UNIVERZITET U ZENICI
POLITEHNIČKI FAKULTET

APLIKACIJA ZA NUMERIČKO DIFERENCIRANJE I NUMERIČKO INTEGRIRANJE

-SEMINARSKI RAD-

STUDENTI
Nejra Čoloman, II-119
Ajla Brdarević, II-120
Zlatko Šarić, II-122

PREDMETNI NASTAVNIK
r. prof. dr. Aleksandar Karač

Zenica (februar, 2025)

Sadržaj

1. UVOD.....2
2. TEORIJSKI DIO.....2
3. APLIKACIJA.....5
4. PRIMJERI PRIMJENE APLIKACIJE.....9
    4.1 Primjer korištenja Eulerove metode.....9
    4.2 Primjer korištenja kod određenog integrala.....9
    4.3 Primjer korištenja kod neodređenog integrala.....9
5. ZAKLJUČCI.....9
6. LITERATURA.....10

1. UVOD

Numeričko diferenciranje i integriranje ključne su tehnike u matematičkoj analizi koje imaju široku primjenu u raznim znanstvenim i inženjerskim područjima. Ove metode omogućuju rješavanje složenih problema koji se ne mogu riješiti analitičkim putem, čime pružaju moćan alat za istraživanje i primjenu matematičkih koncepata.

U okviru ovog seminara, cilj je razviti aplikaciju koja će omogućiti jednostavno i efikasno diferenciranje i integriranje funkcija. Aplikacija je izrađena pomoću PyQt5 okvira za grafičke korisničke interfejse, što omogućava intuitivno i korisnički prihvatljivo iskustvo. Korisnici mogu unositi funkcije te postavljati intervale za numeričko integriranje. Kroz rad će se detaljno predstaviti različite metode numeričkog integriranja i diferenciranja koje su implementirane u aplikaciji. Te metode uključuju trapeznu formulu, Simpsonovu formulu, i metodu srednje tačke. Svaka od ovih metoda ima svoje specifične prednosti i primjene, ovisno o tipu funkcije koja se integrira.

Jedna od ključnih značajki aplikacije je mogućnost vizualizacije funkcija i njihovih derivata, čime se dodatno poboljšava razumijevanje matematičkih koncepata. Korisnici mogu vidjeti grafički prikaz funkcija koje unesu, kao i njihove derivacije i integrale, što olakšava interpretaciju rezultata i razumijevanje procesa numeričkog izračuna. Kroz ovaj rad, pokazat će se kako su ove metode implementirane u aplikaciji i kako se mogu koristiti za rješavanje stvarnih problema numeričkog diferenciranja i integriranja.

2. TEORIJSKI DIO

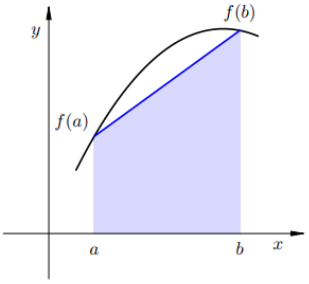
U ovom poglavlju su navedeni osnovni teorijski koncepti koji su praktično implementirani u aplikaciji. Algoritmi, funkcije i metode koje implementiraju navedene teorijske koncepte su objašnjeni u poglavlju 3. Aplikacija.

Numeričko integriranje

- Trapezno pravilo:

Trapezno pravilo je jedna od najjednostavnijih numeričkih metoda za aproksimaciju određenih integrala. Temelji se na zamjeni funkcije pravcem koji prolazi kroz krajnje tačke intervala integracije, čime se površina ispod funkcije aproksimira površinom trapeza.

„Vidimo da je integracijska formula $I_1(f)$ dobivena iz egzaktnosti na svim polinomima stupnja manjeg ili jednakog 1, i glasi $\int_a^b f(x)dx \approx \frac{h}{2}(f(a) + f(b))$. Ta formula zove se trapezna formula. Odakle joj ime? Napišemo li je na malo drugačiji način, kao $\int_a^b f(x)dx \approx \frac{f(a)+f(b)}{2}(b - a)$, odmah ćemo vidjeti da je $(f(a) + f(b))/2$ srednjica, a $b - a$ visina trapeza sa slike.” [1] (slika 1)



Slika 1. Trapezna formula - visina trapeza

Trapezno pravilo koristi se za brzu i jednostavnu procjenu vrijednosti određenog integrala, posebno u situacijama kada eksplicitna analitička funkcija nije dostupna ili kada je ručno integriranje složeno. Ova metoda je posebno korisna u numeričkim proračunima, gdje se integral približno izračunava pomoću diskretnih tačaka. Preciznost trapeznog pravila zavisi od oblika funkcije—što je funkcija glađa i što je interval podjele manji, to će aproksimacija biti tačnija.

- Simpsonova formula

Simpsonova formula je jedna od Newton–Cotesovih formula za numeričku integraciju, koja omogućava precizniju aproksimaciju određenih integrala u poređenju s trapeznim pravilom. Ova metoda se zasniva na interpolaciji funkcije pomoću kvadratnog polinoma, čime se integral približno računa preko tri tačke na datom intervalu.

„... integracijska formula $I_2(f)$ dobivena je iz egzaktnosti na svim polinomima stupnja manjeg ili jednakog 2, i glasi $\int_a^b f(x)dx \approx \frac{h}{3}[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]$. Simpsonova formula ima još jednu prednost. Iako je dobivena iz uvjeta egzaktnosti na vektorskom prostoru polinoma stupnja manjeg ili jednakog 2, ona egzaktno integrira i sve polinome

stupnja 3. Dovoljno je pokazati da egzaktno integrira $f(x) = x^3$. Egzaktni integral jednak je $\int_a^b x^3 dx = \frac{b^4 - a^4}{4}$, …" [2]

Ova metoda se često koristi u numeričkim proračunima jer kombinuje jednostavnost s povećanom preciznošću, naročito kada analitičko rješenje nije dostupno ili je ručno integriranje složeno. Što je funkcija glađa i interval podjele manji, to je aproksimacija tačnija.

- Formula srednje tačke (midpoint formula)

Midpoint pravilo, ili pravilo srednje tačke, jedna je od otvorenih Newton-Cotesovih formula koja ne interpolira rubne tačke intervala, već koristi vrijednost funkcije u sredini intervala za aproksimaciju integrala. Ova metoda je jednostavna i često korištena zbog svoje relativno dobre tačnosti u poređenju s drugim osnovnim numeričkim metodama.

„Vjerojatno najkorištenija i najpoznatija otvorena Newton–Cotesova formula je ona najjednostavnija za m = 0, poznata pod imenom "midpoint formula" (formula srednje točke). Dakle za bismo odredili midpoint formulu, moramo naći koeficijent ω_0: ω_0 takav da je $\int_a^b f(x)dx = \omega_0 f(\frac{a+b}{2})$ egzaktna na vektorskom prostoru polinoma što višeg stupnja. Za $f(x) = 1$, imamo $b - a = \int_a^b 1dx = \omega_0$, odakle odmah slijedi da je $\int_a^b f(x)dx = (b - a)f(\frac{a+b}{2})$." [3]

- Eulerova metoda

Eulerova metoda je ključna numerička tehnika koja se koristi za aproksimaciju rješenja diferencijalnih jednadžbi početnih uvjeta. Iako jednostavna, pruža dobar uvod u koncept numeričkih rješenja diferencijalnih jednadžbi.

„Metoda se zasniva na ideji da se y' u gornjoj jednadžbi zamijeni s podijeljenom razlikom $y'(x) = \frac{y(x+h) - y(x)}{h} + O(h)$, pa rješenje diferencijalne jednadžbe zadovoljava $y(x + h) = y(x) + hy'(x) + O(h^2) = y(x) + hf(x, y(x)) + O(h^2)$." [4]

3. APLIKACIJA

Aplikacija je rađena u programskom jeziku Python, koristeći neke od osnovnih biblioteka kao što su NumPy, PyQt5, Matplotlib i SymPy. Python je odabran zbog svoje pristupačnosti, velikog broja dostupnih biblioteka, i jednostavne sintakse.

NumPy se koristi za pohranu i obradu podataka u formi nizova te za efikasne numeričke proračune. PyQt5 se u aplikaciji koristi za kreiranje prozora, formi, dugmadi i ostalih elemenata sučelja. Matplotlib se koristi za prikazivanje grafova funkcija, što olakšava analizu podataka i vizualizaciju rezultata numeričkih metoda. SymPy se u aplikaciji koristi za manipulaciju matematičkim formulama, simboličko diferenciranje i integriranje funkcija.

Aplikacija se sastoji od nekoliko segmenata:

1. diferenciranje.py - sadrži funkciju diferenciraj u kojoj je implementiran kod za numeričko diferenciranje.

   Kod 1 prikazuje implementaciju Eulerove metode. Funkcija prima jedan argument function_str, koji predstavlja matematičku funkciju u obliku stringa, kao što je x + y ili x * y. dydx(x, y) računa derivaciju funkcije f(x,y) koristeći eval(), zamjenjujući ^ sa **. Početni uvjeti: $x_o = 0\, i\, y_0 = 1;\ x_{end} = 10;$ step_size=0.1. Inicijalizacija listi: x_vals i y_vals čuvaju vrijednosti *x* i *y*. Iteracija: U svakoj petlji računa se nova vrijednost za *y* pomoću Eulerove metode: $y=y+step\_size \cdot dydx(x,y)$ i nova vrijednost za *x*:$x=x+step\_size$. Dodaju se u liste. Povratak rezultata: Funkcija vraća dva NumPy niza sa svim vrijednostima *x* i *y* kao rješenje ODJ-a.

```
def diferenciraj(function_str):
    def dydx(x, y):
        return eval(function_str.replace('^', '**').replace('e', 'np.exp(1)'))
    x0 = 0  # Početna vrijednost za x
    y0 = 1  # Početna vrijednost za y
    x_end = 10  # Krajnja vrijednost za x
    step_size = 0.1  # Veličina koraka
    x_vals = [x0]
    y_vals = [y0]
    x = x0
    y = y0
    while x < x_end:
        y = y + step_size * dydx(x, y)
        x = x + step_size
        x_vals.append(x)
        y_vals.append(y)
    return np.array(x_vals), np.array(y_vals)
```

Kod 1. diferenciranje.py

2. integriranje.py - sadrži funkcije za različite metode numeričkog integriranja, te funkciju za odabir odgovarajuće metode.

Funkcija trapezna_formula (kod 2) implementira, već kroz teoriju spomenutu, trapeznu formulu za numeričko integriranje. Ima ulazne argumente: f (funkcija koju integriramo), a (početna tačka intervala), b (krajnja tačka intervala) i n (broj podintervala). Izračunava veličinu koraka $h = \frac{b-a}{n}$. Inicijalizira početnu sumu kao $0,5 \cdot (f(a) + f(b))$. Iterira kroz unutrašnje tačke intervala for i in range(1, n), dodaje funkcijske vrijednosti u sumu $suma \mathrel{+}= f(a + i * h)$. Konačna aproksimacija je $suma \cdot h$, što daje aproksimaciju vrijednosti određenog integrala na zadanom intervalu.

```
def trapezna_formula(f, a, b, n):
    h = (b - a) / n
    suma = 0.5 * (f(a) + f(b))
    for i in range(1, n):
        suma += f(a + i * h)
    return suma * h
```

Kod 2. Trapezna formula u integriranje.py

Funkcija simpsonova_formula (kod 3) implementira Simpsonovu formulu za numeričku integraciju. Prima četiri argumenta: f (funkcija koju integriramo), a (početna tačka intervala), b (krajnja tačka intervala) i n (broj podintervala). Provjerava da li je broj podintervala paran i povećava ga za 1 ako nije. Izračunava veličinu koraka $h = \frac{b-a}{n}$. Inicijalizira sumu kao $f(a) + f(b)$. Dodaje težine za neparne tačke kroz petlju, računa vrijednosti $4 \cdot f(a + i \cdot h)$, i za parne tačke, računa vrijednosti $2 \cdot f(a + i \cdot h)$. Konačna aproksimacija je $suma \cdot \frac{h}{3}$, što daje aproksimaciju određenog integrala na intervalu [a, b].

```
def simpsonova_formula(f, a, b, n):
    if n % 2 == 1:
        n += 1  # Broj podintervala mora biti paran
    h = (b - a) / n
    suma = f(a) + f(b)
    for i in range(1, n, 2):
        suma += 4 * f(a + i * h)
    for i in range(2, n-1, 2):
        suma += 2 * f(a + i * h)
    return suma * h / 3
```

Kod 3. Simpsonova formula u integriranje.py

Funkcija midpoint_formula (kod 4) implementira Midpoint formulu za numeričku integraciju. Prima četiri argumenta: f (funkcija koju integriramo), a (početna tačka intervala), b (krajnja tačka intervala) i n (broj podintervala). Izračunava veličinu koraka $h = \frac{b-a}{n}$. Inicijalizira sumu kao 0. Prolazi kroz sve podintervale, računa sredinu svakog podintervala $x = a + (i + 0,5) \cdot h$, i dodaje vrijednosti funkcije u sumu. Konačna aproksimacija je *suma*h*, što daje aproksimaciju određenog integrala na intervalu [a,b].

```
def midpoint_formula(f, a, b, n):
    h = (b - a) / n
    suma = 0
    for i in range(n):
        x = a + (i + 0.5) * h
        suma += f(x)
    return suma * h
```

Kod 4. Midpoint formula u integriranje.py

Funkcija odaberi_metodu (kod 5) automatski bira najbolju numeričku metodu na osnovu prepoznatih obrazaca u funkciji. Provjerava da li funkcija f vraća neodređene vrijednosti (NaN) za prosječnu vrijednost između granica a i b. Ako pronađe NaN, vraća poruku o grešci. Ako se u tekstualnom prikazu funkcije prepoznaju polinomski izrazi (x** ili x^), koristi se Simpsonova metoda, jer je optimalna za polinome. Ako se prepoznaju eksponencijalni izrazi (exp ili e**), koristi se metoda srednje tačke (Midpoint formula) koja dobro aproksimira integrale eksponencijalnih funkcija. Ako funkcija ne pripada nijednoj od prepoznatih kategorija, koristi se trapezna metoda kao univerzalno rješenje.

```
def odaberi_metodu(f, a, b, n=1000):
    try:
        test_value = f((a + b) / 2)
        if np.isnan(test_value):
            return None, "Funkcija ima neodređene vrijednosti (NaN)"
        function_str = str(f)
        if 'x**' in function_str or 'x^' in function_str:
            return simpsonova_formula(f, a, b, n), "Koristim Simpsonovu metodu”        if 'exp'
in function_str or 'e**' in function_str:
            return midpoint_formula(f, a, b, n), "Koristim Metodu srednje tačke"
        return trapezna_formula(f, a, b, n), "Koristim Trapeznu metodu"
    except Exception as e:
        return None, f"Greška u odabiru metode: {e}"
```

Kod 5. Odabir metode u integriranje.py

3. gui.py - sadrži kod za GUI aplikacije, te poziva funkcije iz diferenciranje.py i integriranje.py. Ključne funkcionalnosti:

   Grafički korisnički interfejs (GUI):

   - Kreiranje glavnog prozora (MainWindow) i definiranje layout-a koristeći QVBoxLayout, QFormLayout, i QScrollArea.
   - Dodavanje unosa za funkciju, početak i kraj intervala te prikaz rezultata u QLabel.
   - Stilizacija dugmadi (QPushButton) i polja za unos funkcije (QLineEdit).

Diferenciranje funkcije (Eulerova metoda): Klikom na dugme "Eulerova metoda" poziva se metoda self.diferenciraj() koja koristi funkciju diferenciraj iz modula diferenciranje. Rezultati diferenciranja prikazuju se u QLabel.

```
self.dif_button.clicked.connect(self.diferenciraj)
self.layout.addWidget(self.dif_button)
```

Kod 6. Poziv metode za diferenciranje

Numeričko integriranje: Korisnik može odabrati između različitih metoda za numeričko integriranje funkcije:

1. **Trapezna metoda:** Koristi metodu trapezna_formula za numeričko integriranje.

```
self.int_trap_button.clicked.connect(self.trapezna)
self.layout.addWidget(self.int_trap_button)
```

Kod 7. Poziv trapezne metode

2. **Simpsonova metoda:** Poziva simpsonova_formula za preciznije integriranje.

```
self.int_simp_button.clicked.connect(self.simpsonova)
self.layout.addWidget(self.int_simp_button)
```

Kod 8. Poziv Simpsonove metode

3. **Metoda srednje tačke (Midpoint):** Koristi midpoint_formula za aproksimaciju.

```
self.int_mid_button.clicked.connect(self.midpoint)
self.layout.addWidget(self.int_mid_button)
```

Kod 9. Poziv Midpoint metode

4. **Neodređeni integral:** Funkcija koristi sympy.integrate za simboličku integraciju.

```
self.indefinite_int_button.clicked.connect(self.neodredjeni_integral)
self.layout.addWidget(self.indefinite_int_button)
```

Kod 10. Poziv metode za neodređeni integral

Vizuelizacija funkcije: Prikazuje graf funkcije korištenjem biblioteke Matplotlib.

```
        plt.plot(x_vals, y_vals)
        plt.title(f"Graf funkcije: {function_str}")
        plt.xlabel('x')
        plt.ylabel('f(x)')
        plt.grid(True)
        plt.show()
```

Kod 11. Vizuelizacija funkcije

4. main.py - sadrži glavnu metodu za pokretanje aplikacije. Sastoji se od dva dijela - klase i glavne metode.

   U kodu 12 se može vidjeti da klasa MainApp nasljeđuje QMainWindow kako bi se omogućilo korištenje svih metoda i atributa glavnog prozora aplikacije. Postavlja se naslov prozora, te pozicija i veličina prozora. I na kraju se inicijalizira korisnički interface iz GUI modela.

```
class MainApp(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Numeričko Diferenciranje i Integriranje")
        self.setGeometry(100, 100, 600, 400)
        self.ui = Ui_MainWindow(self)
```

Kod 12. Klasa MainApp

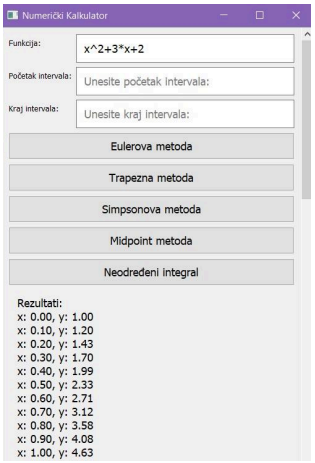
Pored toga main.py ima glavnu metodu, kod 13. Prvo se provjerava da li se skripta pokreće direktno, tj. da se ne importuje kao modul. Zatim se stvara QApplication objekat koji upravlja svim događajima aplikacije, te instanca glavnog prozora. Prikazuje glavni prozor, a zatim ulazi u glavnu događajnu petlju.

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainApp()
    window.show()
    sys.exit(app.exec_())
```

Kod 13. Glavna metoda

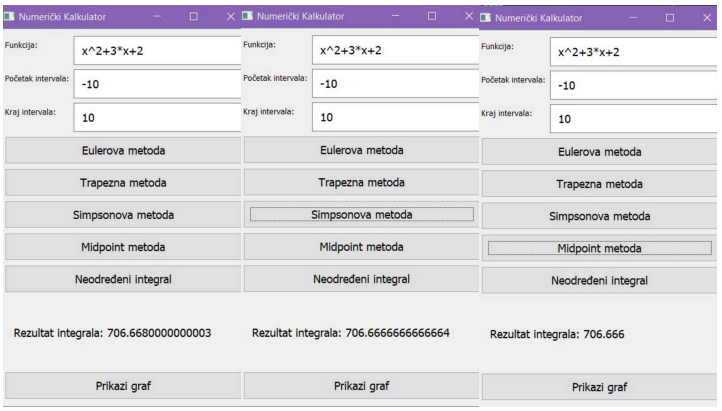
4. PRIMJERI PRIMJENE APLIKACIJE

4.1 Primjer korištenja Eulerove metode



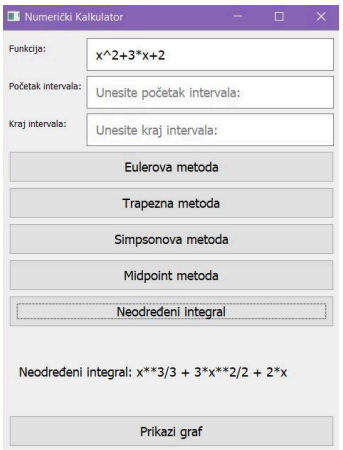
Slika 2. Primjer korištenja Eulerove metode za x^2+3*x+2

4.2 Primjer korištenja kod određenog integrala



Slika 3. Primjer korištenja aplikacije za određeni integral x^2+3*x+2 [-10,10]

4.3 Primjer korištenja kod neodređenog integrala



Slika 4. Primjer korištenja aplikacije za neodređeni integral x^2+3*x+2

5. ZAKLJUČCI

Razvijanje ove aplikacije za numeričko diferenciranje i integriranje omogućilo je ključne uvide u primjenu numeričkih metoda. Implementacija trapezne i Simpsonove formule pokazala je njihove specifične prednosti: trapezna formula je jednostavna i efikasna za manje krivudave funkcije, dok Simpsonova pruža veću preciznost za funkcije koje se mogu aproksimirati kvadratnim polinomima. Metoda srednje tačke bila je korisna za složenije funkcije. Vizualizacija funkcija omogućila je bolju interpretaciju rezultata i povezivanje teorijskih znanja s praktičnom primjenom. Numeričko diferenciranje omogućilo je precizne aproksimacije derivata funkcija koje nisu analitički diferencijabilne.

Zaključno, aplikacija potvrđuje značaj numeričkih metoda u matematičkoj analizi i njihov potencijal za rješavanje stvarnih problema.

6. LITERATURA

[1] Drmač, Z., Hari, V., Marušić, M., Rogina, M., Singer, S. i Singer, S. SVEUČILIŠTE U ZAGREBU PMF - MATEMATIČKI ODJEL Numerička analiza [online] str. 480-481. Dostupno na: https://web.math.pmf.unizg.hr/~rogina/2001096/num_anal.pdf, pregledano 19.1.2025.

[2] Drmač, Z., Hari, V., Marušić, M., Rogina, M., Singer, S. i Singer, S. SVEUČILIŠTE U ZAGREBU PMF - MATEMATIČKI ODJEL Numerička analiza [online] str. 486-487. Dostupno na: https://web.math.pmf.unizg.hr/~rogina/2001096/num_anal.pdf, pregledano 23.1.2025.

[3] Drmač, Z., Hari, V., Marušić, M., Rogina, M., Singer, S. i Singer, S. SVEUČILIŠTE U ZAGREBU PMF - MATEMATIČKI ODJEL Numerička analiza [online] str. 497-498. Dostupno na: https://web.math.pmf.unizg.hr/~rogina/2001096/num_anal.pdf, pregledano 1.2.2025.

[4] Drmač, Z., Hari, V., Marušić, M., Rogina, M., Singer, S. i Singer, S. SVEUČILIŠTE U ZAGREBU PMF - MATEMATIČKI ODJEL Numerička analiza [online] str. 535-536. Dostupno na: https://web.math.pmf.unizg.hr/~rogina/2001096/num_anal.pdf, pregledano 4.2.2025.