Diskretne kaotične mape

Malinović, Adrijan

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

Sveučilišni preddiplomski studij računarstva

DISKRETNE KAOTIČNE MAPE

Završni rad

Adrijan Malinović

Osijek, 2023.

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

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

Osijek, 20.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Adrijan Malinović
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4535, 27.07.2020.
OIB Pristupnika:	67296426450
Mentor:	prof. dr. sc. Davor Vinko
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Diskretne kaotične mape
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak završnog rad:	Zadatak završnog rada je napraviti pregled postojećih diskretnih kaotičnih mapa i usporediti njihova svojstva. Za više detalja javiti se mentoru: davor.vinko@ferit.hr
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda. Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda. Jasnoća pismenog izražavanja: 2 bod/boda. Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	20.09.2023.
Datum potvrde ocjene od strane Odbora:	24.09.2023.

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, 25.09.2023.

Ime i prezime studenta:	Adrijan Malinović
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4535, 27.07.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom **: Diskretne kaotične mape**

izrađen pod vodstvom mentora prof. dr. sc. Davor Vinko

i sumentora ,

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 koj i mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Ovaj rad bavit će se proučavanjem diskretnih kaotičnih mapa, koje predstavljaju posebnu klasu dinamičkih sustava s kaotičnim ponašanjem. Kaos je fenomen koji se javlja u mnogim prirodnim i matematičkim sustavima, karakteriziran visokom osjetljivošću na početne uvjete i nepredvidivim dugoročnim ponašanjem. Diskretne kaotične mape pružaju jednostavan matematički model za proučavanje ovog složenog fenomena.

U okviru teorijske pozadine, pružit će se uvod u teoriju kaosa kako bi se steklo temeljno razumijevanje ovog fenomena. Razmotrit će se osnovni koncepti kao što su osjetljivost na početne uvjete, determinizam i nepredvidivost.

Nadalje, istražit će se dinamički sustavi i diskretne kaotične mape. Objasnit će se njihova struktura i osnovne karakteristike. Uz to, istaknut će se primjeri diskretnih kaotičnih mapa koje se često koriste u istraživanju kaosa, uključujući logističku mapu i Hénonovu mapu, gdje svaka mapa ima specifične osobine koje će se proučiti i analizirati.

Za bolje razumijevanje i vizualiziranje ponašanja diskretnih kaotičnih mapa, implementacija će biti u programskom jeziku Python. Rad će se usredotočiti na jednodimenzionalne i dvodimenzionalne mape, pružajući primjere kodova i dijagrama, koristeći biblioteke *numpy* i *matplotlib*.

Također, proučit će se donekle primjena diskretnih kaotičnih mapa te će njihova važnost biti istaknuta u različitim područjima. Objasnit će se zašto se diskretne kaotične mape mogu koristiti za generiranje pseudo slučajnih brojeva koji su od velikog značaja u različitim područjima, poput računalnih simulacija i sigurnosnih protokola. Razmotriti će se gdje se diskretne kaotične mape mogu primijeniti u kriptografiji, pružajući visoku razinu sigurnosti u algoritmima šifriranja. Osim toga, spomenut će se i gdje se ove mape koriste u modeliranju dinamičkih sustava, omogućavajući razumijevanje i analizu kompleksnih procesa u prirodi, ekonomiji, biologiji i drugim područjima.

1.1. Zadatak završnog rada

Zadatak završnog rada je napraviti pregled postojećih diskretnih kaotičnih mapa i usporediti njihova svojstva.

2. TEORIJSKA POZADINA

Poglavlje 2 pruža teorijsku pozadinu koja je ključna za razumijevanje diskretnih kaotičnih mapa, s fokusom na teoriju kaosa i općenito na koncept kaotičnih mapa. Ovaj dio služi kao uvod u poglavlje, pružajući pregled tema koje će biti obuhvaćene u potpoglavljima 2.1 i 2.2.

U potpoglavlju 2.1, istražiti će se teorija kaosa koja pomaže u razumijevanju temeljnih karakteristika kaotičnih sustava. Ovdje se neće ulaziti previše u matematičke detalje, već će fokus biti na ključnim konceptima i idejama. Razmotrit će se povijest kaosa kao znanstvene discipline i istaknuti značajke poput osjetljivosti na početne uvjete i nepredvidivosti dugoročnog ponašanja. Također, spomenut će se famozni "efekt leptira" koji ilustrira kako male promjene u početnim uvjetima mogu imati značajan utjecaj na konačni rezultat.

U potpoglavlju 2.2, rasprava će se proširiti na općenito shvaćanje kaotičnih mapa. Osim diskretnih kaotičnih mapa koje su fokus ovog rada, postoje i razni drugi dinamički sustavi te postoje i neprekidne kaotične mape koje se opisuju diferencijalnim jednadžbama. Njih se neće spominjati u ovom radu, no on će se kratko dotaknuti dinamičkih sustava jer će nam razumijevanje njihovih karakteristika pomoći u kritičkom promišljanju o diskretnim kaotičnim mapama koje će se proučavati detaljnije.

Kroz ova potpoglavlja, cilj je stvoriti temeljno razumijevanje teorije kaosa i dinamičkih sustava koje će biti od suštinske važnosti za daljnje istraživanje diskretnih kaotičnih mapa. Ova teorijska pozadina omogućit će da se prepoznaju glavna obilježja, obrasci i fenomeni koji se javljaju u kaotičnim sustavima te ih primijeniti na konkretne primjere diskretnih kaotičnih mapa koje će se istražiti kasnije u radu.

2.1. Uvod u teoriju kaosa

U sljedećim odjeljcima, rad će se usredotočiti na temeljne koncepte teorije kaosa koji su relevantni za temu diskretnih kaotičnih mapa. Poznavanje povijesti i temeljnih saznanja o teoriji kaosa bit će od pomoći za dublje istraživanje diskretnih kaotičnih mapa i njihovu primjenu u različitim područjima.

2.1.1. Povijesna vrijednost

Teorija kaosa, kao posebna disciplina u znanosti, ima relativno kratku povijest, no njen utjecaj i značaj su iznimni. Razvoj teorije kaosa započeo je u drugoj polovici 20. stoljeća, kada su znanstvenici počeli proučavati složene sustave i njihovo nepredvidivo ponašanje. Ova nova grana znanosti nastala je kao rezultat istraživanja u različitim područjima, uključujući matematiku, fiziku, biologiju, ekonomiju i računalne znanosti.

Iako su pojedini koncepti koji se vežu uz teoriju kaosa postojali i prije njenog formalnog nastanka, ključni koraci prema otkrivanju i priznanju kaotičnih fenomena dogodili su se u drugoj polovici 20. stoljeća. Jedan od jako važnih trenutaka u povijesti teorije kaosa, prema [1, *Birth of chaos theory, Poincaré and phase space*], je rad francuskog matematičara Henrija Poincaréa s kraja 19. i početka 20. stoljeća. Poincaré je bio pionir u proučavanju dinamičkih sustava, posebno u kontekstu problema trojnih tijela u astronomiji. Njegovo istraživanje je pokazalo da čak i manje razlike u početnim uvjetima mogu dovesti do potpuno različitih rezultata. Ovaj koncept je kasnije postao poznat kao "osjetljivost na početne uvjete" i predstavlja temeljnu karakteristiku kaotičnih sustava.

Ipak, tek u 1960-ima teorija kaosa kao takva počinje dobivati prepoznavanje. Edward Lorenz je bio meteorolog koji je istraživao atmosferske modele pomoću računalnih simulacija. U jednom od svojih eksperimenata, prema [1, *Rebirth of chaos theory, Lorenz and the butterfly effect*], Lorenz je primijetio da je vrlo mala promjena u početnim uvjetima, poput zaokruživanja brojeva na šest decimala, imala dramatičan utjecaj na ishode dugoročne prognoze vremena. Tako je došao do zaključka da leptir koji zamahne krilima negdje u Brazilu može izazvati lančanu reakciju događaja koja dovodi do stvaranja tornada negdje drugdje na svijetu. Ovaj fenomen je popularno nazvan "efekt leptira", a Lorenz je postao pionir u proučavanju determinističke nepredvidivosti.

U idućim desetljećima, prema [1, *The golden age of chaos theory*], teorija kaosa nastavila je rasti i razvijati se. Brojni znanstvenici iz različitih područja, kao što su Mitchell Feigenbaum, David Ruelle, Otto Rössler, Benoît Mandelbrot i mnogi drugi, pridonijeli su razumijevanju i primjeni teorije kaosa.

Važnost teorije kaosa proteže se i izvan znanstvene zajednice. Ona je pružila dublje razumijevanje složenosti svijeta oko nas i pomogla da shvatimo da su neki fenomeni intrinzično nepredvidivi, unatoč njihovoj determinističkoj prirodi. Prema [2, 3], ova spoznaja ima široku primjenu u raznim područjima kao što su biologija i medicina, ekonomija, meteorologija, računalne znanosti i mnoge druge. Teorija kaosa je postala neizostavni alat za proučavanje složenih sustava i razumijevanje njihovih dinamičkih obrazaca.

2.1.2. Temeljna saznanja

Dakle, teorija kaosa je grana znanosti koja proučava složene dinamičke sustave koji pokazuju osjetljivost na početne uvjete i deterministčku nepredvidivost. Ova disciplina temelji se na nekoliko ključnih koncepta i principa koji su ključni za analizu kaotičnih fenomena. U ovom odjeljku istražit će se ta temeljna saznanja o teoriji kaosa.

Konkretnije, istražiti će se neki od pojmova koji se često koriste kada se priča o kaotičnim sustavima te o teoriji kaosa u matematičkom smislu , a to su:

- osjetljivost na početne uvjete
- determinizam
- *fraktali*
- nestabilnost i *bifurkacije*
- *atraktori*

Jedan od osnovnih koncepta teorije kaosa je osjetljivost na početne uvjete. To znači da mali pomaci u početnim uvjetima mogu dovesti do značajnih promjena u konačnom rezultatu sustava. Ovaj koncept je najbolje ilustriran već spomenutim izrazom “efekt leptira”, koji sugerira da jedan jednostavan događaj može imati dalekosežne posljedice. Osjetljivost na početne uvjete objašnjava zašto je predviđanje dugoročnih rezultata u kaotičnim sustavima iznimno teško, čak i ako su poznate početne vrijednosti.

Unatoč svojoj nepredvidljivosti, kaotični sustavi su deterministički. To znači da se njihovo ponašanje u potpunosti određuje matematičkim jednadžbama i početnim uvjetima. Ali ipak, zbog osjetljivosti na početne uvjete, male razlike u ulaznim vrijednostima mogu rezultirati potpuno različitim izlaznim vrijednostima. Ova paradoksalna kombinacija determinizma i nepredvidljivosti čini kaos fascinantnim područjem za istraživanje.

Fraktali su geometrijski oblici koji se ponavljaju na svim skalama različitih reda veličina, odnosno koliko kod mala ili velika bila "reprezentacija" tih oblika i struktura (bilo to prikaz iz nekog dijagrama ili primjer iz stvarnog svijeta), oni nikada neće biti jednostavne strukture poput nekih drugih geometrijskih oblika kao recimo ravna linija (koja je jednake složenosti i njezin prikaz izgledat će "glatko" bez obzira na kojoj brojčanoj skali njezinih parametara ju promatramo). Ovo nije tako kod *fraktala*, čija će struktura uvijek ostati kompliciranа koliko god približimo njihov prikaz. Oni su često povezani s kaosom i mogu se koristiti za vizualizaciju i analizu kaotičnih sustava, a također mogu biti od važnosti u razumijevanju diskretnih kaotičnih mapa jer one često proizvode *fraktalne* obrasce.

Kaotični sustavi često prolaze kroz faze nestabilnosti i *bifurkacija*. *Bifurkacije* su promjene u ponašanju sustava koje se događaju kada se mijenja neki parametar u matematičkom modelu. One mogu rezultirati nastankom novih dinamičkih oblika, poput periodičnih ili kaotičnih oscilacija, što je temeljno za proučavanje diskretnih kaotičnih mapa jer one često proizvode složene oblike ponašanja.

Atraktor je pojam koji u kontekstu kaotičnih sustava predstavlja putanju ili putanje kojima se sustav ima tendenciju kretati tijekom vremena. Atrakcija prema određenom *atraktoru* rezultat je kompleksnih interakcija između komponenti sustava. Diskretne kaotične mape mogu imati različite *atraktore*, uključujući stabilne fiksne točke, periodične putanje ili čak *atraktore* koji pokazuju složenije oblike kao što su *fraktali*.

Ovu su samo neki od pojmova teorije kaosa koji su od značaja za razumijevanje diskretnih kaotičnih mapa. Osjetljivost na početne uvjete objašnjava zašto diskretne kaotične mape mogu generirati nepredvidive rezultate čak i s jednostavnim ulaznim vrijednostima. Determinizam omogućuje spoznaju da su ove mape definirane matematičkim pravilima. *Fraktali* pružaju vizualnu reprezentaciju njihovih složenih oblika. Nestabilnosti i *bifurkacije* omogućuju da pratimo kako se mape mijenjaju s promjenama parametara, a *atraktori* su ključni za razumijevanje putanja kojima se mape mogu kretati.

2.2. Uvod u dinamičke sustave i diskretne kaotične mape

Daljnji dio rada usredotočit će se na dinamičke sustave i diskretne kaotične mape. Dinamički sustavi su matematički modeli koji opisuju promjene stanja tijekom vremena i imaju široku primjenu u mnogim disciplinama. Diskretne kaotične mape su poseban tip dinamičkih sustava koji se koriste za modeliranje i analizu dinamičkih procesa u diskretnim vremenskim koracima. One nam pružaju alate za razumijevanje složenih i nepredvidivih fenomena koji se javljaju u različitim područjima.

Spomenut ćemo njihove karakteristike i reći nešto o njihovom ponašanju, jer je razumijevanje tih koncepta bitno za praktičan dio našeg istraživanja.

2.2.1. O dinamičkim sustavima

Dinamički sustavi su matematički modeli koji opisuju promjene stanja tijekom vremena. Ključni su za razumijevanje prirodnih i drugih procesa koji se odvijaju u svijetu oko nas, a mogu se primijeniti na različite discipline, uključujući fiziku, biologiju, ekonomiju, sociologiju i mnoge druge.

Prema [4, 5], teorija o dinamičkim sustavima uključuje proučavanje njihove strukture, svojstava i evolucije. Važni elementi u njihovoj analizi su stanja, putanje i zakoni koji određuju njihovu promjenu. Stanja sustava predstavljaju vrijednosti varijabli koje opisuju sustav u određenom trenutku, dok putanje predstavljaju evoluciju tih stanja tijekom vremena. Analiza dinamičkih sustava uključuje proučavanje njihovih obilježja kao što su stabilnost, periodičnost, oscilacije i kaos. Što se tiče stabilnosti, ona se odnosi na svojstvo sustava da se vrati u određeno stanje nakon malih promjena. Periodičnost se javlja kada sustav prolazi kroz ponavljajuće cikluse stanja, a oscilacije se manifestiraju kao ponavljajući izmjenični procesi između različitih stanja sustava. Kaotični sustavi predstavljaju poseban tip dinamičkih sustava koji pokazuju osjetljivost na početne uvjete i iznimnu složenost u svojoj evoluciji, a kako je već i rečeno, karakteristično za njih je njihova ekstremna osjetljivost na male promjene u početnim uvjetima, što naravno rezultira nepredvidivim ponašanjem. Ova svojstva često rezultiraju nelinearnim i kompleksnim dinamikama koje su teške za analizu klasičnim matematičkim metodama.

Slijedeći odjeljci i potpoglavlja fokusirat će se isključivo na diskretne kaotične mape, koje predstavljaju posebne vrste diskretnih dinamičkih sustava s izraženom kaotičnom dinamikom.

2.2.2. O diskretnim kaotičnim mapama

Prema [6, 7], diskretne kaotične mape predstavljaju poseban tip kaotičnih sistema koji se koriste za modeliranje i analizu dinamičkih procesa. Ove mape pružaju jednostavan matematički okvir za proučavanje kaotične evolucije sustava u diskretnom vremenskom koraku te se nadovezuju na koncepte dinamičkih sustava, pružajući mogućnost spoznaje složenih i nepredvidivih dinamičkih procesa.

Diskretne kaotične mape nisu ništa drugo nego iterativne matematičke funkcije. One se primjenjuju na početnu vrijednost ili sjeme (eng. *initial value, seed*) i rezultat se koristi kao ulaz za sljedeću iteraciju. Ovaj proces ponavljanja stvara nizove vrijednosti koje predstavljaju evoluciju sustava kroz vremenske korake. Iterativne funkcije mogu biti jednostavne matematičke formule ili kompleksnije kombinacije različitih operacija. Postoji mnogo različitih diskretnih kaotičnih mapa, a ovaj rad će se s nekima od najpoznatijih baviti u sljedećem poglavlju.

Mogu biti jednodimenzionalne, dvodimenzionalne ili imati još više dimenzija. Svaka dimenzija predstavlja jedan parametar čije se stanje prati kroz evoluciju sustava. Osim ovih "evolucijskih" parametara često postoje i neki dodatni parametri koji mogu mijenjati oblik i ponašanje sustava te nam omogućiti proučavanje različitih aspekata kaotične dinamike.

Putanja (eng. *orbit*) u ovom kontekstu predstavlja seriju točaka koja se generira iterativnim procesom. Putanje igraju veoma važnu ulogu u proučavanju obilježja diskretnih kaotičnih mapa jer njihovo ponašanje može biti izuzetno komplicirano. Moguće su različite vrste putanja, uključujući periodične cikluse, kvazi-periodične putanje i kaotične *atraktore*. Periodične putanje su putanje koje se ponavljaju u određenim vremenskim intervalima. One se mogu manifestirati kao stabilni ciklusi s točno definiranim periodom, što znači da se vrijednosti na putanji ponavljaju u pravilnim intervalima. Kvazi-periodične putanje su putanje koje pokazuju neku vrstu redovitosti, ali ne sasvim točno periodičko ponašanje. To mogu biti bliske vrijednosti koje se ponavljaju s manjim odstupanjima. Kaotični *atraktori* su složene putanje koje se karakteriziraju osjetljivošću na početne uvjete. Što nosi sa sobom činjenicu da iako se može početi s vrlo bliskim početnim uvjetima, putanje će s vremenom divergirati jedna od druge i pratiti sasvim različite vrijednosti.

Kaotični *atraktori* obično imaju *fraktalnu* strukturu, tj. ista složena struktura se ponavlja na različitim razinama skaliranja.

Putanje se mogu proučavati i istraživati koristeći različite metode vizualizacije. Jedan od najjednostavnijih načina je grafički prikazivanje putanja u faznom prostoru, gdje se vrijednosti na putanji prikazuju kao točke u višedimenzionalnom prostoru. Osim toga, mogu se koristiti i druge metode, poput *bifurkacijskih* dijagrama, Lyapunovih eksponenata i drugih tehnika za analizu.

3. IMPLEMENTACIJA I VIZUALIZACIJA U PYTHON-U

Sljedeće poglavlje bavit će se implementacijom i vizualizacijom diskretnih kaotičnih mapa. Odnosno, pokazati će se kako se kroz skripte napisane u programskom jeziku Python može simulirati evoluciju kaotičnih mapa i pratiti njihovo stanje tijekom vremena te će se također pokazati kako se to stanje može prikazati pomoću dijagrama. Za ove potrebe koristiti će se biblioteke *NumPy* i *Matplotlib*.

3.1. Logistička mapa

Logistička mapa jedna je od najpoznatijih diskretnih kaotičnih mapa, ali je isto tako i jedna od najjednostavnijih za implementirati. Može se opisati jednostavnom matematičkom funkcijom koja glasi ovako:

$$x_{n+1} = r x_n (1 - x_n)$$

Parametri x_n i x_{n+1} predstavljaju određeni korak u evoluciji funkcije, odnosno trenutni i sljedeći korak (počinjemo od x_0, vrijednost *n* predstavlja broj iteracije), a dodatni parametar *r* se dodjeljuje na početku te o njemu i o vrijednosti x_0 ovisi kakva će biti evolucija, tj. to su početni uvjeti.

To se u Python-u može napraviti vrlo lako, definicijom funkcije koja kao ulaz uzima trenutnu vrijednost x_n i parametra *r* te kao povratnu vrijednost vrati izračunati x_{n+1}.

```
def logistic_map_iteration(r: float, current_value: float) -> float:
    next_value = r * current_value * (1 - current_value)
    return next_value
```

Slika 3.1. Implementacija izračuna jednog koraka iteracije

Međutim, potrebno je više od samo jednog koraka iteracije za smislenu vizualizaciju, jer se postavlja pitanje što se s ovim procesom dogodi kada ga iteriramo određeni broj puta. Može se sada uzeti napravljena funkcija te nju koristiti da bi izračunali cijeli niz uzastopnih koraka iteracije.

Nova funkcija će kao argument također primiti parametar *r*, kao i parametre *seed* i *iterations*, koji označavaju početnu vrijednost x_0 i broj iteracija do kojih će se računati evoluciju. Ideja je da ova nova funkcija koristi prošlu funkciju kako bi izračunala i vratila niz (predstavljen *NumPy* objektom *ndarray*, eng. *N-dimensional array*, u ovom slučaju je to jednodimenzionalni niz) koji predstavlja trenutne vrijednosti x_n u svakom koraku. Kako bi se to učinilo mora se prvo inicijalizirati prazan niz *NumPy* metodom *empty*, kojoj će se kao argument poslati oblik n-dimenzionalnog polja kojeg se želi inicijalizirati te tip podataka koji će biti sadržan u polju. Zatim će se u taj prazan niz staviti početna vrijednost x_0 metodom *append*, koja nadoda predani element predanom nizu te novi niz vrati kao rezultat. Sada se može običnim prolazom kroz petlju slijedno pozivati funkciju sa slike 3.1. i time iterirati logističku mapu, a svaku izračunatu trenutnu vrijednost tijekom svakog koraka nadodati na izlazni niz. Na kraju se samo vrati rezultat.

```
def logistic_map_array(r: float, seed: float, iterations: int) -> np.ndarray:
    empty_array = np.empty(shape=(0,), dtype=float)
    output_array = np.append(empty_array, seed)
    for n in range(iterations):
        output_array = np.append(
            output_array,
            logistic_map_iteration(r, output_array[n])
        )
    return output_array
```

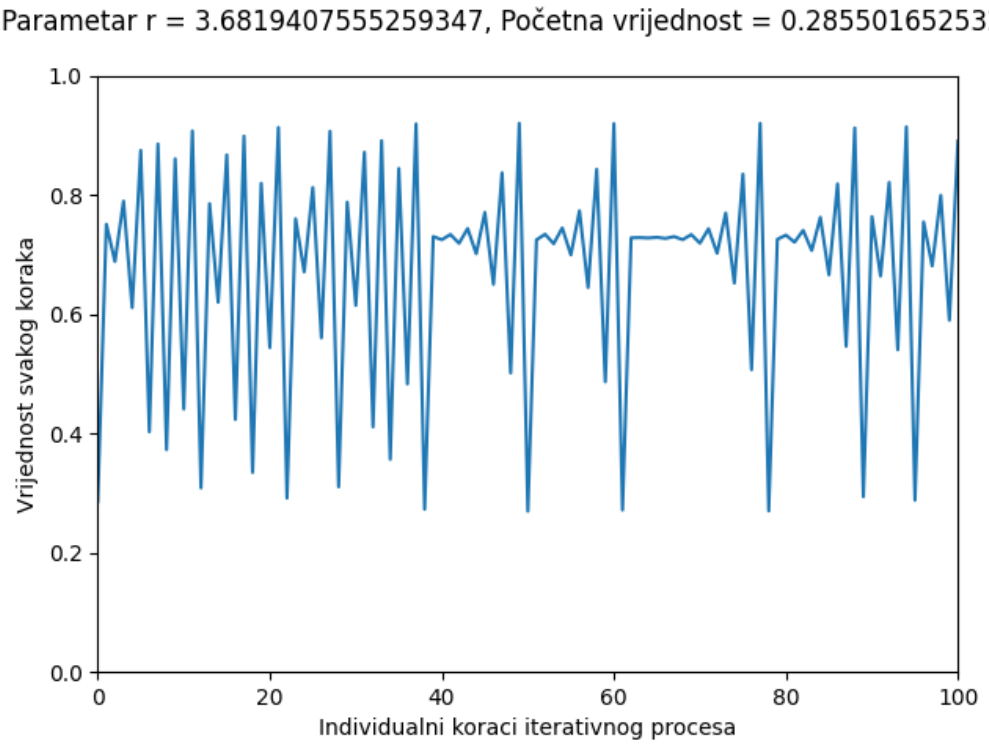
Slika 3.2. Implementacija izračuna evolucije mape nakon određenog broja iteracija

Sada će se izabrati neke početne vrijednosti te vizualizirati vraćeni niz za te parametre, za to će se koristiti *pyplot* modul biblioteke *Matplotlib* pod poznatim aliasom *plt*. Za početak će se nasumično odabrati r i početnu vrijednost koristeći metodu *uniform* modula *random* koji je sadržan u *NumPy* biblioteci. Tu je važno napomenuti da ako se za r izabere vrijednost u intervalu od 0 do 4, tada će vrijednost x_n uvijek ostati ograničena na interval od 0 do 1. Iz toga razloga će se proučavati samo ti slučajevi u kojima je r u intervalu od 0 do 4, jer u suprotnom vrijednosti za skoro sve početne x_0 divergiraju i ispadaju iz intervala u kojemu ih je lako vizualno predočiti. Ovdje će se za r uzeti nasumično vrijednost iz intervala od 3.6 do 4 jer, kao što će se uskoro vidjeti, za dobar dio vrijednosti u tom intervalu logistička mapa odaje naznake kaotičnog ponašanja. Što se tiče početne vrijednosti, iz prethodnih razloga ona će biti iz intervala od 0 do 1, a za broj iteracija će se staviti vrijednost 100. Pozivanjem funkcije *logistic_map_array* s predanim parametrima dobiva se evoluciju koja će se predočiti dijagramom.

Sam linijski dijagram će se prikazati i urediti koristeći *pyplot* modul. Prvo će se metodi *plot* predati generirana evolucija, koja će iscrtati dobivene vrijednosti kao točke gdje x os predstavlja korake iteracije a y os brojčanu vrijednost u svakom koraku (ova metoda stvara dijagram, no on se prikazuje tek kada se pokrene metodom *show*). Metode *xlim* i *ylim* služe kako bi se ograničile x i y osi dijagrami na točno određene intervale, a pomoću *title*, *xlabel* i *ylabel* se uređuje tekst koji će ići na zaglavlje i osi prikazanog dijagrama (u ovom slučaju dodatne informacije o samom dijagramu).

```
random_r = np.random.uniform(3.6, 4)
random_seed = np.random.uniform(0, 1)
iterations = 100

evolution = logistic_map_array(r=random_r, seed=random_seed, iterations=iterations)

plt.plot(evolution)
plt.xlim(0, iterations)
plt.ylim(0, 1)
plt.title(f"Parametar r = {random_r}, Početna vrijednost = {random_seed}\n")
plt.xlabel("Individualni koraci iterativnog procesa")
plt.ylabel("Vrijednost svakog koraka")
plt.show()
```

Slika 3.3. Skripta koja iscrtava linijski dijagram logističke mape za određene uvjete



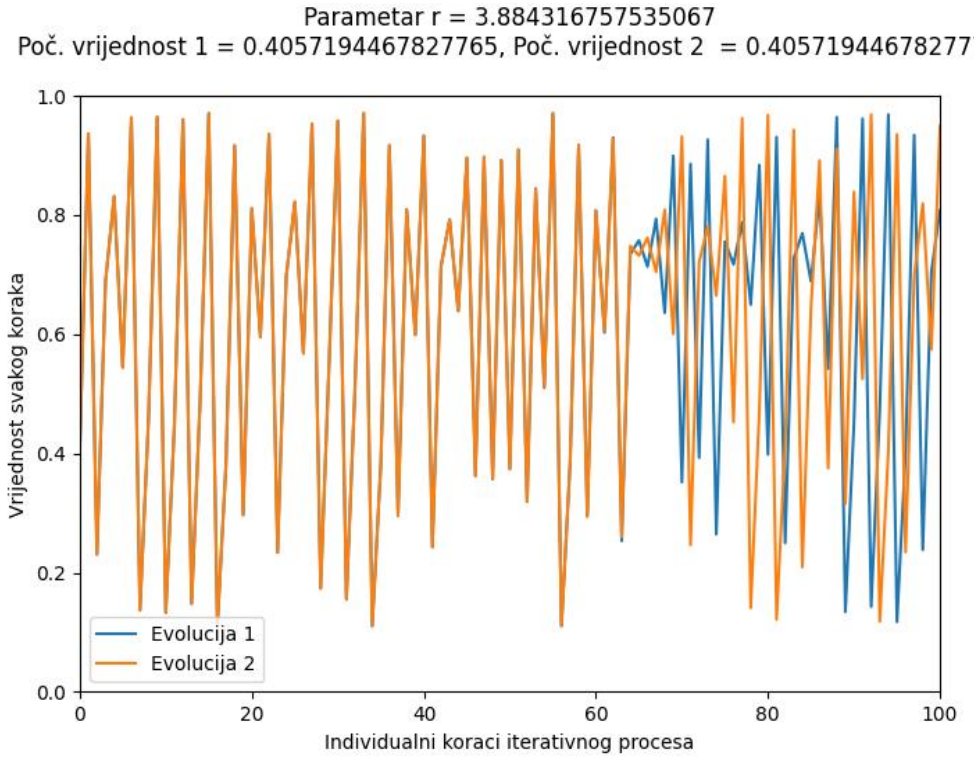
Slika 3.4. Iscrtani dijagram

Sada će se vidjeti što se dogodi kada se iscrtaju dvije evolucije sa veoma malom razlikom u početnom parametru x_0. Prošla skripta će se preurediti na način da se sada generiraju dvije početne vrijednosti, umjesto jedne. Druga će se dobiti tako da se na prvu nadodaju veoma mali pomak (reda 10^{-15}). Zatim će se svaka zasebno iscrtati na graf metodom *plot* te prikladnim oznakama. Također će se metodom *legend* prikazati navedene oznake u donjem lijevom kutu te će se prikazati i početna vrijednost druge evolucije u zaglavlju.

```
random_r = np.random.uniform(3.6, 4)
random_seed_1 = np.random.uniform(0, 1)
random_seed_2 = random_seed_1 + 0.000000000000001
iterations = 100

evolution_1 = logistic_map_array(r=random_r, seed=random_seed_1, iterations=iterations)
evolution_2 = logistic_map_array(r=random_r, seed=random_seed_2, iterations=iterations)

plt.plot(evolution_1, label="Evolucija 1")
plt.plot(evolution_2, label="Evolucija 2")

plt.legend(loc="lower left")
plt.xlim(0, iterations)
plt.ylim(0, 1)
plt.title(f"Parametar r = {random_r}\nPoč. vrijednost 1 = {random_seed_1}, Poč. vrijednost 2  = {random_seed_2}\n")
plt.xlabel("Individualni koraci iterativnog procesa")
plt.ylabel("Vrijednost svakog koraka")
plt.show()
```

Slika 3.5. Preuređena skripta

Sa slike 3.6 jasno se vidi da unatoč razlici na tek petnaestoj decimali početnih vrijednosti, već nakon sedamdesetak iteracija te dvije evolucije jasno divergiraju jedna od druge. Generalno vrijedi da sa što manjom razlikom u početnim uvjetima toliko duže treba sekvencama da divergiraju jedna od druge, no koliko god razlika bila mala, ako proporcionalno dovoljno koraka prođe, s vremenom će sekvence bitni potpuno različite i neusporedive. Naravno, točan broj koraka nakon kojeg odudaraju ovisi i o parametru r i o početnim uvjetima x_0, no donekle točna procjena se može ostvariti i sa informacijom o veličini razmaka u početnim uvjetima.



Slika 3.6. Divergiranje dviju evolucija sa bliskim početnim uvjetom x_0

Sada će se vidjeti kakve putanje nastaju za različite vrijednosti parametra r. Napraviti će se dvije nove funkcije za ovu potrebu, jedna će generirati i vratiti spektar početnih parametara, a druga će generirane parametre vizualno predočiti.

Funkciji za stvaranje nasumičnih parametara predat će se kao argumenti granice parametra *r* (u obliku niza) između kojih ćemo nasumično birati vrijednost te broj uzoraka za svaki interval *r*-a. Funkcija će kao rezultat vratiti uređeni par (u Python-u zvan *tuple*, eng. *N-tuple*, N-torka) dva dvodimenzionalna niza, jedan predstavlja spektar *r*-ova, a drugi spektar vrijednosti x_0. Inicijalizirat će se vrijednosti oba 2D niza pomoću metode *zeros*, koja vraća N-dimenzionalno polje navedenog oblika i tipa podataka (vidi sliku 3.7.) te ga popuni nulama. Oba niza će imati isti oblik, redova će imati koliko je intervala između granica *r*-a, a stupaca onoliko koliko je veličina uzorka. Razlog zašto se metodi *shape* predalo još i treću dimenziju (vrijednost dva) leži u tome da na ovaj način možemo skratiti potreban kod, budući da *zeros* vraća u ovom slučaju 3D niz, a ovdje se on sprema u dvije vrijednosti, Python će ga na automatski otpakirati u te dvije vrijednosti u obliku kakav je potreban (svaka vrijednost će u sebe spremiti odsječak 3D niza, odnosno 2D niz u ovom slučaju). Zatim će se na temelju oblika polja spektra parametara dohvatiti broj redaka i stupaca, te po njima iterirati pomoću dvije petlje. Vrijednosti parametara *r* izabrat će se nasumično (razlog zašto je leži u tome da u ovom konkretnom primjeru točne vrijednosti *r*-a nisu sasvim bitne, budući da se ovdje samo želi demonstrirati kako logistička mapa ima raznovrsno ponašanje ovisno o početnom parametru) tako da svaki red u spektru sadrži vrijednosti analogne odabranim intervalima, odnosno prvi red za prvi interval, drugi red za drugi interval itd. Parametre x_0 izabrat će se nasumično bez obzira na njihovu poziciju u matrici. Prije nego se vrate rezultati, sortirati će se *r*-ove uzlazno po redovima slijeva na desno, radi lakše vizualizacije. Metoda *argsort* sa dolje navedenim argumentima vraća indekse kako bi se sortirale vrijednosti na navedeni način, a pomoću metode *take_along_axis* i izračunatih indeksa se izvrši samo sortiranje.

```python
def logistic_map_parameters(r_boundaries: np.ndarray, sample_size: int) -> tuple[np.ndarray, np.ndarray]:
    r_parameters, initial_parameters = np.zeros(
        shape=(2, len(r_boundaries) - 1, sample_size),
        dtype=float
    )
    row_number, column_number = r_parameters.shape
    for i in range(row_number):
        for j in range(column_number):
            r_parameters[i, j] = np.random.uniform(
                r_boundaries[i], r_boundaries[i + 1]
            )
            initial_parameters[i, j] = np.random.uniform(
                0.0, 1.0
            )
    sorted_indices = np.argsort(r_parameters, axis=1)
    r_parameters = np.take_along_axis(
        r_parameters, sorted_indices, axis=1
    )
    return r_parameters, initial_parameters
```

Slika 3.7. Funkcija koja nam vraća spektar početnih parametara za zadanu veličinu

Povratne vrijednosti funkcije *logistic_map_parameters* će služiti kao argumenti funkcije sa slike 3.8., a navest će se i broj iteracija. Funkcijom *logistic_map_visualisation* će se spektri početnih vrijednosti iscrtati u cijeli niz pod dijagrama, za što će se najprije koristiti metodu *subplots*. Ona će za predane dimenzije inicijalizirati sveukupnu sliku dijagrama (*fig*, eng. *figure*) te omogućiti pristup pojedinom pod dijagramu pomoću varijable *ax* (eng. *axes*) u koju se sprema dvodimenzionalno polje objekata *Axes*, gdje svaki objekt predstavlja jedan pod dijagram. Iteriranjem kroz sve *Axes* objekte za određene indekse retka i stupca, slijedno se računaju putanje logističke mape (koristeći pri tome rezultate prethodne definirane funkcije, analogno za sve indekse retka i stupca) te se iscrtavaju jedan po jedan dijagram. Dodatno, osim same putanje, iscrtat će se metodom *axhline* na svakom dijagramu dvije vodoravne linije koje predstavljaju vrijednosti minimuma i maksimuma (lako se računaju za cijeli niz pomoću *NumPy* metoda *min* i *max*) evolucije, a označit će se i crvenim točkama koordinate početne i krajnje vrijednosti. Metode za crtanje i označavanje dijagrama koriste se analogno na objektima *Axes* i *pyplot*, jedine su iznimke drugačija imena metoda *set_xlim*, *set_ylim* i *set_title* te pozivanje metoda *tight_layout* (ova metoda "stegne" pod dijagrame radi više prostora) i *show*, koje se i dalje pozivaju na samom *pyplot* objektu.

```python
def logistic_map_visualisation(r_parameters: np.ndarray, initial_parameters: np.ndarray, iterations: int) -> None:
    row_number, column_number = r_parameters.shape[0], r_parameters.shape[1]
    fig, ax = plt.subplots(nrows=row_number, ncols=column_number, squeeze=False)
    for i in range(row_number):
        for j in range(column_number):
            calculated_array = logistic_map_array(
                r=r_parameters[i, j],
                seed=initial_parameters[i, j],
                iterations=iterations
            )
            ax[i, j].axhline(
                np.max(calculated_array),
                color="red",
                linestyle="--",
                linewidth=1,
                label="Max"
            )
            ax[i, j].axhline(
                np.min(calculated_array),
                color="red",
                linestyle="--",
                linewidth=1,
                label="Min"
            )
            ax[i, j].plot(calculated_array, color="blue", linewidth=1.5)
            ax[i, j].plot(0, initial_parameters[i, j], color="red", marker="o", label="Početna")
            ax[i, j].plot(iterations, calculated_array[-1], color="red", marker="o", label="Konačna")
            ax[i, j].tick_params(axis='x', labelsize=14)
            ax[i, j].tick_params(axis='y', labelsize=14)
            ax[i, j].set_xlim(0, iterations)
            ax[i, j].set_ylim(0, 1)
            ax[i, j].set_title(
                f"r={r_parameters[i, j]:.4f}Početna={initial_parameters[i, j]:.4f}, Konačna={calculated_array[-1]:.4f}",
                fontsize=16
            )
    plt.tight_layout(pad=0.00)
    plt.show()
```

Slika 3.8. Implementacija funkcije za opširniju vizualizaciju logističke mape

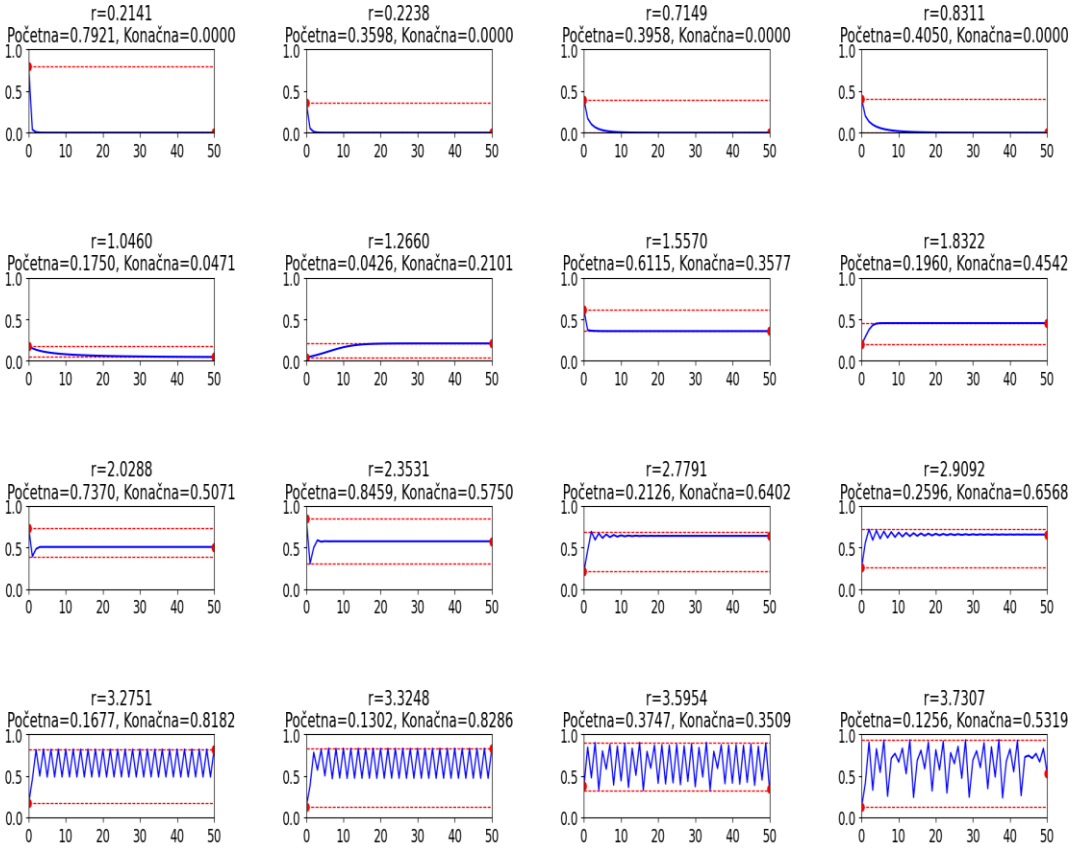
Još samo preostaje testirati prethodno napisani kod, zbog potreba demonstracije za granice intervala nasumično generiranih vrijednosti parametra r uzet će se niz brojeva od 0 do 4, a za veličinu uzorka uzet će se vrijednost 4. Iscrtani dijagram sa tim argumentima sastojat će se od 16 poddijagrama, strukturiranih u 4 retka i 4 stupca. Pozivanjem prvo funkcije *logistic_map_parameters* sa navedenim argumentima, zatim nakon toga funkcije *logistic_map_visualisation* sa vraćenim rezultatima prošlog poziva kao argumentima, dobiva se iscrtani dijagram sa slike 3.10.

```
r_boundaries = np.array(
    [0.0, 1.0, 2.0, 3.0, 4.0]
)
sample_size = 4
r_parameters, initial_parameters = logistic_map_parameters(r_boundaries=r_boundaries, sample_size=sample_size)
logistic_map_visualisation(r_parameters=r_parameters, initial_parameters=initial_parameters, iterations=50)
```

Slika 3.9. Skripta koja koristi dvije novo definirane funkcije

Na slici 3.10. se vidi da logistička mapa ima raznovrsno ponašanje ovisno o početnim uvjetima. Osim kaotičnosti (ne periodične orbite) vide se naznake fiksnih točaka konvergencije te periodičnih oscilacija.

- x_n teži u 0 za sve r u intervalu [0, 1] neovisno o x_0
- x_n se brzo približava vrijednosti $\frac{r-1}{r}$ za sve r u intervalu [1, 2] neovisno o x_0
- x_n se također približava vrijednosti $\frac{r-1}{r}$ za sve r u intervalu [2, 3] s time da su za vrijednosti bliže $r = 3$ karakteristične oscilacije prije nego se dođe do točke konvergencije
- Za r u intervalu [3, 4] se logistička mapa ponaša ili kaotično ili periodično sa dvije ili više vrijednosti



Slika 3.10. Dijagram iscrtan prethodnom skriptom

Iz prethodnog dijagrama ne može se dobiti uvid u dublje razumijevanje logističke mape za vrijednosti $r = [3, 4]$, za tu potrebu će se koristiti *bifurkacijski* dijagram. *Bifurkacijski* dijagram u ovom slučaju neće biti ništa drugo nego obični dijagram raspršenja čija x os predstavlja neprekidni spektar mogućih vrijednosti parametra r, a y os predstavlja sve vrijednosti koje logistička mapa s parametrom r "pogodi" za vrijeme svoje evolucije (za neki predefinirani broj iteracija i početnu vrijednost x_0). Pomoću njega se može vizualno predočiti kako se logistička mapa ponaša za cijeli neprekidni spektar parametra r, odnosno na njemu se točno može vidjeti za koje r-ove se događaju konvergencije te za koje se događaju periodične i ne periodične oscilacije.

Sada će se definirati funkcija koja će služiti za iscrtavanje *bifurkacijskog* dijagrama. Ona će kao parametre primiti minimalnu i maksimalnu vrijednost parametra r, te broj segmenata koliko će se različitih vrijednosti *r*-a uzeti u obzir (vrijednost je neprekidna pa je nemoguće uzeti u obzir potpun spektar). Također će se kao argument predati broj iteracija do kojeg će se računati. Početni uvjet x_0 će se unutar funkcije postaviti na 0.5, budući da se dobije skoro identičan dijagram za bilo koju vrijednost x_0. Metodom *linspace* će se dobiti diskretni spektar *r*-ova između najmanje i najveće vrijednosti, do određene predane preciznosti (predstavlja broj elemenata u 1D polju). Nadalje će se izračunati maksimalni broj točaka koje će dijagram sadržavati, taj broj je potreban kako bi se zauzela statički memorija za niz koji će pohranjivati te točke. Ako se ovo ne napravi na ovaj način već se pokuša inicijalizirati polje nedefinirane veličine i onda se na njega nadodavaju vrijednosti koristeći neke od metoda *append* ili *vstack*, naići će se na probleme što se tiče vremena izvođenja programskog koda. Naime, te metode vraćaju kopiju cijelog polja umjesto da mu samo promjene stanje, a budući da će se ove metode morati pozivati veliki broj puta tijekom izvođenja programa, jako puno vremena će programu oduzeti stvaranje svih ovih silnih objekata. Varijabla *rx_points* će sadržavati sve točke dijagrama raspršenja u obliku 2D polja koji će imati dva stupca (prvi za *r*, drugi za pripadni x_n koji se pojavio negdje u putanji) i redaka onoliko koliko ima točaka u grafu, a taj broj se dobije tako da se pomnoži broj elemenata u *r* spektru sa brojem koji je za jedan veći od broja iteracija (dodaje se jedan jer brojimo i x_0). Definirat će se i varijabla *rx_index* kojom će se pratiti trenutnu poziciju (redak) u polju svih točaka. Budući da će tu koristiti *slicing* tehnika *NumPy* polja kako bi se postavljale vrijednosti, taj indeks je za ovu potrebu nužan. Nakon toga će se u petlji koja prolazi kroz sve *r* vrijednosti izračunati evoluciju za pojedini *r* te spremiti sve dobivene točke (*r*, *x*) u polje koje će se iscrtati metodom *scatter*, čiji će prvi argument biti svaki redak prvog stupca (sve *r* vrijednosti), a drugi argument svaki redak drugog stupca (sve x_n vrijednosti). Na kraju će se samo dodati rešetkasti prikaz (metoda *grid*) te označiti i podesiti dijagram prije prikaza.

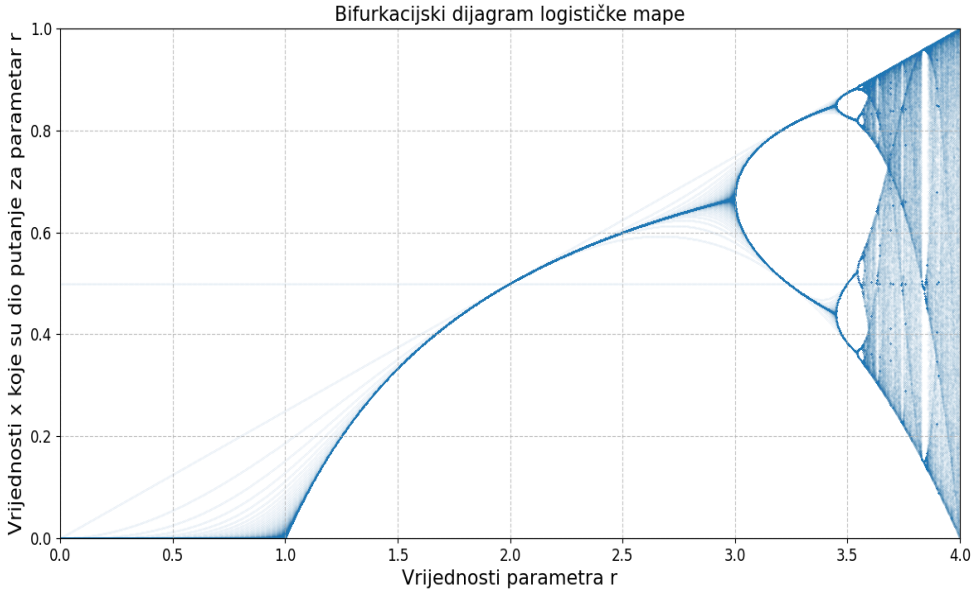
```python
def logistic_map_bifurcation(
    r_min: float = 0.0,
    r_max: float = 4.0,
    r_precision: int = 1000,
    iterations: int = 1000,
) -> None:
    r_values = np.linspace(r_min, r_max, r_precision)
    max_points = r_precision * (iterations + 1)
    rx_points = np.empty(shape=(max_points, 2), dtype=float)
    seed = 0.5
    rx_index = 0
    for r in r_values:
        evolution = logistic_map_array(r=r, seed=seed, iterations=iterations)
        num_points = iterations + 1
        rx_points[rx_index:rx_index + num_points, 0] = r
        rx_points[rx_index:rx_index + num_points, 1] = evolution
        rx_index += num_points
    plt.figure(figsize=(14, 7))
    plt.scatter(rx_points[:, 0], rx_points[:, 1], s=0.001, marker='.')
    plt.grid(True, linestyle='--', alpha=0.75)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.xlim(r_min, r_max)
    plt.ylim(0.0, 1.0)
    plt.title(f"Bifurkacijski dijagram logističke mape", fontsize=16)
    plt.xlabel("Vrijednosti parametra r", fontsize=16)
    plt.ylabel(f"Vrijednosti x koje su dio putanje za parametar r", fontsize=16)
    plt.show()
```

Slika 3.11. Funkcija koja iscrtava bifurkacijski dijagram logističke mape

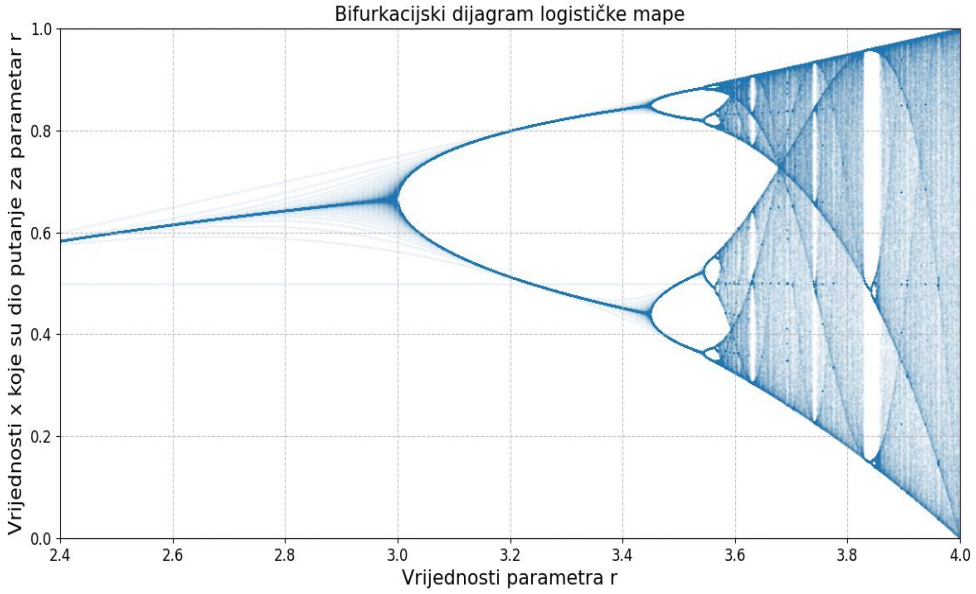
Sada će se nekoliko puta pokrenuti novo definiranu funkciju sa nekoliko različitih skupina parametara te proučiti dobiveni izlaz. Kod *bifurkacijskih* dijagrama (slike 3.12. - 3.17.), puno je lakše prepoznati kako *r* utječe na putanju logističke mape. Najgušće obojani dijelovi dijagrama predstavljaju privlačne (*atraktore*) regije logističke mape kojima vrijednosti x_n teže kroz određenu evoluciju. Različit broj krivulja koje se pojavljuju što se više pomiče udesno po x osi se mogu interpretirati kao broj oscilacija koje se ponavljaju za evolucije mape s pripadnim *r*-om.

- Za *r* u intervalu [0, 3], x_n teži u fiksnu točku
- Za *r* u intervalu [3, ≈3.45], x_n teži u periodičnu putanju veličine 2
- Za *r* u intervalu [≈3.45, ≈3.55], x_n teži u periodičnu putanju veličine 4
- Kako se *r* povećava iznad ≈3.55, x_n teži redom u periodične putanje veličina 8, 16, 32 itd. (udvostručenje perioda), gdje se dužina intervala za svaki broj oscilacija drastično smanjuje (ako proučimo omjere veličina uzastopnih intervala, približavaju se Feigenbaumovoj konstanti $\delta \approx 4.66920$)

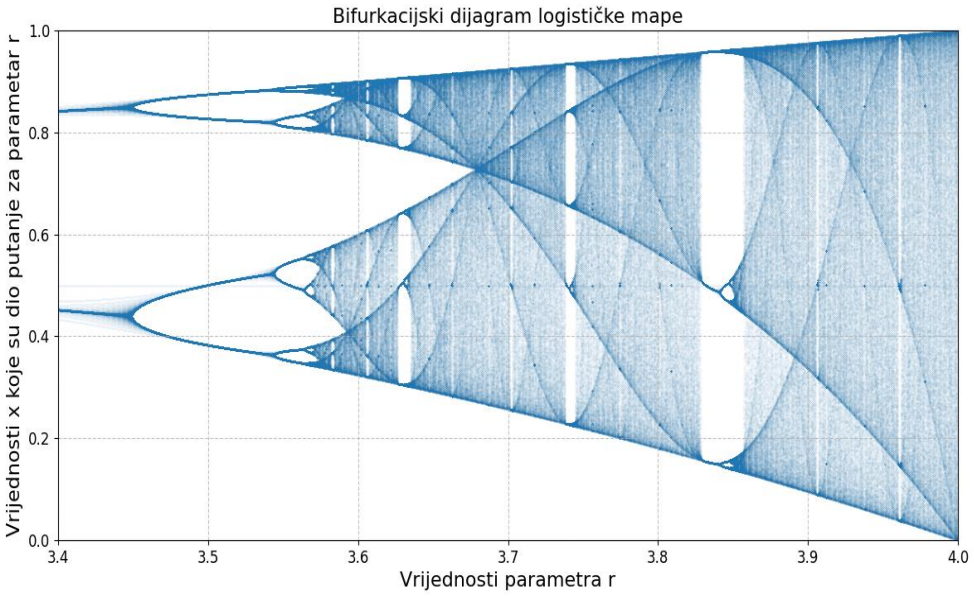
- Od $r \approx 3.56995$ počinje kaos, odnosno ne periodične putanje za većinu početnih uvjeta, iako ovdje možemo pronaći i beskonačno intervala u kojemu se opet vide periodična ponašanja, takozvane "otoke stabilnosti" (uspravne "bijele linije") koji postoje za svaku moguću veličinu perioda (za bilo koji cijeli broj n, može se pronaći r za koji evolucija logističke mape teži periodičnim oscilacijama s periodom n)



Slika 3.12. Bifurkacijski dijagram logističke mape sa osnovnim zadanim argumentima naše funkcije za iscrtavanje

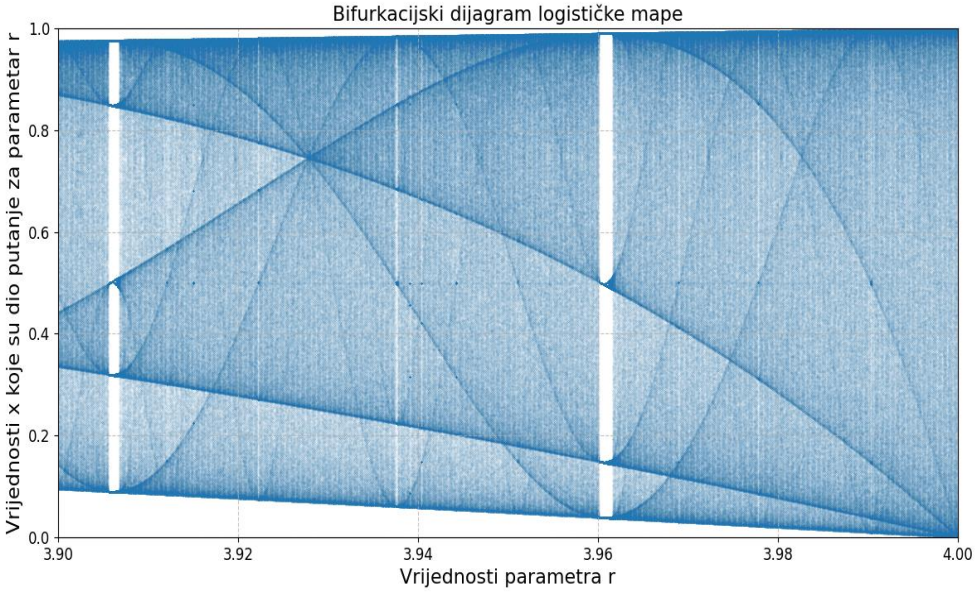


Slika 3.13. Bifurkacijski dijagram od r=2.4

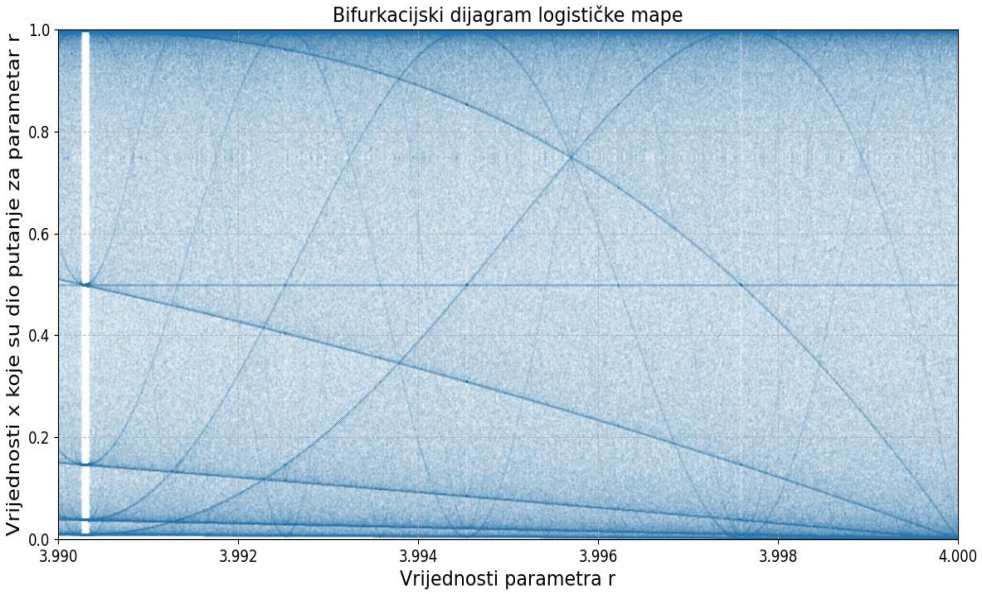


Slika 3.14. Bifurkacijski dijagram od r=3.4

Što manji interval granica parametra *r* se počne promatrati, primijetit će se slične strukture "masnih obojanih područja", odnosno *atraktora*, koliko god u dubinu se išlo, što pokazuje *fraktalnu* strukturu logističke mape (slike 3.15. i 3.16.). Važno je napomenuti da je ono što se vidi samo dio strukture koja je sakrivena u *bifurkacijskom* dijagramu, ovisno o preciznosti i iteracije do koje se gleda. Može se mijenjanjem tih argumenata dobiti uvid u određene dijelove dijagrama, no ipak, nikad se neće moći predočiti baš sve informacije, neovisno koje početne argumente izaberemo.

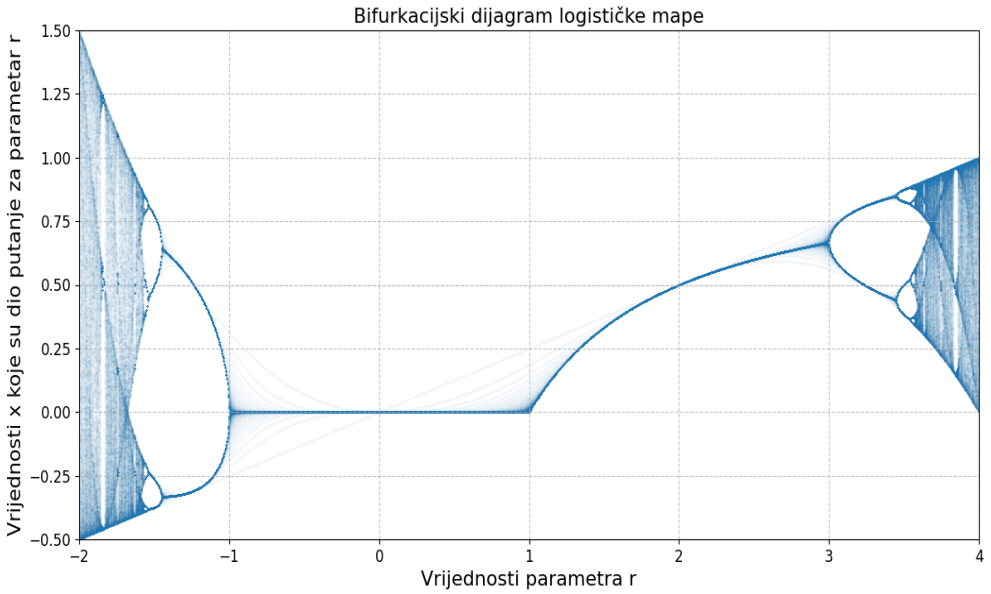


Slika 3.15. Bifurkacijski dijagram od r=3.9, povećan broj iteracija i preciznost



Slika 3.16. Bifurkacijski dijagram od r=3.99, drastično smanjen broj iteracija te drastično povećana preciznost

Moguće je promatrati i negativne vrijednosti za r, u kojem slučaju se dobije dijagram sa slike 3.17. Vidi se da se dobije zrcalna slika strukture koja nastaje za pozitivne vrijednosti, iako su same vrijednosti x_n drugačijeg reda veličine (vrijednosti ispadaju iz intervala [0, 1]). Za vrijednosti r-a veće od 4 ili manje od –2, x_n će divergirati.



Slika 3.17. Bifurkacijski dijagram od r=-2

3.2. Hénonova mapa

Hénonova mapa je najpoznatija dvodimenzionalna diskretna kaotična mapa te se često pojavljuje u polju proučavanja dinamičkih sustava. Može se definirati jednako lako kao i logističku mapu, no ipak je nešto kompleksnija. Ona za razliku od logističke mape ima dva početna parametra (x_0, y_0) čije evoluciju (x_n, y_n) se prate tijekom vremena, a ima i dva dodatna parametra (a, b) koji omogućuju podešavanje ponašanja mape. Funkcije svakog koraka iteracije imaju sljedeći oblik:

$$x_{n+1} = 1 - ax_n^2 + y_n$$

$$y_{n+1} = bx_n$$

Već tu se vidi veća kompleksnost naspram logističke mape, jer u pitanje nije samo veći broj parametara. Dodatni značaj doprinosi to što što x i y ovise jedan o drugome (da nije tako ove dvije funkcije nema smisla promatrati kao "jednu mapu", budući da između njih ne bi bilo nikakve korelacije). Hénonovu mapu može se u Python-u implementirati na sličan način kao i logističku, s nekim izmjenama naravno. Nastojat će se pratiti slična struktura radi lakše čitljivosti.

Sa slike 3.18. se vidi da će funkcija koraka iteracije uzimati parametre a, b, x_n i y_n te koristeći navedene formule računati i vračati uređeni par sljedećih koraka u nizu iteracije (x_{n+1}, y_{n+1}).

```
def henon_map_iteration(a: float, b: float, current_x: float, current_y: float) -> tuple[float, float]:
    next_x = 1 - a * (current_x ** 2) + current_y
    next_y = b * current_x
    return next_x, next_y
```

Slika 3.18. Funkcija koraka iteracije Hénonove mape

Sada će se definirati funkcija izračuna evolucije Hénonove mape, koristeći funkciju koraka iteracije. Ova funkcija kao argumente prima sve potrebne početne uvjete kao i broj iteracija do kojeg se želi računati, a vraća uređeni par dva *NumPy* niza putanja x i y, odnosno njihovu zajedničku evoluciju. Prazna polja koja će sadržavati putanje inicijalizirati će se statički, tako što im se predaju točne potrebne duljine za dimenzije oblika. To se moglo napraviti i pomoću metode *append* kao i kod logističke mape, no budući da se ovdje radi s dvostruko više vrijednosti koje treba izračunati, bolje je to odmah napraviti optimalno kako se ne bi došlo do većih problema sa sporim izvođenjem koda kada se bude crtao bifurkacijski dijagram. Ostatak koda je analogan onome što se radilo kod logističke mape, jedine su razlike to što se ovdje radi sa dva polja za pripadne x_n i y_n vrijednosti te se dodavanje novih vrijednosti u nizove radi s obzirom na točne indekse.

```python
def henon_map_array(
        a: float, b: float,
        seed_x: float, seed_y: float,
        iterations: int
) -> tuple[np.ndarray, np.ndarray]:
    output_array_x = np.empty(shape=(iterations + 1,), dtype=float)
    output_array_y = np.empty(shape=(iterations + 1,), dtype=float)

    output_array_x[0] = seed_x
    output_array_y[0] = seed_y

    for n in range(iterations):
        next_x, next_y = henon_map_iteration(
            a, b,
            output_array_x[n], output_array_y[n]
        )
        output_array_x[n + 1] = next_x
        output_array_y[n + 1] = next_y

    return output_array_x, output_array_y
```

Slika 3.19. Funkcija za izračun evolucije Hénonove mape

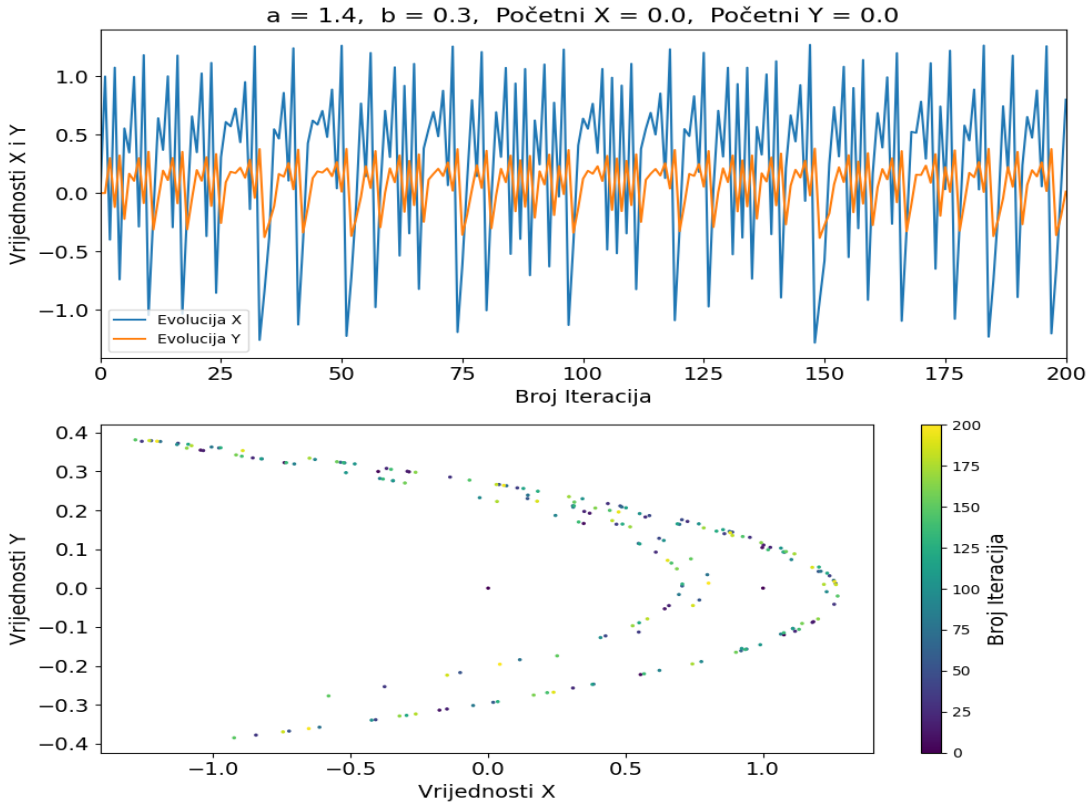
Sada će se ove funkcije testirati te pomoću njih dobiti podatke koji se mogu vizualizirati. Crtat će se također linijske dijagrame, no ovaj put za dvije skupine vrijednosti x i y, može ih se za početak iscrtati na istom pod dijagramu kako bi se lakše usporedili. Dodatno, budući da su sada tu dva niza, može se vidjeti što se dogodi kada se iscrta njihov dijagram raspršenja. Ovakav način vizualizacije može pružiti više konteksta o odnosu x i y te reći nešto o kakvim putanjama se oni zajedno kreću kao točka u dvodimenzionalnom prostoru. U ovom primjeru, za vrijednosti dodatnih parametara a i b, uzet će se klasične vrijednosti 1.4 i 0.3, za ove vrijednosti Hénonova ima kaotična svojstva za dobar dio početnih uvjeta. Za početne vrijednosti x_n i y_n za sada će se uzeti vrijednost 0. Koristeći prošle funkcije za izračun podataka za crtanje te koristeći *Axes* objekte metode *subplots*, na prvom pod dijagramu crtaju se dva linijska dijagrama za pripadne evolucije x i y, a na drugom njihov dijagram raspršenja. Uz sve već viđene potrebne oznake i podešavanja, također će se na dijagramu raspršenja iscrtati stupac za označavanje spektra boja. Ovo, u konačnici, nije potrebno za ovaj dijagram, no korisno je ako se hoće vizualno predočiti koja točka (x, y) je došla iz kojeg dijela evolucije, budući da tu nema "linija" između susjednih točaka koje bi govorile tu informaciju kao

kod linijskog dijagrama. To se može lako postići dodatnim argumentima *c* i *cmap* metode *scatter*, gdje *c* označava polje od 0 do broja iteracija (ovo mapira obojani stupac na te vrijednosti), a *cmap* označava stil boja koji će se koristiti. Pozivanjem metode *colorbar* na objektu koji predstavlja dijagram raspršenja dodaje se stupac sa bojama te ga se vraća kao objekt kako bi mu se mogla postaviti oznaka.

```
a, b, seed_x, seed_y, iterations = 1.4, 0.3, 0.0, 0.0, 200
evolution_x, evolution_y = henon_map_array(a, b, seed_x, seed_y, iterations)
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 9))
ax1.plot(evolution_x, label="Evolucija X")
ax1.plot(evolution_y, label="Evolucija Y")
ax1.tick_params(axis='x', labelsize=14)
ax1.tick_params(axis='y', labelsize=14)
ax1.set_title(f"a = {a},  b = {b},  Početni X = {seed_x},  Početni Y = {seed_y}", fontsize=16)
ax1.set_xlabel("Broj Iteracija", fontsize=14)
ax1.set_ylabel("Vrijednosti X i Y", fontsize=14)
ax1.set_xlim(0, iterations)
ax1.legend(loc="lower left")
scatter_plot = ax2.scatter(
    evolution_x, evolution_y,
    c=np.arange(iterations + 1), cmap='viridis',
    marker='.', s=10
)
ax2.tick_params(axis='x', labelsize=14)
ax2.tick_params(axis='y', labelsize=14)
ax2.set_xlabel("Vrijednosti X", fontsize=14)
ax2.set_ylabel("Vrijednosti Y", fontsize=14)
cbar = plt.colorbar(mappable=scatter_plot)
cbar.set_label('Broj Iteracija', fontsize=14)
plt.tight_layout()
plt.show()
```

Slika 3.20. Opisana skripta

Sa slike 3.21. se prema gornjem pod dijagramu vidi kaotično ponašanje Hénonove mape za navedene parametre. Također, po oblicima linijskih dijagrama vidi se kako su *x* i *y* evolucije u ovom primjeru povezane, ista struktura se ponavlja za obje pojedinačne putanje, no vidi se da se za *y* vrijednosti struktura događa na umanjenom intervalu, odnosno proširenom za *x* vrijednosti. Na donjem pod dijagramu se vidi kakvu strukturu u dvodimenzionalnom prostoru stvaraju točke (*x*, *y*), to je *fraktalna* struktura poznata kao Hénonov *atraktor* za vrijednosti mape $a = 1.4$ i $b = 0.3$.



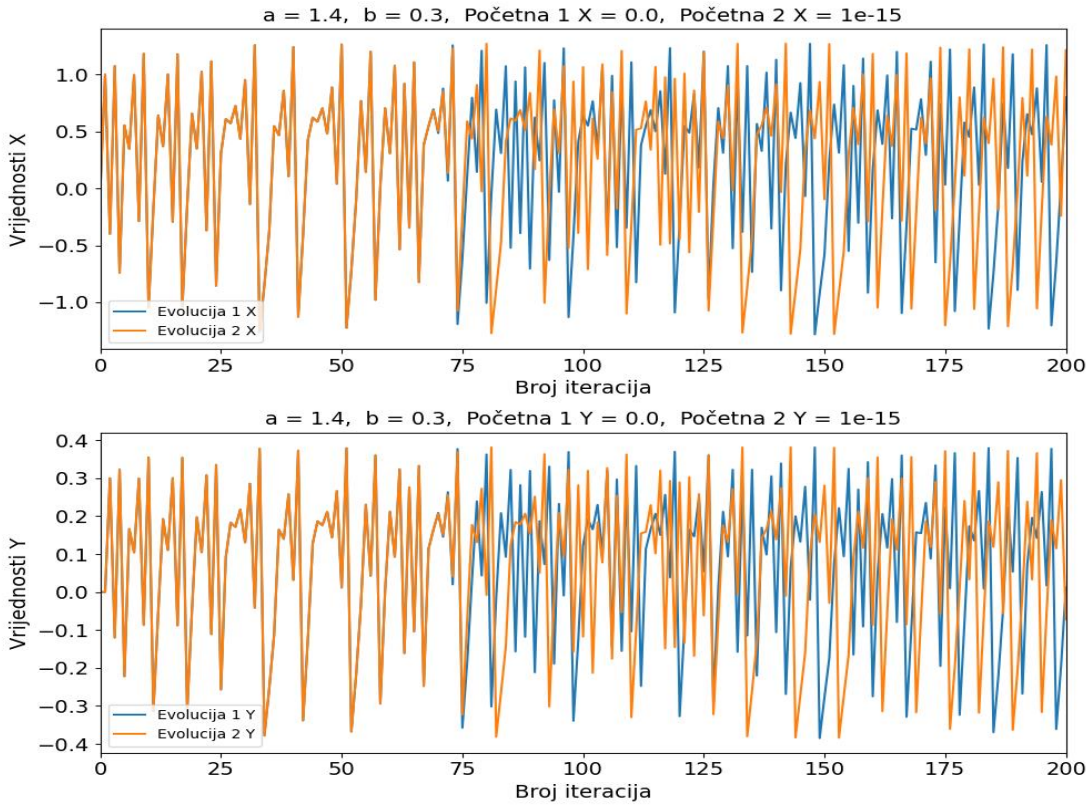
Slika 3.21. Izlaz prethodne skripte

U nastavku će se vizualizirati kako Hénonov *atraktor* izgleda kada ga se preciznije prikaže pomoću većeg broja iteracija, no sada će se prvo pokazati kako se klasična Hénonova mapa ponaša s obzirom na početne uvjete x_0 i y_0. U tu svrhu preuredit će se najprije prethodna skripta prema slici 3.22., prikazat će se posebno dvije putanje *x*, a posebno dvije putanje *y* vrijednosti na odvojene pod dijagrame. Na početne uvjete (x_0, y_0) novo stvorenih putanja x i y dodat će se mala vrijednost reda 10^{-15} te će se prikazati sve potrebne oznake.

```
a, b, seed_1_x, seed_1_y, iterations = 1.4, 0.3, 0.0, 0.0, 200
offset = 0.000000000000001
seed_2_x, seed_2_y = seed_1_x + offset, seed_1_y + offset
evolution_1_x, evolution_1_y = henon_map_array(a, b, seed_1_x, seed_1_y, iterations)
evolution_2_x, evolution_2_y = henon_map_array(a, b, seed_2_x, seed_2_y, iterations)
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 9))
ax1.plot(evolution_1_x, label="Evolucija 1 X")
ax1.plot(evolution_2_x, label="Evolucija 2 X")
ax1.tick_params(axis='x', labelsize=14)
ax1.tick_params(axis='y', labelsize=14)
ax1.set_title(f"a = {a},  b = {b},  Početna 1 X = {seed_1_x},  Početna 2 X = {seed_2_x}", fontsize=14)
ax1.set_xlabel("Broj iteracija", fontsize=14)
ax1.set_ylabel("Vrijednosti X", fontsize=14)
ax1.set_xlim(0, iterations)
ax1.legend(loc="lower left")
ax2.plot(evolution_1_y, label="Evolucija 1 Y")
ax2.plot(evolution_2_y, label="Evolucija 2 Y")
ax2.tick_params(axis='x', labelsize=14)
ax2.tick_params(axis='y', labelsize=14)
ax2.set_title(f"a = {a},  b = {b},  Početna 1 Y = {seed_1_y},  Početna 2 Y = {seed_2_y}", fontsize=14)
ax2.set_xlabel("Broj iteracija", fontsize=14)
ax2.set_ylabel("Vrijednosti Y", fontsize=14)
ax2.set_xlim(0, iterations)
ax2.legend(loc="lower left")
plt.tight_layout()
plt.show()
```

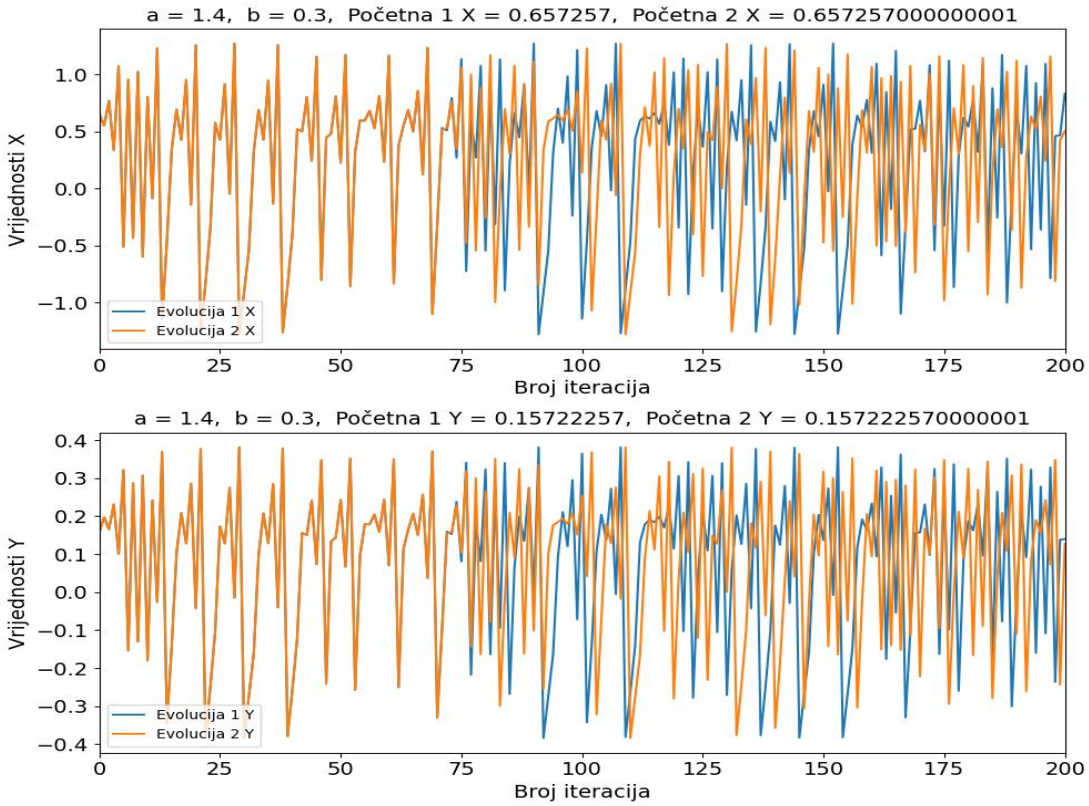
Slika 3.22. Preuređena skripta

Slika 3.23. prikazuje izlaz preuređene skripte. Kao je već spomenuto, za početne vrijednosti dodatnih parametara $a = 1.4$ i $b = 0.3$ mapa ima kaotične, odnosno ne periodične oscilacije, a to je i vidljivo sa same slike. Nakon otprilike 75 iteracija početi pomak od 10^{-15} postane toliko značajan da putanje u potpunosti divergiraju jedna od druge te imaju sasvim različitu evoluciju. Također se ponovno vidi da x i y evolucije imaju istu strukturu putanje na drugačijim skalama, u ovom dijagramu je to još i jasnije nego u prethodnom budući da je y-os drugog pod dijagrama skalirana da vrijednosti ispune cijeli pristupni prostor.



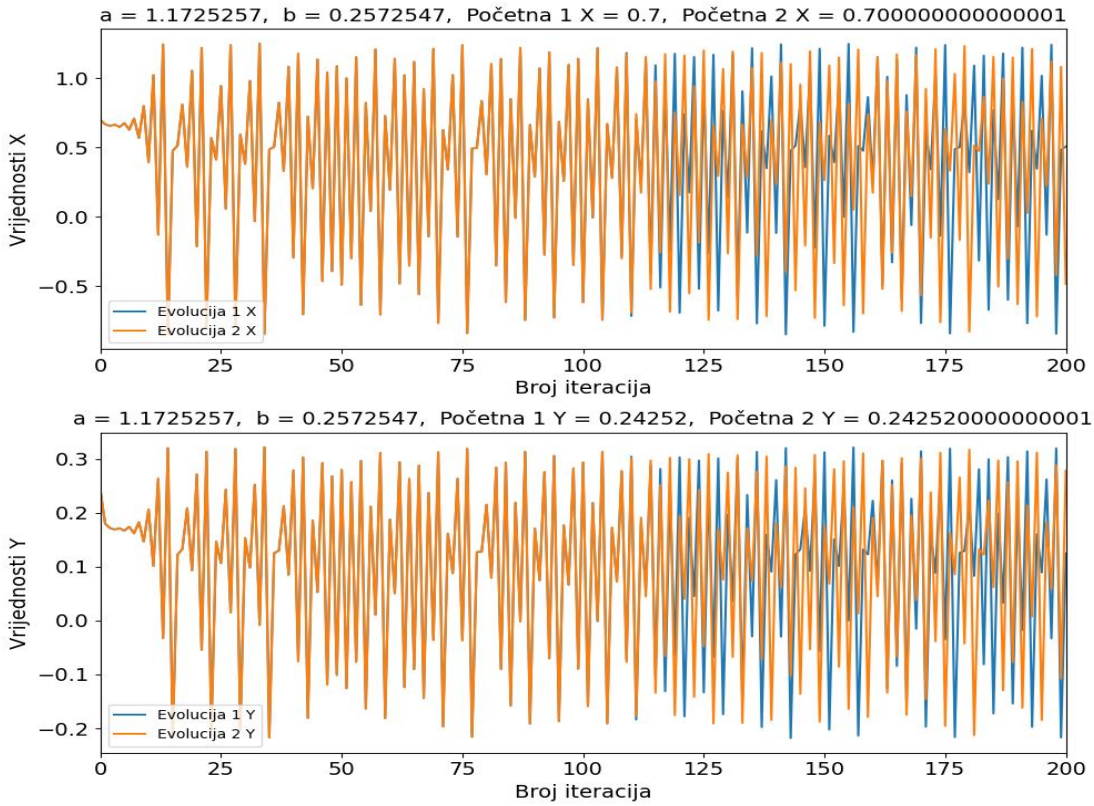
Slika 3.23. Izlaz prethodne skripte

U kaotičnost Hènonove mape za navedene uvjete se može dodatno uvjeriti isprobavajući kao početne parametre neke druge nasumično izabrane vrijednosti, kao što se vidi na primjeru sa slike 3.24. Nakon sličnog broja koraka iteracije, također se jasno vidi divergencija izračunatih putanja. Važno je napomenuti da je kod Hénonove mape, za razliku od logističke, puno utjecajniji izbor parametara x_0 i y_0 (primjerice, za klasičnu mapu sa x_0 = 0 i y_0 = 0.3 vrijednosti divergiraju u beskonačnost), što znači da iako se zna da je mapa kaotična za klasične vrijednosti *a* i *b* svejedno se mora paziti na izbor ostalih početnih parametara ovisno o tome na kakvo ponašanje se cilja.

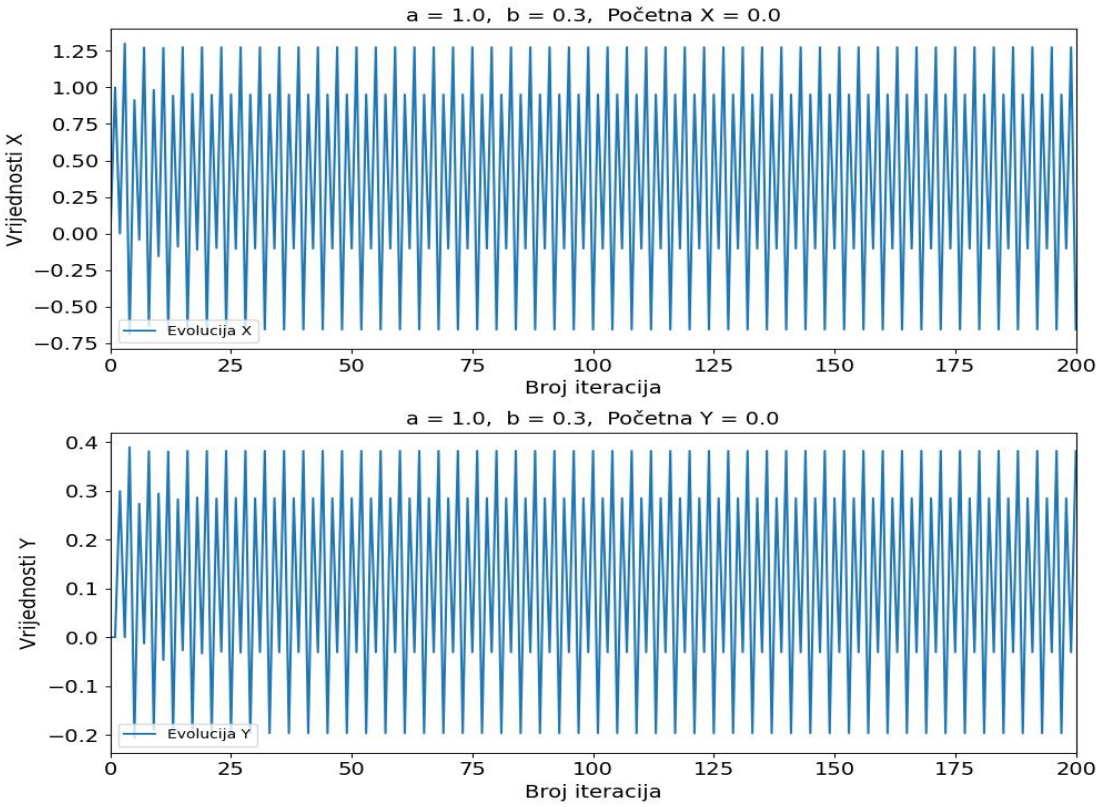


Slika 3.24. Izlaz prethodne skripte sa promijenjenima početnim uvjetima x_0 *i* y_0

Sa slika 3.25 se vidi da mapa može stvarati kaotične putanje i sa drugim početnim parametrima a i b, moglo bi se reći da je nekakvo generalno pravilo da su "sigurne" vrijednosti (vrijednosti za koje evolucije ne divergiraju u beskonačnost, a takve evolucije nema smisla prikazivati dijagramom) za eksperimentiranje a = [1.0, 1.4] i b = [0.0, 0.3], no ovo naravno ovisi o točnim vrijednostima x_0 i y_0. Može se divergencija dogoditi i za te vrijednosti i to nerijetko, no pažljivijim izborom ostalih početnih parametara se to može izbjeći, ako je to cilj. Sa slike 3.26. se također vidi da se mogu naći vrijednosti a i b za koje mapa ima periodična oscilacijska ponašanja, kao što je bio i slučaj kod logističke mape. Sve u svemu, veći uvid u ponašanje Hénonove mape na osnovu početnih uvjeta steći će se kada se budu crtali njezini *bifurkacijski* dijagrami.

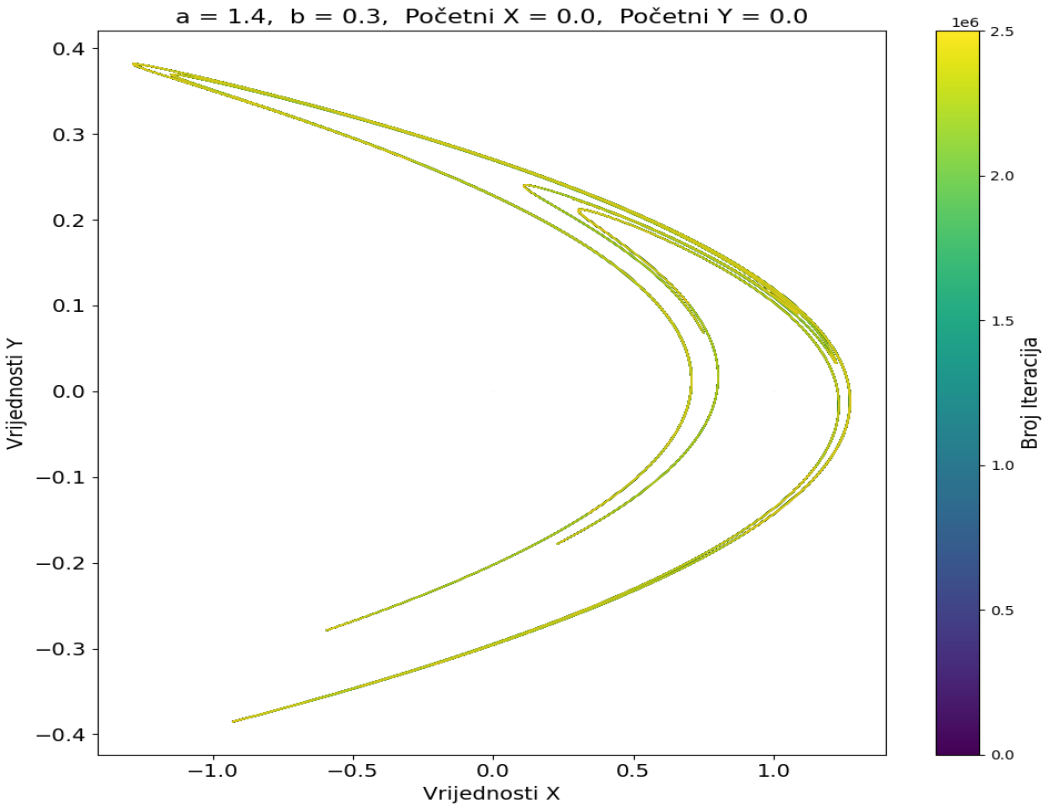


Slika 3.25. Izlaz prethodne skripte sa svim početnim parametrima promijenjenima



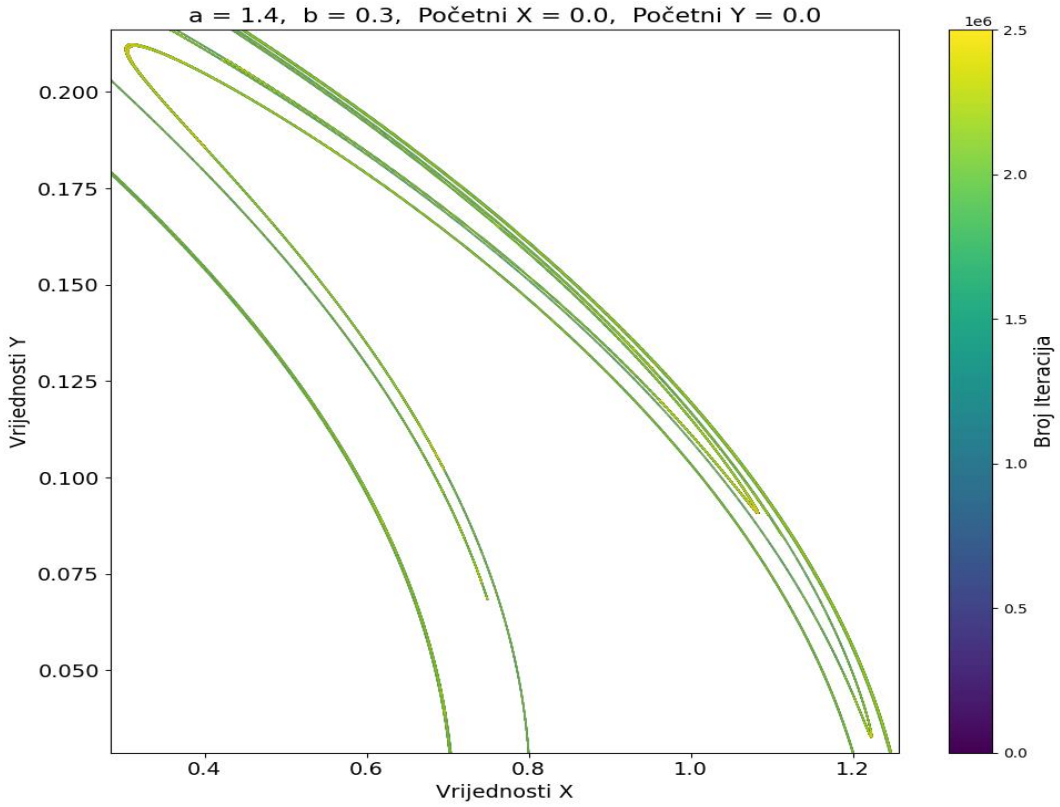
Slika 3.26. Izlaz za samo jedan par putanja kada promijenimo a = 1.0

Što se tiče Hénonovog *atraktora*, sada će se pokazati što se dogodi kada ga se iscrta sa puno većim brojem iteracija. U tu svrhu će se ponovno iskoristiti dio koda skripte sa slika 3.20., onaj dio koji je potreban samo za dijagram raspršenja. Pokazat će se kakva struktura nastaje kada ga se iscrta sa preciznošću od 2,500,000 iteracija, odnosno kakvim putanjama teži kretanje točaka (x_n, y_n) u 2D prostoru.

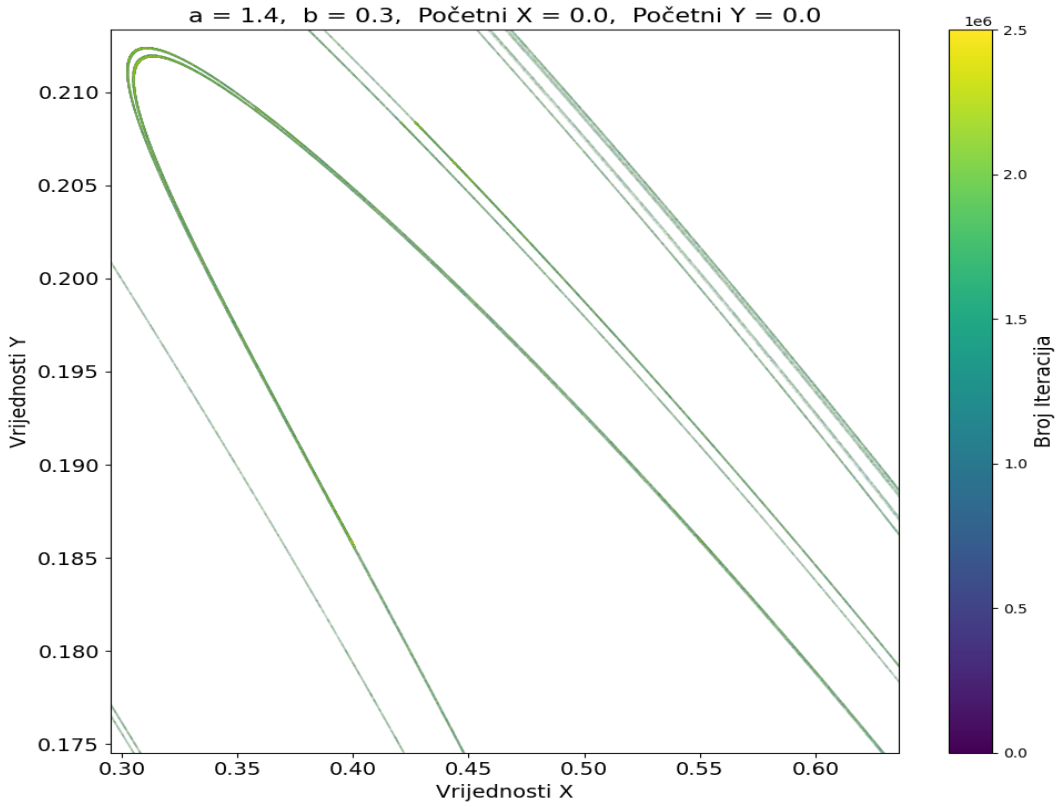


Slika 3.27. Hénonov atraktor za klasične vrijednosti mape

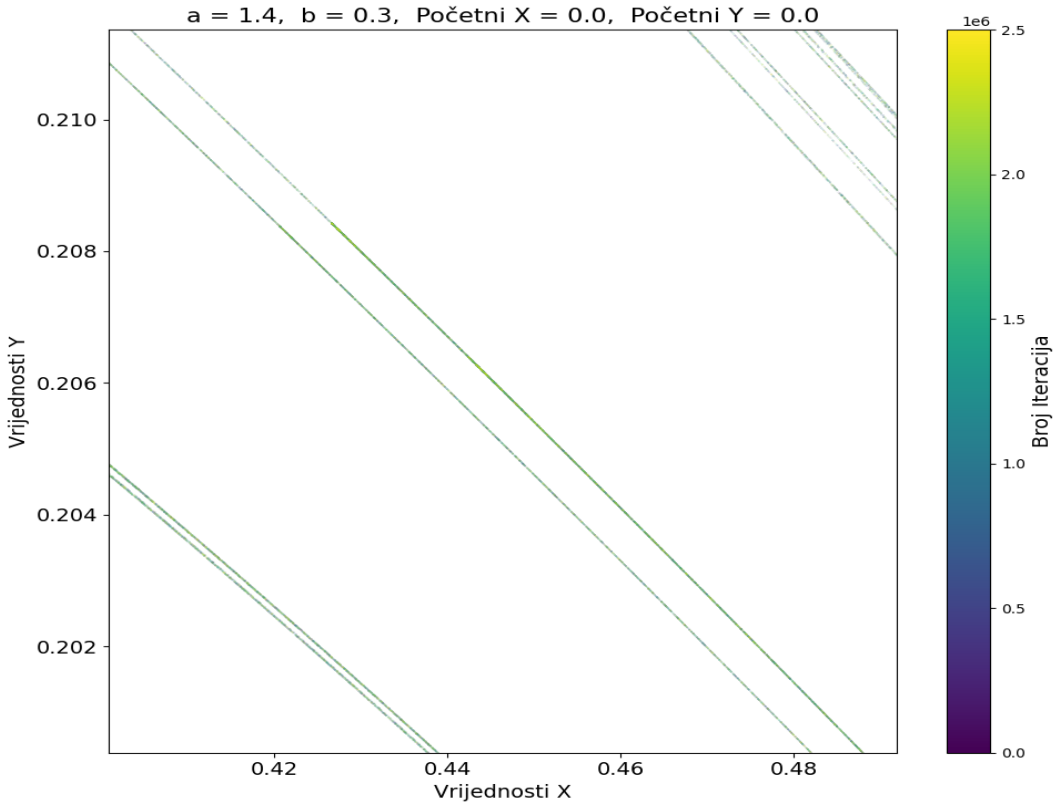
Nakon mnogo iteracija, putanje Hénonove mape u 2D prostoru nalikuju na skupine čudnih krivulja. Naizgled se čini da je to samo nekolicina krivulja, no ako se poveća dobivena slika vidi se da su putanje kojima se kreću vrijednosti zapravo *fraktalne* strukture. Bez obzira koliko god se povećao dobiveni prikaz, uvijek će se vidjeti kako se naizgled jedna krivulja uvećanjem prošili u dvije ili više njih, što se jasno vidi sa slika 3.28 - 3.30. Naravno, u ovom slučaju, nakon nekoliko uvećanja već se gubi struktura tih krivulja (slika 3.30), no to je zato što je potrebno jako puno više iteracija i računalne moći da bi se prikazala detaljnija struktura. Kada bi se u teoriji moglo izračunati "beskonačno iteracija", moglo bi se uvećati sliku koliko god se želi bez gubitka detalja.



Slika 3.28. Uvećanje prethodnog prikaza

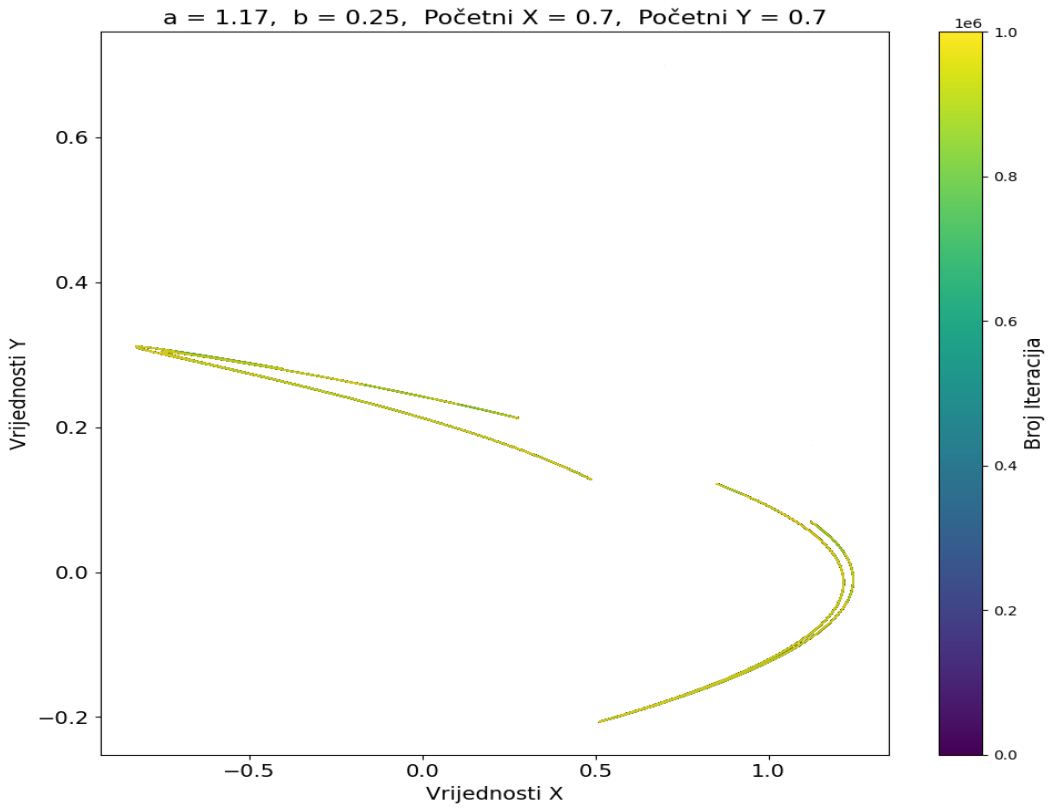


Slika 3.29. Dodatno uvećanje prethodnog prikaza



Slika 3.30. Uvećanje prikaza do granica preciznosti

Slične *atraktorske* strukture dobiju se i za druge vrijednosti početnih parametara gdje mapa ima kaotično ponašanje. U slučaju da se mapa ponaša periodički, onda nema previše smisla crtati dijagram raspršenja jer će se vidjeti svega par točaka (točke između kojih putanja oscilira). Izuzev su slučaji kada se žele vizualizirati periodička ponašanja sa jako velikim periodom, no takve vrijednosti će se teško pronaći "ručno", odnosno pogađanjem parametara. Na slici 3.31. se vidi primjer sličnog *fraktalnog atraktora* za proizvoljno izabrane početne vrijednosti za koje Hénonova mapa ima ne periodičko ponašanje.



Slika 3.31. Atraktor za navedene početne vrijednosti

Kao i kod logističke mape, teško je na osnovu par evolucija za određene parametre zaključiti nešto o općenitom ponašanju mape za sve moguće kombinacije početnih uvjeta. Iz toga razloga će se sada pokazati kako implementirati i iscrtati *bifurkacijski* dijagram Hénonove mape. Tu je važno napomenuti da će se morati uzeti nešto drugačiji pristup nego kod logističke mape, budući da ova mapa ima dva dodatna parametra *a* i *b*. Zbog toga se neće moći samo jednostavno prikazati jedan dijagram za sve moguće spektre dodatnih parametara nego će se morati izabrati jedan od njih čiji će se cijeli spektar prikazati, a drugi od njih koji će se fiksirati na zadanu vrijednost. Alternativno, moglo se crtati trodimenzionalni *bifurkacijski* dijagram i taj problem riješiti na taj način, no zbog jasnoće interpretacije dijagrama čitatelju rada (koji ne može upravljati sa prikazom u stvarnom vremenu i tako iz njega izvući potrebne informacije) izabrat će se prethodno opisana metoda vizualizacija.

Funkcija za ovu namjenu kao argumente će, osim potrebnih početnih uvjeta i broja iteracija, primiti i parametre prema kojima će se znati koju vrijednost se fiksira (*a* ili *b*) te same vrijednosti tih parametara (jedan u obliku broja, drugi u obliku spektra vrijednosti) i preciznost do koje će se računati promjenjivi parametar. Argument *cutoff* služi kako bi se izabralo koliko zadnjih izračunatih vrijednosti će se prikazati, ovo je pogodno jer se time lakše može dobiti uvid u

dugoročno ponašanje mape, no ako se za argument stavi 0 i dalje se može prikazati sve vrijednosti ako to bude potrebno. Parametri *y_limit_1* i *y_limit_1q* zapravo nisu potrebni, no korisno ih je navesti zbog veće kontrole u pravljenju dijagrama. Prvo što će funkcija napraviti je inicijalizirati vektor spektra promjenjivog parametra te izračunati i zauzeti memoriju za polja koja će predstavljati putanje x_n i y_n. To se radi na sličan način kao i kod logističke mape samo će se ovdje uzeti u obzir *cutoff* pri računanju broja točaka koje se dodaju u svakom koraku te će se sve postupke raditi za dva polja umjesto jednog. U petlji će se, na osnovu je li fiksiran *a* ili *b*, dogoditi da se u svakom koraku izračunaju potrebne vrijednosti iteracije Hénonove mape te ih se pravilno spremi u polja koja će držati podatke za iscrtavanje *bifurkacijskog* dijagrama. Nakon što su izračunati *x* i *y* putanje, stvorit će se dva pod dijagrama jedan ispod drugoga. Na prvom će se iscrtati bifurkacijski dijagram x vrijednosti u ovisnosti o promjenjivom parametru, a na drugom y vrijednosti u ovisnosti o promjenjivom parametru.

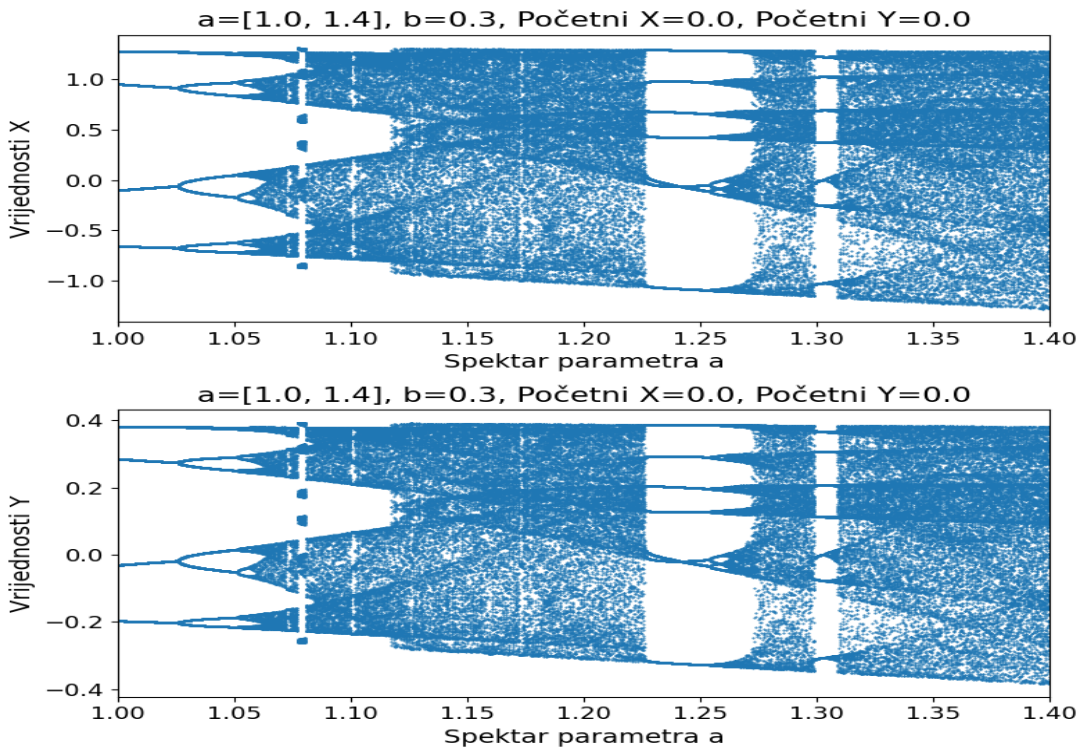
```python
def henon_map_bifurcation(
        fix_a: bool, fixed_parameter_value: float, non_fixed_parameter_range: tuple[float, float],
        non_fixed_parameter_precision: int = 1000, iterations: int = 1000, cutoff: int = 100, seed_x: float = 0,
        seed_y: float = 0, y_limit_1: float = 0, y_limit_2: float = 0
) -> None:
    non_fixed_parameter_values = np.linspace(
        non_fixed_parameter_range[0], non_fixed_parameter_range[1], non_fixed_parameter_precision)
    num_points = min(cutoff, iterations + 1) if cutoff > 0 else iterations + 1
    max_points = non_fixed_parameter_precision * num_points
    bifurcation_data_x = np.empty(shape=(max_points, 2), dtype=float)
    bifurcation_data_y = np.empty(shape=(max_points, 2), dtype=float)
    x_index, y_index = 0, 0
    for non_fixed_value in non_fixed_parameter_values:
        if fix_a:
            a, b, a_or_b = fixed_parameter_value, non_fixed_value, non_fixed_value
        else:
            a, b, a_or_b = non_fixed_value, fixed_parameter_value, non_fixed_value
        x_values, y_values = henon_map_array(a, b, seed_x, seed_y, iterations)
        bifurcation_data_x[x_index:x_index + num_points, 0] = a_or_b
        bifurcation_data_y[y_index:y_index + num_points, 0] = a_or_b
        bifurcation_data_x[x_index:x_index + num_points, 1] = x_values[-num_points:]
        bifurcation_data_y[y_index:y_index + num_points, 1] = y_values[-num_points:]
        x_index, y_index = x_index + num_points, y_index + num_points
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 8))
    ax1.scatter(bifurcation_data_x[:, 0], bifurcation_data_x[:, 1], s=1, marker='.')
    ax2.scatter(bifurcation_data_y[:, 0], bifurcation_data_y[:, 1], s=1, marker='.')
    ax1.tick_params(axis='x', labelsize=13)
    ax1.tick_params(axis='y', labelsize=13)
    ax2.tick_params(axis='x', labelsize=13)
    ax2.tick_params(axis='y', labelsize=13)
    ax1.set_ylabel('Vrijednosti X', fontsize=14)
    ax2.set_ylabel('Vrijednosti Y', fontsize=14)
    lower_bound, upper_bound = non_fixed_parameter_range
    ax1.set_xlim(lower_bound, upper_bound)
    ax2.set_xlim(lower_bound, upper_bound)
    if y_limit_1 != 0:
        ax1.set_ylim(-y_limit_1, y_limit_1)
    if y_limit_2 != 0:
        ax2.set_ylim(-y_limit_2, y_limit_2)
    if fix_a:
        ax1.set_title(
            f'a={fixed_parameter_value}, b=[{lower_bound}, {upper_bound}], Početni X={seed_x}, Početni Y={seed_y}',
            fontsize=16)
        ax2.set_title(
            f'a={fixed_parameter_value}, b=[{lower_bound}, {upper_bound}], Početni X={seed_x}, Početni Y={seed_y}',
            fontsize=16)
        ax1.set_xlabel('Spektar parametra b', fontsize=14)
        ax2.set_xlabel('Spektar parametra b', fontsize=14)
    else:
        ax1.set_title(
            f'a=[{lower_bound}, {upper_bound}], b={fixed_parameter_value}, Početni X={seed_x}, Početni Y={seed_y}',
            fontsize=16)
        ax2.set_title(
            f'a=[{lower_bound}, {upper_bound}], b={fixed_parameter_value}, Početni X={seed_x}, Početni Y={seed_y}',
            fontsize=16)
        ax1.set_xlabel('Spektar parametra a', fontsize=14)
        ax2.set_xlabel('Spektar parametra a', fontsize=14)
    plt.tight_layout()
    plt.show()
```

Slika 3.32. Funkcija za iscrtavanje bifurkacijskog dijagrama Hènonove mape

Na slici 3.34 se vidi primjer izlaza koji se dobije pozivanjem prethodne funkcije sa parametrima sa slike 3.33. Ovo su najpoznatije vrijednosti za koje se tipično crta bifurkacijski dijagram Hénonove mape, a vidi se da je on sličan, ali i nešto kompliciraniji od onoga što se vidjelo kod logističke mape. Mogu se opet prepoznati strukture kao što su tendencije fiksnim točkama konvergencije kao i udvostručavanje perioda sve do "eksplozije" u kaos i ne periodične oscilacije. Ova struktura je također *fraktalna* struktura, a vidi se i ono što je već rečeno za isto ponašanje putanja *x* i *y* na drugačijim skalama.

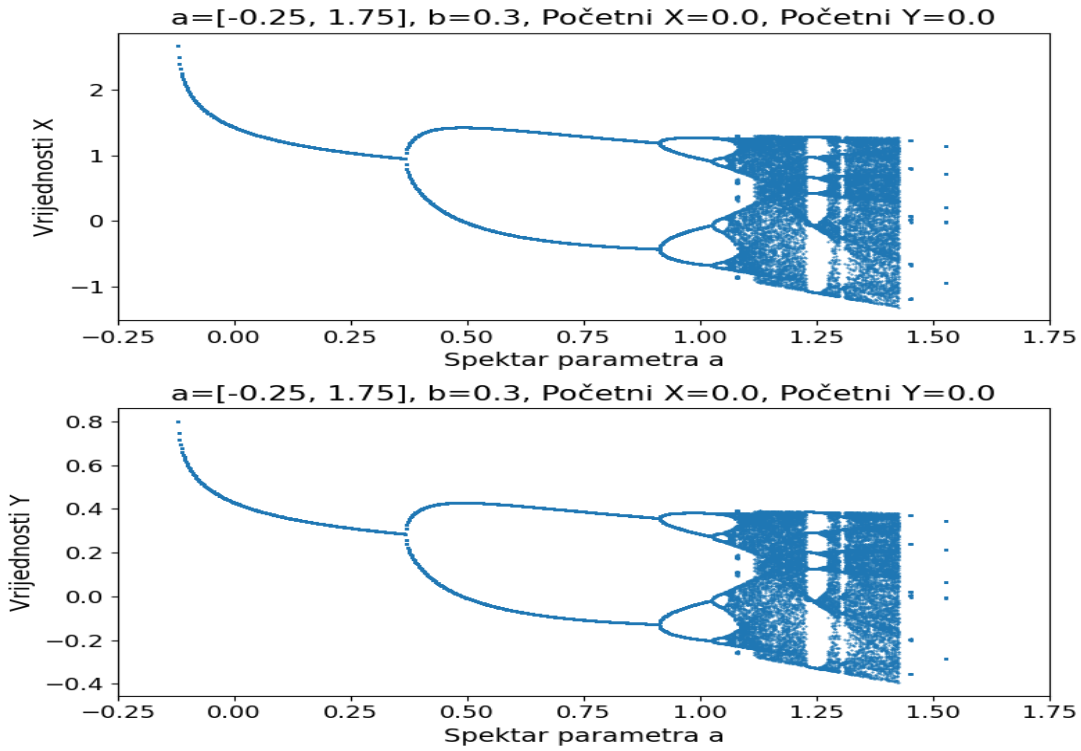
```
henon_map_bifurcation(
        fix_a=False, fixed_parameter_value=0.3,
        non_fixed_parameter_range=(1.0, 1.4),
        non_fixed_parameter_precision=1000,
        iterations=10000, cutoff=100,
        seed_x=0.0, seed_y=0.0,
        y_limit_1=0, y_limit_2=0
)
```

Slika 3.33 Prikaz pozivanja prošle funkcije

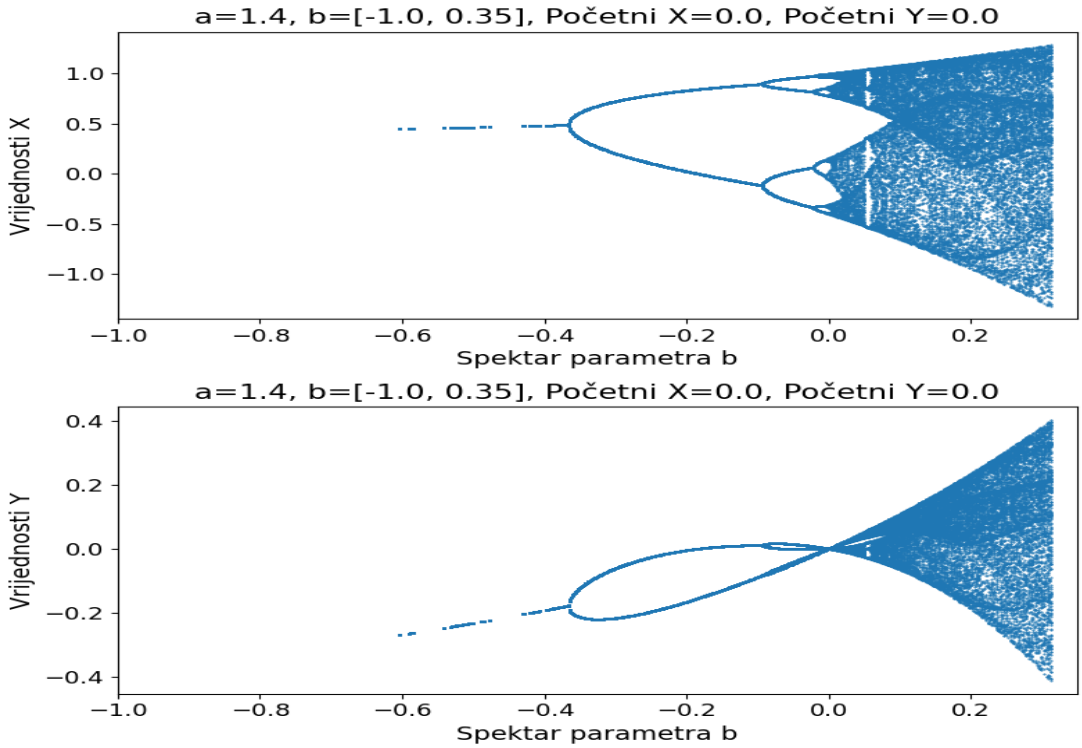


Slika 3.34 Dobiveni bifurkacijski dijagram

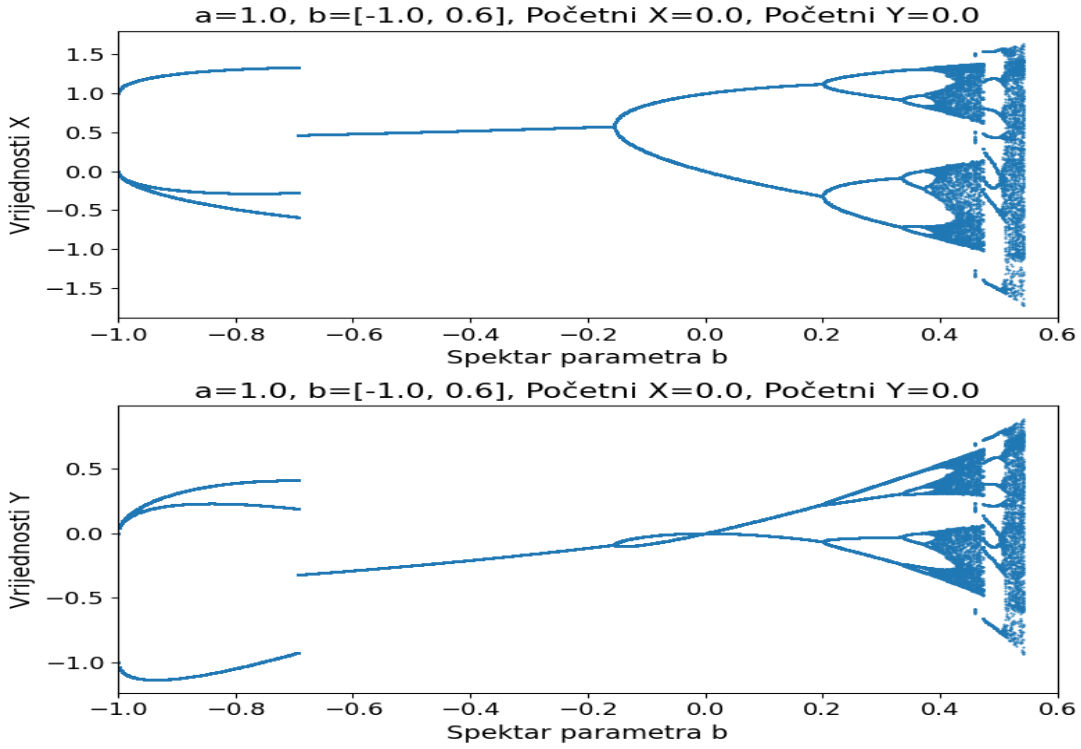
Sa slike 3.35 se vidi da je dijagram sličan gornjem dijagramu, ali za veći interval promatranja dodatnog parametra *a*, vidi se do koje granice je mapa stabilna te kada vrijednosti odlaze u beskonačnost. Na slici 3.36 je *a* fiksiran na 1.4, a *b* promjenjiv te je dobiven drugačiji dijagram, isto tako, za ove konkretne vrijednosti se vidi da se ipak nekada *x* i *y* vrijednosti ponašaju drugačije. Bifurkacijski dijagram *y* vrijednosti ovdje izgleda nalik na dijagram pripadnih *x* vrijednosti, no vidi se da je on deformiran i izobličen, a ovisno o početnim uvjetima mogu se i drugačija ponašanja uočiti (primjerice, za neke uvjete može se ostvariti da su *x* i *y* dijagrami "zrcalni").



Slika 3.35 Dobiveni bifurkacijski dijagram za širi spektar a vrijednosti



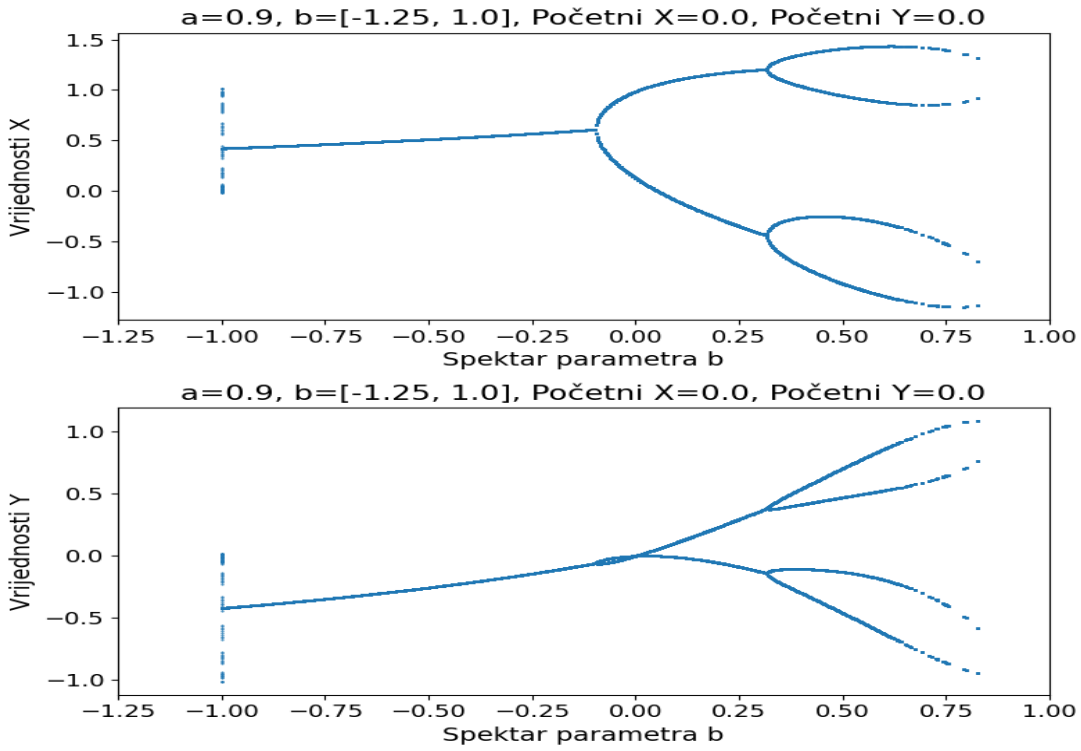
Slika 3.36 Dobiveni bifurkacijski dijagram za navedene parametre



Slika 3.37 Još jedan primjer izobličenog dijagrama

Sve u svemu, vidi se kako ova mapa znatno varira s ponašanjem s obzirom na početne uvjete, čak i toliko da nije dovoljan jedan *bifurkacijski* dijagram, već je očito da ih ona ima mnogo za različite

parametre te ih nije ni moguće sve proučiti u cijelosti. No ipak, kroz par primjera mogao se steći uvid u kompleksno ponašanje Hénonove te povući paralelu s logističkom mapom, koja je znatno jednostavnija za opisati. Dodatan svjedok tome je i slika 3.38., na kojoj je prikazan dijagram za čije vrijednosti spektra jednog parametra mapa uopće nema kaotična svojstva kada je drugi parametar fiksiran na zadanu konstantu.



Slika 3.38. Primjer dijagrama bez kaotičnih oscilacija

4. PRIMJENE DISKRETNIH KAOTIČNIH MAPA I ZAŠTO SU VAŽNE

Završno poglavlje rada kratko će se baviti proučavanjem primjene diskretnih kaotičnih mapa kao i njihovim uporabama u područjima inženjeringa, prema [10]. Objasnit će se zašto se ove mape koriste u generiranju pseudo slučajnih brojeva, istražit će se važnost njihovog korištenja u kriptografiji te će se dotaknuti i njihove primjene u modeliranju dinamičkih sustava, gdje mogu pružiti vrijedne uvide u kompleksno ponašanje stvarnih sustava.

4.1. Generiranje pseudo slučajnih brojeva

Diskretne kaotične mape igraju fascinantnu ulogu u generiranju pseudo-slučajnih brojeva te se u tu svrhu i intenzivno koriste, nudeći jedinstven i deterministički pristup slučajnosti. One su prigodne za generiranje pseudo-slučajnih brojeva baš zato što su računalno učinkovite i deterministčke. Korisnici mogu jednostavno reproducirati isti niz brojeva određivanjem istih početnih uvjeta i parametara, što može biti prednost za aplikacije koje zahtijevaju ponovljivost. Međutim, zbog njihove osjetljivosti na početne uvjete, oni su također osjetljivi na izbor parametara i mogu zahtijevati pažljivo podešavanje kako bi se postigla željena svojstva slučajnosti.

Kada se koriste na odgovarajući način, ključna prednost korištenja diskretnih kaotičnih mapa u ovom kontekstu je njihova sposobnost da proizvedu nizove koji pokazuju statistička svojstva slična pravim nasumičnim brojevima, dok su opet potpuno deterministički i ponovljivi uz iste početne uvjete i parametre. To ih čini vrijednim alatima za generiranje pseudo-slučajnih brojeva u širokom rasponu primjena koje pronalaze u raznim domenama gdje je potrebna slučajnost.

4.2. Kriptografija

U području kriptografije, diskretne kaotične mape igraju ključnu ulogu u poboljšanju sigurnosti digitalnih komunikacija i zaštite podataka. Kaotične mape nude izvrstan izvor nepredvidivih vrijednosti, što ih čini idealnim za generiranje kriptografskih ključeva i osiguravanje sigurnih procesa šifriranja i dešifriranja. Ugrađena osjetljivost na početne uvjete u kaotičnim mapama jamči da čak i manje promjene u početnoj vrijednosti rezultiraju potpuno drugačijim nizovima

vrijednosti, što napadačima na kriptosustave čini iznimno izazovnim predviđanje ili probijanje ključa obrnutim inženjeringom.

Kriptografski sustavi koji se temelje na kaosu, primjerice poput raznih generatora kaotičnog toka ključeva temeljenih na logističkoj mapi, implementirani su kako bi osigurali prijenos informacija, čuvajući osjetljive podatke u raznim aplikacijama u rasponu primjena od sigurne razmjene poruka do *online* financijskih transakcija. Deterministička priroda kaotičnih mapa u kombinaciji s njihovim pseudo slučajnim izlazom pruža snažan temelj za moderne kriptografske protokole, pridonoseći stalno razvijajućem polju kibernetičke sigurnosti.

4.3. Modeliranje dinamičkih sustava

Diskretne kaotične mape također su se pojavile kao alati u području modeliranja i simulacija dinamičkih sustava, omogućujući znanstvenicima i inženjerima proučavanje složenih pojava u stvarnom svijetu. One se često koriste za uvođenje kontrolirane slučajnosti u numeričke simulacije, povećavajući na taj način realističnost modela. Primjerice, u vremenskoj prognozi, kaotični nizovi mogu se koristiti za simulaciju turbulentne i nepredvidive prirode atmosferskih uvjeta, omogućujući time meteorolozima točnija predviđanja. U populacijskoj dinamici, već spomenuti kaotični pseudo slučajni brojevi pomažu modelirati stohastičnost koja je svojstvena stopama nataliteta, širenju bolesti i drugim varijablama, pomažući istraživačima u razumijevanju dinamike ekosustava.

Osim toga, diskretne kaotične karte koriste se za inicijalizaciju kaotičnih sustava u fizici i inženjerstvu, kao što su kaotični oscilatori ili kaotični krugovi, omogućujući istraživanje kaotičnog ponašanja i njegove primjene u poljima kao što su obrada signala i sigurna komunikacija. Njihova deterministička, ali kaotična priroda omogućuje preciznu ravnotežu između realizma i kontrole u simulaciji složenih sustava, što ih čini neprocjenjivim za istraživače koji traže uvid u dinamičke procese.

5. ZAKLJUČAK

Ovaj rad bavio se diskretnim kaotičnim mapama, koje su posebni tipovi dinamičkih sustava sa kaotičnim ponašanjem. Reklo se nešto o povijesti i o osnovnim konceptima matematičke teorije kaosa te napravljen osvrt na dinamičke sustave u općenitijem smislu, a i spomenule su se neke generalne činjenice i svojstva diskretnih kaotičnih mapa kako bi se približila tema čitatelju te mu se pružio validan kontekst u ostvarivanju lakšeg razumijevanja konkretne implementacije i vizualizacije u programskom jeziku Python. Kao konkretnim primjerima, rad se bavio logističkom i Hénonovom mapom, koje su dvije najpoznatije diskretne kaotične mape u području dinamičkih sustava i inženjeringa, s ciljem da se naučeno znanje u vizualizaciji i implementaciji ovih mapa može koristiti u daljnje znanstvene i istraživačke svrhe, ili pak služiti kao paralela pri daljnjem proučavanju drugih kaotičnih sustava i funkcija. U konačnici se rad kratko osvrnuo na raznovrsnu primjenu diskretnih kaotičnih mapa u drugim inženjerskim područjima i strukama, ne bi li potaknuo čitatelja na daljnju intrigu i zanimanje za ove veoma fascinantne i korisne matematičke funkcije. Kroz ovaj rad se pokazalo da su diskretne kaotične mape izuzetno moćan i jednostavan alat te da mogu biti korisne u širokom spektru potreba.

LITERATURA

[1] C., Oestreicher, A history of chaos theory [online], Dialogues in Clinical Neuroscience, br. 3, sv. 9, str 279-289, 2007., dostupno na: 10.31887/DCNS.2007.9.3/coestreicher [pristupljeno 30. lipnja 2023.]

[2] Britannica, The Editors of Encyclopaedia, "chaos theory". Encyclopedia Britannica, dostupno na https://www.britannica.com/science/chaos-theory, [pristupljeno 30. lipnja 2023.]

[3] H.R., Biswas, M., Hasan, S.K., Bala, Chaos theory and it's applications on our real life [online], dostupno na:

https://bu.ac.bd/uploads/BUJ1V5I12/6.%20Hena%20Rani%20Biswas.pdf [pristupljeno 30. lipnja 2023.]

[4] D.K., Arrowsmith, C.M., Place, 1990., An introduction to dynamical systems [online], dostupno na:

https://books.google.hr/books?hl=hr&lr=&id=HOU6q6wHCncC&oi=fnd&pg=PP10&dq=dynamical+systems&ots=qSOmb5Con-&sig=fRrZWe-qcACw4xhp3hODDSGAsIA&redir_esc=y#v=onepage&q=dynamical%20systems&f=false [pristupljeno 30. lipnja 2023.]

[5] S., Sternberg, 2013., Dynamical Systems [online], dostupno na:

https://books.google.hr/books?hl=hr&lr=&id=T2uTAwAAQBAJ&oi=fnd&pg=PP1&dq=dynamical+systems&ots=i5VHk4z9MK&sig=1WZmgX8nVb-dz48UnxP-vlbmBRw&redir_esc=y#v=onepage&q=dynamical%20systems&f=false [pristupljeno 30. lipnja 2023.]

[6] Martelli, M., 2011., Introduction to Discrete Dynamical Systems and Chaos (Wiley Series in Discrete Mathematics and Optimization). John Wiley & Sons, dostupno na: https://books.google.hr/books?hl=hr&lr=&id=QM45FTTqXScC&oi=fnd&pg=PR7&dq=discrete+dynamical+systems&ots=sdCpiwiZNS&sig=aHuZqAUB8LHitwGe4Yf-wSS_pQQ&redir_esc=y#v=onepage&q=discrete%20dynamical%20systems&f=false [pristupljeno 30. kolovoza 2023.]

[7] Steven Strogatz, Course MAE5790, YouTube, Nonlinear Dynamics and Chaos Cornell University, dostupno na:

https://www.youtube.com/watch?v=ycJEoqmQvwg&list=PLbN57C5Zdl6j_qJA-pARJnKsmROzPnO9V [pristupljeno 30. kolovoza 2023.]

[8] Ausloos, M., 2006., The Logistic Map and the Route to Chaos: From the Beginnings to Modern Applications (ilustrirano izdanje). Understanding Complex Systems, Springer Science & Business Media, dostupno na: https://books.google.hr/books?hl=hr&lr=&id=8pDLfRv2RVcC&oi=fnd&pg=PA3&dq=logistic+map&ots=0YObRzXSIJ&sig=3dbTLPvdsuxUydgD0jpJ1RjHrdI&redir_esc=y#v=onepage&q=logistic%20map&f=false [pristupljeno 30. kolovoza 2023.]

[9] Z. Galias, "Dynamics of the Hénon Map in the Digital Domain," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 70, no. 1, pp. 388-398, Jan. 2023, doi: 10.1109/TCSI.2022.3217139., dostupno na: https://ieeexplore.ieee.org/abstract/document/9933045 [pristupljeno 30. kolovoza 2023.]

[10] Bahi, J. M., & Guyeux, C. (2013). Discrete Dynamical Systems and Chaotic Machines: Theory and Applications (ilustrirano izdanje). Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series., dostupno na: https://books.google.hr/books?hl=hr&lr=&id=odJDWmTuUtAC&oi=fnd&pg=PP1&dq=discrete+dynamical+systems&ots=Trj-Y-tDFs&sig=2QXVLXzdRf1Y1MBlNN8Qy8BtjBY&redir_esc=y#v=onepage&q=discrete%20dynamical%20systems&f=false [pristupljeno 30. kolovoza 2023.]

SAŽETAK

Rad "Diskretne kaotične mape" služi kao uvod u proučavanje diskretnih kaotičnih mapa i relevantne pozadine nužne za njihovo razumijevanje. Cilj rada je bio istražiti teoriju kaosa te dinamičke sustave kao i proučiti primjene tih mapa u različitim područjima poput generiranja pseudo slučajnih brojeva, kriptografije i modeliranja dinamičkih sustava. Teorijska pozadina obuhvaća uvod u teoriju kaosa, s osnovnim konceptima kao što su osjetljivost na početne uvjete, determinizam i nepredvidljivost. Prikazana je povijesna važnost teorije kaosa, pri čemu su Henri Poincaré i Edward Lorenz pioniri u tom području. Rad također spominje druge ključne koncepte poput determinizma, *fraktala*, nestabilnosti, *bifurkacija*, *atraktora* i putanja, a istražena je i tema implementacije i vizualizacije diskretnih kaotičnih mapa u programskom jeziku Python.

KLJUČNE RIJEČI

Teorija kaosa, dinamički sustavi, kaotične mape, diskretne kaotične mape, determinizam, atraktori, fraktali, oscilacije, putanje, periodičnost, kaos, osjetljivost na početne uvjete, linijski dijagram, dijagram raspršenja, bifurkacijski dijagram, logistička mapa, Hénonova mapa, Python, NumPy, Matplotlib, pseudo slučajni brojevi, kriptografija, modeliranje dinamičkih sustava

ABSTRACT

The paper "Discrete Chaotic Maps" serves as an introduction to the study of discrete chaotic maps and the relevant background necessary for their understanding. The aim of the paper was to explore chaos theory and dynamical systems, as well as to examine the applications of these maps in various areas such as generating pseudo random numbers, cryptography, and the modeling of dynamical systems. The theoretical background includes an introduction to chaos theory, covering fundamental concepts such as sensitivity to initial conditions, determinism, and unpredictability. The historical significance of chaos theory is presented, with Henri Poincaré and Edward Lorenz being pioneers in the field. The paper also mentions other key concepts such as determinism, fractals, instabilities, bifurcations, attractors and orbits, and then also explores the topic of implementing and visualizing discrete chaotic maps in the Python programming language.

KEYWORDS

Chaos theory, dynamical systems, chaotic maps, discrete chaotic maps, determinism, attractors, fractals, oscillations, orbits, periodicity, chaos, sensitivity to initial conditions, line plot, scatter plot, bifurcation diagram, logistic map, Hénon map, Python, NumPy, Matplotlib, pseudo random numbers, cryptography, modeling of dynamical systems.

ŽIVOTOPIS

Adrijan Malinović, rođen 30. ožujka 2001. godine u Osijeku. Započeo svoje obrazovanje u osnovnoj školi Augusta Šenoe u Osijeku, gdje je pokazao iznimnu predanost i uspjeh tijekom svih školskih godina. Nakon završetka osnovne škole, nastavio svoje obrazovanje u III. Gimnaziji Osijek, gdje je proučavao matematičke i informatičke discipline. Tijekom srednjoškolskog obrazovanja, također je postigao odlične rezultate. Nakon uspješno završene srednje škole, upisao se na Fakultet Elektrotehnike, Računarstva i Informacijskih Tehnologija u Osijeku, smjer računarstvo. Odlučio je slijediti svoju strast prema računalima i tehnologiji te se posvetiti dubljem razumijevanju ovog područja.

_____

Potpis autora

PRILOZI

Video materijali:

- Dynamical Systems and Chaos
- Chaos: The Science of the Butterfly Effect
- This equation will change how you see the world (the logistic map)

https://www.youtube.com/playlist?list=PLF0b3ThojznQwpDEClMZmHssMsuPnQxZT

https://www.youtube.com/watch?v=fDek6cYijxI

https://www.youtube.com/watch?v=ovJcsL7vyrk