UNIWERSYTET GDAŃSKI

WYDZIAŁ MATEMATYKI, FIZYKI I INFORMATYKI

Anna Dąbkowska

Daniel Jarzymowski

Maciej Ossowski

Kierunek studiów: **Modelowanie matematyczne i analiza danych**
Przedmiot: **Numeryczne modelowanie układów dynamicznych**

Projekt zaliczeniowy

Gdańsk 2023

Spis treści

Wprowadzenie 2

1 Zadanie 1 **3**

2 Zadanie 2 **4**

3 Zadanie 3 **7**

4 Zadanie 4 **10**

Wprowadzenie

Praca powstała na podstawie wiedzy, którą nabyliśmy podczas zajęć laboratoryjnych. Samodzielnie opracowaliśmy wszystkie cztery zadania.

Rozdział 1

Zadanie 1

Treść zadania: Poniższa macierz jest macierzą Lesliego dla pewnej populacji insektów podzielonej na cztery przedziały wiekowe.

$$\begin{pmatrix} 0 & 3 & 3 & 3 \\ 0.2 & 0 & 0 & 0 \\ 0 & 0.29 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \end{pmatrix}$$

Znajdź rozkład populacji po 10 latach oraz opisz jej zachowanie po długim okresie czasu (stwierdzając, czy będzie się ona rozwijać, kurczyć, czy stabilizować), jeśli rozkład początkowy to $(1000, 2000, 1500, 500)$.

Rozwiązanie:

Rozdział 2

Zadanie 2

Treść zadania: Dana jest następująca funkcja:

$$f(x) = 3.5x - 3.5x^2$$

Znajdź jej punkty stałe oraz punkty okresowe o okresie 2. Wykonaj to algebraicznie oraz zilustruj na wykresach. Zweryfikuj stabilność punktów stałych.

Rozwiązanie:

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import Symbol, solve, lambdify

# algebraiczne wyznaczenie punktów stałych i okresowych

mu = 3.5

# funkcja logistyczna
def logistic(mu, x):
    return mu*x*(1-x)

# złożenie funkcji logistycznej
def logistic2(mu, x):
    return logistic(mu, logistic(mu,x))


# punkty stałe
d = Symbol('d', real=True)
pkt_stale = solve(logistic(mu,d)-d, d)
print('Punkty stałe:', pkt_stale)

# punkty okresowe o okresie 2
# wykorzystujemy funkcję jednokrotnie złożoną
pkt_okresowe = solve(logistic2(mu,d)-d, d)
print('\nPunkty okresowe:', pkt_okresowe)

# ilustracja

def logistic_cobweb(mu, x0, N):
```

```python

    t = np.linspace(0, 1, 200)
    ax.plot(t, logistic(mu, t), 'k', lw=2)
    x = x0

    for i in range(N):
        y=logistic(mu, x)
        ax.plot([x,x],[x,y], 'k', lw=0.9)
        ax.plot([x,y],[y,y], 'k', lw=0.9)
        ax.plot([x], [y], 'ok', ms=8, alpha=(i+1)/N)
        x = y
    plt.title(f'$\mu$={mu}, $x_0$={x0}')

fig, ax = plt.subplots(figsize=(8,7), dpi=150)

logistic_cobweb(mu, .15, 10)
x = np.linspace(0, 1, 200)
ax.plot(x, x, c='red')

ax.set_xlabel('x', fontsize = 10)
ax.set_ylabel('Logistic(x)', fontsize = 10)
ax.tick_params(labelsize = 10)


for i in range(1,4):
    ax.axvline(pkt_okresowe[i], linestyle='--', alpha=.75, linewidth=1, c='green')
for i in range(1,2):
     ax.axvline(pkt_stale[i], linestyle=':', alpha=.25, linewidth=5, c='purple')

ax.set_ylim([0,1])
okr = plt.scatter(pkt_okresowe, [0,0,0,0], c='green', clip_on=False, alpha=.75)
st = plt.scatter(pkt_stale, [0,0], clip_on=False, facecolors='none', edgecolors='purple', s=120, alpha=.45)


plt.legend([okr, st], ['punkty okresowe o okresie 2', 'punkty stałe'])

# Weryfikacja stabilnosci punktów stałych
# z kryterium różniczkowania:
# sprawdzenie, czy pochodna funkcji w punkcie stałym
# jest mniejsza niż 1 co do modułu

x = Symbol('x')
f = mu*x*(1-x)

f_prime=f.diff(x)
print('Pochodna: ', f_prime)

# uczynienie pochodnej f funkcją podstawialną
f_prime=lambdify(x, f_prime)
```

```python

print('\nPunkt', pkt_stale[0], ': ', f_prime(pkt_stale[0]))
print('Punkt', pkt_stale[1], ': ', f_prime(pkt_stale[1]))
# dla obu z tych punktów moduł pochodnej jest większy niż 1,
# więc oba te punkty są niestabilne.
```

Rozdział 3

Zadanie 3

Treść zadania: Wygeneruj diagram bifurkacji dla funkcji:

$$G(x, \alpha) = e^{-\alpha x^2} - 0.5$$

Na jego podstawie oszacuj wartości dwóch najmniejszych bifurkacji b_1, b_2. Dobierając pewien parametr α taki, że $b_1 < \alpha < b_2$, wyświetl iteracje $G^0, \ldots, G^{20}$ orbity układu, której wyrazem początkowym G^0 jest $x_0 = 0$ oraz zinterpretuj długofalowe zachowanie takiej orbity.

Rozwiązanie:

```
import numpy as np
import matplotlib.pyplot as plt
from math import e

# Diagram bifurkacji

def gauss(alpha, beta,  x):
    return e**(-alpha*x**2) + beta

beta = -.5
alpha_values = np.linspace(0,10,2000)
y = []

for alpha in alpha_values:
    x = 0.1
    for i in range(500):
        x = gauss(alpha, beta, x)
    for i in range(50):
        x = gauss(alpha, beta, x)
        y.append([alpha, x])


y = np.array(y)
fig, ax = plt.subplots(figsize=(8,6), dpi=150)
plt.plot(y[:,0], y[:,1], 'k.', markersize=0.03)

plt.xlabel('$\\alpha$', fontsize=10)
plt.ylabel('x', fontsize=10)
```

```
plt.title('Diagram bifurkacji dla funkcji $G(x,\\alpha,\\beta)=e
    ^{-\\alpha x^2}-\\beta$ \n $\\beta$=-0.5, $x_0$=0.1')

# pierwsza bifurkacja zdaje się być w przybliżeniu
# b1 = 1.9, b2 = 3.95
# przyjmę następującą wartosć alfa
alpha = 2.08

# Iteracje orbity układu,
# interpretacja długofalowego zachowania orbity

def cobweb(alpha, beta, x0, n, ax = None):
    t = np.linspace(-1, 1, 500)
    ax.plot(t, gauss(alpha, beta, t), 'k', lw=2)
    ax.plot([-1,1], [-1, 1], 'r', lw=2) # funkcja y=x
    x = x0

    for i in range(n):
        y = gauss(alpha, beta, x)
        ax.plot([x,x], [x,y], 'k', lw=1)
        ax.plot([x,y], [y,y], 'k', lw=1)
        ax.plot([x], [y], 'ok', ms=10, alpha=(i+1)/n)
        x = y

    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_title(f'Wykres orbity układu \n$\\alpha$={alpha}, $\\
    beta={beta}$, $x_0$={x0}')
    ax.grid()



def gauss_iterations(alpha, beta, x0, n, ax=None):
    x = x0
    x_list = [x0]

    for i in range(n-1):
        x = gauss(alpha, beta, x)
        x_list.append(x)

    plt.plot(x_list, 'o:r')
    ax.set_xlabel('nr iteracji')
    ax.set_title(f'Wartosci orbity układu \n$\\alpha$={alpha}, $\\
    beta={beta}$, $x_0$={x0}')
    ax.set_xticks(np.arange(20))
    ax.grid()


# punkt startowy x0=0, liczba iteracji n=20
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,6), sharey=True,
    dpi=150)
cobweb(alpha, beta, 0, 20, ax = ax1)
gauss_iterations(alpha, beta, 0, 20, ax=ax2)

```

```
# Interpretacja: układ taki będzie miał dwa stabilne
# punkty okresowe o okresie 2 (dla zadanego alfa są to
# ok. 0.14 oraz ok. 0.43).
```

Rozdział 4

Zadanie 4

Treść zadania: Przeprowadź grę w chaos, generując trójkąt Sierpińskiego. Zamiast środków odcinków użyj jawnych wzorów na kontrakcje oraz prawdopodobieństwa z rozkładu klasycznego. Używając stałych Lipschitza zastosowanych kontrakcji, podaj wartość wymiaru fraktalnego uzyskanego zbioru.

Rozwiązanie:

```python
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm
from math import sqrt
from math import log

# Kontrakcje potrzebne do utworzenia Trójkąta Sierpińskiego
def f1(x, y):
    return (x/2, y/2)

def f2(x, y):
    return ((x+1)/2, y/2)

def f3(x, y):
    return (x/2 + 1/4, y/2 + sqrt(3)/4)


IFS = [f1, f2, f3]
num = 300000

# Szerokość i wysokość rysunku
width = height = 1000
fern = np.zeros((width, height))
x, y = 0, 0

for i in range(num):
    f = np.random.choice(IFS, p = [1/3, 1/3, 1/3])
    # prawdopodobieństwa muszą sumować się do 1
    x, y = f(x, y)
    cx, cy = int( x*width), int(y*height)
    fern[cy, cx] = 1 # transpozycja konieczna

```

```
fig, ax = plt.subplots(figsize = (12,12))
plt.imshow(fern[::-1, :], cmap = cm.Blues)
ax.axis('off')

# Obliczamy wymiar samopodobieństwa.
# Ponieważ cały Trójkąt jest podobny do siebie w skali 2
# (stała Lipschitza), a powtarza się trzykrotnie,
# zatem wymiar fraktalny liczymy w następujący sposób:
L = 1/2
dim = log(3)/log(1/L)
print('Wymiar fraktalny zbioru generowanego przez Trójkąt Sierpiń
   skiego:',dim)
```