Goodwin-Griffith-genetic-oscillator-model

Krzysztof Olech

25 stycznia 2023

Kontrola genów Model zaproponowany przez Griffitha, w którym X to koncentracja pewnego białka proporcjonalna do aktywności opisywanego genu, a Y to koncentracja odpowiedniego mRNA,

$$\dot{X} = -\alpha X + Y$$

$$\dot{Y} = \frac{x^2}{1 + X^2} - \beta Y$$

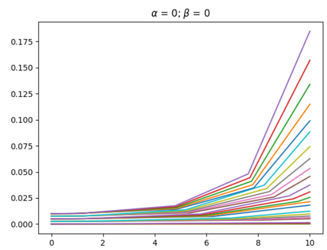
Spis treści

1 Pkty stale

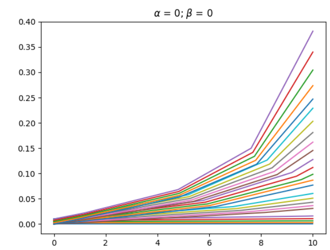
Analize naszego ukladu zaczynamy od analizy pktow stalych. Korzystajac z Wolframa mozemy doknac analizy naszych rownan.

1.1 Rowzionzania Trywialne

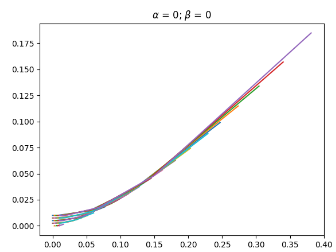
α i $\beta = 0$



Wykresy zachowania X



Wykresy zachowania Y



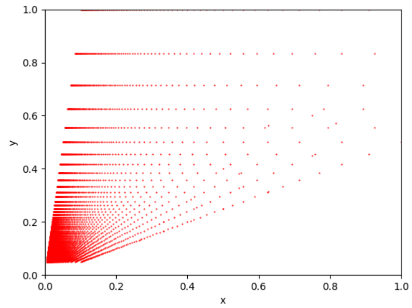
Wykres Fazowy

Mozemy zaobserwowac ze nasze rozwionzania tylko i wylocznie dla x i y = 0 wiec mozemy wysnuac wniosek ze jest to chwiejny pkt stacionarny postaramy sie potem to udowdonic kozystajac z analizy jakobianu.

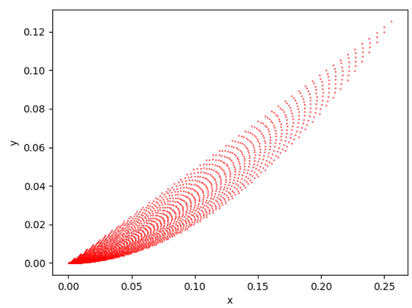
1.2 Rozwionzania nie trywialne

1.2.1 Przypadek pierwszy

$$X = -\frac{\sqrt{1-4\alpha^2\beta^2}-1}{2\beta} \qquad Y = -\frac{\sqrt{1-4\alpha^2\beta^2}-1}{2\beta}$$



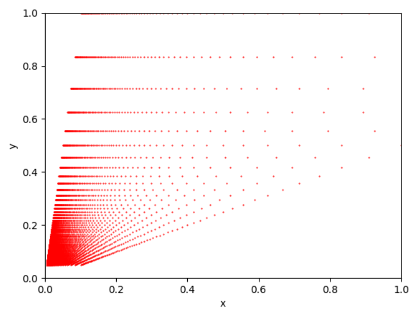
a i b in 0.5 do 10



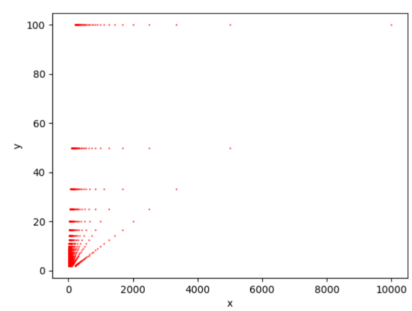
a i b in od 0.01 do 0.5

1.2.2 Przypadek drogi

$$X = \frac{\sqrt{1-4\alpha^2\beta^2}+1}{2\beta} \qquad Y = \frac{\sqrt{1-4\alpha^2\beta^2}+1}{2\beta}$$



a i b in 0.5 do 10



a i b in 0.01 do 0.5

Zastanawiające jest to że dla większych wartości α i β pkt stabilne dla obu równań są dokradnie takie same.

1.3 Jakobian

$$\begin{Bmatrix} -\alpha & 1 \\ \frac{2x}{(x^2+1)^2} & \beta \end{Bmatrix}$$

Potencjalne pkt zerujące równania:
$-\alpha X + Y = 0 \Rightarrow Y = \alpha X$
$\frac{x^2}{1+X^2} - \beta Y = 0 \Rightarrow \frac{x^2}{1+X^2} = \beta Y$
$\frac{x^2}{1+x^2} = \beta\alpha x$
rozwiązania:
trywialne x= 0
nie trywialne dla α i $\beta \neq 0$
$x = \frac{1-\sqrt{1-4\alpha^2\beta^2}}{2\alpha\beta}$
$x = \frac{\sqrt{1-4\alpha^2\beta^2}-1}{2\alpha\beta}$

1.3.1 Rozwiązania dla $x = \frac{1-\sqrt{1-4\alpha^2\beta^2}}{2\alpha\beta}$

$$\lambda_1 =$$

$(16\ \alpha^3\beta^2 + 16\alpha^2\beta^3 + 8\alpha\sqrt{1-4\alpha^2\beta^2} + 8\beta\sqrt{1-4\alpha^2\beta^2}$

-$\sqrt{(-16\alpha^3\beta^2 - 16\alpha^2\beta^3 - 8\alpha\sqrt{1-4\alpha^2\beta^2} - 8\beta\sqrt{1-4\alpha^2\beta^2} + 8\alpha + 8\beta)^2}$

$\overline{-4(1024\alpha^5\beta^5 - 768\alpha^3\beta^3 - 128\alpha\beta\sqrt{1-4\alpha^2\beta^2}) - 256\alpha^5\beta^5\sqrt{1-4\alpha^2\beta^2}}$

$\overline{+512\alpha^3\beta^3\sqrt{1-4\alpha^2\beta^2} + 128\alpha\beta))} - 8\alpha - 8\beta$

/(2 (2 - 2 $\sqrt{1-4\alpha^2\beta^2})^2$)

$$\lambda_2 =$$

$(16\ \alpha^3\beta^2 + 16\alpha^2\beta^3 + 8\alpha\sqrt{1-4\alpha^2\beta^2} + 8\beta\sqrt{1-4\alpha^2\beta^2}$

+ $\sqrt{(-16\alpha^3\beta^2 - 16\alpha^2\beta^3 - 8\alpha\sqrt{1-4\alpha^2\beta^2} - 8\beta\sqrt{1-4\alpha^2\beta^2} + 8\alpha + 8\beta)^2}$

$\overline{-4(1024\alpha^5\beta^5 - 768\alpha^3\beta^3 - 128\alpha\beta\sqrt{1-4\alpha^2\beta^2} - 256\alpha^5\beta^5\sqrt{1-4\alpha^2\beta^2} + 512\alpha^3\beta^3\sqrt{1-4\alpha^2\beta^2} + 128\alpha\beta)}$

- 8 $\alpha - 8\beta)$

/(2 (2 - 2 $\sqrt{1-4\alpha^2\beta^2})^2$)

1.3.2 Rozwiązania dla $x = \frac{\sqrt{1-4\alpha^2\beta^2}-1}{2\alpha\beta}$

$$\lambda_1 =$$

$(16\ \alpha^3\beta^2 + 16\alpha^2\beta^3 + 8\alpha\sqrt{1-4\alpha^2\beta^2} + 8\beta\sqrt{1-4\alpha^2\beta^2}$

- $\sqrt{(-16\alpha^3\beta^2 - 16\alpha^2\beta^3 - 8\alpha\sqrt{1-4\alpha^2\beta^2} - 8\beta\sqrt{1-4\alpha^2\beta^2} + 8\alpha + 8\beta)^2}$

$\overline{-4(-512\alpha^5\beta^5 - 256\alpha^3\beta^3 - 128\alpha\beta\sqrt{1-4\alpha^2\beta^2} + 256\alpha^5\beta^5\sqrt{1-4\alpha^2\beta^2} + 128\alpha\beta)}$

- $8\ \alpha - 8\beta)/(2(2-2\sqrt{1-4\alpha^2\beta^2})^2)$

$$\lambda_2 =$$

$(16\ \alpha^3\beta^2 + 16\alpha^2\beta^3 + 8\alpha\sqrt{1-4\alpha^2\beta^2} + 8\beta\sqrt{1-4\alpha^2\beta^2}$

$+\ \sqrt{(-16\alpha^3\beta^2 - 16\alpha^2\beta^3 - 8\alpha\sqrt{1-4\alpha^2\beta^2} - 8\beta\sqrt{1-4\alpha^2\beta^2} + 8\alpha + 8\beta)^2}$

$\overline{-4(-512\alpha^5\beta^5 - 256\alpha^3\beta^3 - 128\alpha\beta\sqrt{1-4\alpha^2\beta^2} + 256\alpha^5\beta^5\sqrt{1-4\alpha^2\beta^2} + 128\alpha\beta)}$

- $8\ \alpha - 8\beta)/(2(2-2\sqrt{1-4\alpha^2\beta^2})^2)$

Niestety rozwiązanie Tych równań jest bardzo zmożone i cienkie dlatego nie wykonamy analizy stabilności Kotow stałych z ich pomocą lecz wykonamy to numerycznie.

Próba znalezienia numerycznego rozwiązania nie udała się ze względu na niska ilość algorytmów umożliwiających analizę równań uwikłanych. Dostępne są za peywalem na matematice.

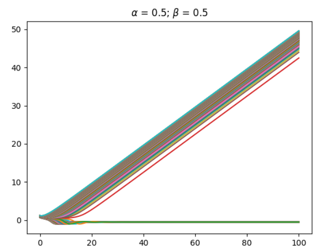
załącznik 1.

2 analiza

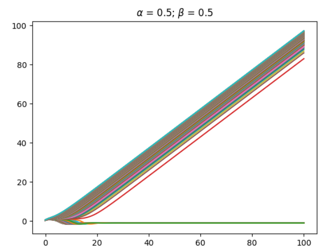
Pierwsze wykresy zostały stworzone dla szerokiego zakresu pkt x i y dla kilku stałych a i b Załącznik 2.

2.1 $\alpha = 0.5$ $\beta = 0.5$

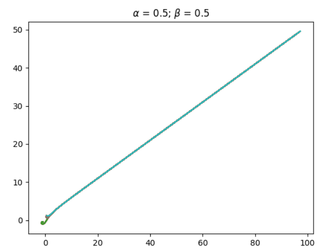
x i y początkowe z zakresu od 0.1 do 1 (wsystkei kombinacje co około 0.01)



Wykresy zachowania X



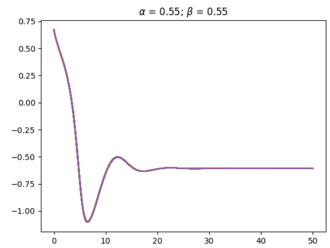
Wykresy zachowania Y



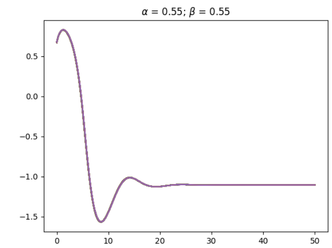
Wykres Fazowy

2.2 $\alpha = 0.55$ $\beta = 0.55$

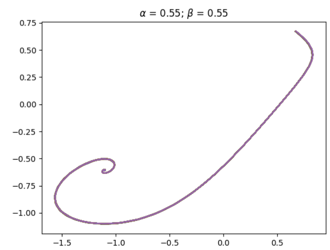
x i y początkowe z zakresu od 0.7 do 0.71



Wykresy zachowania X



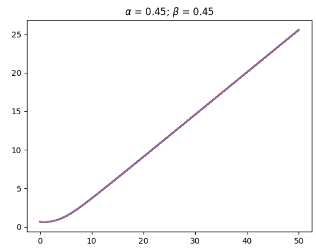
Wykresy zachowania Y



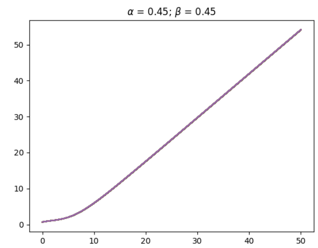
Wykres Fazowy

2.3 $\alpha = 0.45$ $\beta = 0.45$

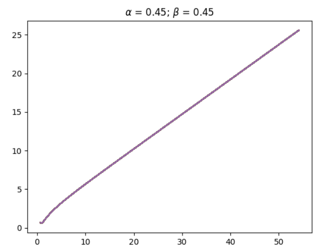
x i y początkowe z zakresu od 0.7 do 0.71



Wykresy zachowania X



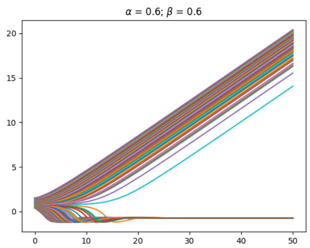
Wykresy zachowania Y



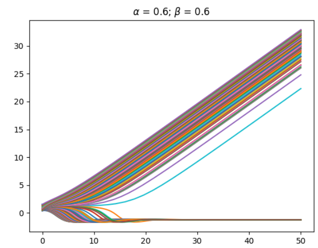
Wykres Fazowy

2.4 $\alpha = 0.6$ $\beta = 0.6$

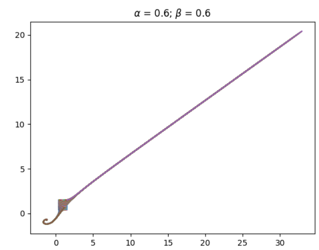
x i y początkowe z zakresu od 0.5 do 1



Wykresy zachowania X



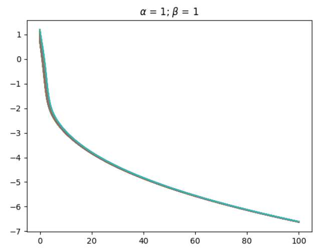
Wykresy zachowania Y



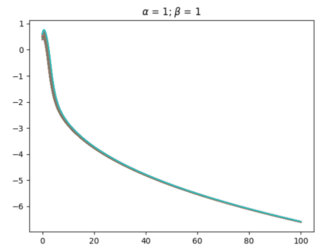
Wykres Fazowy

2.5 $\alpha = 1$ $\beta = 1$

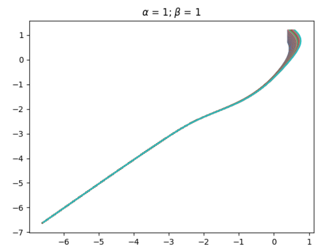
x i y początkowe z zakresu od 0.5 do 1



Wykresy zachowania X



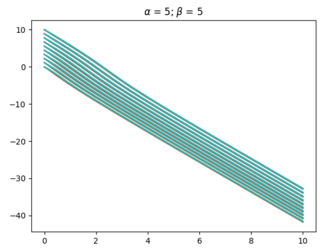
Wykresy zachowania Y



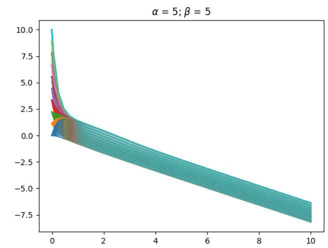
Wykres Fazowy

2.6 $\alpha = 5$ $\beta = 5$

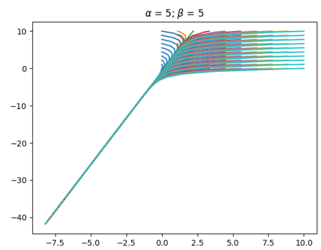
x i y początkowe z zakresu od 0 do 10



Wykresy zachowania X



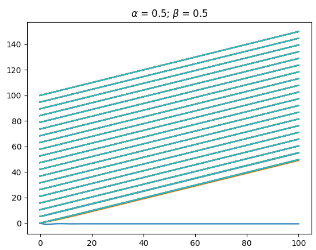
Wykresy zachowania Y



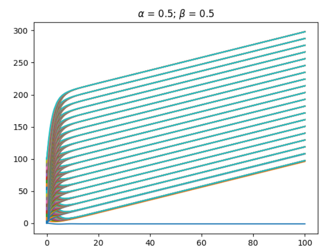
Wykres Fazowy

2.7 $\alpha = 0.5\ \beta = 0.5$

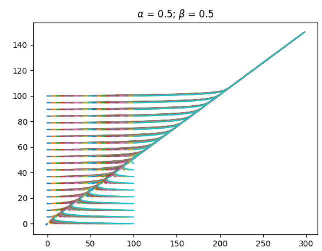
x i y początkowe z zakresu od 0 do 100



Wykresy zachowania X



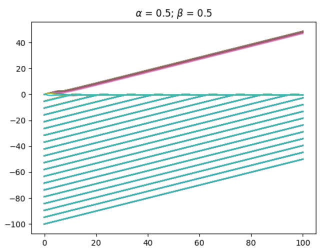
Wykresy zachowania Y



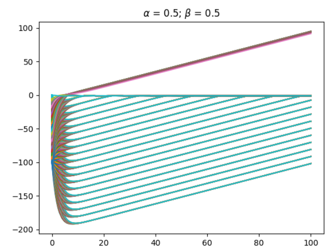
Wykres Fazowy

2.8 $\alpha = 0.5$ $\beta = 0.5$

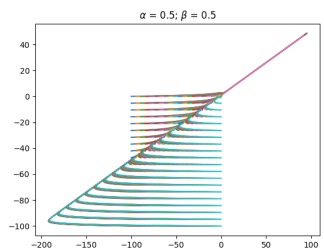
x i y początkowe z zakresu od-100 do 0 - nie rzeczywisty zakres stężenie białka ujemne



Wykresy zachowania X



Wykresy zachowania Y



Wykres Fazowy

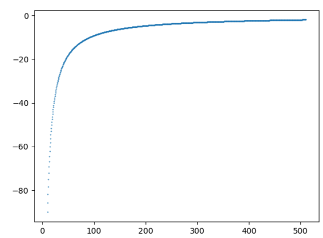
2.9 Wnioski

Analizując nie rzeczywiste wyniki możemy ustalić ze na pewno po stronie ujemnej pkt x y = 0 jest atraktorem dla nie za duzych dodatnich zmiennych x i y jest atraktorem również po dodatniej stronie.

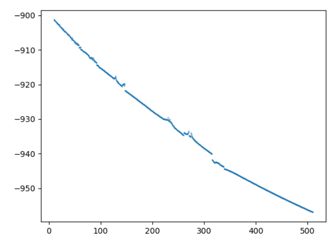
3 Bifurkacje

Poszukiwanie pktu podejrzanego o Bifurkacje. Po napisaniu Kodu umozliwiajacego liczenie mi Bifurkacje załącznik 3 zaczełem szukac czy istnieje pkt w któym wystompi Bifurkacja.

3.1 α w zakresie od 0 do 500

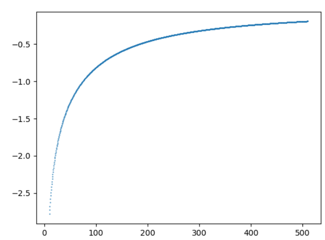


Wykresy zachowania X

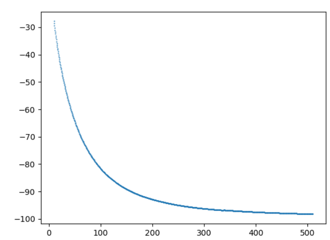


Wykresy zachowania Y

3.2 β w zakresie od 0 do 500

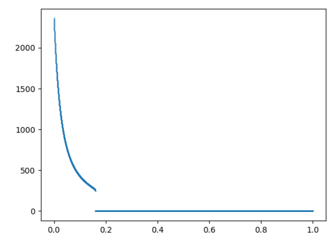


Wykresy zachowania X

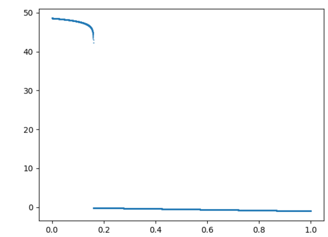


Wykresy zachowania Y

3.3 α w zakresie od 0 do 1, 10k pkt

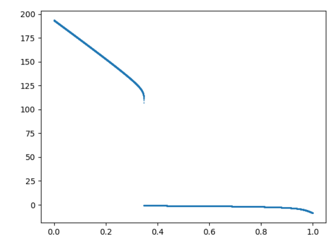


Wykresy zachowania X

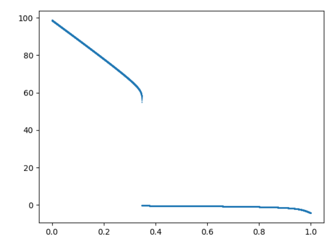


Wykresy zachowania Y

3.4 β w zakresie od 0 do 1, 10k pkt



Wykresy zachowania X



Wykresy zachowania Y

3.5 Wnioski

Możemy zaobserwować brak bifurkacji dla sprawdzanych parametrów oraz możemy zaobserwować ze istnieją nie ciągłości funkcji dla kilku pktów w szczególności w okolicach $\beta = 0.3$ oraz $\alpha = 0.18$ które mogły by sugerować możliwość zaistnienia bifurkacji w okolicach tamtego pkt niestety nie udało mi się ich wyznaczyć a z uwagi na brak analitycznego określenia pktów stałych nie byłem w stanie dokładniej ich określić zęby stwierdzić czy są możliwe.

4 Wnioski

Niestety z uwagi na niska znajomość procesów biologicznych z mojej strony nie dokona dobrze zrozumiałem model Biologiczny który był symulowany. Wiem ze zmienne x i y określają stężenia określonych protein. Za to parametry α i β określają prędkości wymiany określonych protein. W głównej publikacji rozważano dużo bardziej złożone równania niż te które ja rozważałem. Możliwe ze brak oscylacji lub jej bardzo szybkie tłumienie sprowadza się do tego lub za niskiej dokładności pakietu użytego do całkowania.

5 Załączniki

https://github.com/KrOlech/Goodwin-Griffith-genetic-oscillator-model

5.1 Załącznik 0

Kod źródłowy 1: Uniwersalne Funkcje.

```
import numpy as np
from numba import jit
from scipy.integrate import odeint, solve_ivp

@jit(nopython=True)
def xDot(x, y, alfa):
    return - alfa * x + y


@jit(nopython=True)
def yDot(x, beta):
    x2 = x * x
    return x2 / (1 + x2) - beta


def dfun(r, t, alfa, beta):
    tab = np.zeros((2, 2))
    tab[0, 0] = - alfa
    tab[0, 1] = 1
    tab[1, 0] = 2 * r[0] / pow(r[0] * r[0] + 1, 2)
    tab[1, 1] = - beta

    return tab


def model(t, r, alfa, beta):
    return [xDot(*r, alfa), yDot(r[0], beta)]
```

5.2 Załącznik 1

Kod źródłowy 2: Pkty Zerowe Funkcji.

```
from numpy import arange
from cmath import sqrt
import matplotlib.pyplot as plt
from numba import jit


@jit(nopython=True)
def pkt1X(m4ab, a, b):
    return - (m4ab - 1) / (a * b)


@jit(nopython=True)
def pkt1Y(m4ab, a, b):
    return - (m4ab - 1) / b


@jit(nopython=True)
def pkt2X(m4ab, a, b):
    return (m4ab + 1) / (a * b)


@jit(nopython=True)
def pkt2Y(m4ab, a, b):
    return (m4ab + 1) / b


def wykres(f1, f2, name):
    for a in arange(0.01, 0.5, 0.01):
        for b in arange(0.01, 0.5, 0.01):
            m4ab = sqrt(1 - 4 * a * a * b * b)
            b2 = 2 * b
            plt.plot(f1(m4ab, a, b2), f2(m4ab, a, b2), marker='.', c='r', markersize=1)
    plt.xlabel("x")
    plt.ylabel("y")
    #plt.ylim(0, 1.0)
    #plt.xlim(0, 1.0)
    plt.savefig(name + '.png')
    plt.cla()
    plt.close()


def main():
    wykres(pkt1X, pkt1Y, 'ujemneNNN')
    wykres(pkt2X, pkt2Y, 'dodatnieNNN')


if __name__ == "__main__":
    main()
```

5.3 Załącznik 1.5

Kod źródłowy 3: Pkty Zerowe Funkcji Próba okreslenia stabilnosci.

```python
from scipy.optimize import root


def fun1(a,b):

    ab = a * b
    a2 = a * a
    b2 = b * b
    a3 = a2 * a
    b3 = b2 * b
    a5 = a2 * a3
    b5 = b2 * b3
    ab55 = a5 * b5
    ab32 = a3 * b2
    ab33 = a3 * b3
    a8 = 8 * a
    b8 = 8 * b
    a2_16 = 16 * a2
    ab128 = 128 * ab
    ab32_16 = 16 * ab32
    pierwiastek = pow(1 - 4 * a2 * b2, 1 / 2)
    iner = (-ab32_16 - a2_16 * b3 - a8 * pierwiastek - b8 * pierwiastek + a8 + b8)
    return (ab32_16
            + a2_16 * b3
            + a8 * pierwiastek
            + b8 * pierwiastek +
            pow(iner * iner
                - 4 * (1024 * ab55
                       - 768 * ab33
                       - ab128 * pierwiastek
                       - 256 * ab55 * pierwiastek
                       + 512 * ab33 * pierwiastek
                       + ab128)
                - a8 - b8
                , 1 / 2)
            / (2 * pow(2 - 2 * pierwiastek, 1 / 2)))


def main():
    print(root(fun1, 10, 10))


if __name__ == "__main__":
    main()
```

5.4 Załącznik 2

Kod źródłowy 4: Pierwsze sprawdzenie dla stałego α i β.

```python
from concurrent.futures import ThreadPoolExecutor

import matplotlib.pyplot as plt
from numpy import linspace
from scipy.integrate import solve_ivp

from UniversalFun import model


def main(r):
    sol = solve_ivp(model, [0, 100], r[1], args=r[0], dense_output=True, method="BDF")

    return sol['t'], sol['y']


def multiThredFunInstaPlot(r):
    with ThreadPoolExecutor(max_workers=10) as executor:
        for i, result in enumerate(executor.map(main, r)):
            plt.plot(result)
            plt.savefig(str(r[i]) + '.png')
            plt.cla()
            plt.close()


def multiThredFun(r, pltx, plty, pltxy):
    with ThreadPoolExecutor(max_workers=10) as executor:
        for result in executor.map(main, r):
            t, y = result
            pltx.plot(t, y[1])
            plty.plot(t, y[0])
            pltxy.plot(y[0], y[1])


def staticAB(a=1.0, b=1.0, n=linspace(0, 10, 20), l=linspace(0, 10, 20)):
    r = []

    for y in n:
        for x in l:
            r.append(((a, b), (x, y)))

    pltX, axX = plt.subplots()
    pltY, axY = plt.subplots()
    pltXY, azXY = plt.subplots()

    multiThredFun(r, axX, axY, azXY)

    axX.set_title(r"α = " + str(a) + r"; β = " + str(b))
    pltX.savefig(f"DuzyZakres_2={a}_b={b}_X.png")

    axY.set_title(r"α = " + str(a) + r"; β = " + str(b))
    pltY.savefig(f"DuzyZakres_2={a}_b={b}_Y.png")

    azXY.set_title(r"α = " + str(a) + r"; β = " + str(b))
    pltXY.savefig(f"DuzyZakres_2={a}_b={b}_XY.png")
```

```python


if __name__ == "__main__":
    a = float(input("podaj a: "))
    b = float(input("podaj b: "))
    x0 = float(input("podaj x0: "))
    x1 = float(input("podaj x1: "))
    nx = int(input("podaj ilosc pkt na x: "))
    y0 = float(input("podaj y0: "))
    y1 = float(input("podaj y1: "))
    ny = int(input("podaj ilosc pkt na y: "))

    n = linspace(x0, x1, nx)
    l = linspace(y0, y1, ny)
    staticAB(a, b, n, l)
```

5.5 Załącznik 3

Kod źródłowy 5: Bifurkacje.

```
import matplotlib.pyplot as plt

from scipy.integrate import odeint, solve_ivp
from numpy import linspace
from concurrent.futures import ThreadPoolExecutor
from UniversalFun import model


def mainOneel(r):
    sol = solve_ivp(model, [0, 10], r[1], args=r[0], dense_output=True, method="BDF")

    return sol['y'], r[2]


def multiThredFunOneEl(r):
    x = []
    y = []
    t = []

    with ThreadPoolExecutor(max_workers=10) as executor:
        for result in executor.map(mainOneel, r):
            x.append(result[0][0][-1])
            y.append(result[0][1][-1])
            t.append(result[1])

    return x, y, t


def bifurkacja(aw, bw, xw, yw, zmienna='a', offset=0.5, npkt=10000):
    A, B, X, Y = [aw], [bw], [xw], [yw]
    if zmienna == 'a':
        A = linspace(A[0] - offset, A[0] + offset, npkt)
        tabelka = [(A, "a")]
    elif zmienna == 'b':
        B = linspace(B[0] - offset, B[0] + offset, npkt)
        tabelka = [(B, "b")]
    elif zmienna == 'ab':
        A = linspace(A[0] - offset, A[0] + offset, npkt)
        B = linspace(B[0] - offset, B[0] + offset, npkt)
        tabelka = [(A, "a"), (B, "b")]
    else:
        print(f"Podano nie obslugiwany parametr {zmienna}, dopuscalne wartosci to: \"a\" \"b
        return

    z = []

    for i, a in enumerate(A):
        for j, b in enumerate(B):
            for k, x in enumerate(X):
                for l, y in enumerate(Y):
                    z.append(((a, b), (x, y), (i, j, k, l)))

    _x, y, time = multiThredFunOneEl(z)

```

```
    for w, name in zip([_x, y], ["x", "y"]):
        for t in tabelka:
            plt.plot(t[0], w, linewidth=0, markersize=1, marker='.')
            plt.savefig(f"a={aw}, b={bw}, x={xw}, y={yw}, z={zmienna}, i= {t[1]}, j= {name}.
            plt.cla()
            plt.close()


if __name__ == "__main__":
    a = float(input("podaj a0: "))
    b = float(input("podaj b0: "))
    x0 = float(input("podaj x0: "))
    y0 = float(input("podaj y0: "))

    zmienna = input("podaj woku ktorej zmiennej szukamy Bifurkacji\n Dopuszcalne wartosci \"

    offset = float(input("podaj plus minus ile bedziemy zmieniac zakres parametru: "))

    n = int(input("podaj ile pktow w tym zakresie chcemy rozwazyc: "))

    bifurkacja(a, b, x0, y0, zmienna, offset, n)
```

}

5.6 Uruchomienie

Załącznik zerowy jest konieczny w ramach odpalenia załącznika 2 i 3.

Żeby wykonac kazdy z plików nalezy zainstalwoac Pythona (w wykonaniu był uzywany python 3.10 ale program powinien działac poprawnie równiez na wersji 3.8 i wyzej) Oraz pakiety do Pythona: -numpy -Scipy -Matplotlib

Programy pracują wielowątkowo, Dokładniej wieloprocesowo gdyż niestety jak na razie Python nie wspiera w pełni wielowontkowosci planują to zmienić w ramach wersji 3.12 lub 3.13. Ilość procesów została ustalona na 10 jako ze tyle pozwalał mój procesor bez problemów z używaniem komputera w trakcie działania programu. W celu zmienienia tego należy edytować odpowiedni parametr w programie. Nie powiano być problemu z włączaniem programu z nisza ilością rdzeni lecz mogą pojawić się komunikaty o błędach i wyniki mogą być nie kompletne.

Każdy z pod programów wykona i zapisze do pliku wykresy o nazwach w określonej konwencji.