

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Marija Ilić

Upravljanje Newtonovim fraktalom

ZAVRŠNI RAD

Pula, rujan, 2025. godine

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Marija Ilić

Upravljanje Newtonovim fraktalom

ZAVRŠNI RAD

JMBAG: 0246071515, izvanredni student
Studijski smjer: Informatika

Kolegij: Matematika 1
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske znanosti

Mentor: doc. dr. sc. Siniša Miličić

Pula, rujan, 2025. godine

Sadržaj

1	Uvod	1
2	Newtonova metoda	2
2.1	Načelo rada metode	2
2.2	Značaj i primjena metode	2
3	Newtonov fraktal	4
3.1	Definicija	4
3.2	Prikaz	4
4	Upravljanje Newtonovim fraktalom	6
5	Implementacija	7
5.1	Programsko okruženje	7
5.2	Algoritamska modularnost	7
5.3	Strategije konstrukcije funkcije	8
5.4	Načini vizualizacije	10
6	Galerija	13
6.1	Hipoteza 1	13
6.2	Hipoteza 2	14
6.3	Hipoteza 3	15
6.4	Hipoteza 4	17
7	Zaključak	20

1 Uvod

Matematika nije samo jezik znanosti, već i izvor estetskih i vizualno fascinantnih struktura. Jedan od primjera gdje se apstraktni numerički postupci pretvaraju u složene i neponovljive uzorke jest Newtonova metoda, poznata po svojoj učinkovitosti u određivanju nultočaka funkcija. Kada se ova metoda proširi na kompleksnu ravninu, pojavljuju se Newtonovi fraktali. Oblici koji na prvi pogled djeluju kaotično, a zapravo u sebi kriju duboku matematičku logiku.

Ovaj rad istražuje temeljna načela Newtonove metode, njezinu primjenu u generiranju fraktalnih struktura te povezanost između numeričke analize i vizualizacije dinamičkih sustava. Osim matematičkog aspekta, naglasak je stavljen i na estetsku dimenziju fraktala, koji se sve češće koriste kao ilustracija ljepote i složenosti prirodnih procesa. Na taj se način rad nalazi na sjecištu između stroge analitičke discipline i kreativne vizualne reprezentacije.

2 Newtonova metoda

Newtonova metoda (ili Newton–Raphsonova metoda) jedna je od najpoznatijih iterativnih metoda za numeričko nalaženje nultočaka funkcija. Povjesno ju je razvio Isaac Newton 1669. godine, a kasnije unaprijedio Joseph Raphson 1690. godine. Metoda se temelji na linearom aproksimiranju funkcije pomoću tangente i računanju njenog presjeka s osi x , čime se postupno poboljšava početna aproksimacija rješenja [1].

2.1 Načelo rada metode

Za zadanu diferencijabilnu funkciju $f(x)$ i početnu točku x_0 , nova aproksimacija nultočke računa se formulom

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad f'(x_n) \neq 0.$$

Ova formula proizlazi iz jednadžbe tangente u točki $(x_n, f(x_n))$. Presjek tangente s osi x daje bolju procjenu nultočke, a ponavljanjem procesa nastaje niz vrijednosti x_0, x_1, x_2, \dots koji u povoljnim slučajevima konvergira prema stvarnom rješenju.

Važno je naglasiti da metoda ne konvergira uvijek. Ako se odabere početna vrijednost u kojoj je derivacija $f'(x)$ jednaka nuli, postupak se prekida jer tangenta ne siječe os x . Također, kod višestrukih nultočkih konvergencija može biti linearne, a ne kvadratne, što značajno usporava računanje [1].

2.2 Značaj i primjena metode

Newtonova metoda je važna iz nekoliko razloga:

1. **Brza konvergencija** – za jednostavne nultočke metoda konvergira kvadratno, što znači da se broj točnih decimala otprilike udvostručuje u svakoj iteraciji. To je daleko brže od linearnih metoda poput bisekcije [2].
2. **Opća primjenjivost** – metoda se može koristiti za mnoge vrste nelinearnih jednadžbi, ne samo za polinome [1].
3. **Osnovna metoda u numeričkoj analizi** – služi kao polazište za razvoj brojnih drugih metoda, uključujući modificirane Newtonove metode i kvazi-Newtonove postupke [3].
4. **Veza s dinamičkim sustavima** – analiza ponašanja iteracija Newtonove metode otkriva bogatu dinamiku, uključujući cikluse i kaotično ponašanje. Upravo ta svojstva omogućila su otkriće fraktalnih struktura u kompleksnoj ravnini [1, 4].

5. **Primjene u praksi** – Newtonova metoda široko se koristi u znanosti i inženjerstvu, primjerice pri rješavanju nelinearnih jednadžbi, optimizaciji, računalnim simulacijama i u algoritmima za aproksimaciju korijena složenih funkcija [5].

3 Newtonov fraktal

Newtonov fraktal je geometrijski objekt koji nastaje kada Newtonovu metodu primijenimo u kompleksnoj ravnini. Dok u realnom slučaju metoda tipično konvergira prema jednoj od nultočaka funkcije, u kompleksnom slučaju situacija je složenija. Različite početne točke mogu konvergirati prema različitim nultočkama, a granice između tih područja imaju izrazito nepravilnu i složenu strukturu. Upravo te granice tvore fraktalne obrasce [1, 4].

3.1 Definicija

Za kompleksnu funkciju $f(z)$, Newtonova iteracija definirana je kao

$$N(z) = z - \frac{f(z)}{f'(z)}.$$

Za svaku početnu točku $z_0 \in \mathbb{C}$, niz iteracija $N^n(z_0)$ (gdje n označava broj primjena funkcije) može konvergirati prema jednoj od nultočaka funkcije f .

Područje privlačenja (eng. *basin of attraction*) za nultočku r definirano je kao skup svih početnih točaka z_0 iz kojih Newtonova metoda konvergira prema r :

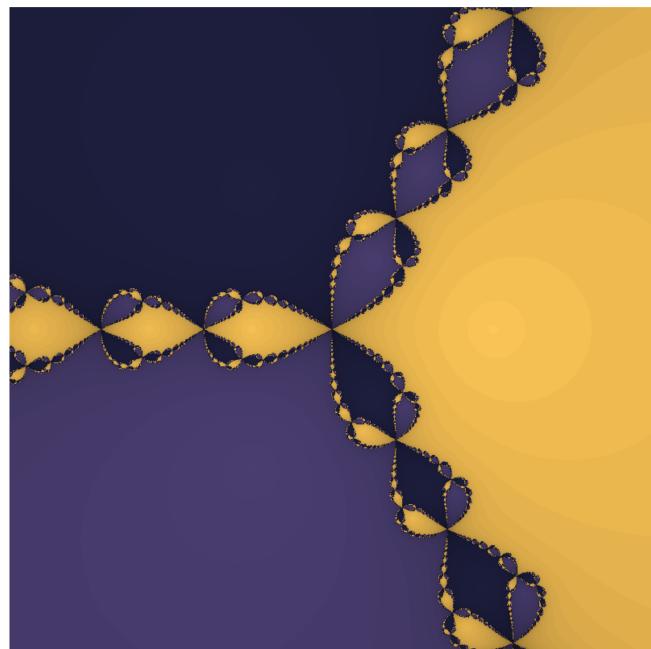
$$B(r) = \{z_0 \in \mathbb{C} \mid N^n(z_0) \rightarrow r\}.$$

Granice između različitih područja privlačenja su izrazito složene i imaju slijednu strukturu. Upravo one čine **Newtonov fraktal** [1, 6].

3.2 Prikaz

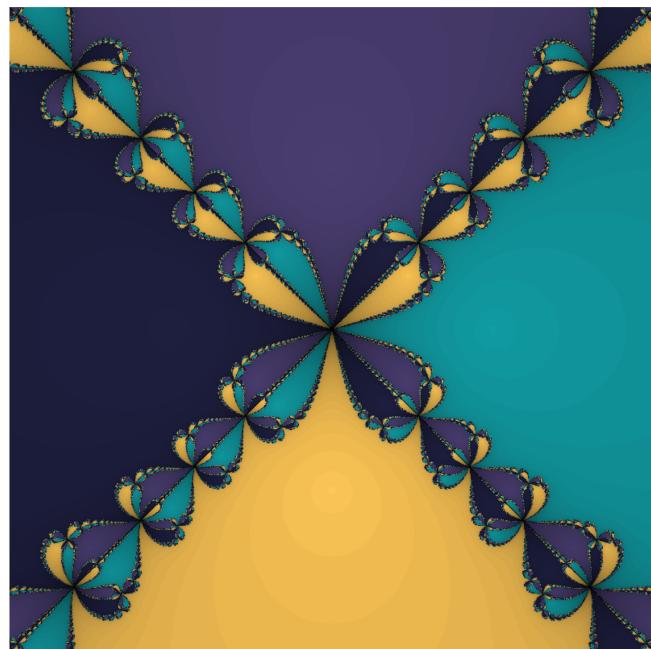
Vizualno, Newtonov fraktal konstruira se tako da svakoj početnoj točki z_0 u kompleksnoj ravnini pridružimo boju ovisno o tome prema kojoj nultočki konvergira. Boje označavaju različite nultočke funkcije, dok intenzitet boje može označavati brzinu konvergencije.

Primjerice, za polinom $f(z) = z^3 - 1$ postoje tri različite nultočke: $1, -\frac{1}{2} + \frac{\sqrt{3}}{2}i$ i $-\frac{1}{2} - \frac{\sqrt{3}}{2}i$. Prikaz Newtonova fraktala u ovom slučaju daje karakterističnu trobojnu strukturu, gdje svaka boja odgovara jednoj od nultočaka [1].



Slika 1: Newtonov fraktal za polinom $f(z) = z^3 - 1$

Za složenije polinome poput $f(z) = z^4 - 1$ ili $f(z) = z^5 - 1$, fraktalne strukture postaju još bogatije i kompleksnije. Granice između područja privlačenja pokazuju beskonačnu složenost i samosličnost, što su tipična obilježja fraktala [7].



Slika 2: Newtonov fraktal za polinom $f(z) = z^4 - 1$

4 Upravljanje Newtonovim fraktalom

Upravljanje Newtonovim fraktalom odnosi se na postupke kojima se može usmjeravati ili mijenjati način na koji se fraktal generira i vizualno interpretira.

To uključuje:

1. **Odabir funkcije** – izbor funkcije direktno određuje oblik i složenost frakata. Primjerice, jednostavna funkcija $f(z) = z^3 - 1$ generira klasični trobojni fraktal, dok složeniji izrazi daju bogatije strukture.
2. **Načini vizualizacije** – ključni aspekt u radu s fraktalima je vizualna interpretacija. Prikaz u boji omogućuje intuitivno razumijevanje područja konvergencije i brzine iteracija, dok prikaz rubova naglašava složenost prijelaznih zona. Time se može “usmjeriti pažnja” promatrača na globalnu strukturu ili na fine detalje fraktalne granice.
3. **Iterativna kontrola konvergencije** – broj dopuštenih iteracija utječe na preciznost i izgled frakata. Stroži kriteriji naglašavaju finije detalje, dok blaži daju bržu, ali manje preciznu sliku.
4. **Strategije konstrukcije funkcije** – kako je prikazano implementacijom u nastavku, fraktal se može graditi iz unaprijed poznatih nultočaka, iz nultočaka derivacije ili iz analitičkog izraza funkcije. Upravljanje u ovom smislu znači odabrati najprikladniji način ovisno o dostupnim informacijama i cilju istraživanja.
5. **Algoritamska modularnost** – kroz objektno orientiranu arhitekturu moguće je jednostavno upravljati različitim fazama postupka: generiranjem funkcije, provođenjem iteracija i vizualizacijom rezultata. To omogućuje eksperimentiranje, kombiniranje metoda i proširivanje sustava dodatnim funkcionalnostima.

U širem kontekstu, upravljanje Newtonovim fraktalom predstavlja spoj matematike i informatike. Sa stajališta matematike, ono pruža uvid u dinamiku iterativnih metoda i složenost kaotičnih sustava. Sa stajališta informatike, riječ je o problemskom pristupu gdje se naglašava modularno programiranje i vizualizacija podataka.

5 Implementacija

5.1 Programsко okruženje

Implementacija je realizirana u programskom jeziku **Python**, zbog njegove bogate podrške za znanstvene izračune i vizualizaciju. Python nudi fleksibilno okruženje koje omogućuje kombiniranje numeričkih i simboličkih metoda, što je ključno za izvođenje iterativnih postupaka Newtonove metode u kompleksnoj ravnini.

Za potrebe implementacije korištene su tri ključne biblioteke:

- **NumPy** – za učinkovite numeričke operacije, rad s polinomima i evaluaciju funkcija nad velikim skupovima podataka [8].
- **SymPy** – za simboličku obradu funkcija koje se zatim prevode u numerički oblik primjenjiv u NumPy okruženju [9].
- **Matplotlib** – za vizualizaciju rezultata, odnosno prikaz generiranih fraktala [10].

Kao razvojno okruženje korišten je **Jupyter Notebook**, koji omogućuje spajanje koda, dokumentacije i vizualizacija u jedinstvenom formatu. Ovaj pristup se pokazao posebno pogodnim jer vizualna komponenta ima jednaku važnost kao i numerički proračuni.

5.2 Algoritamska modularnost

Kod je organiziran u skladu s principima **objektno orijentiranog programiranja (OOP)**. Cilj je postići jasnu modularnost. Svaka komponenta sustava odgovorna je za jednu logičku cjelinu: definiranje funkcije, izvođenje iteracija ili vizualizaciju rezultata. Takva organizacija olakšava održavanje i buduće proširenje sustava, primjerice dodavanjem novih metoda vizualizacije ili alternativnih strategija konstrukcije funkcija.

Struktura implementacije obuhvaća nekoliko ključnih klasa:

- **PolynomialBuilder** – zadužen za konstrukciju funkcije i njezine derivacije. Ovisno o dostupnim informacijama (nultočke funkcije, nultočke derivacije ili analitički izraz), omogućuje generiranje polinoma i pripadajuće derivacije.
- **Fractal** – implementira logiku Newtonove metode kroz centralnu metodu *_compute_newton_convergence*. Ova klasa koordinira proces iteracija, prati konvergenciju prema pojedinim nultočkama i generira numeričke podatke potrebne za prikaz fraktala.

- **Colorizer** – pretvara rezultate iteracija u vizualni prikaz dodjeljujući svakoj nultočki jedinstvenu boju, uz mogućnost naglašavanja brzine konvergencije kroz intenzitet boje.
- **EdgeDetector** – fokusiran na detekciju rubova između područja konvergencije, čime se naglašava fraktalna struktura granica.

Ovakva podjela omogućuje prirodno mapiranje matematičkih i računalnih aspekata problema u jasno razdvojene komponente.

5.3 Strategije konstrukcije funkcije

Newtonov fraktal može se generirati na više načina, ovisno o tome koje informacije o funkciji imamo unaprijed. Ključna ideja u svim slučajevima je ista: primijeniti Newtonovu metodu nad točkama kompleksne ravnine i analizirati njihovu konvergenciju.

U implementaciji, unutar klase **PolynomialBuilder**, podržane su tri strategije:

Generiranje Newtonovog fraktala iz zadanih nultočaka funkcije

Kreće se od poznatih nultočaka funkcije. Na temelju njih gradi se pripadajući polinom i njegova derivacija. Prednost ovog pristupa je u tome što se nultočke unaprijed znaju.

```
@staticmethod
def create_for_function_zeros(
    zeros: List[complex]
) -> Tuple[np.poly1d, np.poly1d]:
    poly_coeffs = np.poly(zeros)
    dpoly_coeffs = np.polyder(poly_coeffs)

    return np.poly1d(poly_coeffs), np.poly1d(dpoly_coeffs)
```

Generiranje Newtonovog fraktala iz zadanih nultočaka derivacije funkcije

Polazi se od poznatih nultočaka prve derivacije funkcije. Na temelju njih rekonstruira se derivacija, a zatim integracijom pripadajuća funkcija. Budući da nultočke same funkcije u početku nisu poznate, potrebno ih je naknadno izračunati.

```
@staticmethod
def create_for_derivate_zeros(
    zeros: List[complex],
    f_const: int=0,
```

```

    f_scale: int=1,
) -> Tuple[np.poly1d, np.poly1d]:
    dpoly_coeffs = f_scale * np.poly(zeros)
    poly_coeffs = np.polyint(dpoly_coeffs, k=f_const)

    return np.poly1d(poly_coeffs), np.poly1d(dpoly_coeffs)

```

Generiranje Newtonovog fraktala iz zadane funkcije

Kao polazište uzima se analitički izraz funkcije. Uz pomoć simboličke obrade generira se i derivacija, nakon čega se obje transformiraju u numerički oblik. Nultočke se zatim određuju računanjem.

```

@staticmethod
def create_for_function_expression(
    f_expr: Union[sp.Expr, Callable[[np.ndarray], np.ndarray]]
) -> Tuple[np.poly1d, np.poly1d]:
    z = sp.Symbol('z')
    poly = sp.Poly(f_expr, z)

    poly_coeffs = list(map(float, poly.all_coeffs()))
    dpoly_coeffs = np.polyder(poly_coeffs)

    return np.poly1d(poly_coeffs), np.poly1d(dpoly_coeffs)

```

U svim slučajevima, nakon što se definiraju funkcija, derivacija i nultočke, centralna metoda *_compute_newton_convergence* iz klase **Fractal** provodi iteracije prema formuli:

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

```

def _compute_newton_convergence(
    self,
    iterations: np.ndarray | None = None,
    zero_indices: np.ndarray | None = None
) -> None:
    converged = np.zeros(self.Z.shape, dtype=bool)

    for iteration in range(MAX_ITERATIONS):
        not_converged = ~converged
        if np.all(converged):
            break

        self.Z[not_converged] = (
            self.Z[not_converged] - self.f(self.Z[not_converged])
            / self.df(self.Z[not_converged])
        )

```

```

for index, zero in enumerate(self.f_zeros):
    mask = (np.abs(self.Z - zero) < TOLERANCE)
    & not_converged

    converged[mask] = True

    if iterations is not None:
        iterations[mask] = iteration
    if zero_indices is not None:
        zero_indices[mask] = index

```

Iteracija traje sve dok se ne postigne konvergencija ili dok se ne dosegne maksimalan broj koraka. Rezultati se zatim koriste za generiranje vizualizacije.

5.4 Načini vizualizacije

Vizualizacija je ključni korak u prezentaciji Newtonovih fraktala, jer se složenost metode najbolje očituje kroz geometrijske strukture koje nastaju. Implementirana su dva osnovna pristupa: prikaz u boji i prikaz rubova.

Prikaz u boji

Svaka nultočka funkcije dobiva jedinstvenu boju. Sve točke kompleksne ravnine koje konvergiraju prema istoj nultočki boje se tom bojom. Dodatno, broj iteracija potreban za postizanje konvergencije može modulirati nijansu ili intenzitet boje, čime se dobiva dodatna dimenzija informacija. Ovaj prikaz kroz metodu *get_colored* realizira klasa **Colorizer**.

```

@staticmethod
def get_colored(
    Z: np.ndarray,
    f_zeros: np.ndarray,
    iterations: np.ndarray
) -> np.ndarray:
    image = np.zeros(Z.shape + (3,))

    for index, zero in enumerate(f_zeros):
        mask = np.abs(Z - zero) < TOLERANCE

        color = np.array(Colorizer._get_color(index).value)

        normalized_iterations = iterations[mask] / MAX_ITERATIONS
        brightness = 1 - normalized_iterations[:, np.newaxis]

        image[mask] = color * brightness

    return image

```

Prikaz rubova

Umjesto da boji cijela područja, ovaj pristup naglašava granice između različitih regija konvergencije. Upravo na tim prijelazima dolazi do izražaja fraktalna složenost sustava. Klasa **EdgeDetector** implementira dva algoritma za identifikaciju rubova:

1. **Metoda susjeda** – uspoređuje susjedne točke u mreži i označava rubove tamo gdje se njihova konvergencija razlikuje.

```
def get_edges_neighbor(self) -> np.ndarray:
    indices_arr = self.zero_indices
    padded_indices_arr = np.pad(
        indices_arr,
        pad_width=1,
        constant_values=-1
    )

    mask = np.zeros(padded_indices_arr.shape, dtype=bool)

    # svi susjedni smjerovi: gore, dolje, lijevo, desno
    for shift, axis in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
        shifted = np.roll(padded_indices_arr, shift=shift, axis=axis)
        diff = (shifted != padded_indices_arr) \
            & (padded_indices_arr != -1) \
            & (shifted != -1)
        mask |= diff

    # maska bez paddinga
    return mask[1:-1, 1:-1]
```

2. **Metoda kružnice** – promatra širi skup susjednih točaka oko jedne točke i identificira rub ako među njima postoji različite nultočke konvergencije.

```
def get_edges_circle(
    self,
    radius: float=0.005,
    circle_points: int=4
) -> np.ndarray:
    offsets = self._get_circle_subpixel_offsets(radius, circle_points)
    mask = np.zeros(self.Z.shape, dtype=bool)

    indices_y, indices_x = np.meshgrid(
        np.arange(self.Z.shape[0]),
        np.arange(self.Z.shape[1]),
        indexing='ij'
    )
```

```

for vertical, horizontal in offsets:
    shifted_indices_y = indices_y - vertical
    shifted_indices_x = indices_x - horizontal

    shifted = map_coordinates(
        self.zero_indices.astype(float),
        [shifted_indices_y, shifted_indices_x],
        order=1,
        mode='constant',
        cval=-1
    )

    diff = (shifted != self.zero_indices)
    & (self.zero_indices != -1)
    & (shifted != -1)
    mask |= diff

return mask

```

Oba pristupa omogućuju detaljniju analizu složenih struktura koje nastaju između područja stabilne konvergencije.

6 Galerija

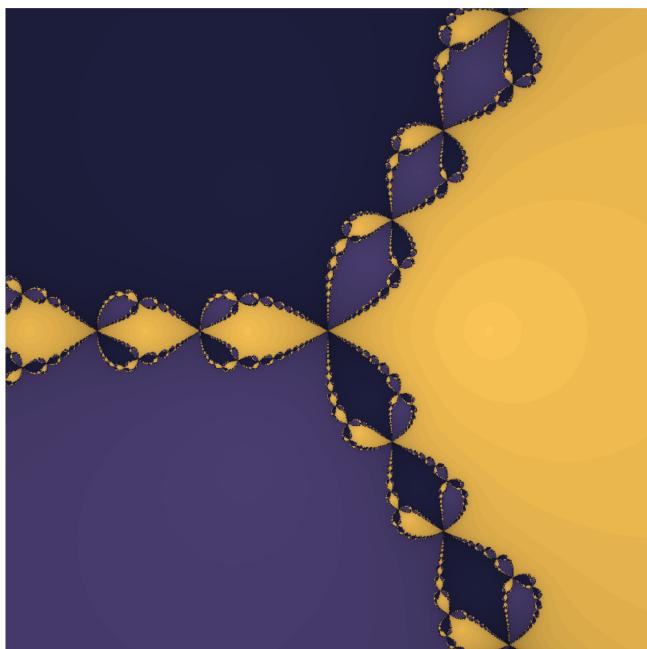
6.1 Hipoteza 1

Povećanjem stupnja polinoma raste broj područja privlačenja i složenost rubova fraktala.

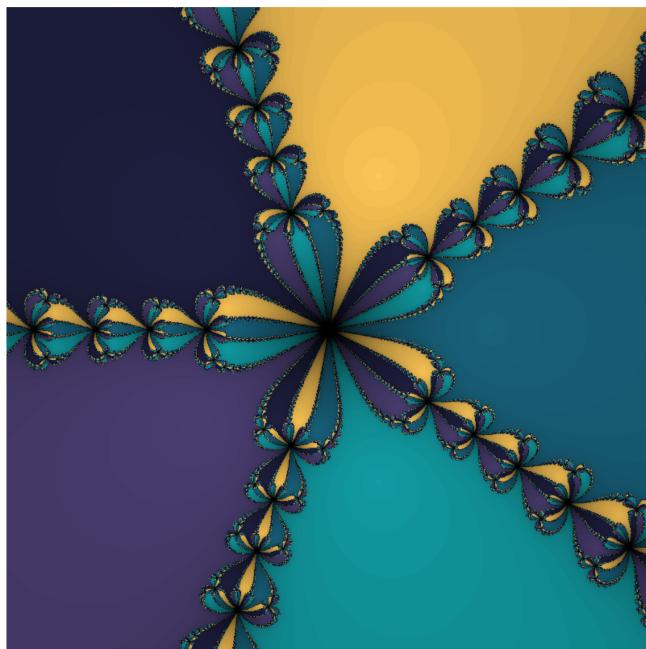
Na generiranim fraktalima može se uočiti da se broj područja privlačenja doista podudara s brojem nultočaka promatranog polinoma. Tako je, primjerice, na *Slici 3*, koja prikazuje slučaj polinoma $f(z) = z^3 - 1$, vidljivo tri simetrično raspoređena područja privlačenja. Granice između njih pokazuju određenu razvedenost, ali njihova struktura ostaje relativno jednostavna.

Nasuprot tome, *Slika 4* prikazuje fraktal dobiven za polinom $f(z) = z^5 - 1$. U ovom slučaju pojavljuje se pet područja privlačenja, raspoređenih simetrično oko ishodišta, dok su granice između njih znatno složenije i čine gusto isprepletenu mrežu. Ovaj porast složenosti uočljiv je upravo kao posljedica povećanja stupnja polinoma.

Na temelju ovih opažanja može se zaključiti da hipoteza vrijedi. S rastom stupnja polinoma raste i broj područja privlačenja, a rubovi fraktala postaju sve složeniji.



Slika 3: Newtonov fraktal za polinom trećeg stupnja: $f(z) = z^3 - 1$



Slika 4: Newtonov fraktal za polinom petog stupnja: $f(z) = z^5 - 1$

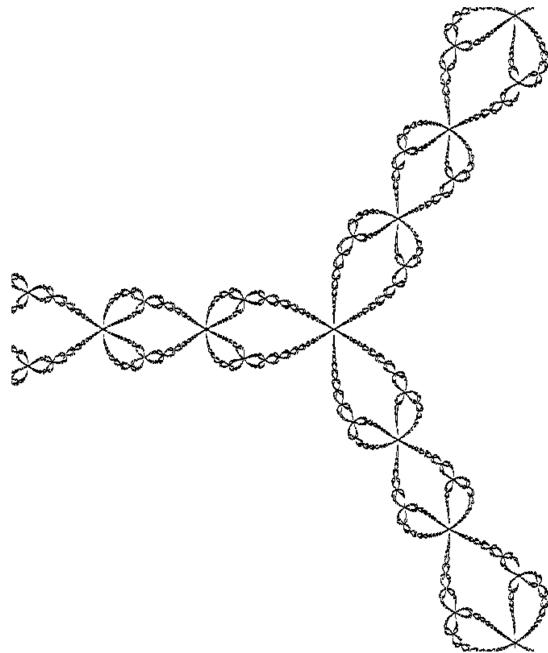
6.2 Hipoteza 2

Različite metode detekcije rubova utječe na detaljnost prikaza granica između područja privlačenja.

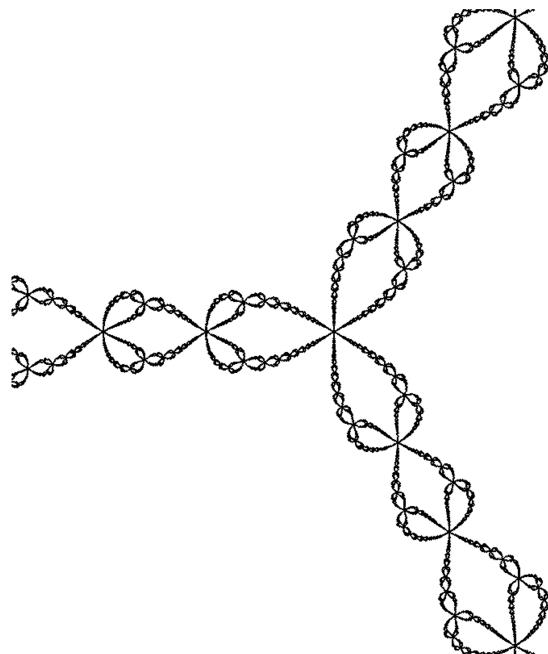
Usporedba generiranih fraktala pokazuje da odabrana metoda detekcije rubova ne mijenja samu složenost granica, već utječe na to koliko su ti rubovi detaljno prikazani. Na *Slici 5*, gdje je primijenjena metoda susjeda, granice su jasno naglašene i jednostavne za praćenje. Takav prikaz omogućuje brz uvid u osnovnu strukturu fraktala, ali sitnije nepravilnosti i razvedenosti na rubovima ostaju neprimijećene.

Na *Slici 6*, kod koje je korištena metoda kružnice, isti rubovi postaju detaljnije istaknuti. Ova metoda otkriva manje razlike i dodatne slojeve razvedenosti, čime fraktalna mreža granica djeluje bogatije i vizualno složenije, iako je sama struktura rubova u osnovi ista.

Na temelju ovih opažanja može se zaključiti da hipoteza vrijedi. Izbor metode detekcije rubova ne utječe na samu složenost fraktala, ali bitno određuje razinu detaljnosti s kojom su rubovi prikazani.



Slika 5: Rubovi Newtonovog fraktala, metoda susjeda: $f(z) = z^3 - 1$



Slika 6: Rubovi Newtonovog fraktala, metoda kružnice: $f(z) = z^3 - 1$

6.3 Hipoteza 3

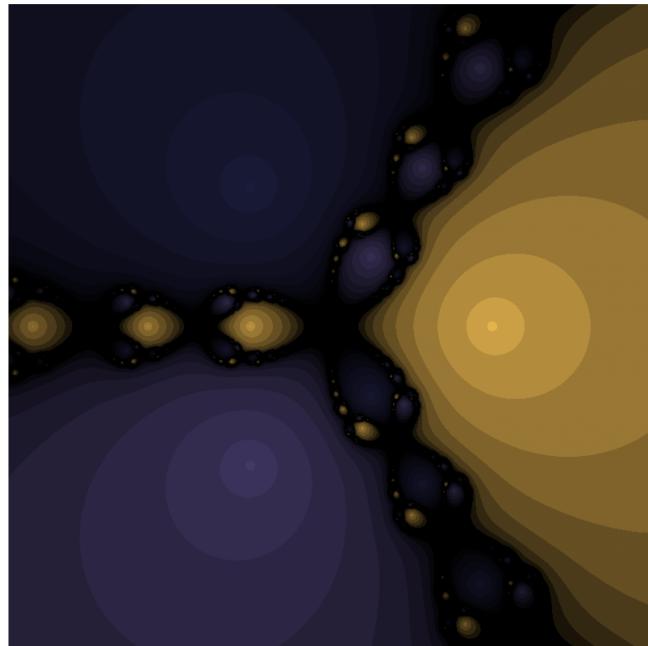
Promjenom maksimalnog broja dopuštenih iteracija nad Newtonovom metodom mijenja se detaljnost prikaza fraktala.

Rezultati generiranja fraktala pokazuju da broj dopuštenih iteracija izravno utječe na razinu detalja koji su vidljivi u prikazu. Na *Slici 7*, gdje je maksimalan broj iteracija postavljen na 10, prikazana su osnovna područja privlačenja, no granice među njima ostaju grube i manje razvedene. Manji detalji u blizini rubova nisu vidljivi zbog nedostatne preciznosti.

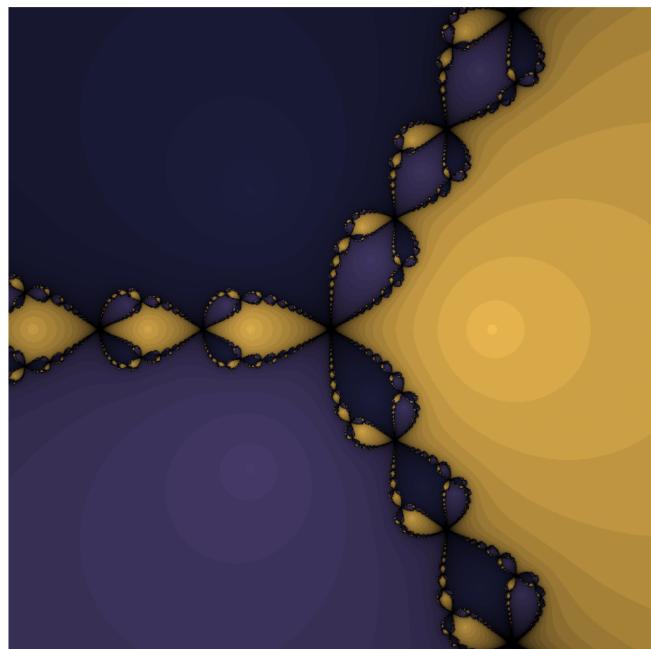
Povećanjem broja iteracija na 20, kao što je prikazano na *Slici 8*, granice postaju jasnije. Fraktal je precizniji, a rubovi pokazuju više razvedenosti nego u prethodnom slučaju.

Kod maksimalnog broja iteracija od 50, prikazanog na *Slici 9*, fraktal poprima vrlo detaljan izgled. Granice područja privlačenja postaju izrazito razvedene i uočavaju se sitne strukture koje ranije nisu bile vidljive.

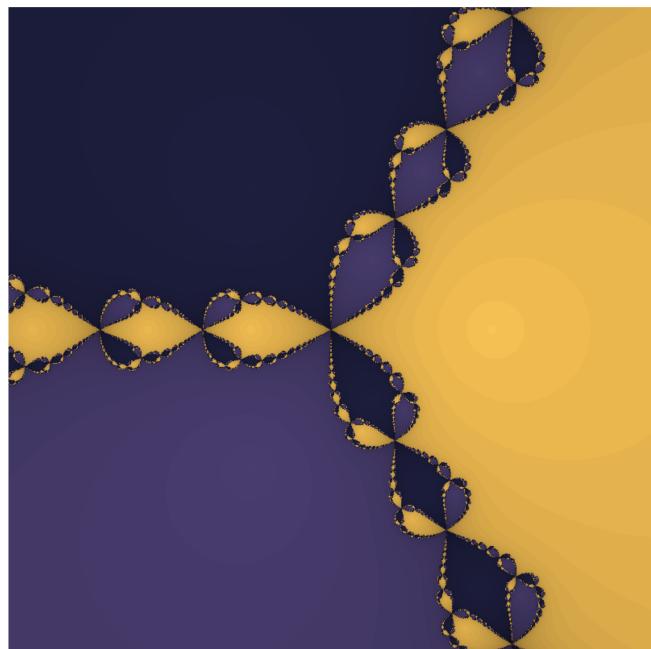
Ovi rezultati potvrđuju hipotezu da povećanjem maksimalnog broja iteracija raste detaljinost prikaza fraktala. Rubovi postaju sve jasniji i otkrivaju finije strukture koje čine fraktalnu mrežu.



Slika 7: Newtonov fraktal s maksimalnim brojem iteracija 10: $f(z) = z^3 - 1$



Slika 8: Newtonov fraktal s maksimalnim brojem iteracija 20: $f(z) = z^3 - 1$



Slika 9: Newtonov fraktal s maksimalnim brojem iteracija 50: $f(z) = z^3 - 1$

6.4 Hipoteza 4

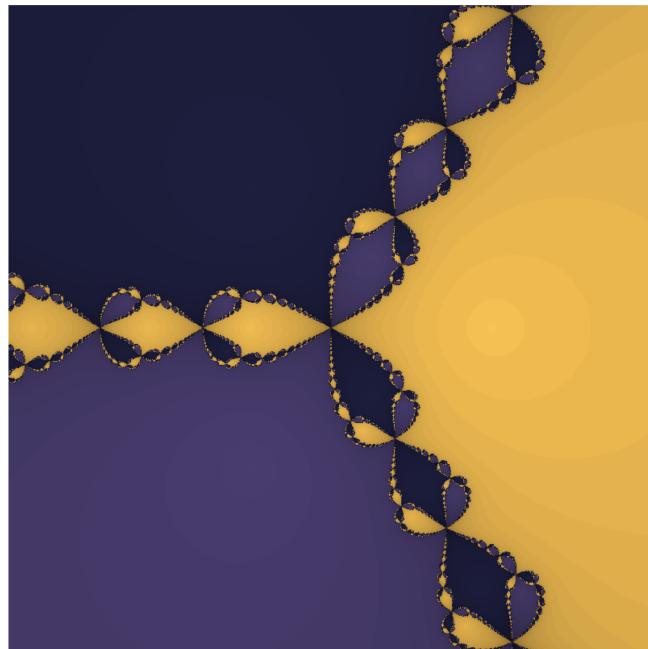
Promjenom konstante pri integriranju dolazi do promjene u prikazu fraktala, pri čemu se uočava efekt povećanja u središnjem dijelu.

Rezultati generiranja fraktala pokazuju da odabir različite konstante pri integriranju značajno utječe na način na koji se fraktal vizualno prikazuje. Na *Slici 10*, kod konstante -1 , struktura fraktala oko središta ostaje kompaktna i pregledna, bez izraženog naglašavanja detalja u centru.

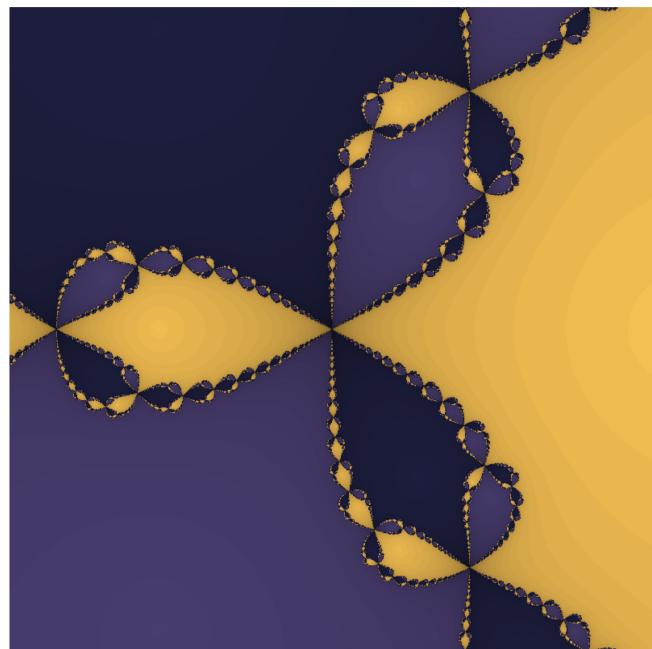
Kod konstante -10 , prikazane na *Slici 11*, uočava se efekt povećanja u središnjem dijelu u odnosu na prethodnu sliku. Centralne strukture postaju izraženije i uzimaju veći dio prikaza, dok se okolni detalji povlače u drugi plan.

Ovaj efekt dodatno se pojačava kod konstante -30 , prikazane na *Slici 12*, gdje fraktal u središtu dominira prikazom i stvara dojam intenzivnog uvećanja.

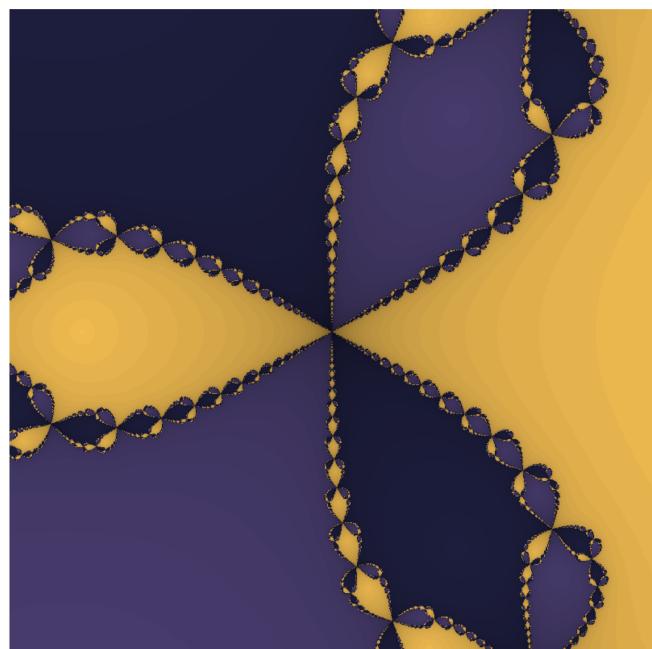
Na temelju ovih opažanja potvrđuje se hipoteza da promjena konstante pri integriranju ne mijenja samu strukturu fraktala, već utječe na njegov prikaz tako da dolazi do vizualnog povećanja u središnjem dijelu.



Slika 10: Newtonov fraktal s konstantom integracije -1



Slika 11: Newtonov fraktal s konstantom integracije -10



Slika 12: Newtonov fraktal s konstantom integracije -30

7 Zaključak

Newtonova metoda, premda nastala iz potrebe za praktičnim i brzim rješavanjem jednadžbi, pokazuje svoju drugu, neočekivanu stranu kada se promatra kroz prizmu fraktalne geometrije. Dobiveni Newtonovi fraktali ne predstavljaju samo ilustraciju djelovanja algoritma, već otkrivaju složenost graničnih područja u kompleksnoj ravnini, gdje mala promjena početnog uvjeta vodi prema radikalno različitim ishodima. Time se potvrđuje da jednostavna iterativna pravila mogu generirati strukture izuzetne složenosti, što je ujedno i temelj teorije dinamičkih sustava. Rad je pokazao kako se matematička metoda može transformirati u vizualni prikaz koji ne samo da ima znanstvenu vrijednost, nego i estetsku privlačnost. Buduća istraživanja mogu uključivati usporedbe Newtonovih fraktala s drugim vrstama fraktala, kao i analizu njihove primjene u računalnoj grafici i vizualizaciji podataka.

Literatura

- [1] A. Burton. Newton's method and fractals. Whitman College, 2009. <https://www.whitman.edu/documents/Academics/Mathematics/burton.pdf>.
- [2] T. Sauer. *Numerical Analysis*. Pearson, 2006.
- [3] A. Galántai. The theory of newton's method. *Journal of Computational and Applied Mathematics*, 124(1–2):25–44, 2000.
- [4] R. L. Devaney. *A First Course in Chaotic Dynamical Systems*. Westview Press, 1992.
- [5] B. T. Polyak. Newton's method and its use in optimization. *European Journal of Operational Research*, 181(3):1086–1096, 2007.
- [6] P. Cayley. Desiderata and suggestions: No. 3. the newton-fourier imaginary problem. *American Journal of Mathematics*, 2(1):97–97, 1879.
- [7] H.-O. Peitgen and P. H. Richter. *The Beauty of Fractals: Images of Complex Dynamical Systems*. Springer, 1986.
- [8] C. R. Harris, K. J. Millman, S. J. van der Walt, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.
- [9] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. Sympy: Symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.
- [10] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

Popis slika

1	Newtonov fraktal za polinom $f(z) = z^3 - 1$	5
2	Newtonov fraktal za polinom $f(z) = z^4 - 1$	5
3	Newtonov fraktal za polinom trećeg stupnja: $f(z) = z^3 - 1$	13
4	Newtonov fraktal za polinom petog stupnja: $f(z) = z^5 - 1$	14
5	Rubovi Newtonovog fraktala, metoda susjeda: $f(z) = z^3 - 1$	15
6	Rubovi Newtonovog fraktala, metoda kružnice: $f(z) = z^3 - 1$. . .	15
7	Newtonov fraktal s maksimalnim brojem iteracija 10: $f(z) = z^3 - 1$	16
8	Newtonov fraktal s maksimalnim brojem iteracija 20: $f(z) = z^3 - 1$	17
9	Newtonov fraktal s maksimalnim brojem iteracija 50: $f(z) = z^3 - 1$	17
10	Newtonov fraktal s konstantom integracije -1	18
11	Newtonov fraktal s konstantom integracije -10	19
12	Newtonov fraktal s konstantom integracije -30	19