56 val c = view.findViewById<EditText>(R.id.editTextJ1).text.toString().toDouble()
57 val d = view.findViewById<EditText>(R.id.editTextK1).text.toString().toDouble()
58 val h = view.findViewById<EditText>(R.id.editTextH1).text.toString().toInt()
```

Slika 4.7. Kod za spremanje korisničkog unosa

```
60 graph.removeAllSeries()
61 val series =
62    LineGraphSeries<DataPoint>()
63 val ahelp =
64    a.toDouble()
65 val bhelp = b.toDouble()
66 var j = ahelp
67 while (j - 1 <= bhelp + 1) {
68     val y = c * sin(d * j)
69     val datapoint = DataPoint(j, y)
70     series.appendData(datapoint, true, 10000)
71     j += 0.1
72 }
73 graph.viewport.setMinX(ahelp - 1)
74 graph.viewport.setScalableY(true)
75 graph.addSeries(series)
```

Slika 4.8. Kod za uzorkovanje i grafički prikaz funkcije

Na slici 4.8. vidljiv je kod kojim se uzorkuje odabrana funkcija u odabranim granicama i grafički se prikazuje. Prva linija koda je potrebna ako će korisnik raditi uzastopna računanja, njome se brišu prethodno prikazane funkcije. Nakon toga potrebno je inicijalizirati novu varijablu tipa *LineGraphSeries* u koju će se spremiti sve točke uzorkovanja. Uz pomoć *while* petlje vrši se uzorkovanje funkcije i dodaju se u varijablu *series* kako bi se funkcija mogla grafički prikazati. Na kraju su 2 linije koda kojima se povećava preglednost grafa te zadnja linija kojom se spremljene točke prikazuju na grafu u obliku krivulje.

```
78 if (h > (b - a)) {
79     view.findViewById<TextView>(R.id.textViewSolutionTrap1).setText("")
80     view.findViewById<TextView>(R.id.textViewSolutionSimpson1)
81         .setText("Pogreška! Nepravilan unos. Provjeri granice i korak
82          integriranja!")
83     view.findViewById<TextView>(R.id.textViewSolutionExact1).setText("")
84 }
```

Slika 4.9. Kod za provjeru granica i koraka integriranja

Na slici 4.9. je zaštita od nepravilnog unosa granica ili koraka integriranja. Ako su granice i korak integriranja u redu, onda se izvodi ostatak koda.

```
86  var trap = 0.0
87  trap += ((c * sin(a.toDouble() * d)) + (c * sin(b.toDouble() * d))) / 2
88  for (i in a + h..b - h step h) {
89      trap += c * sin(i.toDouble() * d)
90  }
91  trap *= h
92  view.findViewById<TextView>(R.id.textViewSolutionTrap1)
93      .setText("Rješenje preko trapezoidne formule: ${trap.toFloat()}")
94  val series2 = LineGraphSeries<DataPoint>()
95  var help = a
96  val num2 = b - a + 1
97  while (help <= b) {
98      val y = c * sin(d * help)
99      val datapoint = DataPoint(help.toDouble(), y)
100     series2.appendData(datapoint, true, num2)
101     help += h
102 }
103 series2.setColor(Color.RED)
104 graph.addSeries(series2)
```

Slika 4.10. Kod za trapeznu formulu i za grafički prikaz iste

Na slici 4.10. vidljiv je kod vezan za trapezno pravilo. Linijama koda 86-93 se dobiva brojčano rješenje integrala koristeći trapeznu formulu, prema (3-6), te prikazuje na *TextView* elementu. Linijama koda 94-104 grafički prikazujemo aproksimaciju originalne funkcije uz pomoć trapeznog pravila.

```
105 if (((b - a) / h) % 2 == 0) {
106     var simpson = 0.0
107     simpson += c * sin(a.toDouble() * d) + c * sin(b.toDouble() * d)
108     for (i in a + h..b - h step 2 * h) {
109         simpson += 4 * (c * sin(i.toDouble() * d))
110     }
111     for (i in a + 2 * h..b - 2 * h step 2 * h) {
112         simpson += 2 * (c * sin(i.toDouble() * d))
113     }
114     simpson *= h.toDouble() / 3
115     view.findViewById<TextView>(R.id.textViewSolutionSimpson1)
116       .setText("Rješenje preko Simpsonove formule: ${simpson.toFloat()}")
```

Slika 4.11. Kod za računanje integrala uz pomoć Simpsonovog pravila

Na slici 4.11. se nalazi kod kojim se dobiva brojčano rješenje integrala koristeći Simpsonovo pravilo, prema (3-12), te prikazuje na *TextView* elementu.

```
117 val series3 = LineGraphSeries<DataPoint>()
118 var help2 = ahelp
119 while (help2.equals(bhelp) == false) {
120     val x1 = help2
121     val y1 = c * sin(d * help2)
122     val x2 = help2 + h
123     val y2 = c * sin(d * (help2 + h))
124     val x3 = help2 + 2 * h
125     val y3 = c * sin(d * (help2 + 2 * h))
126     val denominator = (x1 - x2) * (x1 - x3) * (x2 - x3)
127     val coeff1 =
128         (x3 * (y2 - y1) + x2 * (y1 - y3) + x1 * (y3 - y2)) / denominator
129     val coeff2 =
130         (x3 * x3 * (y1 - y2) + x2 * x2 * (y3 - y1) + x1 * x1 * (y2 - y3))
131         / denominator
132     val coeff3 =
133         (x2 * x3 * (x2 - x3) * y1 + x3 * x1 * (x3 - x1) * y2 + x1 * x2 *
134         (x1 - x2) * y3) / denominator
135
136     var k = help2
137     while (k <= help2 + 2 * h) {
138         val yhelp = coeff1 * (k.pow(2)) + (coeff2 * k) + coeff3
139         val datapoint = DataPoint(k, yhelp)
140         series3.appendData(datapoint, true, 10000)
141         k += 0.01
142     }
143     help2 += 2 * h
144 }
145 series3.setColor(Color.GREEN)
146 graph.addSeries(series3)
```

Slika 4.12. Kod za računanje jednadžbe parabola Simpsonovog pravila i za grafički prikaz istih

Na slici 4.12. prikazan je kod kojim se grafički prikazuje aproksimacija originalne funkcije uz pomoć Simpsonovog pravila. Ovdje je vidljiva *while* petlja koja se može rastaviti na tri dijela.

Prvi dio (linije koda 120-125) predstavlja računanje koordinata tri točke, koje su po *x*-osi međusobno udaljene za korak izračuna *h*. One su potrebne za sljedeći dio.

Za jednadžbu parabole koja prolazi kroz tri prethodno izračunate točke potrebna je formula:

$$y = Ax^2 + Bx + C \tag{4-1}$$

gdje su:

$$A = \frac{(x_3 * (y_2 - y_1) + x_2 * (y_1 - y_3) + x_1 * (y_3 - y_2))}{n} \tag{4-2}$$

$$B = \frac{(x_3^2 * (y_1 - y_2) + x_2^2 * (y_3 - y_1) + x_1^2 * (y_2 - y_3))}{n} \tag{4-3}$$

$$C = \frac{(x_2 * x_3 * (x_2 - x_3) * y_1 + x_3 * x_1 * (x_3 - x_1) * y_2 + x_1 * x_2 * (x_1 - x_2) * y_3)}{n} \tag{4-4}$$

$$n = (x_1 - x_2)(x_1 - x_3)(x_2 - x_3) \tag{4-5}$$

Linije koda 126-134 prikazuju računanje koeficijenata *A*,*B*,*C*.

Linije koda 136-142 predstavljaju dodavanje točaka u varijablu *series3* uz pomoć koje ćemo na kraju prikazati aproksimiranu funkciju. Dakle, ova tri dijela koda ponavljaju se za svaka dva susjedna segmenta, zato se na liniji 143 varijabla *help2* uvećava za 2*h*, sve dok se ne postigne uvjet za kraj *while* petlje. Zadnjom linijom koda prikazuju se sve točke na grafu i time se dobiva aproksimirana funkcija.

```
148 } else {
149     view.findViewById<TextView>(R.id.textViewSolutionSimpson1)
150         .setText("Neparan broj podintervala.")
151 }
```

Slika 4.13. Kod koji se izvršava u slučaju neparnog broja segmenata

U slučaju da je neparan broj segmenata, Simpsonovo pravilo se ne može koristiti pa će tako biti i ispisano u *TextView* elementu za to rješenje.

```
153 val newtonL =
154     ((c * cos(a.toDouble() * d)) / d - (c * cos(b.toDouble() * d)) / d)
155 view.findViewById<TextView>(R.id.textViewSolutionExact1)
156     .setText("Rješenje preko Newton-Leibnizove formule:
157     ${newtonL.toFloat()}")
```

Slika 4.14. Kod za rješenje integrala uz pomoć Newton – Leibnizove formule

Točno rješenje, preko Newton – Leibnizove formule, prema (3-2), računa se uz pomoć koda na slici 4.14..

4.3. Testiranje aplikacije i usporedba s već dostupnom tehnologijom

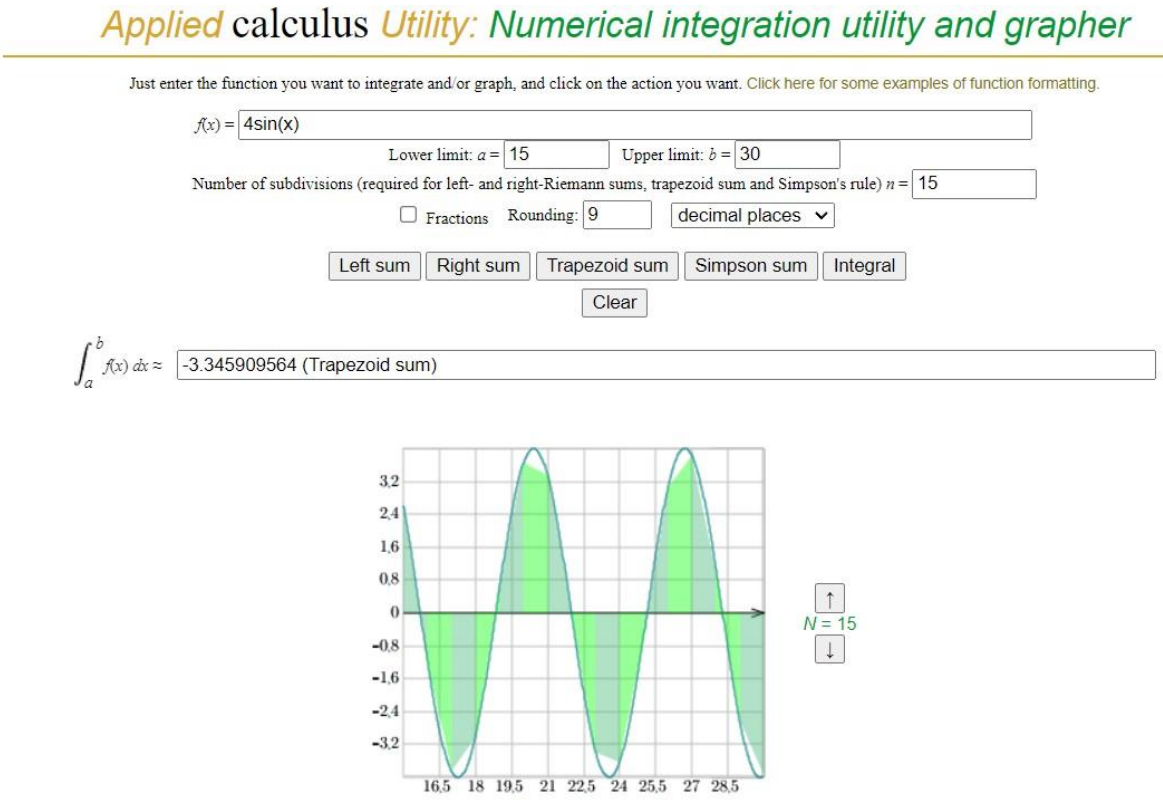
Sinusna funkcija, unos 1:

$a = 0, \quad b = 30,$

$j = 2, \quad k = 1,$

$h = 1$

Slika 4.15. Unos 1 za sinusnu funkciju i rješenje - aplikacija



Slika 4.16. Unos 1 za sinusnu funkciju i rješenje – web stranica

Rezultat Simpsonovog pravila : 1.70214385

Rezultat Newton – Leibnizove formule: 1.6914971

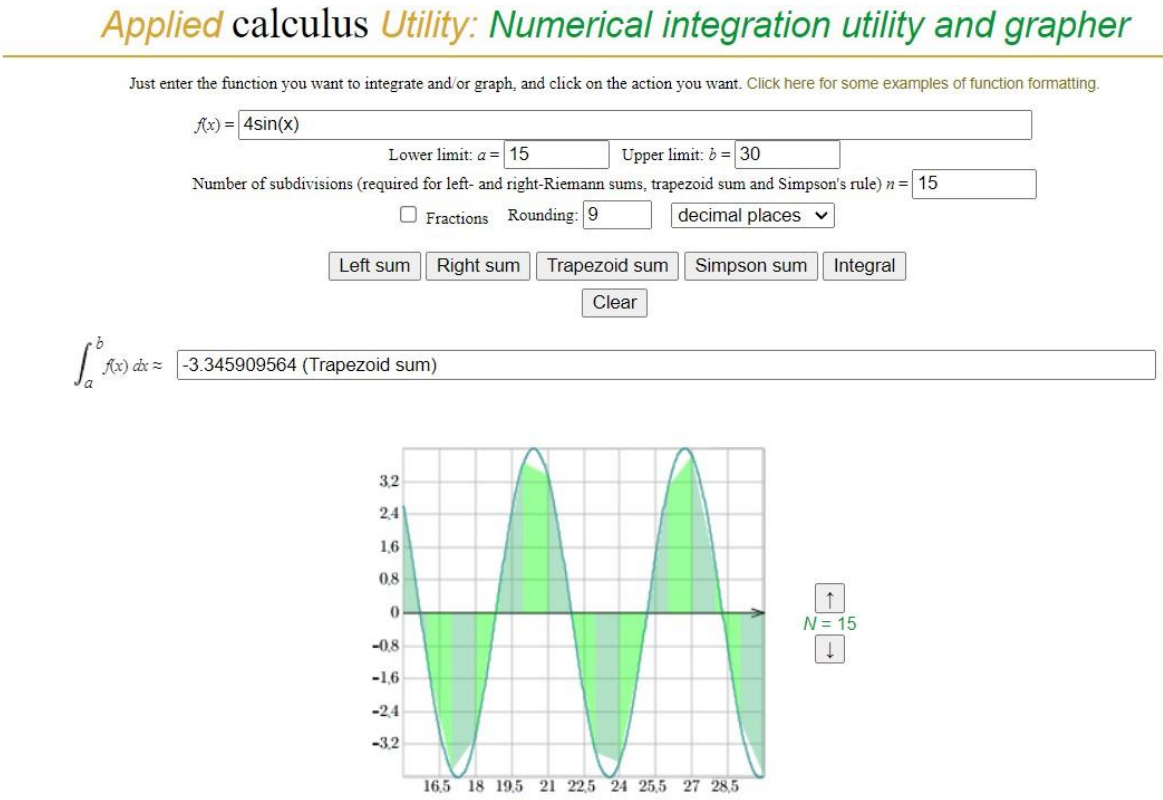
Sinusna funkcija, unos 2:

$a = 15,\ \ b = 30,$

$j = 4,\quad k = 1,$

$h = 1$

Slika 4.17. Unos 2 za sinusnu funkciju i rješenje - aplikacija



Slika 4.18. Unos 2 za sinusnu funkciju i rješenje – web stranica

Rezultat Simpsonovog pravila: neparan broj podintervala

Rezultat Newton – Leibnizove formule: -3.655757451

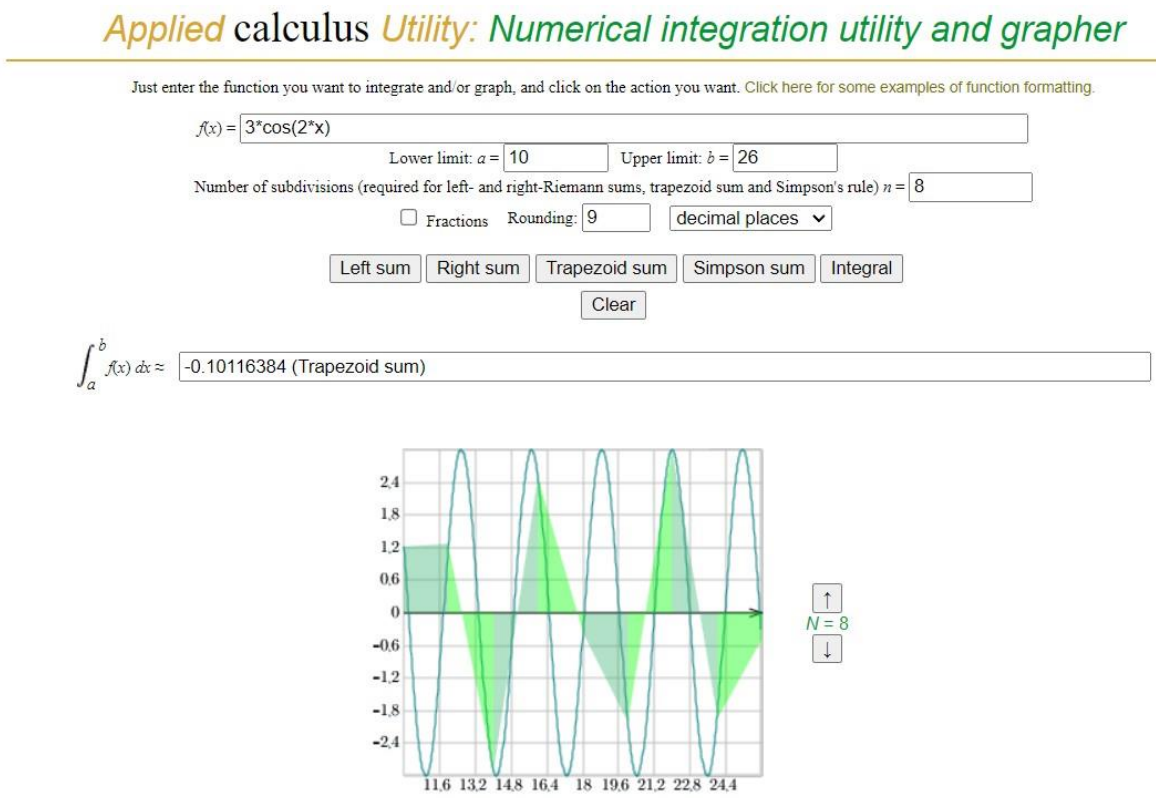
Kosinusna funkcija, unos 1:

$a = 10, \quad b = 26,$

$j = 3, \quad k = 2,$

$h = 2$

Slika 4.19. Unos 1 za kosinusnu funkciju i rješenje - aplikacija



Slika 4.20. Unos 1 za kosinusnu funkciju i rješenje – web stranica

Rezultat Simpsonovog pravila: -0.262162693

Rezultat Newton – Leibnizove formule: 0.110523512

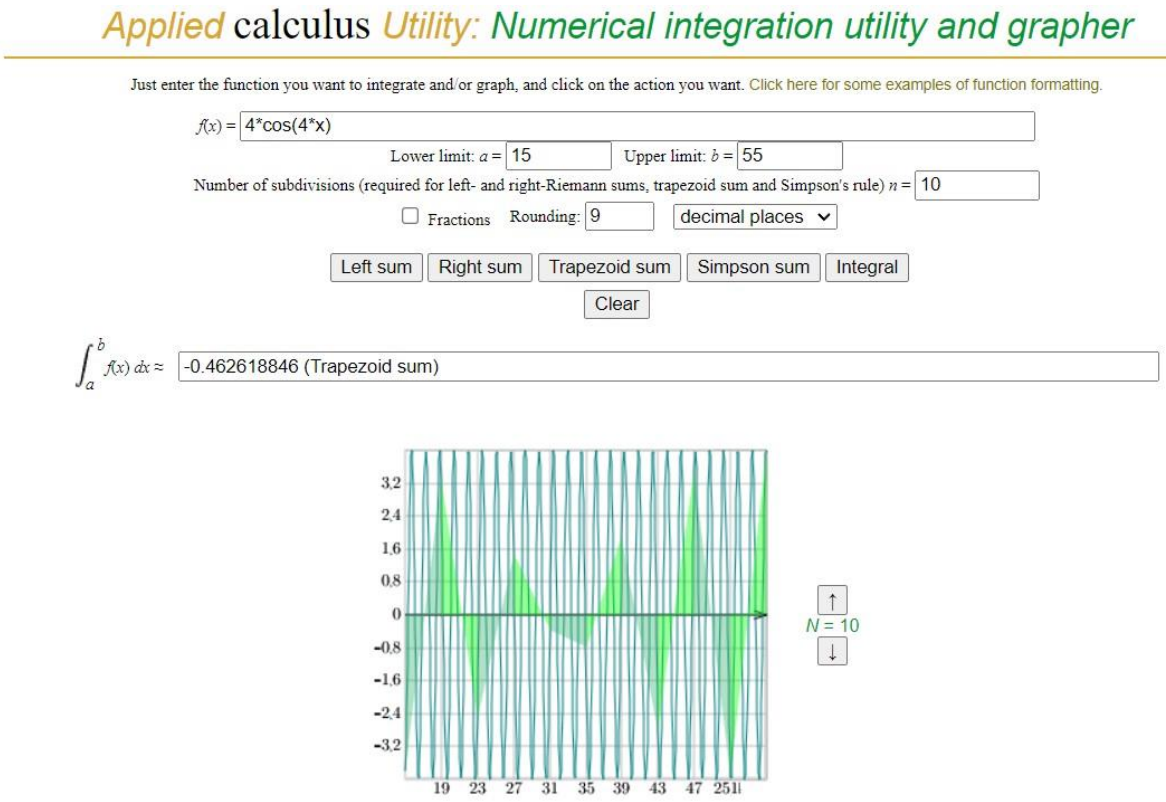
Kosinusna funkcija, unos 2:

$a = 15,\ \ b = 55,$

$j = 4,\quad k = 4,$

$h = 4$

Slika 4.21. Unos 2 za kosinusnu funkciju i rješenje - aplikacija



Slika 4.22. Unos 2 za kosinusnu funkciju i rješenje – web stranica

Rezultat Simpsonovog pravila: -7.592512206

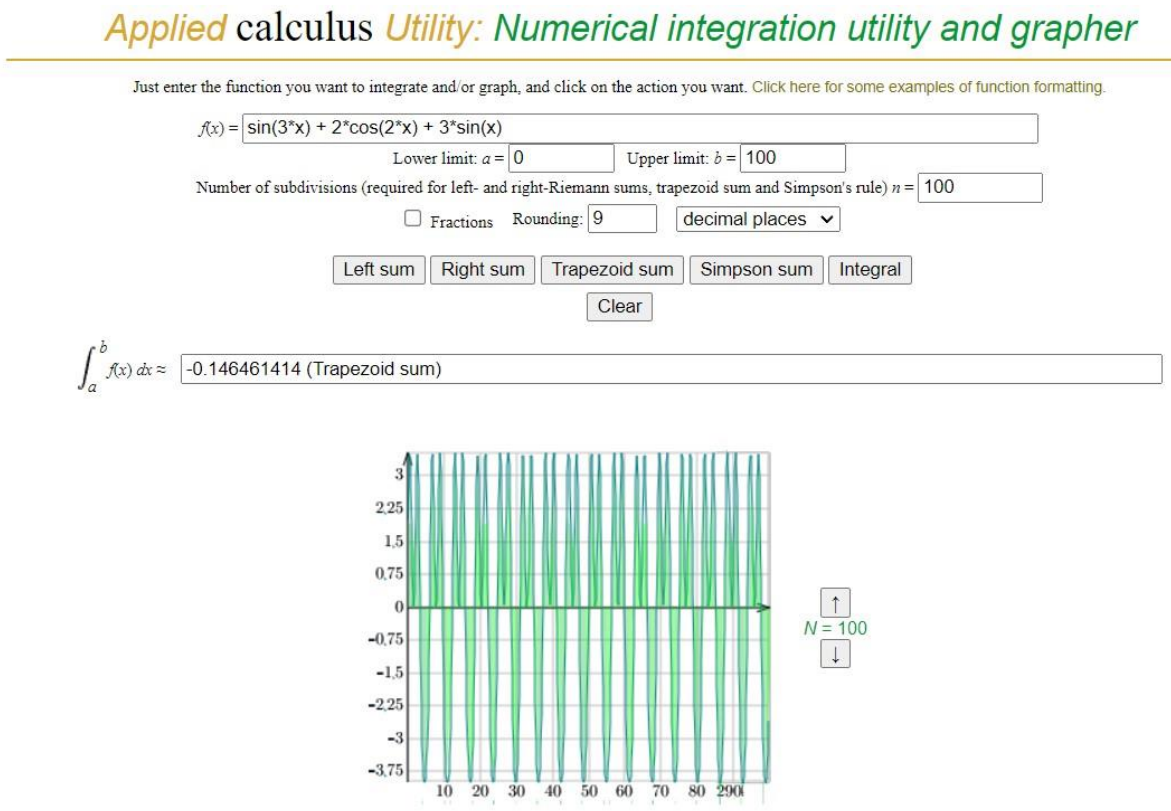
Rezultat Newton – Leibnizove formule: 0.393209334

Složena funkcija, unos 1:

$a = 15,\ \ b = 55,$

$h = 4$

Slika 4.23. Unos 1 za složenu funkciju i rješenje - aplikacija



Slika 4.24. Unos 1 za složenu funkciju i rješenje – web stranica

Rezultat Simpsonovog pravila: 1.839955314

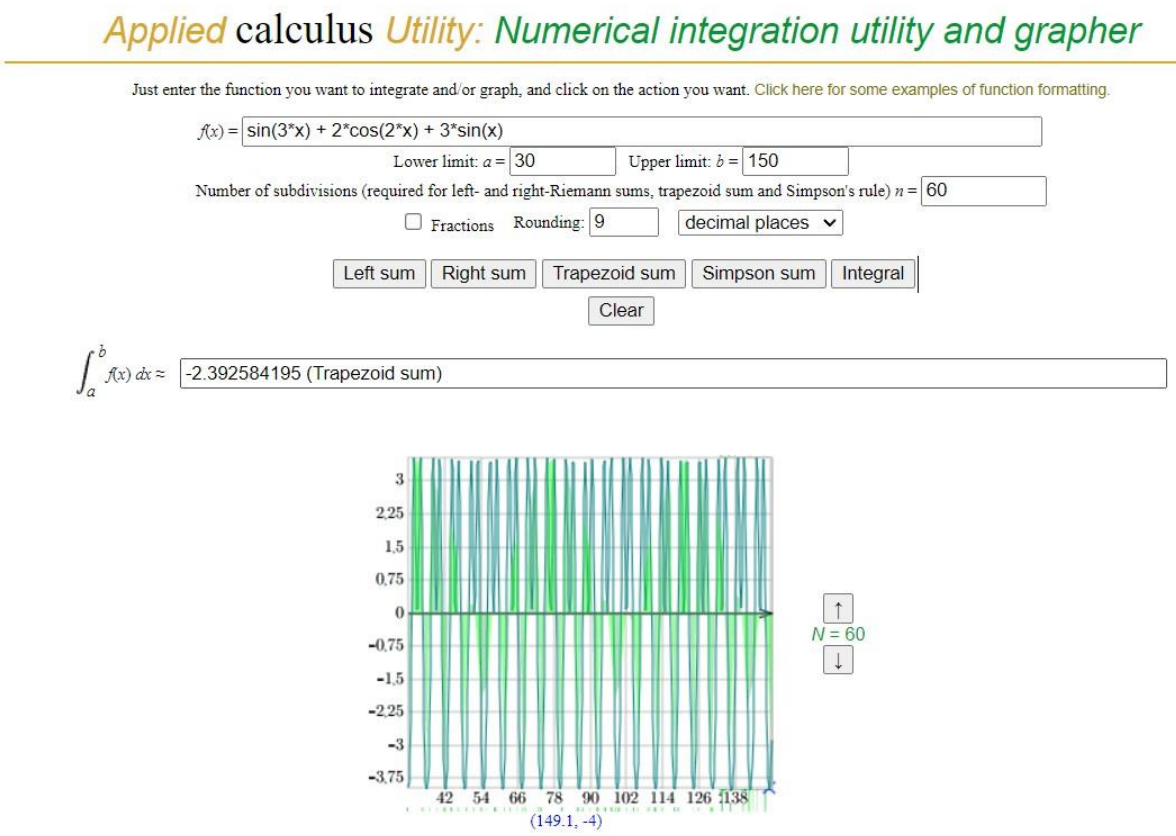
Rezultat Newton – Leibnizove formule: -0.119555041

Složena funkcija, unos 2:

$a = 30,\ \ b = 150,$

$h = 2$

Slika 4.25. Unos 2 za složenu funkciju i rješenje - aplikacija



Slika 4.26. Unos 2 za složenu funkciju i rješenje – web stranica

Rezultat Simpsonovog pravila: -2.242451545

Rezultat Newton – Leibnizove formule: -2.235916839

Zbog jednostavnosti prikaza rezultata, odabran je kalkulator sa poveznice: „https://www.zweigmedia.com/RealWorld/integral/integral.html“. Iz priloženog (slike 4.15. – 4.26.) vidi se da se rezultati poklapaju, osim malih razlika do kojih dolazi kod zaokruživanja brojeva. Grafički prikazi se također poklapaju, s tim da je na stranici prikaz precizniji.

5. ZAKLJUČAK

Tema ovog završnog rada bila je numerička integracija i njena sinteza s Android platformom. Što se tiče izrade završnog rada, materijala o matematičkoj teoriji iza numeričke integracije nije bilo teško pronaći, kako na internetu, tako i u različitim udžbenicima i knjigama.

Sama izrada aplikacije je bila izazovnija, ali Android Studio se istaknuo kao vrlo učinkovito razvojno sučelje. Kotlin kao programski jezik je također bio dosta jednostavan, a library GraphView koji je korišten za grafički prikaz funkcija je dobro dokumentiran.

Nakon ovog završnog rada jasnije je zašto je numerička integracija bitna, a to je za slučajeve kad uobičajeni način rješavanja integrala nije moguć, posebno za korisnike koji nemaju pristup naprednim matematičkim softverima. Kako je Android jedna od najpristupačnijih platformi u današnjem svijetu, mislim da bi ova aplikacija, nakon još poboljšanja, mogla biti koristan alat.

Glavni nedostatci aplikacije su nemogućnost unosa decimalnih brojeva za granice integriranja, parametre funkcija te na kraju i sam korak izračunavanja, koji je bitan za trapezno i Simpsonovo pravilo. Također, bilo bi preglednije da korisnik umjesto mogućnosti odabira koraka izračunavanja ima mogućnost odabira broja segmenata na koje se dijeli funkcija integriranja. Te implementacije su otežali tipovi podataka. Preciznije, u *for* petlji, u obliku koji je korišten u aplikaciji, nije moguće koristiti decimalne brojeve kao korak iteracije. Kako se u *GraphView* biblioteci točke spremaju sa dvije koordinate koje su tipa *double*, to je predstavljao veliki problem. Potencijalno rješenje je preopterećivanje klasa i funkcija iz *GraphView* biblioteke.

LITERATURA

[1] Numerical integration. Wikipedia; 2024
https://en.wikipedia.org/wiki/Numerical_integration [16.09.2024.]

[2] R. Scitovski, Numerička matematika, Odjel za matematiku Sveučilišta u Osijeku, Osijek, 2004.

[3] Trapezoidal rule. Wikipedia; 2024
https://en.wikipedia.org/wiki/Trapezoidal_rule [16.09.2024.]

[4] Simpson's rule. Wikipedia; 2024
https://en.wikipedia.org/wiki/Simpson%27s_rule [16.09.2024.]

SAŽETAK

U ovom završnom radu objašnjena je matematička teorija iza numeričke integracije, točnije, teorija vezana za određeni integral, trapezno i Simpsonovo pravilo, a zatim i primjenjena na izradu Android aplikacije. To je odrađeno u Android Studio razvojnom sučelju, u Kotlin programskom jeziku, uz pomoć GraphView biblioteke. Cijeli postupak izrade aplikacije detaljno je opisan slikama izgleda aplikacije, slikama koda i tekstualnim objašnjenjima. U aplikaciji korisnik može odabrati jednu od tri ponuđene funkcije: sinusna, kosinusna i složena funkcija. Nakon odabira korisnik može mijenjati funkcije unošenjem različitih parametara, što će utjecati na krajnja rješenja i grafički prikaz tih funkcija. U slučaju nekakvih nejasnoća, u aplikaciji postoje kratke upute o korištenju aplikacije. Kako bi se osigurala točnost rješenja dobivenih korištenjem aplikacije, uspoređena su sa rješenjima dobivenim na sličnim alatima sa drugih poveznica.

Ključne riječi: Android, Numerička integracija, Simpsonovo pravilo, Trapezno pravilo,

ABSTRACT

Android application for solving integrals using numerical integration

In this final paper the mathematical theory behind numerical integration, more precisely, behind the trapezoidal and Simpson's rule is explained and then applied to the development of an Android application. This was done in the Android Studio integrated development environment using Kotlin programming language, with the help of the GraphView library. The whole process is described in detail with pictures of the application layout, pictures of code and textual explanations. In the application the user can choose between three functions: sine, cosine and composite function. After the choice, the user can change functions by entering different parameters, which affect the end solutions and graphical representations of said functions. In case of unclarities, there are short instructions implemented into the application. To ensure the accuracy of the solutions given by the application, they were compared to solutions given by similar tools from other links.

Keywords: Android, Numerical integration, Simpson's rule, Trapesoidal rule